

Complex Analysis

December 17, 2025

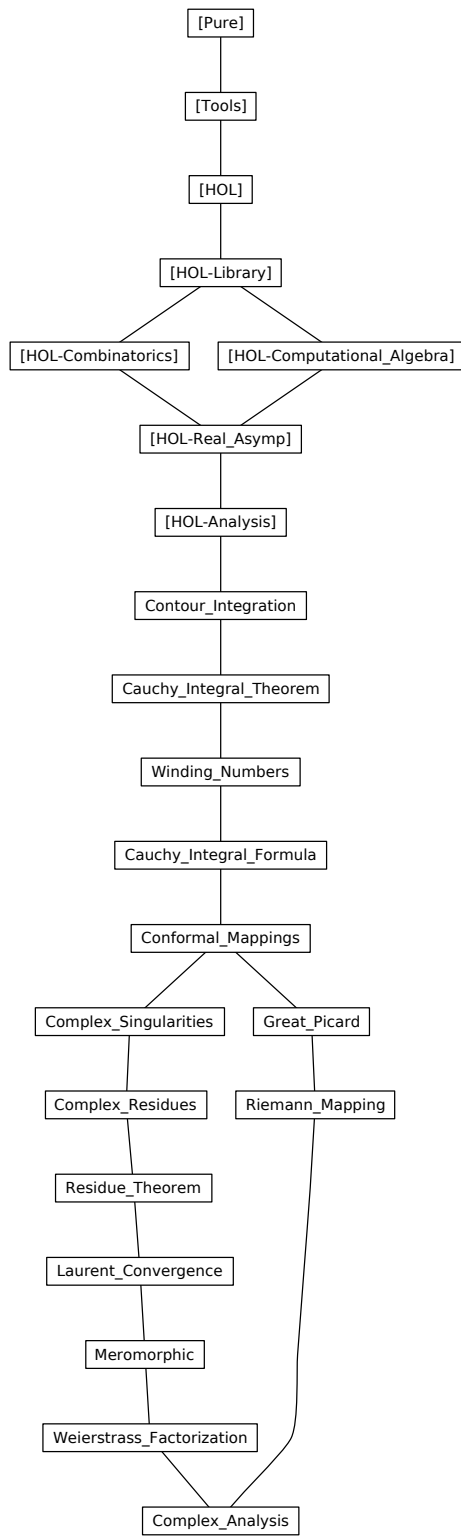
Contents

1	Contour integration	6
1.1	Definition	6
1.2	Reversing a path	9
1.3	Joining two paths together	10
1.4	Shifting the starting point of a (closed) path	14
1.5	More about straight-line paths	17
1.6	Relation to subpath construction	18
1.7	Cauchy's theorem where there's a primitive	22
1.8	Arithmetical combining theorems	24
1.9	Operations on path integrals	25
1.10	Arithmetic theorems for path integrability	27
1.11	Reversing a path integral	30
1.12	Reversing the order in a double path integral	32
1.13	Partial circle path	35
1.14	Special case of one complete circle	45
1.15	Uniform convergence of path integral	48
2	Complex Path Integrals and Cauchy's Integral Theorem	51
2.1	The key quadrisection step	53
2.2	Cauchy's theorem for triangles	55
2.3	Version needing function holomorphic in interior only	60
2.4	Version allowing finite number of exceptional points	66
2.5	Cauchy's theorem for an open starlike set	68
2.6	Cauchy's theorem for a convex set	71
2.7	Generalize integrability to local primitives	74
2.8	Homotopy forms of Cauchy's theorem	84
3	Winding numbers	88
3.1	Definition	89
3.1.1	Useful sufficient conditions for the winding number to be positive	97

3.2	The winding number is an integer	99
3.3	Continuity of winding number and invariance on connected sets	105
3.4	The winding number is constant on a connected region	109
3.5	Winding number is zero "outside" a curve	110
3.6	More winding number properties	117
3.7	Winding number for a triangle	121
3.8	Winding numbers for simple closed paths	124
3.9	Winding number for rectangular paths	141
4	Cauchy's Integral Formula	144
4.1	Proof	144
4.2	General stepping result for derivative formulas	146
4.3	Existence of all higher derivatives	151
4.4	Morera's theorem	155
4.5	Combining theorems for higher derivatives including Leibniz rule	157
4.6	A holomorphic function is analytic, i.e. has local power series	165
4.7	The Liouville theorem and the Fundamental Theorem of Algebra	168
4.8	Weierstrass convergence theorem	169
4.9	Some more simple/convenient versions for applications	176
4.10	On analytic functions defined by a series	177
4.11	Equality between holomorphic functions, on open ball then connected set	182
4.12	Some basic lemmas about poles/singularities	183
4.13	General, homology form of Cauchy's theorem	187
4.14	Cauchy's inequality and more versions of Liouville	201
4.15	Complex functions and power series	205
5	Conformal Mappings and Consequences of Cauchy's Integral Theorem	211
5.1	Analytic continuation	212
5.2	Open mapping theorem	215
5.3	Maximum modulus principle	219
5.4	Factoring out a zero according to its order	221
5.5	Entire proper functions are precisely the non-trivial polynomials	230
5.6	Relating invertibility and nonvanishing of derivative	232
5.6.1	Hence a nice clean inverse function theorem	235
5.6.2	Holomorphism of covering maps and lifts.	236
5.7	The Schwarz Lemma	238
5.8	The Schwarz reflection principle	241
5.9	Bloch's theorem	247
6	The Great Picard Theorem and its Applications	256

6.1	Schottky's theorem	256
6.2	The Little Picard Theorem	265
6.3	The Arzelà–Ascoli theorem	270
6.3.1	Montel's theorem	273
6.4	Some simple but useful cases of Hurwitz's theorem	278
6.5	The Great Picard theorem	283
7	Moebius functions, Equivalents of Simply Connected Sets, Riemann Mapping Theorem	297
7.1	Moebius functions are biholomorphisms of the unit disc . . .	297
7.2	A big chain of equivalents of simple connectedness for an open set	299
7.3	A further chain of equivalences about components of the com- plement of a simply connected set	315
7.4	Further equivalences based on continuous logs and sqrts . . .	324
7.5	More Borsukian results	326
7.6	Finally, the Riemann Mapping Theorem	327
7.7	Applications to Winding Numbers	328
7.8	The winding number defines a continuous logarithm for the path itself	328
7.9	Winding number equality is the same as path/loop homotopy in $\mathbb{C} - 0$	331
7.10	Non-essential singular points	334
7.11	Isolated singularities	339
7.12	The order of non-essential singularities (i.e. removable singu- larities or poles)	358
7.13	Isolated points	387
7.14	Isolated zeros	388
7.15	Definition of residues	391
7.16	Poles and residues of some well-known functions	403
8	The Residue Theorem, the Argument Principle and Rouché's Theorem	403
8.1	Cauchy's residue theorem	405
8.2	The argument principle	414
8.3	Coefficient asymptotics for generating functions	419
8.4	Rouche's theorem	422
8.5	More Laurent expansions	479
8.6	Formal convergence versus analytic convergence	480
8.7	Remove singular points	481
8.8	Meromorphicity	489
8.9	Nice meromorphicity	496
8.10	Closure properties and proofs for individual functions	500
8.11	Meromorphic functions and zorder	509

8.12	More on poles and zeros	513
9	The Weierstraß Factorisation Theorem	522
9.1	The elementary factors	522
9.2	Infinite products of elementary factors	526
9.3	Writing a quotient as an exponential	532
9.4	Constructing the sequence of zeros	535
9.5	The factorisation theorem for holomorphic functions	541
9.6	The factorisation theorem for meromorphic functions	548



1 Contour integration

theory *Contour_Integration*

imports *HOL-Analysis.Analysis*

begin

lemma *hopital_complex_simple*:

assumes $(f \text{ has_field_derivative } f') \text{ (at } z)$

assumes $(g \text{ has_field_derivative } g') \text{ (at } z)$

assumes $f z = 0 \ g z = 0 \ g' \neq 0 \ f' / g' = c$

shows $((\lambda w. f w / g w) \longrightarrow c) \text{ (at } z)$

proof –

have *eventually* $(\lambda w. w \neq z) \text{ (at } z)$

by *(auto simp: eventually_at_filter)*

hence *eventually* $(\lambda w. ((f w - f z) / (w - z)) / ((g w - g z) / (w - z)) = f w / g w) \text{ (at } z)$

by *eventually_elim (simp add: assms field_split_simps)*

moreover **have** $((\lambda w. ((f w - f z) / (w - z)) / ((g w - g z) / (w - z))) \longrightarrow f' / g') \text{ (at } z)$

by *(intro tendsto_divide has_field_derivativeD assms)*

ultimately **have** $((\lambda w. f w / g w) \longrightarrow f' / g') \text{ (at } z)$

by *(blast intro: Lim_transform_eventually)*

with *assms* **show** *?thesis* **by** *simp*

qed

1.1 Definition

This definition is for complex numbers only, and does not generalise to line integrals in a vector field

definition *has_contour_integral* :: $(\text{complex} \Rightarrow \text{complex}) \Rightarrow \text{complex} \Rightarrow (\text{real} \Rightarrow \text{complex}) \Rightarrow \text{bool}$

(infixr $\langle \text{has}'_contour_integral \rangle$ 50)

where $(f \text{ has_contour_integral } i) \ g \equiv$

$((\lambda x. f(g x) * \text{vector_derivative } g \text{ (at } x \text{ within } \{0..1\}))$

$\text{has_integral } i) \ \{0..1\}$

definition *contour_integrable_on*

(infixr $\langle \text{contour}'_integrable_on \rangle$ 50)

where $f \text{ contour_integrable_on } g \equiv \exists i. (f \text{ has_contour_integral } i) \ g$

definition *contour_integral*

where $\text{contour_integral } g \ f \equiv \text{SOME } i. (f \text{ has_contour_integral } i) \ g \ \vee \neg f \text{ contour_integrable_on } g \wedge i=0$

lemma *not_integrable_contour_integral*: $\neg f \text{ contour_integrable_on } g \implies \text{contour_integral } g \ f = 0$

unfolding *contour_integrable_on_def* *contour_integral_def* **by** *blast*

lemma *contour_integral_unique*: $(f \text{ has_contour_integral } i) \ g \implies \text{contour_integral}$

$g \ f = i$
unfolding *contour_integral_def has_contour_integral_def contour_integrable_on_def*
using *has_integral_unique* **by** *blast*

lemma *has_contour_integral_cong*:
assumes $\bigwedge z. z \in \text{path_image } g \implies f \ z = f' \ z \ g = g' \ c = c'$
shows $(f \text{ has_contour_integral } c) \ g \longleftrightarrow (f' \text{ has_contour_integral } c') \ g'$
unfolding *has_contour_integral_def* *assms(2,3)*
by (*intro has_integral_cong*) (*auto simp: assms path_image_def intro!: assms(1)*)

lemma *has_contour_integral_eqpath*:
 $\llbracket (f \text{ has_contour_integral } y) \ p; f \text{ contour_integrable_on } \gamma;$
 $\text{contour_integral } p \ f = \text{contour_integral } \gamma \ f \rrbracket$
 $\implies (f \text{ has_contour_integral } y) \ \gamma$
using *contour_integrable_on_def contour_integral_unique* **by** *auto*

lemma *has_contour_integral_integral*:
 $f \text{ contour_integrable_on } i \implies (f \text{ has_contour_integral } (\text{contour_integral } i \ f))$
 i
by (*metis contour_integral_unique contour_integrable_on_def*)

lemma *has_contour_integral_unique*:
 $(f \text{ has_contour_integral } i) \ g \implies (f \text{ has_contour_integral } j) \ g \implies i = j$
using *has_integral_unique*
by (*auto simp: has_contour_integral_def*)

lemma *has_contour_integral_translate*:
 $(f \text{ has_contour_integral } I) \ ((+) \ z \circ g) \longleftrightarrow ((\lambda x. f \ (x + z)) \text{ has_contour_integral } I) \ g$
by (*simp add: has_contour_integral_def add_ac*)

lemma *contour_integrable_translate*:
 $f \text{ contour_integrable_on } ((+) \ z \circ g) \longleftrightarrow (\lambda x. f \ (x + z)) \text{ contour_integrable_on } g$
by (*simp add: contour_integrable_on_def has_contour_integral_translate*)

lemma *contour_integral_translate*:
 $\text{contour_integral } ((+) \ z \circ g) \ f = \text{contour_integral } g \ (\lambda x. f \ (x + z))$
by (*simp add: contour_integral_def contour_integrable_translate has_contour_integral_translate*)

lemma *has_contour_integral_integrable*: $(f \text{ has_contour_integral } i) \ g \implies f \text{ contour_integrable_on } g$
using *contour_integrable_on_def* **by** *blast*

Show that we can forget about the localized derivative.

lemma *has_integral_localized_vector_derivative*:
 $((\lambda x. f \ (g \ x) * \text{vector_derivative } p \ (\text{at } x \text{ within } \{a..b\}))) \text{ has_integral } i) \ \{a..b\}$
 \longleftrightarrow
 $((\lambda x. f \ (g \ x) * \text{vector_derivative } p \ (\text{at } x))) \text{ has_integral } i) \ \{a..b\}$

```

proof –
  have *:  $\{a..b\} - \{a,b\} = \text{interior } \{a..b\}$ 
    by (simp add: atLeastAtMost_diff_ends)
  show ?thesis
    by (rule has_integral_spike_eq [of  $\{a,b\}$ ]) (auto simp: at_within_interior [of
      _  $\{a..b\}$ ])
qed

lemma integrable_on_localized_vector_derivative:
   $(\lambda x. f (g x) * \text{vector\_derivative } p \text{ (at } x \text{ within } \{a..b\})) \text{ integrable\_on } \{a..b\}$ 
 $\longleftrightarrow$ 
   $(\lambda x. f (g x) * \text{vector\_derivative } p \text{ (at } x)) \text{ integrable\_on } \{a..b\}$ 
  by (simp add: integrable_on_def has_integral_localized_vector_derivative)

lemma has_contour_integral:
   $(f \text{ has\_contour\_integral } i) g \longleftrightarrow$ 
   $((\lambda x. f (g x) * \text{vector\_derivative } g \text{ (at } x)) \text{ has\_integral } i) \{0..1\}$ 
  by (simp add: has_integral_localized_vector_derivative has_contour_integral_def)

lemma contour_integrable_on:
   $f \text{ contour\_integrable\_on } g \longleftrightarrow$ 
   $(\lambda t. f(g t) * \text{vector\_derivative } g \text{ (at } t)) \text{ integrable\_on } \{0..1\}$ 
  by (simp add: has_contour_integral integrable_on_def contour_integrable_on_def)

lemma has_contour_integral_mirror_iff:
  assumes valid_path g
  shows  $(f \text{ has\_contour\_integral } I) (-g) \longleftrightarrow ((\lambda x. -f (-x)) \text{ has\_contour\_integral } I) g$ 
proof –
  from assms have g piecewise_differentiable_on  $\{0..1\}$ 
    by (auto simp: valid_path_def piecewise_C1_imp_differentiable)
  then obtain S where finite S and S:  $\bigwedge x. x \in \{0..1\} - S \implies g \text{ differentiable at } x \text{ within } \{0..1\}$ 
    unfolding piecewise_differentiable_on_def by blast
  have S': g differentiable at x if  $x \in \{0..1\} - (\{0, 1\} \cup S)$  for x
    proof –
      from that have  $x \in \text{interior } \{0..1\}$  by auto
      with S[of x] that show ?thesis by (auto simp: at_within_interior[of _  $\{0..1\}$ ])
    qed
  have  $(f \text{ has\_contour\_integral } I) (-g) \longleftrightarrow$ 
     $((\lambda x. f (-g x) * \text{vector\_derivative } (-g) \text{ (at } x)) \text{ has\_integral } I) \{0..1\}$ 
    by (simp add: has_contour_integral)
  also have ...  $\longleftrightarrow ((\lambda x. -f (-g x) * \text{vector\_derivative } g \text{ (at } x)) \text{ has\_integral } I) \{0..1\}$ 
    by (intro has_integral_spike_finite_eq[of S  $\cup \{0, 1\}$ ])
    (insert  $\langle \text{finite } S \rangle$  S', auto simp: o_def fun_Compl_def)
  also have ...  $\longleftrightarrow ((\lambda x. -f (-x)) \text{ has\_contour\_integral } I) g$ 
    by (simp add: has_contour_integral)
  finally show ?thesis .

```


qed

lemma *contour_integral_on_mirror_iff*:
 assumes *valid_path g*
 shows $f \text{ contour_integrable_on } (-g) \longleftrightarrow (\lambda x. -f (-x)) \text{ contour_integrable_on } g$
 by (auto simp: contour_integrable_on_def has_contour_integral_mirror_iff assms)

lemma *contour_integral_mirror*:
 assumes *valid_path g*
 shows $\text{contour_integral } (-g) f = \text{contour_integral } g (\lambda x. -f (-x))$
proof (cases *f contour_integrable_on (-g)*)
 case True with *contour_integral_unique* *assms* **show** ?thesis
 by (auto simp: contour_integrable_on_def has_contour_integral_mirror_iff)
 next
 case False **then show** ?thesis
 by (simp add: *assms* contour_integral_on_mirror_iff not_integrable_contour_integral)
 qed

1.2 Reversing a path

lemma *has_contour_integral_reversepath*:
 assumes *valid_path g* and *f*: $(f \text{ has_contour_integral } i) \ g$
 shows $(f \text{ has_contour_integral } (-i)) \ (\text{reversepath } g)$
proof –
 { fix *S x*
 assume *xs*: $g \text{ C1_differentiable_on } (\{0..1\} - S) \ x \notin (-) \ 1 \wedge S \ 0 \leq x \leq 1$
 have $\text{vector_derivative } (\lambda x. g (1 - x)) \ (\text{at } x \text{ within } \{0..1\}) =$
 $-\text{vector_derivative } g \ (\text{at } (1 - x) \text{ within } \{0..1\})$
proof –
 obtain *f'* where *f'*: $(g \text{ has_vector_derivative } f') \ (\text{at } (1 - x))$
 using *xs*
 by (force simp: has_vector_derivative_def C1_differentiable_on_def)
 have $(g \circ (\lambda x. 1 - x)) \text{ has_vector_derivative } -1 *_{\mathbb{R}} f' \ (\text{at } x)$
 by (intro vector_diff_chain_within has_vector_derivative_at_within [OF
f] derivative_eq_intros | simp)+
 then have *mf'*: $((\lambda x. g (1 - x)) \text{ has_vector_derivative } -f') \ (\text{at } x)$
 by (simp add: o_def)
 show ?thesis
 using *xs*
 by (auto simp: vector_derivative_at_within_ivl [OF *mf'*] vector_derivative_at_within_ivl
 [OF *f*])
 qed
 } **note** * = *this*
 obtain *S* where *S*: *continuous_on* $\{0..1\} \ g$ *finite* *S* $g \text{ C1_differentiable_on } \{0..1\} - S$
 using *assms* **by** (auto simp: valid_path_def piecewise_C1_differentiable_on_def)
 have $((\lambda x. - (f (g (1 - x)) * \text{vector_derivative } g \ (\text{at } (1 - x) \text{ within } \{0..1\})))$
has_integral -i)

```

    {0..1}
    using has_integral_affinity01 [where m= -1 and c=1, OF f [unfolded
has_contour_integral_def]]
    by (simp add: has_integral_neg)
    then show ?thesis
    using S
    unfolding reversepath_def has_contour_integral_def
    by (rule_tac S = ( $\lambda x. 1 - x$ ) ' S in has_integral_spike_finite) (auto simp: *)
qed

```

```

lemma contour_integrable_reversepath:
  valid_path g  $\implies$  f contour_integrable_on g  $\implies$  f contour_integrable_on
(reversepath g)
  using has_contour_integral_reversepath contour_integrable_on_def by blast

```

```

lemma contour_integrable_reversepath_eq:
  valid_path g  $\implies$  (f contour_integrable_on (reversepath g)  $\longleftrightarrow$  f contour_integrable_on
g)
  using contour_integrable_reversepath valid_path_reversepath by fastforce

```

```

lemma contour_integral_reversepath:
  assumes valid_path g
  shows contour_integral (reversepath g) f = - (contour_integral g f)
proof (cases f contour_integrable_on g)
  case True then show ?thesis
    by (simp add: assms contour_integral_unique has_contour_integral_integral
has_contour_integral_reversepath)
  next
  case False then have  $\neg$  f contour_integrable_on (reversepath g)
    by (simp add: assms contour_integrable_reversepath_eq)
  with False show ?thesis by (simp add: not_integrable_contour_integral)
qed

```

1.3 Joining two paths together

```

lemma has_contour_integral_join:
  assumes (f has_contour_integral i1) g1 (f has_contour_integral i2) g2
    valid_path g1 valid_path g2
  shows (f has_contour_integral (i1 + i2)) (g1 +++ g2)
proof -
  obtain s1 s2
    where s1: finite s1  $\forall x \in \{0..1\} - s1. g1$  differentiable at x
      and s2: finite s2  $\forall x \in \{0..1\} - s2. g2$  differentiable at x
    using assms
  by (auto simp: valid_path_def piecewise_C1_differentiable_on_def C1_differentiable_on_eq)
  have 1: (( $\lambda x. f (g1 x) * \text{vector\_derivative } g1 \text{ (at } x)$ ) has_integral i1) {0..1}
  and 2: (( $\lambda x. f (g2 x) * \text{vector\_derivative } g2 \text{ (at } x)$ ) has_integral i2) {0..1}
  using assms
  by (auto simp: has_contour_integral)

```

```

  have i1: (( $\lambda x. (2 * f (g1 (2 * x))) * \text{vector\_derivative } g1 \text{ (at (2 * x))}$ ) has_integral
i1) {0..1/2}
  and i2: (( $\lambda x. (2 * f (g2 (2 * x - 1))) * \text{vector\_derivative } g2 \text{ (at (2 * x - 1))}$ )
has_integral i2) {1/2..1}
  using has_integral_affinity01 [OF 1, where m= 2 and c=0, THEN has_integral_cmul
[where c=2]]
  has_integral_affinity01 [OF 2, where m= 2 and c=-1, THEN has_integral_cmul
[where c=2]]
  by (simp_all only: image_affinity_atLeastAtMost_div_diff, simp_all add:
scaleR_conv_of_real mult_ac)
  have g1: vector_derivative ( $\lambda x. \text{if } x \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)$ )
(at z) =
    2 *R vector_derivative g1 (at (z * 2))
  if 0 ≤ z * 2 < 1 * 2 ∉ s1 for z
proof (rule vector_derivative_at [OF has_vector_derivative_transform_within])
  show 0 < |z - 1/2|
  using that by auto
  have (( $\lambda x. 2 * \text{vector\_derivative } g1 \text{ (at (2 * x))}$ ) (at z)
  by (simp add: has_vector_derivative_def has_derivative_def bounded_linear_mult_left)
  moreover have (g1 has_vector_derivative vector_derivative g1 (at (z * 2)))
  (at (2 * z))
  using s1 that by (auto simp: algebra_simps vector_derivative_works)
  ultimately
  show (( $\lambda x. g1 (2 * x)$ ) has_vector_derivative 2 *R vector_derivative g1 (at
(z * 2))) (at z)
  by (intro vector_diff_chain_at [simplified o_def])
qed (use that in ⟨simp_all add: dist_real_def abs_if_split: if_split_asm⟩)

  have g2: vector_derivative ( $\lambda x. \text{if } x \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)$ )
(at z) =
    2 *R vector_derivative g2 (at (z * 2 - 1))
  if 1 < z * 2 ≤ 1 * 2 - 1 ∉ s2 for z
proof (rule vector_derivative_at [OF has_vector_derivative_transform_within])
  show 0 < |z - 1/2|
  using that by auto
  have (( $\lambda x. 2 * x - 1$ ) has_vector_derivative 2) (at z)
  by (simp add: has_vector_derivative_def has_derivative_def bounded_linear_mult_left)
  moreover have (g2 has_vector_derivative vector_derivative g2 (at (z * 2 -
1))) (at (2 * z - 1))
  using s2 that by (auto simp: algebra_simps vector_derivative_works)
  ultimately
  show (( $\lambda x. g2 (2 * x - 1)$ ) has_vector_derivative 2 *R vector_derivative g2
(at (z * 2 - 1))) (at z)
  by (intro vector_diff_chain_at [simplified o_def])
qed (use that in ⟨simp_all add: dist_real_def abs_if_split: if_split_asm⟩)

  have (( $\lambda x. f ((g1 +++ g2) x) * \text{vector\_derivative } (g1 +++ g2) \text{ (at } x)$ ) has_integral
i1) {0..1/2}
  proof (rule has_integral_spike_finite [OF _ i1])

```

```

  show finite (insert (1/2) ((*) 2 - 's1))
    using s1 by (force intro: finite_vimageI [where h = (*)2] inj_onI)
  qed (auto simp add: joinpaths_def scaleR_conv_of_real mult_ac g1)
  moreover have (( $\lambda x. f ((g1 +++ g2) x) * \text{vector\_derivative } (g1 +++ g2) (at x)$ ) has_integral i2) {1/2..1}
  proof (rule has_integral_spike_finite [OF _ _ i2])
    show finite (insert (1/2) (( $\lambda x. 2 * x - 1$ ) - 's2))
      using s2 by (force intro: finite_vimageI [where h =  $\lambda x. 2*x-1$ ] inj_onI)
    qed (auto simp add: joinpaths_def scaleR_conv_of_real mult_ac g2)
  ultimately
  show ?thesis
    by (simp add: has_contour_integral has_integral_combine [where c = 1/2])
  qed

```

```

lemma contour_integrable_joinI:
  assumes f contour_integrable_on g1 f contour_integrable_on g2
    valid_path g1 valid_path g2
  shows f contour_integrable_on (g1 +++ g2)
  using assms
  by (meson has_contour_integral_join contour_integrable_on_def)

```

```

lemma contour_integrable_joinD1:
  assumes f contour_integrable_on (g1 +++ g2) valid_path g1
  shows f contour_integrable_on g1
proof -
  obtain s1
    where s1: finite s1  $\forall x \in \{0..1\} - s1. g1$  differentiable at x
  using assms by (auto simp: valid_path_def piecewise_C1_differentiable_on_def
    C1_differentiable_on_eq)
  have ( $\lambda x. f ((g1 +++ g2) (x/2)) * \text{vector\_derivative } (g1 +++ g2) (at (x/2))$ )
    integrable_on {0..1}
    using assms integrable_affinity [of _ 0 1/2 1/2 0] integrable_on_subbox
    [where a=0 and b=1/2]
    by (fastforce simp: contour_integrable_on)
  then have *: ( $\lambda x. (f ((g1 +++ g2) (x/2)) / 2) * \text{vector\_derivative } (g1 +++ g2) (at (x/2))$ )
    integrable_on {0..1}
    by (auto dest: integrable_cmul [where c=1/2] simp: scaleR_conv_of_real)
  have g1: vector_derivative ( $\lambda x. \text{if } x*2 \leq 1 \text{ then } g1 (2*x) \text{ else } g2 (2*x - 1)$ )
    (at (z/2)) =
    2 *R vector_derivative g1 (at z)
    if 0 < z z < 1 z  $\notin$  s1 for z
  proof (rule vector_derivative_at [OF has_vector_derivative_transform_within])
    show 0 < |(z - 1)/2|
      using that by auto
    have §: (( $\lambda x. x * 2$ ) has_vector_derivative 2) (at (z/2))
      using s1 by (auto simp: has_vector_derivative_def has_derivative_def
        bounded_linear_mult_left)
    have (g1 has_vector_derivative vector_derivative g1 (at z)) (at z)
      using s1 that by (auto simp: vector_derivative_works)

```

```

    then show (( $\lambda x. g1 (2 * x)$ ) has_vector_derivative  $2 *_R$  vector_derivative  $g1$ 
    (at  $z$ )) (at ( $z/2$ ))
      using vector_diff_chain_at [OF §] by (auto simp: field_simps o_def)
    qed (use that in <simp_all add: field_simps dist_real_def abs_if_split: if_split_asm>)
    have fin01: finite ( $\{0, 1\} \cup s1$ )
      by (simp add: s1)
    show ?thesis
      unfolding contour_integrable_on
      by (intro integrable_spike_finite [OF fin01 _ *]) (auto simp: joinpaths_def
scaleR_conv_of_real g1)
  qed

```

lemma contour_integrable_joinD2:

```

  assumes f contour_integrable_on ( $g1 +++ g2$ ) valid_path  $g2$ 
  shows f contour_integrable_on  $g2$ 
proof -
  obtain s2
    where s2: finite s2  $\forall x \in \{0..1\} - s2. g2$  differentiable at  $x$ 
    using assms by (auto simp: valid_path_def piecewise_C1_differentiable_on_def
C1_differentiable_on_eq)
  have ( $\lambda x. f ((g1 +++ g2) (x/2 + 1/2)) * \text{vector\_derivative } (g1 +++ g2) (at$ 
( $x/2 + 1/2$ ))) integrable_on  $\{0..1\}$ 
    using assms integrable_affinity [of _  $1/2::\text{real } 1\ 1/2\ 1/2$ ]
      integrable_on_subbox [where  $a=1/2$  and  $b=1$ ]
    by (fastforce simp: contour_integrable_on image_affinity_atLeastAtMost_diff)
  then have *: ( $\lambda x. (f ((g1 +++ g2) (x/2 + 1/2))/2) * \text{vector\_derivative } (g1$ 
 $+++ g2) (at (x/2 + 1/2)))$ 
    integrable_on  $\{0..1\}$ 
    by (auto dest: integrable_cmul [where  $c=1/2$ ] simp: scaleR_conv_of_real)
  have  $g2: \text{vector\_derivative } (\lambda x. \text{if } x*2 \leq 1 \text{ then } g1 (2*x) \text{ else } g2 (2*x - 1))$ 
(at ( $z/2 + 1/2$ )) =
     $2 *_R \text{vector\_derivative } g2 (at z)$ 
    if  $0 < z < 1$   $z \notin s2$  for  $z$ 
  proof (rule vector_derivative_at [OF has_vector_derivative_transform_within])
    show  $0 < |z/2|$ 
      using that by auto
    have §: (( $\lambda x. x * 2 - 1$ ) has_vector_derivative  $2$ ) (at (( $1 + z$ )/2))
      using s2 by (auto simp: has_vector_derivative_def has_derivative_def
bounded_linear_mult_left)
    have ( $g2$  has_vector_derivative vector_derivative  $g2 (at z)$ ) (at  $z$ )
      using s2 that by (auto simp: vector_derivative_works)
    then show (( $\lambda x. g2 (2*x - 1)$ ) has_vector_derivative  $2 *_R$  vector_derivative
 $g2 (at z)$ ) (at ( $z/2 + 1/2$ ))
      using vector_diff_chain_at [OF §] by (auto simp: field_simps o_def)
    qed (use that in <simp_all add: field_simps dist_real_def abs_if_split: if_split_asm>)
  have fin01: finite ( $\{0, 1\} \cup s2$ )
    by (simp add: s2)
  show ?thesis
    unfolding contour_integrable_on

```

by (intro integrable_spike_finite [OF fin01 _ *]) (auto simp: joinpaths_def
scaleR_conv_of_real g2)
qed

lemma *contour_integrable_join* [simp]:
 $\llbracket \text{valid_path } g1; \text{valid_path } g2 \rrbracket$
 $\implies f \text{ contour_integrable_on } (g1 \mathrel{+++} g2) \longleftrightarrow f \text{ contour_integrable_on } g1$
 $\wedge f \text{ contour_integrable_on } g2$
using *contour_integrable_joinD1* *contour_integrable_joinD2* *contour_integrable_joinI*
by *blast*

lemma *contour_integral_join* [simp]:
 $\llbracket f \text{ contour_integrable_on } g1; f \text{ contour_integrable_on } g2; \text{valid_path } g1; \text{valid_path } g2 \rrbracket$
 $\implies \text{contour_integral } (g1 \mathrel{+++} g2) f = \text{contour_integral } g1 f + \text{contour_integral } g2 f$
by (simp add: *has_contour_integral_integral* *has_contour_integral_join* *contour_integral_unique*)

1.4 Shifting the starting point of a (closed) path

lemma *has_contour_integral_shiftpath*:
assumes *f*: (*f* *has_contour_integral* *i*) *g* *valid_path* *g*
and *a*: $a \in \{0..1\}$
shows (*f* *has_contour_integral* *i*) (*shiftpath* *a* *g*)
proof –
obtain *S*
where *S*: *finite* *S* **and** *g*: $\forall x \in \{0..1\} - S. g \text{ differentiable at } x$
using *assms* **by** (auto simp: *valid_path_def* *piecewise_C1_differentiable_on_def* *C1_differentiable_on_eq*)
have *: $((\lambda x. f(g\ x) * \text{vector_derivative } g \text{ (at } x)) \text{ has_integral } i) \{0..1\}$
using *assms* **by** (auto simp: *has_contour_integral*)
then have *i*: $i = \text{integral } \{a..1\} (\lambda x. f(g\ x) * \text{vector_derivative } g \text{ (at } x)) +$
 $\text{integral } \{0..a\} (\lambda x. f(g\ x) * \text{vector_derivative } g \text{ (at } x))$
by (smt (verit, ccfv_threshold) *Henstock_Kurzweil_Integration.integral_combine* *a* *add.commute* *atLeastAtMost_iff* *has_integral_iff*)
have *vd1*: $\text{vector_derivative } (\text{shiftpath } a\ g) \text{ (at } x) = \text{vector_derivative } g \text{ (at } (x + a))$
if $0 \leq x + a < 1$ *x* $\notin (\lambda x. x - a) \text{ ` } S$ **for** *x*
unfolding *shiftpath_def*
proof (rule *vector_derivative_at* [OF *has_vector_derivative_transform_within*])
have $((\lambda x. g(x + a)) \text{ has_vector_derivative } \text{vector_derivative } g \text{ (at } (a + x)))$
 $(\text{at } x)$
proof (rule *vector_diff_chain_at* [of _ 1, simplified *o_def* *scaleR_one*])
show $((\lambda x. x + a) \text{ has_vector_derivative } 1) \text{ (at } x)$
by (rule *derivative_eq_intros* | *simp*) +
have *g* *differentiable* *at* $(x + a)$
using *g* *a* **that** **by** *force*
then show (*g* *has_vector_derivative* *vector_derivative* *g* *at* $(a + x))$ *at* $(x$

```

+ a))
  by (metis add.commute vector_derivative_works)
qed
then show (( $\lambda x. g (a + x)$ ) has_vector_derivative vector_derivative g (at (x
+ a))) (at x)
  by (auto simp: field_simps)
show 0 < dist (1 - a) x
  using that by auto
qed (use that in ‹auto simp: dist_real_def›)

have vd2: vector_derivative (shiftpath a g) (at x) = vector_derivative g (at (x
+ a - 1))
  if  $x \leq 1$   $1 < x + a$   $x \notin (\lambda x. x - a + 1) \text{ ' } S$  for x
  unfolding shiftpath_def
proof (rule vector_derivative_at [OF has_vector_derivative_transform_within])
  have (( $\lambda x. g (x + a - 1)$ ) has_vector_derivative vector_derivative g (at
(a+x-1))) (at x)
  proof (rule vector_diff_chain_at [of _ 1, simplified o_def scaleR_one])
    show (( $\lambda x. x + a - 1$ ) has_vector_derivative 1) (at x)
    by (rule derivative_eq_intros | simp)+
    have g differentiable at (x+a-1)
    using g a that by force
    then show (g has_vector_derivative vector_derivative g (at (a+x-1))) (at
(x + a - 1))
    by (metis add.commute vector_derivative_works)
  qed
  then show (( $\lambda x. g (a + x - 1)$ ) has_vector_derivative vector_derivative g (at
(x + a - 1))) (at x)
  by (auto simp: field_simps)
show 0 < dist (1 - a) x
  using that by auto
qed (use that in ‹auto simp: dist_real_def›)

have va1: ( $\lambda x. f (g x) * \text{vector\_derivative } g \text{ (at } x)$ ) integrable_on ({a..1})
  using * a by (fastforce intro: integrable_subinterval_real)
have v0a: ( $\lambda x. f (g x) * \text{vector\_derivative } g \text{ (at } x)$ ) integrable_on ({0..a})
  using * a by (force intro: integrable_subinterval_real)
have finite ({1 - a}  $\cup (\lambda x. x - a) \text{ ' } S$ )
  using S by blast
then have (( $\lambda x. f (\text{shiftpath } a \text{ } g \text{ } x) * \text{vector\_derivative } (\text{shiftpath } a \text{ } g) \text{ (at } x)$ )
has_integral integral {a..1} ( $\lambda x. f (g x) * \text{vector\_derivative } g \text{ (at } x)$ )) {0..1
- a}
  apply (rule has_integral_spike_finite
[where f =  $\lambda x. f(g(a+x)) * \text{vector\_derivative } g \text{ (at } (a+x))$ ])
  subgoal
  using a by (simp add: vd1) (force simp: shiftpath_def add.commute)
  subgoal
  using has_integral_affinity [where m=1 and c=a] integrable_integral [OF
va1]

```

```

    by (force simp add: add.commute)
  done
moreover
have finite ( $\{1 - a\} \cup (\lambda x. x - a + 1) \text{ ` } S$ )
  using  $S$  by blast
then have (( $\lambda x. f$  (shiftpath  $a$   $g$   $x$ ) * vector_derivative (shiftpath  $a$   $g$ ) (at  $x$ ))
  has_integral integral  $\{0..a\}$  ( $\lambda x. f$  ( $g$   $x$ ) * vector_derivative  $g$  (at  $x$ )))
 $\{1 - a..1\}$ 
  apply (rule has_integral_spike_finite
    [where  $f = \lambda x. f(g(a+x-1)) * \text{vector\_derivative } g \text{ (at}(a+x-1))$ ])
  subgoal
    using  $a$  by (simp add: vd2) (force simp: shiftpath_def add.commute)
  subgoal
    using has_integral_affinity [where  $m=1$  and  $c=a-1$ , simplified, OF integrable_integral [OF v0a]]
    by (force simp add: algebra_simps)
  done
ultimately show ?thesis
  using  $a$ 
  by (auto simp: i has_contour_integral intro: has_integral_combine [where  $c = 1-a$ ])
qed

```

```

lemma has_contour_integral_shiftpath_D:
  assumes ( $f$  has_contour_integral  $i$ ) (shiftpath  $a$   $g$ )
    valid_path  $g$  pathfinish  $g = \text{pathstart } g$   $a \in \{0..1\}$ 
  shows ( $f$  has_contour_integral  $i$ )  $g$ 
proof -
  obtain  $S$ 
  where  $S$ : finite  $S$  and  $g$ :  $\forall x \in \{0..1\} - S. g$  differentiable at  $x$ 
  using assms by (auto simp: valid_path_def piecewise_C1_differentiable_on_def
    C1_differentiable_on_eq)
  { fix  $x$ 
    assume  $x$ :  $0 < x < 1$   $x \notin S$ 
    then have  $gx$ :  $g$  differentiable at  $x$ 
      using  $g$  by auto
    have  $\S$ : shiftpath  $(1 - a)$  (shiftpath  $a$   $g$ ) differentiable at  $x$ 
      using assms  $x$ 
      by (intro differentiable_transform_within [OF  $gx$ , of min  $x$   $(1-x)$ ])
        (auto simp: dist_real_def shiftpath_shiftpath_abs_if_split: if_split_asm)
    have vector_derivative  $g$  (at  $x$  within  $\{0..1\}$ ) =
      vector_derivative (shiftpath  $(1 - a)$  (shiftpath  $a$   $g$ )) (at  $x$  within  $\{0..1\}$ )
    apply (rule vector_derivative_at_within_ivl
      [OF has_vector_derivative_transform_within_open
        [where  $f = (\text{shiftpath } (1 - a) (\text{shiftpath } a \text{ } g))$  and  $S =$ 
 $\{0 < .. < 1\} - S$ ]])
    using  $S$  assms  $x$   $\S$ 
    apply (auto simp: finite_imp_closed open_Diff shiftpath_shiftpath
      at_within_interior [of _  $\{0..1\}$ ] vector_derivative_works

```



```

[symmetric])
  done
} note vd = this
have fi: (f has_contour_integral i) (shiftpath (1 - a) (shiftpath a g))
  using assms by (auto intro!: has_contour_integral_shiftpath)
show ?thesis
  unfolding has_contour_integral_def
  proof (rule has_integral_spike_finite [of {0,1}  $\cup$  S, OF _ _ fi [unfolded
has_contour_integral_def]])
    show finite ({0, 1}  $\cup$  S)
    by (simp add: S)
  qed (use S assms vd in  $\langle$ auto simp: shiftpath_shiftpath $\rangle$ )
qed

```

```

lemma has_contour_integral_shiftpath_eq:
  assumes valid_path g pathfinish g = pathstart g a  $\in$  {0..1}
  shows (f has_contour_integral i) (shiftpath a g)  $\longleftrightarrow$  (f has_contour_integral
i) g
  using assms has_contour_integral_shiftpath has_contour_integral_shiftpath_D
by blast

```

```

lemma contour_integrable_on_shiftpath_eq:
  assumes valid_path g pathfinish g = pathstart g a  $\in$  {0..1}
  shows f contour_integrable_on (shiftpath a g)  $\longleftrightarrow$  f contour_integrable_on g
  using assms contour_integrable_on_def has_contour_integral_shiftpath_eq by
auto

```

```

lemma contour_integral_shiftpath:
  assumes valid_path g pathfinish g = pathstart g a  $\in$  {0..1}
  shows contour_integral (shiftpath a g) f = contour_integral g f
  using assms
by (simp add: contour_integral_def contour_integrable_on_def has_contour_integral_shiftpath_eq)

```

1.5 More about straight-line paths

```

lemma has_contour_integral_linepath:
  shows (f has_contour_integral i) (linepath a b)  $\longleftrightarrow$ 
    (( $\lambda x$ . f(linepath a b x) * (b - a)) has_integral i) {0..1}
  by (simp add: has_contour_integral)

```

```

lemma has_contour_integral_trivial [iff]: (f has_contour_integral 0) (linepath a
a)
  by (simp add: has_contour_integral_linepath)

```

```

lemma has_contour_integral_trivial_iff [simp]: (f has_contour_integral i) (linepath
a a)  $\longleftrightarrow$  i=0
  using has_contour_integral_unique by blast

```

```

lemma contour_integral_trivial [simp]: contour_integral (linepath a a) f = 0

```

using *has_contour_integral_trivial* *contour_integral_unique* **by** *blast*

1.6 Relation to subpath construction

lemma *has_contour_integral_subpath_refl* [*iff*]: (*f* *has_contour_integral* 0) (*subpath* *u* *u* *g*)
by (*simp* *add*: *has_contour_integral_subpath_def*)

lemma *contour_integrable_subpath_refl* [*iff*]: *f* *contour_integrable_on* (*subpath* *u* *u* *g*)
using *has_contour_integral_subpath_refl* *contour_integrable_on_def* **by** *blast*

lemma *contour_integral_subpath_refl* [*simp*]: *contour_integral* (*subpath* *u* *u* *g*) *f* = 0
by (*simp* *add*: *contour_integral_unique*)

lemma *has_contour_integral_subpath*:
assumes *f*: *f* *contour_integrable_on* *g* **and** *g*: *valid_path* *g*
and *uv*: *u* ∈ {0..1} *v* ∈ {0..1} *u* ≤ *v*
shows (*f* *has_contour_integral* *integral* {*u..v*} (λ*x*. *f*(*g* *x*) * *vector_derivative* *g* (at *x*)))
 (*subpath* *u* *v* *g*)
proof (*cases* *v=u*)
case *True*
then show ?*thesis*
using *f* **by** (*simp* *add*: *contour_integrable_on_def* *subpath_def* *has_contour_integral*)
next
case *False*
obtain *S* **where** *S*: ∧*x*. *x* ∈ {0..1} − *S* ⇒ *g* *differentiable* at *x* **and** *fs*: *finite* *S*
using *g* **unfolding** *piecewise_C1_differentiable_on_def* *C1_differentiable_on_eq* *valid_path_def* **by** *blast*
have §: (λ*t*. *f* (*g* *t*) * *vector_derivative* *g* (at *t*)) *integrable_on* {*u..v*}
using *contour_integrable_on* *f* *integrable_on_subinterval* *uv* **by** *fastforce*
then have *: ((λ*x*. *f* (*g* ((*v* − *u*) * *x* + *u*)) * *vector_derivative* *g* (at ((*v* − *u*) * *x* + *u*)))
 has_integral (1 / (*v* − *u*)) * *integral* {*u..v*} (λ*t*. *f* (*g* *t*) * *vector_derivative* *g* (at *t*)))
 {0..1}
using *uv* *False* **unfolding** *has_integral_integral*
apply *simp*
apply (*drule* *has_integral_affinity* [**where** *m=v−u* **and** *c=u*, *simplified*])
apply (*simp_all* *add*: *image_affinity_atLeastAtMost_div_diff* *scaleR_conv_of_real*)
apply (*simp* *add*: *divide_simps*)
done

have *vd*: *vector_derivative* (λ*x*. *g* ((*v* − *u*) * *x* + *u*)) (at *x*) = (*v* − *u*) *_R *vector_derivative* *g* (at ((*v* − *u*) * *x* + *u*))
if *x* ∈ {0..1} *x* ∉ (λ*t*. (*v* − *u*) *_R *t* + *u*) − ‘*S* **for** *x*

```

proof (rule vector_derivative_at [OF vector_diff_chain_at [simplified o_def]])
  show (( $\lambda x. (v - u) * x + u$ ) has_vector_derivative  $v - u$ ) (at  $x$ )
  by (intro derivative_eq_intros | simp)+
qed (use S uv mult_left_le [of  $x$   $v - u$ ] that in ⟨auto simp: vector_derivative_works⟩)

```

```

have fin: finite (( $\lambda t. (v - u) *_R t + u$ ) - 'S)
  using fs by (auto simp: inj_on_def False finite_vimageI)
show ?thesis
  unfolding subpath_def has_contour_integral
  apply (rule has_integral_spike_finite [OF fin])
  using has_integral_cmul [OF *, where  $c = v - u$ ] fs assms
  by (auto simp: False vd scaleR_conv_of_real)
qed

```

lemma contour_integrable_subpath:

```

assumes f contour_integrable_on g valid_path g  $u \in \{0..1\}$   $v \in \{0..1\}$ 
shows f contour_integrable_on (subpath u v g)
by (smt (verit, ccfv_threshold) assms contour_integrable_on_def contour_integrable_reversepath_eq
  has_contour_integral_subpath reversepath_subpath valid_path_subpath)

```

lemma has_integral_contour_integral_subpath:

```

assumes f contour_integrable_on g valid_path g  $u \in \{0..1\}$   $v \in \{0..1\}$   $u \leq v$ 
shows (( $\lambda x. f(g\ x) * \text{vector\_derivative } g \text{ (at } x)$ )
  has_integral contour_integral (subpath u v g) f) { $u..v$ }
  (is (?fg has_integral _))

```

proof –

```

  have (?fg has_integral integral { $u..v$ } ?fg) { $u..v$ }
    using assms contour_integrable_on integrable_on_subinterval by fastforce
  then show ?thesis
    by (metis (full_types) assms contour_integral_unique has_contour_integral_subpath)
qed

```

lemma contour_integral_subcontour_integral:

```

assumes f contour_integrable_on g valid_path g  $u \in \{0..1\}$   $v \in \{0..1\}$   $u \leq v$ 
shows contour_integral (subpath u v g) f =
  integral { $u..v$ } ( $\lambda x. f(g\ x) * \text{vector\_derivative } g \text{ (at } x)$ )
using assms has_contour_integral_subpath contour_integral_unique by blast

```

lemma contour_integral_subpath_combine_less:

```

assumes f contour_integrable_on g valid_path g  $u \in \{0..1\}$   $v \in \{0..1\}$   $w \in \{0..1\}$ 
   $u < v$   $v < w$ 
shows contour_integral (subpath u v g) f + contour_integral (subpath v w g)

```

f =

```

  contour_integral (subpath u w g) f
by (smt (verit) Henstock_Kurzweil_Integration.integral_combine assms
  has_integral_contour_integral_subpath has_integral_iff)

```

lemma contour_integral_subpath_combine:

```

assumes  $f$  contour_integrable_on  $g$  valid_path  $g$   $u \in \{0..1\}$   $v \in \{0..1\}$   $w \in \{0..1\}$ 
shows  $\text{contour\_integral } (\text{subpath } u \ v \ g) \ f + \text{contour\_integral } (\text{subpath } v \ w \ g)$ 
 $f =$ 
 $\text{contour\_integral } (\text{subpath } u \ w \ g) \ f$ 
proof (cases  $u \neq v \wedge v \neq w \wedge u \neq w$ )
  case True
    have *:  $\text{subpath } v \ u \ g = \text{reversepath}(\text{subpath } u \ v \ g) \wedge$ 
 $\text{subpath } w \ u \ g = \text{reversepath}(\text{subpath } u \ w \ g) \wedge$ 
 $\text{subpath } w \ v \ g = \text{reversepath}(\text{subpath } v \ w \ g)$ 
    by (auto simp: reversepath_subpath)
    have  $u < v \wedge v < w \vee$ 
 $u < w \wedge w < v \vee$ 
 $v < u \wedge u < w \vee$ 
 $v < w \wedge w < u \vee$ 
 $w < u \wedge u < v \vee$ 
 $w < v \wedge v < u$ 
    using True assms by linarith
    with assms show ?thesis
    using contour_integral_subpath_combine_less [of  $f \ g \ u \ v \ w$ ]
 $\text{contour\_integral\_subpath\_combine\_less}$  [of  $f \ g \ u \ w \ v$ ]
 $\text{contour\_integral\_subpath\_combine\_less}$  [of  $f \ g \ v \ u \ w$ ]
 $\text{contour\_integral\_subpath\_combine\_less}$  [of  $f \ g \ v \ w \ u$ ]
 $\text{contour\_integral\_subpath\_combine\_less}$  [of  $f \ g \ w \ u \ v$ ]
 $\text{contour\_integral\_subpath\_combine\_less}$  [of  $f \ g \ w \ v \ u$ ]
    by (elim disjE) (auto simp: * contour_integral_reversepath contour_integrable_subpath
 $\text{valid\_path\_subpath\_algebra\_simps}$ )
  next
    case False
    with assms show ?thesis
    by (metis add.right_neutral contour_integral_reversepath contour_integral_subpath_refl
 $\text{diff\_0 eq\_diff\_eq add\_0 reversepath\_subpath valid\_path\_subpath}$ )
qed

```

lemma *contour_integral_integral*:

$\text{contour_integral } g \ f = \text{integral } \{0..1\} \ (\lambda x. f \ (g \ x) * \text{vector_derivative } g \ (\text{at } x))$

by (*simp add: contour_integral_def integral_def has_contour_integral contour_integrable_on*)

lemma *contour_integral_cong*:

assumes $g = g' \wedge x. x \in \text{path_image } g \implies f \ x = f' \ x$

shows $\text{contour_integral } g \ f = \text{contour_integral } g' \ f'$

unfolding *contour_integral_integral* **using** *assms*

by (*intro integral_cong*) (*auto simp: path_image_def*)

lemma *contour_integral_spike_finite_simple_path*:

assumes *finite* A *simple_path* $g \ g = g' \wedge x. x \in \text{path_image } g - A \implies f \ x = f' \ x$

shows $\text{contour_integral } g \ f = \text{contour_integral } g' \ f'$

```

unfolding contour_integral_integral
proof (rule integral_spike)
  have finite (g - 'A  $\cap$  {0<.. $<1$ }) using ‹simple_path g› ‹finite A›
  by (intro finite_vimage_IntI simple_path_inj_on) auto
  hence finite ({0, 1}  $\cup$  g - 'A  $\cap$  {0<.. $<1$ }) by auto
  thus negligible ({0, 1}  $\cup$  g - 'A  $\cap$  {0<.. $<1$ }) by (rule negligible_finite)
next
  fix x assume x  $\in$  {0..1} - ({0, 1}  $\cup$  g - 'A  $\cap$  {0<.. $<1$ })
  hence g x  $\in$  path_image g - A by (auto simp: path_image_def)
  with assms show f' (g' x) * vector_derivative g' (at x) = f (g x) * vector_derivative g (at x)
  by simp
qed

```

Contour integral along a segment on the real axis

```

lemma has_contour_integral_linepath_Reals_iff:
  fixes a b :: complex and f :: complex  $\Rightarrow$  complex
  assumes a  $\in$  Reals b  $\in$  Reals Re a < Re b
  shows (f has_contour_integral I) (linepath a b)  $\longleftrightarrow$ 
    (( $\lambda$ x. f (of_real x)) has_integral I) {Re a..Re b}
proof -
  have [simp]: of_real (Re a) = a of_real (Re b) = b and a  $\neq$  b
  using assms by (simp_all add: complex_eq_iff)
  have (( $\lambda$ x. f (of_real x)) has_integral I) (cbox (Re a) (Re b))  $\longleftrightarrow$ 
    (( $\lambda$ x. f (a + b * of_real x - a * of_real x)) has_integral I /R (Re b - Re a)) {0..1}
  by (subst has_integral_affinity_iff [of Re b - Re a _ Re a, symmetric])
    (insert assms, simp_all add: field_simps scaleR_conv_of_real)
  also have ( $\lambda$ x. f (a + b * of_real x - a * of_real x)) =
    ( $\lambda$ x. (f (a + b * of_real x - a * of_real x) * (b - a)) /R (Re b - Re a))
  using ‹a  $\neq$  b› by (auto simp: field_simps fun_eq_iff scaleR_conv_of_real)
  also have (... has_integral I /R (Re b - Re a)) {0..1}  $\longleftrightarrow$ 
    (( $\lambda$ x. f (linepath a b x) * (b - a)) has_integral I) {0..1} using assms
  by (subst has_integral_cmul_iff) (auto simp: linepath_def scaleR_conv_of_real algebra_simps)
  also have ...  $\longleftrightarrow$  (f has_contour_integral I) (linepath a b)
  unfolding has_contour_integral_def
  using has_contour_integral_def has_contour_integral_linepath by presburger
  finally show ?thesis by simp
qed

```

```

lemma contour_integrable_linepath_Reals_iff:
  fixes a b :: complex and f :: complex  $\Rightarrow$  complex
  assumes a  $\in$  Reals b  $\in$  Reals Re a < Re b
  shows (f contour_integrable_on linepath a b)  $\longleftrightarrow$ 
    ( $\lambda$ x. f (of_real x)) integrable_on {Re a..Re b}
  using has_contour_integral_linepath_Reals_iff[OF assms, of f]
  by (auto simp: contour_integrable_on_def integrable_on_def)

```

```

lemma contour_integral_linepath_Reals_eq:
  fixes a b :: complex and f :: complex  $\Rightarrow$  complex
  assumes a  $\in$  Reals b  $\in$  Reals Re a < Re b
  shows contour_integral (linepath a b) f = integral {Re a..Re b} ( $\lambda x$ . f (of_real x))
proof (cases f contour_integrable_on linepath a b)
  case True
  thus ?thesis
    by (metis assms has_contour_integral_integral
        has_contour_integral_linepath_Reals_iff integral_unique)
  next
  case False
  thus ?thesis
    by (simp add: assms contour_integrable_linepath_Reals_iff
        not_integrable_contour_integral not_integrable_integral)
qed

```

1.7 Cauchy's theorem where there's a primitive

```

lemma contour_integral_primitive_lemma:
  fixes f :: complex  $\Rightarrow$  complex and g :: real  $\Rightarrow$  complex
  assumes a  $\leq$  b
    and  $\bigwedge x. x \in S \implies (f \text{ has\_field\_derivative } f' x) \text{ (at } x \text{ within } S)$ 
    and g piecewise_differentiable_on {a..b}  $\bigwedge x. x \in \{a..b\} \implies g x \in S$ 
  shows (( $\lambda x. f'(g x) * \text{vector\_derivative } g \text{ (at } x \text{ within } \{a..b\})$ )
    has_integral (f(g b) - f(g a))) {a..b}
proof -
  obtain K where finite K and K:  $\forall x \in \{a..b\} - K. g \text{ differentiable (at } x \text{ within } \{a..b\})$ 
    and cg: continuous_on {a..b} g
    using assms by (auto simp: piecewise_differentiable_on_def)
  have continuous_on (g ' {a..b}) f
    using assms by (metis DERIV_continuous_on continuous_on_subset image_subsetI)
  then have cfg: continuous_on {a..b} ( $\lambda x. f (g x)$ )
    by (rule continuous_on_compose [OF cg, unfolded o_def])
  { fix x::real
    assume a: a < x and b: x < b and xk: x  $\notin$  K
    then have g differentiable at x within {a..b}
      using K by (simp add: differentiable_at_withinI)
    then have (g has_vector_derivative vector_derivative g (at x within {a..b}))
      (at x within {a..b})
      by (simp add: vector_derivative_works has_field_derivative_def scaleR_conv_of_real)
    then have gdiff: (g has_derivative ( $\lambda u. u * \text{vector\_derivative } g \text{ (at } x \text{ within } \{a..b\})$ ))
      (at x within {a..b})
      by (simp add: has_vector_derivative_def scaleR_conv_of_real)
    have (f has_field_derivative (f' (g x))) (at (g x) within g ' {a..b})
      using assms by (metis a atLeastAtMost_iff b DERIV_subset_image_subset_iff less_eq_real_def)
  }

```

```

    then have fdiff: (f has_derivative (*) (f' (g x))) (at (g x) within g ` {a..b})
      by (simp add: has_field_derivative_def)
    have ((λx. f (g x)) has_vector_derivative f' (g x) * vector_derivative g (at x
within {a..b})) (at x within {a..b})
      using diff_chain_within [OF gdiff fdiff]
      by (simp add: has_vector_derivative_def scaleR_conv_of_real o_def mult_ac)
  } then show ?thesis
    using assms cfg
    by (force simp: at_within_Icc_at intro: fundamental_theorem_of_calculus_interior_strong
[OF ‹finite K›])
qed

```

```

lemma contour_integral_primitive:
  assumes  $\bigwedge x. x \in S \implies (f \text{ has\_field\_derivative } f' x) \text{ (at } x \text{ within } S)$ 
    and valid_path g path_image g  $\subseteq S$ 
  shows (f' has_contour_integral (f(pathfinish g) - f(pathstart g))) g
  using assms
  apply (simp add: valid_path_def path_image_def pathfinish_def pathstart_def
has_contour_integral_def)
  apply (auto intro!: piecewise_C1_imp_differentiable contour_integral_primitive_lemma
[of 0 1 S])
  done

```

```

corollary Cauchy_theorem_primitive:
  assumes  $\bigwedge x. x \in S \implies (f \text{ has\_field\_derivative } f' x) \text{ (at } x \text{ within } S)$ 
    and valid_path g path_image g  $\subseteq S$  pathfinish g = pathstart g
  shows (f' has_contour_integral 0) g
  using assms by (metis diff_self contour_integral_primitive)

```

```

lemma contour_integrable_continuous_linepath:
  assumes continuous_on (closed_segment a b) f
  shows f contour_integrable_on (linepath a b)
proof -
  have continuous_on (closed_segment a b) (λx. f x * (b - a))
    by (rule continuous_intros | simp add: assms)+
  then have continuous_on {0..1} (λx. f (linepath a b x) * (b - a))
    by (metis (no_types, lifting) continuous_on_compose continuous_on_cong
continuous_on_linepath linepath_image_01 o_apply)
  then have (λx. f (linepath a b x)
    * vector_derivative (linepath a b) (at x within {0..1}))
    integrable_on {0..1}
    by (metis (no_types, lifting) continuous_on_cong integrable_continuous_real
vector_derivative_linepath_within)
  then show ?thesis
    by (simp add: contour_integrable_on_def has_contour_integral_def integrable_on_def
[symmetric])
qed

```

lemma *has_field_der_id*: $((\lambda x. x^2/2) \text{ has_field_derivative } x) \text{ (at } x)$
by (rule *has_derivative_imp_has_field_derivative*)
(rule *derivative_intros* | *simp*)⁺

lemma *contour_integral_id* [*simp*]: $\text{contour_integral (linepath } a \ b) (\lambda y. y) = (b^2 - a^2)/2$
using *contour_integral_primitive* [of *UNIV* $\lambda x. x^2/2$ $\lambda x. x$ *linepath* *a* *b*] *contour_integral_unique*
by (*simp add: has_field_der_id*)

lemma *contour_integrable_on_const* [*iff*]: $(\lambda x. c) \text{ contour_integrable_on (linepath } a \ b)$
by (*simp add: contour_integrable_continuous_linepath*)

lemma *contour_integrable_on_id* [*iff*]: $(\lambda x. x) \text{ contour_integrable_on (linepath } a \ b)$
by (*simp add: contour_integrable_continuous_linepath*)

1.8 Arithmetical combining theorems

lemma *has_contour_integral_neg*:
 $(f \text{ has_contour_integral } i) \ g \implies ((\lambda x. -(f \ x)) \text{ has_contour_integral } (-i)) \ g$
by (*simp add: has_integral_neg has_contour_integral_def*)

lemma *has_contour_integral_add*:
 $\llbracket (f1 \text{ has_contour_integral } i1) \ g; (f2 \text{ has_contour_integral } i2) \ g \rrbracket$
 $\implies ((\lambda x. f1 \ x + f2 \ x) \text{ has_contour_integral } (i1 + i2)) \ g$
by (*simp add: has_integral_add has_contour_integral_def algebra_simps*)

lemma *has_contour_integral_diff*:
 $\llbracket (f1 \text{ has_contour_integral } i1) \ g; (f2 \text{ has_contour_integral } i2) \ g \rrbracket$
 $\implies ((\lambda x. f1 \ x - f2 \ x) \text{ has_contour_integral } (i1 - i2)) \ g$
by (*simp add: has_integral_diff has_contour_integral_def algebra_simps*)

lemma *has_contour_integral_lmul*:
 $(f \text{ has_contour_integral } i) \ g \implies ((\lambda x. c * (f \ x)) \text{ has_contour_integral } (c*i)) \ g$
by (*simp add: has_contour_integral_def algebra_simps has_integral_mult_right*)

lemma *has_contour_integral_rmul*:
 $(f \text{ has_contour_integral } i) \ g \implies ((\lambda x. (f \ x) * c) \text{ has_contour_integral } (i*c)) \ g$
by (*simp add: mult.commute has_contour_integral_lmul*)

lemma *has_contour_integral_div*:
 $(f \text{ has_contour_integral } i) \ g \implies ((\lambda x. f \ x / c) \text{ has_contour_integral } (i/c)) \ g$
by (*simp add: field_class.field_divide_inverse*) (*metis has_contour_integral_rmul*)

lemma *has_contour_integral_eq*:
 $\llbracket (f \text{ has_contour_integral } y) \ p; \bigwedge x. x \in \text{path_image } p \implies f \ x = g \ x \rrbracket \implies (g$

has_contour_integral y) p
 by (metis (mono_tags, lifting) *has_contour_integral_def* *has_integral_eq* *image_eqI* *path_image_def*)

lemma *has_contour_integral_bound_linepath*:

assumes (f *has_contour_integral* i) (*linepath* a b)

$0 \leq B$ and B : $\bigwedge x. x \in \text{closed_segment } a \ b \implies \text{norm}(f \ x) \leq B$

shows $\text{norm } i \leq B * \text{norm}(b - a)$

proof –

have $\text{norm } i \leq (B * \text{norm}(b - a)) * \text{measure lborel } (\text{cbox } 0 \ (1::\text{real}))$

proof (rule *has_integral_bound*

[of $\lambda x. f(\text{linepath } a \ b \ x) * \text{vector_derivative}(\text{linepath } a \ b) \text{ (at } x \text{ within } \{0..1\})$])

show $\text{cmod}(f(\text{linepath } a \ b \ x) * \text{vector_derivative}(\text{linepath } a \ b) \text{ (at } x \text{ within } \{0..1\}))$

$\leq B * \text{cmod}(b - a)$

if $x \in \text{cbox } 0 \ 1$ for $x::\text{real}$

using that *box_real*(2) *norm_mult*

by (metis B *linepath_in_path* *mult_right_mono* *norm_ge_zero* *vector_derivative_linepath_within*)

qed (use *assms* *has_contour_integral_def* in *auto*)

then show ?thesis

by (*auto simp: content_real*)

qed

lemma *has_contour_integral_const_linepath*: $((\lambda x. c) \text{ has_contour_integral } c * (b - a))(\text{linepath } a \ b)$

unfolding *has_contour_integral_linepath*

by (metis *content_real* *diff_0_right* *has_integral_const_real* *lambda_one* of *real_1* *scaleR_conv_of_real* *zero_le_one*)

lemma *has_contour_integral_0*: $((\lambda x. 0) \text{ has_contour_integral } 0) \ g$

by (*simp add: has_contour_integral_def*)

lemma *has_contour_integral_is_0*:

$(\bigwedge z. z \in \text{path_image } g \implies f \ z = 0) \implies (f \text{ has_contour_integral } 0) \ g$

by (rule *has_contour_integral_eq* [OF *has_contour_integral_0*]) *auto*

lemma *has_contour_integral_sum*:

$\llbracket \text{finite } s; \bigwedge a. a \in s \implies (f \ a \text{ has_contour_integral } i \ a) \ p \rrbracket$

$\implies ((\lambda x. \text{sum } (\lambda a. f \ a \ x) \ s) \text{ has_contour_integral } \text{sum } i \ s) \ p$

by (*induction s* rule: *finite_induct*) (*auto simp: has_contour_integral_0 has_contour_integral_add*)

1.9 Operations on path integrals

lemma *contour_integral_const_linepath* [*simp*]: *contour_integral* (*linepath* a b)

$(\lambda x. c) = c * (b - a)$

by (rule *contour_integral_unique* [OF *has_contour_integral_const_linepath*])

lemma *contour_integral_neg*: *contour_integral* $g \ (\lambda z. -f \ z) = -\text{contour_integral}$

$g \ f$
by (simp add: contour_integral_integral)

lemma contour_integral_add:

$f1 \text{ contour_integrable_on } g \implies f2 \text{ contour_integrable_on } g \implies \text{contour_integral } g \ (\lambda x. f1 \ x + f2 \ x) =$
 $\text{contour_integral } g \ f1 + \text{contour_integral } g \ f2$
by (simp add: contour_integral_unique has_contour_integral_integral has_contour_integral_add)

lemma contour_integral_diff:

$f1 \text{ contour_integrable_on } g \implies f2 \text{ contour_integrable_on } g \implies \text{contour_integral } g \ (\lambda x. f1 \ x - f2 \ x) =$
 $\text{contour_integral } g \ f1 - \text{contour_integral } g \ f2$
by (simp add: contour_integral_unique has_contour_integral_integral has_contour_integral_diff)

lemma contour_integral_lmul:

shows $f \text{ contour_integrable_on } g$
 $\implies \text{contour_integral } g \ (\lambda x. c * f \ x) = c * \text{contour_integral } g \ f$
by (simp add: contour_integral_unique has_contour_integral_integral has_contour_integral_lmul)

lemma contour_integral_rmul:

shows $f \text{ contour_integrable_on } g$
 $\implies \text{contour_integral } g \ (\lambda x. f \ x * c) = \text{contour_integral } g \ f * c$
by (simp add: contour_integral_unique has_contour_integral_integral has_contour_integral_rmul)

lemma contour_integral_div:

shows $f \text{ contour_integrable_on } g$
 $\implies \text{contour_integral } g \ (\lambda x. f \ x / c) = \text{contour_integral } g \ f / c$
by (simp add: contour_integral_unique has_contour_integral_integral has_contour_integral_div)

lemma contour_integral_eq:

$(\bigwedge x. x \in \text{path_image } p \implies f \ x = g \ x) \implies \text{contour_integral } p \ f = \text{contour_integral } p \ g$
using contour_integral_cong contour_integral_def **by** fastforce

lemma contour_integral_eq_0:

$(\bigwedge z. z \in \text{path_image } g \implies f \ z = 0) \implies \text{contour_integral } g \ f = 0$
by (simp add: has_contour_integral_is_0 contour_integral_unique)

lemma contour_integral_bound_linepath:

shows
 $\llbracket f \text{ contour_integrable_on } (\text{linepath } a \ b);$
 $0 \leq B; \bigwedge x. x \in \text{closed_segment } a \ b \implies \text{norm}(f \ x) \leq B \rrbracket$
 $\implies \text{norm}(\text{contour_integral } (\text{linepath } a \ b) \ f) \leq B * \text{norm}(b - a)$
by (meson has_contour_integral_bound_linepath has_contour_integral_integral)

lemma contour_integral_0 [simp]: $\text{contour_integral } g \ (\lambda x. 0) = 0$

by (simp add: contour_integral_unique has_contour_integral_0)

lemma *contour_integral_sum*:

$\llbracket \text{finite } s; \bigwedge a. a \in s \implies (f\ a) \text{ contour_integrable_on } p \rrbracket$
 $\implies \text{contour_integral } p\ (\lambda x. \text{sum } (\lambda a. f\ a\ x)\ s) = \text{sum } (\lambda a. \text{contour_integral } p\ (f\ a))\ s$
by (*auto simp: contour_integral_unique has_contour_integral_sum has_contour_integral_integral*)

lemma *contour_integrable_eq*:

$\llbracket f \text{ contour_integrable_on } p; \bigwedge x. x \in \text{path_image } p \implies f\ x = g\ x \rrbracket \implies g \text{ contour_integrable_on } p$
unfolding *contour_integrable_on_def*
by (*metis has_contour_integral_eq*)

1.10 Arithmetic theorems for path integrability

lemma *contour_integrable_neg*:

$f \text{ contour_integrable_on } g \implies (\lambda x. -(f\ x)) \text{ contour_integrable_on } g$
using *has_contour_integral_neg contour_integrable_on_def* **by** *blast*

lemma *contour_integrable_add*:

$\llbracket f1 \text{ contour_integrable_on } g; f2 \text{ contour_integrable_on } g \rrbracket \implies (\lambda x. f1\ x + f2\ x) \text{ contour_integrable_on } g$
using *has_contour_integral_add contour_integrable_on_def*
by *fastforce*

lemma *contour_integrable_diff*:

$\llbracket f1 \text{ contour_integrable_on } g; f2 \text{ contour_integrable_on } g \rrbracket \implies (\lambda x. f1\ x - f2\ x) \text{ contour_integrable_on } g$
using *has_contour_integral_diff contour_integrable_on_def*
by *fastforce*

lemma *contour_integrable_lmul*:

$f \text{ contour_integrable_on } g \implies (\lambda x. c * f\ x) \text{ contour_integrable_on } g$
using *has_contour_integral_lmul contour_integrable_on_def*
by *fastforce*

lemma *contour_integrable_rmul*:

$f \text{ contour_integrable_on } g \implies (\lambda x. f\ x * c) \text{ contour_integrable_on } g$
using *has_contour_integral_rmul contour_integrable_on_def*
by *fastforce*

lemma *contour_integrable_div*:

$f \text{ contour_integrable_on } g \implies (\lambda x. f\ x / c) \text{ contour_integrable_on } g$
using *has_contour_integral_div contour_integrable_on_def*
by *fastforce*

lemma *contour_integrable_sum*:

$\llbracket \text{finite } s; \bigwedge a. a \in s \implies (f\ a) \text{ contour_integrable_on } p \rrbracket$
 $\implies (\lambda x. \text{sum } (\lambda a. f\ a\ x)\ s) \text{ contour_integrable_on } p$
unfolding *contour_integrable_on_def* **by** (*metis has_contour_integral_sum*)

```

lemma contour_integrable_neg_iff:
   $(\lambda x. -f x) \text{ contour\_integrable\_on } g \longleftrightarrow f \text{ contour\_integrable\_on } g$ 
  using contour_integrable_neg[of f g] contour_integrable_neg[of  $\lambda x. -f x$  g] by
  auto

lemma contour_integrable_lmul_iff:
   $c \neq 0 \implies (\lambda x. c * f x) \text{ contour\_integrable\_on } g \longleftrightarrow f \text{ contour\_integrable\_on } g$ 
  using contour_integrable_lmul[of f g c] contour_integrable_lmul[of  $\lambda x. c * f x$  g inverse c]
  by (auto simp: field_simps)

lemma contour_integrable_rmul_iff:
   $c \neq 0 \implies (\lambda x. f x * c) \text{ contour\_integrable\_on } g \longleftrightarrow f \text{ contour\_integrable\_on } g$ 
  using contour_integrable_rmul[of f g c] contour_integrable_rmul[of  $\lambda x. c * f x$  g inverse c]
  by (auto simp: field_simps)

lemma contour_integrable_div_iff:
   $c \neq 0 \implies (\lambda x. f x / c) \text{ contour\_integrable\_on } g \longleftrightarrow f \text{ contour\_integrable\_on } g$ 
  using contour_integrable_rmul_iff[of inverse c] by (simp add: field_simps)

lemma uniform_limit_contour_integral_linepath:
  assumes u: uniform_limit (path_image (linepath a b)) f g F
  assumes c:  $\bigwedge n. \text{continuous\_on } (\text{path\_image } (\text{linepath } a \ b)) \ (f \ n)$ 
  assumes [simp]:  $F \neq \text{bot}$ 
  obtains I J where
     $\bigwedge n. (f \ n \text{ has\_contour\_integral } I \ n) \ (\text{linepath } a \ b)$ 
     $(g \text{ has\_contour\_integral } J) \ (\text{linepath } a \ b)$ 
     $(I \longrightarrow J) \ F$ 
proof (rule uniform_limit_integral)
  note [continuous_intros] = continuous_on_compose2[OF c]

  show uniform_limit {0..1}  $(\lambda x \ t. f \ x \ (\text{linepath } a \ b \ t) * (b - a))$ 
     $(\lambda t. g \ (\text{linepath } a \ b \ t) * (b - a)) \ F$ 
proof (rule uniform_limit_intros)
  show uniform_limit {0..1}  $(\lambda x \ t. f \ x \ (\text{linepath } a \ b \ t))$ 
     $(\lambda t. g \ (\text{linepath } a \ b \ t)) \ F$ 
    using u unfolding path_image_def by (rule uniform_limit_compose') auto
qed

show continuous_on {0..1}  $(\lambda t. f \ n \ (\text{linepath } a \ b \ t) * (b - a))$  for n
  by (intro continuous_intros; unfold path_image_def) auto

fix I J

```

```

assume  $I: \bigwedge n. ((\lambda t. f\ n\ (\text{linepath}\ a\ b\ t) * (b - a))\ \text{has\_integral}\ I\ n)\ \{0..1\}$ 
and  $J: ((\lambda t. g\ (\text{linepath}\ a\ b\ t) * (b - a))\ \text{has\_integral}\ J)\ \{0..1\}$ 
and  $\text{lim}: (I \longrightarrow J)\ F$ 
show ?thesis
by (rule that[of I J]) (use I J lim in <auto simp: has_contour_integral>)
qed auto

```

lemma *contour_integral_sums_linepath*:

assumes $u: \text{uniform_limit}\ (\text{closed_segment}\ a\ b)\ (\lambda N\ w. \sum_{n < N}. f\ n\ w)\ g\ \text{sequentially}$

assumes $c: \bigwedge n. \text{continuous_on}\ (\text{closed_segment}\ a\ b)\ (f\ n)$

obtains J **where**

$(g\ \text{has_contour_integral}\ J)\ (\text{linepath}\ a\ b)$

$(\lambda n. \text{contour_integral}\ (\text{linepath}\ a\ b)\ (f\ n))\ \text{sums}\ J$

proof (*rule uniform_limit_contour_integral_linepath*)

show $\text{uniform_limit}\ (\text{path_image}\ (\text{linepath}\ a\ b))\ (\lambda N\ w. \sum_{n < N}. f\ n\ w)\ g\ \text{sequentially}$

using u **by** *simp*

next

show $\text{continuous_on}\ (\text{path_image}\ (\text{linepath}\ a\ b))\ (\lambda w. \sum_{n < N}. f\ n\ w)$ **for** N

by (*intro continuous_intros continuous_on_subset[OF c]*) *simp_all*

next

fix $I\ J$

assume $1: \bigwedge N. ((\lambda w. \sum_{n < N}. f\ n\ w)\ \text{has_contour_integral}\ I\ N)\ (\text{linepath}\ a\ b)$

assume $2: (g\ \text{has_contour_integral}\ J)\ (\text{linepath}\ a\ b)$ **and** $3: (I \longrightarrow J)\ \text{sequentially}$

have $4: I = (\lambda N. (\sum_{n < N}. \text{contour_integral}\ (\text{linepath}\ a\ b)\ (f\ n)))$

proof

fix $N :: \text{nat}$

have $f\ n\ \text{contour_integrable_on}\ (\text{linepath}\ a\ b)$ **for** n

by (*intro contour_integrable_continuous_linepath assms*)

hence $((\lambda w. \sum_{n < N}. f\ n\ w)\ \text{has_contour_integral}$

$(\sum_{n < N}. \text{contour_integral}\ (\text{linepath}\ a\ b)\ (f\ n)))\ (\text{linepath}\ a\ b)$

using c **by** (*intro has_contour_integral_sum*) (*simp_all add: has_contour_integral_integral*)

with $1[\text{of } N]$ **show** $I\ N = (\sum_{n < N}. \text{contour_integral}\ (\text{linepath}\ a\ b)\ (f\ n))$

using *contour_integral_unique* **by** *metis*

qed

have $5: (\lambda n. \text{contour_integral}\ (\text{linepath}\ a\ b)\ (f\ n))\ \text{sums}\ J$

using $1\ 2\ 3\ 4$ **unfolding** *sums_def* **by** *blast*

from *that[OF 2 5]* **show** *?thesis* .

qed auto

lemma *contour_integral_linepath_same_Re*:

assumes $\text{Re}\ z = c\ \text{Re}\ z' = c\ \text{Im}\ z = a\ \text{Im}\ z' = b\ a < b$

shows $\text{contour_integral}\ (\text{linepath}\ z\ z')\ f =$

$i * \text{integral}\ \{a..b\}\ (\lambda x. f\ (\text{Complex}\ c\ x))$

proof –

```

have zz': z = Complex c a z' = Complex c b
  using assms by (auto simp: complex_eq_iff)
have contour_integral (linepath z z') f =
  (z' - z) * integral {0..1} (λx. f (linepath z z' x))
  by (simp add: contour_integral_integral)
also have z' - z = i * of_real (b - a)
  by (simp add: zz' Complex_eq algebra_simps)
also have integral {0..1} (λx. f (linepath z z' x)) =
  integral {0..1} (λx. f (Complex c (linepath a b x)))
  by (simp add: linepath_def Complex_eq scaleR_conv_of_real algebra_simps
zz')
also have ... = integral {0..(b - a) / (b - a)} (λx. f (Complex c (a + (b - a)
* x)))
  using ‹a < b› by (simp add: algebra_simps linepath_def)
also have {0..(b - a) / (b - a)} = (λx. x / (b - a)) ' {0..b - a}
  using ‹a < b› by simp
also have integral ... (λx. f (Complex c (a + (b - a) * x))) =
  integral {a-a..b-a} (λx. f (Complex c (x + a))) / of_real (b - a)
  using ‹a < b› by (subst integral_stretch_real) (auto simp: scaleR_conv_of_real
add_ac)
also have ... = integral {a..b} (λx. f (Complex c x)) / of_real (b - a)
  by (subst integral_shift_real_ivl) (rule refl)
finally show ?thesis
  using ‹a < b› by simp
qed

```

1.11 Reversing a path integral

```

lemma has_contour_integral_reverse_linepath:
  (f has_contour_integral i) (linepath a b)
  ⇒ (f has_contour_integral (-i)) (linepath b a)
  using has_contour_integral_reversepath valid_path_linepath by fastforce

```

```

lemma contour_integral_reverse_linepath:
  continuous_on (closed_segment a b) f ⇒ contour_integral (linepath a b) f =
  - (contour_integral (linepath b a) f)
  using contour_integral_reversepath by fastforce

```

Splitting a path integral in a flat way.*)

```

lemma has_contour_integral_split:
  assumes f: (f has_contour_integral i) (linepath a c) (f has_contour_integral j)
  (linepath c b)
  and k: 0 ≤ k k ≤ 1
  and c: c - a = k *R (b - a)
  shows (f has_contour_integral (i + j)) (linepath a b)
proof (cases k = 0 ∨ k = 1)
case True
then show ?thesis
  using assms by auto

```

```

next
case False
then have k: 0 < k k < 1
  using assms by auto
have c': c = k *R (b - a) + a
  by (metis diff_add_cancel c)
have bc: (b - c) = (1 - k) *R (b - a)
  by (simp add: algebra_simps c')
{ assume *: ((λx. f ((1 - x) *R a + x *R c) * (c - a)) has_integral i) {0..1}
  have ∧x. (x / k) *R a + ((k - x) / k) *R a = a
  using False by (simp add: field_split_simps flip: real_vector.scale_left_distrib)
  then have ∧x. ((k - x) / k) *R a + (x / k) *R c = (1 - x) *R a + x *R b
  using False by (simp add: c' algebra_simps)
  then have ((λx. f ((1 - x) *R a + x *R b) * (b - a)) has_integral i) {0..k}
  using k has_integral_affinity01 [OF *, of inverse k 0]
  by (force dest: has_integral_cmul [where c = inverse k]
      simp add: divide_simps mult.commute [of _ k] image_affinity_atLeastAtMost
  c)
} note fi = this
{ assume *: ((λx. f ((1 - x) *R c + x *R b) * (b - c)) has_integral j) {0..1}
  have **: ∧x. (((1 - x) / (1 - k)) *R c + ((x - k) / (1 - k)) *R b) = ((1 -
x) *R a + x *R b)
  using k
  apply (simp add: c' scaleR_conv_of_real divide_simps)
  apply (simp add: distrib_right distrib_left right_diff_distrib left_diff_distrib)
  done
  have ((λx. f ((1 - x) *R a + x *R b) * (b - a)) has_integral j) {k..1}
  using k has_integral_affinity01 [OF *, of inverse(1 - k) -(k/(1 - k))]
  apply (simp add: divide_simps mult.commute [of _ 1 - k] image_affinity_atLeastAtMost
  ** bc)
  apply (auto dest: has_integral_cmul [where k = (1 - k) *R j and c =
inverse (1 - k)])
  done
}
then show ?thesis
  using f k unfolding has_contour_integral_linepath
  by (simp add: linepath_def has_integral_combine [OF _ _ fi])
qed

lemma continuous_on_closed_segment_transform:
  assumes f: continuous_on (closed_segment a b) f
  and k: 0 ≤ k k ≤ 1
  and c: c - a = k *R (b - a)
  shows continuous_on (closed_segment a c) f
proof -
  have c': c = (1 - k) *R a + k *R b
  using c by (simp add: algebra_simps)
  have closed_segment a c ⊆ closed_segment a b
  by (metis c' ends_in_segment(1) in_segment(1) k subset_closed_segment)

```

```

    then show continuous_on (closed_segment a c) f
      by (rule continuous_on_subset [OF f])
qed

```

lemma *contour_integral_split*:

```

  assumes f: continuous_on (closed_segment a b) f
    and k:  $0 \leq k \leq 1$ 
    and c:  $c - a = k *_R (b - a)$ 
  shows contour_integral(linepath a b) f = contour_integral(linepath a c) f +
    contour_integral(linepath c b) f
proof -
  have c':  $c = (1 - k) *_R a + k *_R b$ 
    using c by (simp add: algebra_simps)
  have closed_segment a c  $\subseteq$  closed_segment a b
    by (metis c' ends_in_segment(1) in_segment(1) k subset_closed_segment)
  moreover have closed_segment c b  $\subseteq$  closed_segment a b
    by (metis c' ends_in_segment(2) in_segment(1) k subset_closed_segment)
  ultimately
  have continuous_on (closed_segment a c) f continuous_on (closed_segment c
    b) f
    by (auto intro: continuous_on_subset [OF f])
  then have (f has_contour_integral
    contour_integral (linepath a c) f + contour_integral (linepath c b) f)
    (linepath a b)
    by (meson c contour_integrable_continuous_linepath
      has_contour_integral_integral has_contour_integral_split k)
  then show ?thesis
    by (metis contour_integral_unique)
qed

```

lemma *contour_integral_split_linepath*:

```

  assumes f: continuous_on (closed_segment a b) f
    and c:  $c \in \text{closed\_segment } a \ b$ 
  shows contour_integral(linepath a b) f = contour_integral(linepath a c) f +
    contour_integral(linepath c b) f
  using c by (auto simp: closed_segment_def algebra_simps intro!: contour_integral_split
    [OF f])

```

1.12 Reversing the order in a double path integral

The condition is stronger than needed but it's often true in typical situations

```

lemma fst_im_cbox [simp]:  $\text{cbox } c \ d \neq \{\}$   $\implies$   $(\text{fst } \text{cbox } (a,c) \ (b,d)) = \text{cbox } a \ b$ 
  by (auto simp: cbox_Pair_eq)

```

```

lemma snd_im_cbox [simp]:  $\text{cbox } a \ b \neq \{\}$   $\implies$   $(\text{snd } \text{cbox } (a,c) \ (b,d)) = \text{cbox } c \ d$ 
  by (auto simp: cbox_Pair_eq)

```

proposition *contour_integral_swap*:


```

assumes fcon: continuous_on (path_image g × path_image h) (λ(y1,y2). f y1
y2)
and vp: valid_path g valid_path h
and gvcon: continuous_on {0..1} (λt. vector_derivative g (at t))
and hvcon: continuous_on {0..1} (λt. vector_derivative h (at t))
shows contour_integral g (λw. contour_integral h (f w)) =
contour_integral h (λz. contour_integral g (λw. f w z))
proof -
have gcon: continuous_on {0..1} g and hcon: continuous_on {0..1} h
using assms by (auto simp: valid_path_def piecewise_C1_differentiable_on_def)
have fgh1: λx. (λt. f (g x) (h t)) = (λ(y1,y2). f y1 y2) ∘ (λt. (g x, h t))
by (rule ext) simp
have fgh2: λx. (λt. f (g t) (h x)) = (λ(y1,y2). f y1 y2) ∘ (λt. (g t, h x))
by (rule ext) simp
have fcon_im1: λx. 0 ≤ x ⇒ x ≤ 1 ⇒ continuous_on ((λt. (g x, h t)) ‘
{0..1}) (λ(x, y). f x y)
by (rule continuous_on_subset [OF fcon]) (auto simp: path_image_def)
have fcon_im2: λx. 0 ≤ x ⇒ x ≤ 1 ⇒ continuous_on ((λt. (g t, h x)) ‘
{0..1}) (λ(x, y). f x y)
by (rule continuous_on_subset [OF fcon]) (auto simp: path_image_def)
have continuous_on (cbox (0, 0) (1, 1::real)) ((λx. vector_derivative g (at x))
∘ fst)
continuous_on (cbox (0, 0) (1::real, 1)) ((λx. vector_derivative h (at x)) ∘
snd)
by (rule continuous_intros | simp add: gvcon hvcon)+
then have gvcon': continuous_on (cbox (0, 0) (1, 1::real)) (λz. vector_derivative
g (at (fst z)))
and hvcon': continuous_on (cbox (0, 0) (1::real, 1)) (λx. vector_derivative
h (at (snd x)))
by auto
have continuous_on ((λx. (g (fst x), h (snd x))) ‘ cbox (0,0) (1,1)) (λ(y1, y2).
f y1 y2)
by (auto simp: path_image_def intro: continuous_on_subset [OF fcon])
then have continuous_on (cbox (0, 0) (1, 1)) ((λ(y1, y2). f y1 y2) ∘ (λw. ((g
∘ fst) w, (h ∘ snd) w)))
by (intro gcon hcon continuous_intros | simp)+
then have fgh: continuous_on (cbox (0, 0) (1, 1)) (λx. f (g (fst x)) (h (snd
x)))
by auto
have integral {0..1} (λx. contour_integral h (f (g x)) * vector_derivative g (at
x)) =
integral {0..1} (λx. contour_integral h (λy. f (g x) y * vector_derivative g
(at x)))
proof (rule integral_cong [OF contour_integral_rmul [symmetric]])
have λx. x ∈ {0..1} ⇒
continuous_on {0..1} (λxa. f (g x) (h xa))
by (subst fgh1) (rule fcon_im1 hcon continuous_intros | simp)+
then show λx. x ∈ {0..1} ⇒ f (g x) contour_integrable_on h
unfolding contour_integrable_on

```

```

    using continuous_on_mult hvcon integrable_continuous_real by blast
  qed
  also have ... = integral {0..1}
    (λy. contour_integral g (λx. f x (h y) * vector_derivative h (at
y)))
  unfolding contour_integral_integral
  apply (subst integral_swap_continuous [where 'a = real and 'b = real, of 0
0 1 1, simplified])
  subgoal
    by (rule fgh gvcon' hvcon' continuous_intros | simp add: split_def)+
    by (simp add: mult.commute mult.left_commute)
  also have ... = contour_integral h (λz. contour_integral g (λw. f w z))
  unfolding contour_integral_integral integral_mult_left [symmetric]
  by (simp add: algebra_simps)
  finally show ?thesis
  by (simp add: contour_integral_integral)
qed

```

```

lemma valid_path_negatepath: valid_path γ ⇒ valid_path (uminus ∘ γ)
  unfolding o_def using piecewise_C1_differentiable_neg valid_path_def by
blast

```

```

lemma has_contour_integral_negatepath:
  assumes γ: valid_path γ and cint: ((λz. f (- z)) has_contour_integral - i) γ
  shows (f has_contour_integral i) (uminus ∘ γ)
proof -
  obtain S where cont: continuous_on {0..1} γ and finite S and diff: γ C1_differentiable_on
{0..1} - S
  using γ by (auto simp: valid_path_def piecewise_C1_differentiable_on_def)
  have ((λx. - (f (- γ x) * vector_derivative γ (at x within {0..1}))) has_integral
i) {0..1}
  using cint by (auto simp: has_contour_integral_def dest: has_integral_neg)
  then
  have ((λx. f (- γ x) * vector_derivative (uminus ∘ γ) (at x within {0..1}))
has_integral i) {0..1}
  proof (rule rev_iffD1 [OF _ has_integral_spike_eq])
    show negligible S
    by (simp add: ⟨finite S⟩ negligible_finite)
    show f (- γ x) * vector_derivative (uminus ∘ γ) (at x within {0..1}) =
      - (f (- γ x) * vector_derivative γ (at x within {0..1}))
    if x ∈ {0..1} - S for x
  proof -
    have vector_derivative (uminus ∘ γ) (at x within cbox 0 1) = - vec-
tor_derivative γ (at x within cbox 0 1)
    proof (rule vector_derivative_within_cbox)
      show (uminus ∘ γ has_vector_derivative - vector_derivative γ (at x within
cbox 0 1)) (at x within cbox 0 1)
      using that unfolding o_def
      by (metis C1_differentiable_on_eq UNIV_I diff differentiable_subset

```

```

has_vector_derivative_minus subsetI that vector_derivative_works)
  qed (use that in auto)
  then show ?thesis
    by simp
  qed
qed
then show ?thesis by (simp add: has_contour_integral_def)
qed

```

```

lemma contour_integrable_negatepath:
  assumes  $\gamma$ : valid_path  $\gamma$  and  $\pi$ :  $(\lambda z. f(-z))$  contour_integrable_on  $\gamma$ 
  shows  $f$  contour_integrable_on  $(\text{uminus} \circ \gamma)$ 
  by (metis  $\gamma$  add.inverse_inverse contour_integrable_on_def has_contour_integral_negatepath
 $\pi$ )

```

```

lemma C1_differentiable_polynomial_function:
  fixes  $p :: \text{real} \Rightarrow 'a::\text{euclidean\_space}$ 
  shows polynomial_function  $p \implies p$  C1_differentiable_on  $S$ 
  by (metis continuous_on_polynomial_function C1_differentiable_on_def has_vector_derivative_polynomial_function)

```

```

lemma valid_path_polynomial_function:
  fixes  $p :: \text{real} \Rightarrow 'a::\text{euclidean\_space}$ 
  shows polynomial_function  $p \implies \text{valid\_path } p$ 
  by (force simp: valid_path_def piecewise_C1_differentiable_on_def continuous_on_polynomial_function C1_differentiable_polynomial_function)

```

```

lemma valid_path_subpath_trivial [simp]:
  fixes  $g :: \text{real} \Rightarrow 'a::\text{euclidean\_space}$ 
  shows  $z \neq g\ x \implies \text{valid\_path } (\text{subpath } x\ x\ g)$ 
  by (simp add: subpath_def valid_path_polynomial_function)

```

1.13 Partial circle path

```

definition part_circlepath ::  $[\text{complex}, \text{real}, \text{real}, \text{real}] \Rightarrow \text{complex}$ 
  where  $\text{part\_circlepath } z\ r\ s\ t \equiv \lambda x. z + \text{of\_real } r * \exp(i * \text{of\_real } (\text{linepath } s\ t\ x))$ 

```

```

lemma pathstart_part_circlepath [simp]:
  pathstart(part_circlepath  $z\ r\ s\ t$ ) =  $z + r * \exp(i * s)$ 
  by (metis part_circlepath_def pathstart_def pathstart_linepath)

```

```

lemma pathfinish_part_circlepath [simp]:
  pathfinish(part_circlepath  $z\ r\ s\ t$ ) =  $z + r * \exp(i * t)$ 
  by (metis part_circlepath_def pathfinish_def pathfinish_linepath)

```

```

lemma reversepath_part_circlepath [simp]:
  reversepath(part_circlepath  $z\ r\ s\ t$ ) = part_circlepath  $z\ r\ t\ s$ 
  unfolding part_circlepath_def reversepath_def linepath_def
  by (auto simp: algebra_simps)

```

```

lemma has_vector_derivative_part_circlepath [derivative_intros]:
  ((part_circlepath z r s t) has_vector_derivative
    (i * r * (of_real t - of_real s) * exp(i * linepath s t x)))
    (at x within X)
  unfolding part_circlepath_def linepath_def scaleR_conv_of_real
  by (rule has_vector_derivative_real_field derivative_eq_intros | simp)+

lemma differentiable_part_circlepath:
  part_circlepath c r a b differentiable at x within A
  using has_vector_derivative_part_circlepath[of c r a b x A] differentiableI_vector
  by blast

lemma vector_derivative_part_circlepath:
  vector_derivative (part_circlepath z r s t) (at x) =
    i * r * (of_real t - of_real s) * exp(i * linepath s t x)
  using has_vector_derivative_part_circlepath vector_derivative_at by blast

lemma vector_derivative_part_circlepath01:
  [0 ≤ x; x ≤ 1]
  ⇒ vector_derivative (part_circlepath z r s t) (at x within {0..1}) =
    i * r * (of_real t - of_real s) * exp(i * linepath s t x)
  using has_vector_derivative_part_circlepath
  by (auto simp: vector_derivative_at_within_ivl)

lemma valid_path_part_circlepath [simp]: valid_path (part_circlepath z r s t)
  unfolding valid_path_def
  by (auto simp: C1_differentiable_on_eq vector_derivative_works vector_derivative_part_circlepath
    has_vector_derivative_part_circlepath
    intro!: C1_differentiable_imp_piecewise continuous_intros)

lemma path_part_circlepath [simp]: path (part_circlepath z r s t)
  by (simp add: valid_path_imp_path)

proposition path_image_part_circlepath:
  assumes s ≤ t
  shows path_image (part_circlepath z r s t) = {z + r * exp(i * of_real x) | x.
    s ≤ x ∧ x ≤ t}
  proof -
    { fix z::real
      assume 0 ≤ z ≤ 1
      with ⟨s ≤ t⟩ have ∃ x. (exp (i * linepath s t z) = exp (i * of_real x)) ∧ s ≤ x
      ∧ x ≤ t
      apply (rule_tac x=(1 - z) * s + z * t in exI)
      apply (simp add: linepath_def scaleR_conv_of_real algebra_simps)
      by (metis (no_types) affine_ineq mult.commute mult_left_mono)
    }
  moreover
  { fix z

```

```

    assume  $s \leq z \leq t$ 
    then have  $z + \text{of\_real } r * \exp(i * \text{of\_real } z) \in (\lambda x. z + \text{of\_real } r * \exp(i * \text{of\_real } x)) \text{ ` } \{0..1\}$ 
    apply (rule_tac  $x = (z - s) / (t - s)$  in image_eqI)
    apply (simp add: linepath_def scaleR_conv_of_real divide_simps exp_eq)
    apply (auto simp: field_split_simps)
  done
}
ultimately show ?thesis
by (fastforce simp add: path_image_def part_circlepath_def)
qed

```

```

lemma path_image_part_circlepath':
  path_image (part_circlepath  $z$   $r$   $s$   $t$ ) =  $(\lambda x. z + r * \text{cis } x) \text{ ` } \text{closed\_segment } s \text{ } t$ 
  by (metis (no_types, lifting) ext cis_conv_exp image_image linepath_image_01
    part_circlepath_def path_image_def)

```

```

lemma path_image_part_circlepath_subset:
   $\llbracket s \leq t; 0 \leq r \rrbracket \implies \text{path\_image}(\text{part\_circlepath } z \text{ } r \text{ } s \text{ } t) \subseteq \text{sphere } z \text{ } r$ 
  by (auto simp: path_image_part_circlepath sphere_def dist_norm algebra_simps
    norm_mult)

```

```

lemma in_path_image_part_circlepath:
  assumes  $w \in \text{path\_image}(\text{part\_circlepath } z \text{ } r \text{ } s \text{ } t)$   $s \leq t$   $0 \leq r$ 
  shows  $\text{norm}(w - z) = r$ 
  by (smt (verit) assms dist_norm mem_Collect_eq norm_minus_commute path_image_part_circlepath_subsetD)

```

```

lemma path_image_part_circlepath_subset':
  assumes  $r \geq 0$ 
  shows  $\text{path\_image}(\text{part\_circlepath } z \text{ } r \text{ } s \text{ } t) \subseteq \text{sphere } z \text{ } r$ 
  by (smt (verit) assms path_image_part_circlepath_subset reversepath_part_circlepath
    reversepath_simps(2))

```

```

lemma part_circlepath_cnj:  $\text{cnj}(\text{part\_circlepath } c \text{ } r \text{ } a \text{ } b \text{ } x) = \text{part\_circlepath } (\text{cnj } c) \text{ } r \text{ } (-a) \text{ } (-b) \text{ } x$ 
  by (simp add: part_circlepath_def exp_cnj linepath_def algebra_simps)

```

```

lemma contour_integrable_on_compose_cnj_iff:
  assumes valid_path  $\gamma$ 
  shows  $f \text{ contour\_integrable\_on } (\text{cnj} \circ \gamma) \longleftrightarrow (\text{cnj} \circ f \circ \text{cnj}) \text{ contour\_integrable\_on } \gamma$ 

```

proof –

```

  from assms obtain  $S$  where  $S$ :  $\text{finite } S \wedge \gamma \text{ C1\_differentiable\_on } \{0..1\} - S$ 
  unfolding valid_path_def piecewise_C1_differentiable_on_def by blast
  have  $f \text{ contour\_integrable\_on } (\text{cnj} \circ \gamma) \longleftrightarrow$ 
     $((\lambda t. \text{cnj}(\text{cnj}(f(\text{cnj}(\gamma \text{ } t)))) * \text{vector\_derivative } \gamma \text{ (at } t))) \text{ integrable\_on } \{0..1\}$ 
  unfolding contour_integrable_on_o_def

```

```

proof (intro integrable_spike_finite_eq [OF S(1)])
  fix t :: real assume t ∈ {0..1} - S
  hence γ differentiable at t
    using S(2) by (meson C1_differentiable_on_eq)
  hence vector_derivative (λx. cnj (γ x)) (at t) = cnj (vector_derivative γ (at
t))
    by (rule vector_derivative_cnj)
  thus f (cnj (γ t)) * vector_derivative (λx. cnj (γ x)) (at t) =
    cnj (cnj (f (cnj (γ t))) * vector_derivative γ (at t))
    by simp
qed
also have ... ⟷ ((λt. cnj (f (cnj (γ t))) * vector_derivative γ (at t)) inte-
grable_on {0..1})
  by (rule integrable_on_cnj_iff)
also have ... ⟷ (cnj ∘ f ∘ cnj) contour_integrable_on γ
  by (simp add: contour_integrable_on_o_def)
finally show ?thesis .
qed

```

lemma contour_integral_cnj:

```

assumes valid_path γ
shows contour_integral (cnj ∘ γ) f = cnj (contour_integral γ (cnj ∘ f ∘ cnj))
proof -
  from assms obtain S where S: finite S γ C1_differentiable_on {0..1} - S
  unfolding valid_path_def piecewise_C1_differentiable_on_def by blast
  have contour_integral (cnj ∘ γ) f =
    integral {0..1} (λt. cnj (f (cnj (γ t))) * vector_derivative γ (at t)))
  unfolding contour_integral_integral
proof (intro integral_spike)
  fix t assume t ∈ {0..1} - S
  hence γ differentiable at t
    using S(2) by (meson C1_differentiable_on_eq)
  hence vector_derivative (λx. cnj (γ x)) (at t) = cnj (vector_derivative γ (at
t))
    by (rule vector_derivative_cnj)
  thus cnj (cnj (f (cnj (γ t))) * vector_derivative γ (at t)) =
    f ((cnj ∘ γ) t) * vector_derivative (cnj ∘ γ) (at t)
    by (simp add: o_def)
qed (use S(1) in auto)
also have ... = cnj (integral {0..1} (λt. cnj (f (cnj (γ t))) * vector_derivative
γ (at t)))
  by (subst integral_cnj [symmetric]) auto
also have ... = cnj (contour_integral γ (cnj ∘ f ∘ cnj))
  by (simp add: contour_integral_integral)
finally show ?thesis .
qed

```

lemma contour_integral_negatepath:

assumes valid_path γ

```

  shows contour_integral (uminus ∘ γ) f = -(contour_integral γ (λx. f (-x)))
(is ?lhs = ?rhs)
proof (cases f contour_integrable_on (uminus ∘ γ))
  case True
  hence *: (f has_contour_integral ?lhs) (uminus ∘ γ)
    using has_contour_integral_integral by blast
  have ((λz. f (-z)) has_contour_integral - contour_integral (uminus ∘ γ) f)
    (uminus ∘ (uminus ∘ γ))
    by (rule has_contour_integral_negatepath) (use * assms in auto)
  hence ((λx. f (-x)) has_contour_integral - ?lhs) γ
    by (simp add: o_def)
  thus ?thesis
    by (simp add: contour_integral_unique)
next
  case False
  hence ¬(λz. f (-z)) contour_integrable_on γ
    using contour_integrable_negatepath[of γ f] assms by auto
  with False show ?thesis
    by (simp add: not_integrable_contour_integral)
qed

lemma contour_integral_bound_part_circlepath:
  assumes f contour_integrable_on part_circlepath c r a b
  assumes B ≥ 0 r ≥ 0 ∧ x. x ∈ path_image (part_circlepath c r a b) ⇒ norm
(f x) ≤ B
  shows norm (contour_integral (part_circlepath c r a b) f) ≤ B * r * |b - a|
proof -
  let ?I = integral {0..1} (λx. f (part_circlepath c r a b x) * i * of_real (r * (b
- a))) *
    exp (i * linepath a b x))
  have norm ?I ≤ integral {0..1} (λx::real. B * 1 * (r * |b - a|) * 1)
  proof (rule integral_norm_bound_integral, goal_cases)
    case 1
    with assms(1) show ?case
      by (simp add: contour_integrable_on vector_derivative_part_circlepath mult_ac)
  next
    case (3 x)
    with assms(2-) show ?case unfolding norm_mult norm_of_real abs_mult
      by (intro mult_mono) (auto simp: path_image_def)
  qed auto
  also have ?I = contour_integral (part_circlepath c r a b) f
    by (simp add: contour_integral_integral vector_derivative_part_circlepath
mult_ac)
  finally show ?thesis by simp
qed

lemma has_contour_integral_part_circlepath_iff:
  assumes a < b
  shows (f has_contour_integral I) (part_circlepath c r a b) ↔

```

```

      (( $\lambda t. f (c + r * cis t) * r * i * cis t$ ) has_integral I) {a..b}
proof -
  have (f has_contour_integral I) (part_circlepath c r a b)  $\longleftrightarrow$ 
    (( $\lambda x. f (part\_circlepath\ c\ r\ a\ b\ x) * vector\_derivative (part\_circlepath\ c\ r$ 
a b)
      (at x within {0..1})) has_integral I) {0..1}
  unfolding has_contour_integral_def ..
  also have ...  $\longleftrightarrow$  (( $\lambda x. f (part\_circlepath\ c\ r\ a\ b\ x) * r * (b - a) * i *$ 
      cis (linepath a b x)) has_integral I) {0..1}
  by (intro has_integral_cong, subst vector_derivative_part_circlepath01)
    (simp_all add: cis_conv_exp)
  also have ...  $\longleftrightarrow$  (( $\lambda x. f (c + r * exp (i * linepath (of\_real\ a) (of\_real\ b) x))$ 
*)
      r * i * exp (i * linepath (of\_real a) (of\_real b) x) *
      vector_derivative (linepath (of\_real a) (of\_real b))
      (at x within {0..1})) has_integral I) {0..1}
  by (intro has_integral_cong, subst vector_derivative_linepath_within)
    (auto simp: part_circlepath_def cis_conv_exp of_real_linepath [symmetric])
  also have ...  $\longleftrightarrow$  (( $\lambda z. f (c + r * exp (i * z)) * r * i * exp (i * z)$ ) has_contour_integral
I)
      (linepath (of\_real a) (of\_real b))
  by (simp add: has_contour_integral_def)
  also have ...  $\longleftrightarrow$  (( $\lambda t. f (c + r * cis t) * r * i * cis t$ ) has_integral I) {a..b}
using assms
  by (subst has_contour_integral_linepath_Reals_iff) (simp_all add: cis_conv_exp)
  finally show ?thesis .
qed

lemma contour_integrable_part_circlepath_iff:
  assumes a < b
  shows f contour_integrable_on (part_circlepath c r a b)  $\longleftrightarrow$ 
    ( $\lambda t. f (c + r * cis t) * r * i * cis t$ ) integrable_on {a..b}
  using assms by (auto simp: contour_integrable_on_def integrable_on_def
    has_contour_integral_part_circlepath_iff)

lemma contour_integral_part_circlepath_eq:
  assumes a < b
  shows contour_integral (part_circlepath c r a b) f =
    integral {a..b} ( $\lambda t. f (c + r * cis t) * r * i * cis t$ )
proof (cases f contour_integrable_on part_circlepath c r a b)
  case True
  hence ( $\lambda t. f (c + r * cis t) * r * i * cis t$ ) integrable_on {a..b}
  using assms by (simp add: contour_integrable_part_circlepath_iff)
  with True show ?thesis
  using has_contour_integral_part_circlepath_iff[OF assms]
    contour_integral_unique has_integral_integrable_integral by blast
next
  case False
  hence  $\neg(\lambda t. f (c + r * cis t) * r * i * cis t)$  integrable_on {a..b}

```



```

    using assms by (simp add: contour_integrable_part_circlepath_iff)
  with False show ?thesis
    by (simp add: not_integrable_contour_integral not_integrable_integral)
qed

```

```

lemma contour_integral_part_circlepath_reverse:
  contour_integral (part_circlepath c r a b) f = -contour_integral (part_circlepath
c r b a) f
  by (metis contour_integral_reversepath reversepath_part_circlepath valid_path_part_circlepath)

```

```

lemma contour_integral_part_circlepath_reverse':
  b < a  $\implies$  contour_integral (part_circlepath c r a b) f =
    -contour_integral (part_circlepath c r b a) f
  by (rule contour_integral_part_circlepath_reverse)

```

```

lemma finite_bounded_log: finite {z::complex. norm z  $\leq$  b  $\wedge$  exp z = w}
proof (cases w = 0)
  case True then show ?thesis by auto
next
  case False
  have *: finite {x. cmod ((2 * real_of_int x * pi) * i)  $\leq$  b + cmod (Ln w)}
  proof (simp add: norm_mult finite_int_iff_bounded_le)
    have abs ' {x. 2 * |real_of_int x| * pi  $\leq$  b + cmod (Ln w)}
       $\subseteq$  {.. $\lfloor$ (b + cmod (Ln w)) / (2 * pi) $\rfloor$ }
    by (auto simp: field_split_simps le_floor_iff)
    then show  $\exists k. \text{abs ' } \{x. 2 * |of\_int x| * pi \leq b + cmod (Ln w)\} \subseteq \{..k\}$ 
      by blast
  qed
  have [simp]:  $\bigwedge P f. \{z. P z \wedge (\exists n. z = f n)\} = f ' \{n. P (f n)\}$ 
    by blast
  have finite {z. cmod z  $\leq$  b  $\wedge$  exp z = exp (Ln w)}
    using norm_add_leD by (fastforce intro: finite_subset [OF _ *] simp: exp_eq)
  then show ?thesis
    using False by auto
qed

```

```

lemma finite_bounded_log2:
  fixes a::complex
  assumes a  $\neq$  0
  shows finite {z. norm z  $\leq$  b  $\wedge$  exp(a*z) = w}
proof -
  have *: finite (( $\lambda z. z / a$ ) ' {z. cmod z  $\leq$  b * cmod a  $\wedge$  exp z = w})
    by (rule finite_imageI [OF finite_bounded_log])
  show ?thesis
    by (rule finite_subset [OF _ *]) (force simp: assms norm_mult)
qed

```

```

lemma has_contour_integral_bound_part_circlepath_strong:
  assumes fi: (f has_contour_integral i) (part_circlepath z r s t)

```

```

    and finite k and le:  $0 \leq B$   $0 < r$   $s \leq t$ 
    and B:  $\bigwedge x. x \in \text{path\_image}(\text{part\_circlepath } z \ r \ s \ t) - k \implies \text{norm}(f \ x) \leq B$ 
    shows  $\text{cmod } i \leq B * r * (t - s)$ 
  proof -
    consider  $s = t \mid s < t$  using  $\langle s \leq t \rangle$  by linarith
    then show ?thesis
    proof cases
      case 1 with fi [unfolded has_contour_integral]
        have  $i = 0$  by (simp add: vector_derivative_part_circlepath)
        with assms show ?thesis by simp
      next
        case 2
        have [simp]:  $|r| = r$  using  $\langle r > 0 \rangle$  by linarith
        have [simp]:  $\text{cmod } (\text{of\_real } t - \text{of\_real } s) = t - s$ 
          by (metis 2 abs_of_pos diff_gt_0 iff_gt norm_of_real of_real_diff)
        have finite (part_circlepath z r s t -  $\{y\} \cap \{0..1\}$ ) if  $y \in k$  for y
        proof -
          let ?w =  $(y - z) / \text{of\_real } r / \exp(i * \text{of\_real } s)$ 
          have fin: finite (of_real -  $\{z. \text{cmod } z \leq 1 \wedge \exp(i * \text{of\_real } (t - s) * z) =$ 
            ?w $\}$ )
            using  $\langle s < t \rangle$ 
            by (intro finite_vimageI [OF finite_bounded_log2]) (auto simp: inj_of_real)
          show ?thesis
            unfolding part_circlepath_def linepath_def vimage_def
            using le
            by (intro finite_subset [OF _ fin]) (auto simp: algebra_simps scaleR_conv_of_real
              exp_add exp_diff)
          qed
          then have fin01: finite ((part_circlepath z r s t) -  $k \cap \{0..1\}$ )
            by (rule finite_finite_vimage_IntI [OF  $\langle \text{finite } k \rangle$ ])
          have **:  $((\lambda x. \text{if } (\text{part\_circlepath } z \ r \ s \ t \ x) \in k \text{ then } 0$ 
            else  $f(\text{part\_circlepath } z \ r \ s \ t \ x) *$ 
              vector_derivative (part_circlepath z r s t) (at x)) has_integral
            i)  $\{0..1\}$ 
            by (rule has_integral_spike [OF negligible_finite [OF fin01]]) (use fi has_contour_integral
              in auto)
          have *:  $\bigwedge x. \llbracket 0 \leq x; x \leq 1; \text{part\_circlepath } z \ r \ s \ t \ x \notin k \rrbracket \implies \text{cmod } (f$ 
            (part_circlepath z r s t x))  $\leq B$ 
            by (auto intro!: B [unfolded path_image_def image_def])
          show ?thesis
            using has_integral_bound [where 'a=real, simplified, OF _ **]
            using assms le * 2  $\langle r > 0 \rangle$  by (auto simp add: norm_mult vector_derivative_part_circlepath)
          qed
        qed
      qed
    corollary contour_integral_bound_part_circlepath_strong:
      assumes f_contour_integrable_on part_circlepath z r s t
      and finite k and  $0 \leq B$   $0 < r$   $s \leq t$ 
      and  $\bigwedge x. x \in \text{path\_image}(\text{part\_circlepath } z \ r \ s \ t) - k \implies \text{norm}(f \ x) \leq B$ 

```

shows $\text{cmod } (\text{contour_integral } (\text{part_circlepath } z \ r \ s \ t) \ f) \leq B * r * (t - s)$
using *assms* $\text{has_contour_integral_bound_part_circlepath_strong}$ $\text{has_contour_integral_integral}$
by *blast*

lemma *has_contour_integral_bound_part_circlepath*:
 $\llbracket (f \text{ has_contour_integral } i) (\text{part_circlepath } z \ r \ s \ t);$
 $0 \leq B; 0 < r; s \leq t;$
 $\bigwedge x. x \in \text{path_image}(\text{part_circlepath } z \ r \ s \ t) \implies \text{norm}(f \ x) \leq B \rrbracket$
 $\implies \text{norm } i \leq B * r * (t - s)$
by (*auto intro: has_contour_integral_bound_part_circlepath_strong*)

lemma *contour_integrable_continuous_part_circlepath*:
 $\text{continuous_on } (\text{path_image } (\text{part_circlepath } z \ r \ s \ t)) \ f$
 $\implies f \text{ contour_integrable_on } (\text{part_circlepath } z \ r \ s \ t)$
unfolding *contour_integrable_on* *has_contour_integral_def* *vector_derivative_part_circlepath*
path_image_def
by (*best intro: integrable_continuous_real_path_part_circlepath [unfolded path_def]*
continuous_intros
 $\text{continuous_on_compose2 [where } g=f, \text{ OF } _ \text{ order_refl]}$)

lemma *simple_path_part_circlepath*:
 $\text{simple_path}(\text{part_circlepath } z \ r \ s \ t) \longleftrightarrow (r \neq 0 \wedge s \neq t \wedge |s - t| \leq 2 * \pi)$
proof (*cases* $r = 0 \vee s = t$)
case *True*
then show *?thesis*
unfolding *part_circlepath_def* *simple_path_def* *loop_free_def*
by (*rule disjE*) (*force intro: bexI [where* $x = 1/4$ *]* *bexI [where* $x = 1/3$ *])* +
next
case *False* **then have** $r \neq 0 \ s \neq t$ **by** *auto*
have $*$: $\bigwedge x \ y \ z \ s \ t. i * ((1 - x) * s + x * t) = i * (((1 - y) * s + y * t)) + z \longleftrightarrow$
 $i * (x - y) * (t - s) = z$
by (*simp add: algebra_simps*)
have *abs01*: $\bigwedge x \ y :: \text{real}. 0 \leq x \wedge x \leq 1 \wedge 0 \leq y \wedge y \leq 1$
 $\implies (x = y \vee x = 0 \wedge y = 1 \vee x = 1 \wedge y = 0 \longleftrightarrow |x - y| \in$
 $\{0, 1\})$
by *auto*
have $*$: $\bigwedge x \ y. (\exists n. (\text{complex_of_real } x - \text{of_real } y) * (\text{of_real } t - \text{of_real } s)$
 $= 2 * (\text{of_int } n * \text{of_real } \pi)) \longleftrightarrow$
 $(\exists n. |x - y| * (t - s) = 2 * (\text{of_int } n * \pi))$
by (*force simp: algebra_simps abs_if dest: arg_cong [where* $f=\text{Re}$ *]* *arg_cong*
 $[\text{where } f=\text{complex_of_real}]$
 $\text{intro: exI [where } x = -n \text{ for } n]$)
have $1: |s - t| \leq 2 * \pi$
if $\bigwedge x. 0 \leq x \wedge x \leq 1 \implies (\exists n. x * (t - s) = 2 * (\text{real_of_int } n * \pi)) \longrightarrow x$
 $= 0 \vee x = 1$
proof (*rule ccontr*)
assume $\neg |s - t| \leq 2 * \pi$
then have $*$: $\bigwedge n. t - s \neq \text{of_int } n * |s - t|$
using *False that* $[\text{of } 2 * \pi / |t - s|]$

```

    by (simp add: abs_minus_commute divide_simps)
  show False
    using * [of 1] * [of -1] by auto
qed
have 2:  $|s - t| = |2 * (\text{real\_of\_int } n * \pi) / x|$  if  $x \neq 0$   $x * (t - s) = 2 * (\text{real\_of\_int } n * \pi)$  for  $x$ 
proof -
  have  $t - s = 2 * (\text{real\_of\_int } n * \pi) / x$ 
    using that by (simp add: field_simps)
  then show ?thesis by (metis abs_minus_commute)
qed
have abs_away:  $\bigwedge P. (\forall x \in \{0..1\}. \forall y \in \{0..1\}. P |x - y|) \longleftrightarrow (\forall x :: \text{real}. 0 \leq x \wedge x \leq 1 \longrightarrow P x)$ 
  by force
have  $\bigwedge x n. [|s \neq t; |s - t| \leq 2 * \pi; 0 \leq x; x < 1; x * (t - s) = 2 * (\text{real\_of\_int } n * \pi)] \implies x = 0$ 
  by (rule ccontr) (auto simp: 2 field_split_simps abs_mult dest: of_int_leD)
then
show ?thesis using False
  apply (simp add: simple_path_def loop_free_def)
  apply (simp add: part_circlepath_def linepath_def exp_eq ** abs01 del: Set.insert_iff)
  apply (subst abs_away)
  apply (auto simp: 1)
  done
qed

lemma arc_part_circlepath:
  assumes  $r \neq 0$   $s \neq t$   $|s - t| < 2 * \pi$ 
  shows arc (part_circlepath z r s t)
proof -
  have *:  $x = y$  if eq:  $i * (\text{linepath } s \ t \ x) = i * (\text{linepath } s \ t \ y) + 2 * \text{of\_int } n * \text{of\_real } \pi * i$ 
    and  $x: x \in \{0..1\}$  and  $y: y \in \{0..1\}$  for  $x \ y \ n$ 
  proof (rule ccontr)
    assume  $x \neq y$ 
    have  $(\text{linepath } s \ t \ x) = (\text{linepath } s \ t \ y) + 2 * \text{of\_int } n * \text{complex\_of\_real } \pi$ 
    by (metis add_divide_eq_iff complex_i_not_zero mult.commute nonzero_mult_div_cancel_left eq)
    then have  $s * y + t * x = s * x + (t * y + \text{of\_int } n * (\pi * 2))$ 
      by (force simp: algebra_simps linepath_def dest: arg_cong [where f=Re])
    with  $\langle x \neq y \rangle$  have st:  $s - t = (\text{of\_int } n * (\pi * 2) / (y - x))$ 
      by (force simp: field_simps)
    have  $|\text{real\_of\_int } n| < |y - x|$ 
      using assms  $\langle x \neq y \rangle$  by (simp add: st abs_mult field_simps)
    then show False
      using assms  $x \ y \ st$  by (auto dest: of_int_lessD)
  qed

```

```

then have inj_on (part_circlepath z r s t) {0..1}
  using assms by (force simp add: part_circlepath_def inj_on_def exp_eq)
then show ?thesis
  by (simp add: arc_def)
qed

```

1.14 Special case of one complete circle

```

definition circlepath :: [complex, real, real]  $\Rightarrow$  complex
  where circlepath z r  $\equiv$  part_circlepath z r 0 (2*pi)

```

```

lemma circlepath: circlepath z r = ( $\lambda x$ . z + r * exp(2 * of_real pi * i * of_real x))
  by (simp add: circlepath_def part_circlepath_def linepath_def algebra_simps)

```

```

lemma pathstart_circlepath [simp]: pathstart (circlepath z r) = z + r
  by (simp add: circlepath_def)

```

```

lemma pathfinish_circlepath [simp]: pathfinish (circlepath z r) = z + r
  by (simp add: circlepath_def) (metis exp_two_pi_i mult.commute)

```

```

lemma circlepath_minus: circlepath z (-r) x = circlepath z r (x + 1/2)

```

```

proof -
  have z + of_real r * exp (2 * pi * i * (x + 1/2)) =
    z + of_real r * exp (2 * pi * i * x + pi * i)
  by (simp add: divide_simps) (simp add: algebra_simps)
  also have ... = z - r * exp (2 * pi * i * x)
  by (simp add: exp_add)
  finally show ?thesis
  by (simp add: circlepath_path_image_def sphere_def dist_norm)
qed

```

```

lemma circlepath_add1: circlepath z r (x+1) = circlepath z r x
  using circlepath_minus [of z r x+1/2] circlepath_minus [of z -r x]
  by (simp add: add.commute)

```

```

lemma circlepath_add_half: circlepath z r (x + 1/2) = circlepath z r (x - 1/2)
  using circlepath_add1 [of z r x-1/2]
  by (simp add: add.commute)

```

```

lemma path_image_circlepath_minus_subset:
  path_image (circlepath z (-r))  $\subseteq$  path_image (circlepath z r)

```

```

proof -
  have  $\exists x \in \{0..1\}. \text{circlepath } z \text{ } r \text{ } (y + 1/2) = \text{circlepath } z \text{ } r \text{ } x$ 
    if  $0 \leq y \leq 1$  for y
  proof (cases y  $\leq 1/2$ )
    case False
    with that show ?thesis
      by (force simp: circlepath_add_half)

```

```

qed (use that in force)
then show ?thesis
  by (auto simp add: path_image_def image_def circlepath_minus)
qed

```

```

lemma path_image_circlepath_minus: path_image (circlepath z (-r)) = path_image
(circlepath z r)
  using path_image_circlepath_minus_subset by fastforce

```

```

lemma has_vector_derivative_circlepath [derivative_intros]:
  ((circlepath z r) has_vector_derivative (2 * pi * i * r * exp (2 * of_real pi * i *
x)))
  (at x within X)
  unfolding circlepath_def scaleR_conv_of_real
  by (rule derivative_eq_intros) (simp add: algebra_simps)

```

```

lemma vector_derivative_circlepath:
  vector_derivative (circlepath z r) (at x) =
    2 * pi * i * r * exp(2 * of_real pi * i * x)
  using has_vector_derivative_circlepath vector_derivative_at by blast

```

```

lemma vector_derivative_circlepath01:
  [0 ≤ x; x ≤ 1]
  ⇒ vector_derivative (circlepath z r) (at x within {0..1}) =
    2 * pi * i * r * exp(2 * of_real pi * i * x)
  using has_vector_derivative_circlepath
  by (auto simp: vector_derivative_at_within_ivl)

```

```

lemma valid_path_circlepath [simp]: valid_path (circlepath z r)
  by (simp add: circlepath_def)

```

```

lemma path_circlepath [simp]: path (circlepath z r)
  by (simp add: valid_path_imp_path)

```

```

lemma path_image_circlepath_nonneg:
  assumes 0 ≤ r shows path_image (circlepath z r) = sphere z r
proof -
  have *: x ∈ (λu. z + (cmod (x - z)) * exp (i * (of_real u * (of_real pi * 2))))
  ‘ {0..1} for x
  proof (cases x = z)
    case True then show ?thesis by force
  next
    case False
    define w where w = x - z
    then have w ≠ 0 by (simp add: False)
    have **: ∧ t. [Re w = cos t * cmod w; Im w = sin t * cmod w] ⇒ w = of_real
(cmod w) * exp (i * t)
    using cis_conv_exp_complex_eq_iff by auto
    obtain t where 0 ≤ t < 2*pi Re(w/norm w) = cos t Im(w/norm w) = sin t

```

```

    apply (rule sincos_total_2pi [of Re(w/(norm w)) Im(w/(norm w))])
    by (auto simp add: divide_simps ‹w ≠ 0› cmod_power2 [symmetric])
  then
  show ?thesis
    using False ** w_def ‹w ≠ 0›
    by (rule_tac x=t / (2*pi) in image_eqI) (auto simp add: field_simps)
qed
show ?thesis
  unfolding circlepath_path_image_def sphere_def dist_norm
  by (force simp: assms algebra_simps norm_mult norm_minus_commute intro:
*)
qed

```

```

lemma path_image_circlepath [simp]:
  path_image (circlepath z r) = sphere z |r|
  using path_image_circlepath_minus
  by (force simp: path_image_circlepath_nonneg abs_if)

```

```

lemma has_contour_integral_bound_circlepath_strong:
  ‹(f has_contour_integral i) (circlepath z r);
  finite k; 0 ≤ B; 0 < r;
  ∧x. ‹norm(x - z) = r; x ∉ k› ⇒ norm(f x) ≤ B›
  ⇒ norm i ≤ B*(2*pi*r)
  unfolding circlepath_def
  by (auto simp: algebra_simps in_path_image_part_circlepath dest!: has_contour_integral_bound_part_circlepath)

```

```

lemma has_contour_integral_bound_circlepath:
  ‹(f has_contour_integral i) (circlepath z r);
  0 ≤ B; 0 < r; ∧x. norm(x - z) = r ⇒ norm(f x) ≤ B›
  ⇒ norm i ≤ B*(2*pi*r)
  by (auto intro: has_contour_integral_bound_circlepath_strong)

```

```

lemma contour_integrable_continuous_circlepath:
  continuous_on (path_image (circlepath z r)) f
  ⇒ f contour_integrable_on (circlepath z r)
  by (simp add: circlepath_def contour_integrable_continuous_part_circlepath)

```

```

lemma simple_path_circlepath: simple_path(circlepath z r) ⟷ (r ≠ 0)
  by (simp add: circlepath_def simple_path_part_circlepath)

```

```

lemma notin_path_image_circlepath [simp]: cmod (w - z) < r ⇒ w ∉ path_image
(circlepath z r)
  by (simp add: sphere_def dist_norm norm_minus_commute)

```

```

lemma contour_integral_circlepath:
  assumes r > 0
  shows contour_integral (circlepath z r) (λw. 1 / (w - z)) = 2 * of_real pi * i
proof (rule contour_integral_unique)
  show ((λw. 1 / (w - z)) has_contour_integral 2 * of_real pi * i) (circlepath z

```

```

r)
  unfolding has_contour_integral_def using assms has_integral_const_real [of
  _ 0 1]
  apply (subst has_integral_cong)
  apply (simp add: vector_derivative_circlepath01)
  apply (force simp: circlepath)
  done
qed

```

1.15 Uniform convergence of path integral

Uniform convergence when the derivative of the path is bounded, and in particular for the special case of a circle.

proposition *contour_integral_uniform_limit:*

```

assumes ev_fint: eventually (λn::'a. (f n) contour_integrable_on γ) F
  and ul_f: uniform_limit (path_image γ) f l F
  and noleB: ∧t. t ∈ {0..1} ⇒ norm (vector_derivative γ (at t)) ≤ B
  and γ: valid_path γ
  and [simp]: ¬ trivial_limit F
shows l contour_integrable_on γ ((λn. contour_integral γ (f n)) ⟶ con-
tour_integral γ l) F
proof -
  have 0 ≤ B by (meson noleB [of 0] atLeastAtMost_iff norm_ge_zero order_refl
order_trans zero_le_one)
  { fix e::real
    assume 0 < e
    then have 0 < e / (|B| + 1) by simp
    then have §: ∀F n in F. ∀x∈path_image γ. cmod (f n x - l x) < e / (|B| +
1)
      using ul_f [unfolded uniform_limit_iff dist_norm] by auto
    obtain a where fga: ∧x. x ∈ {0..1} ⇒ cmod (f a (γ x) - l (γ x)) < e /
(|B| + 1)
      and inta: (λt. f a (γ t) * vector_derivative γ (at t)) integrable_on
{0..1}
    using eventually_happens [OF eventually_conj [OF ev_fint §]]
    by (fastforce simp: contour_integrable_on_path_image_def)
    have ∃h. (∀x∈{0..1}. cmod (l (γ x) * vector_derivative γ (at x) - h x) ≤ e)
    ∧ h integrable_on {0..1}
    proof (intro exI conjI ballI)
      show cmod (l (γ x) * vector_derivative γ (at x) - f a (γ x) * vector_derivative
γ (at x)) ≤ e
        if x ∈ {0..1} for x
      proof -
        have cmod (l (γ x) * vector_derivative γ (at x) - f a (γ x) * vector_derivative
γ (at x)) ≤ B * e / (|B| + 1)
          using noleB [OF that] fga [OF that] ‹0 ≤ B› ‹0 < e›
          by (fastforce simp: mult_ac dest: mult_mono [OF less_imp_le] simp add:
norm_mult_left_diff_distrib [symmetric] norm_minus_commute divide_simps)
        also have ... ≤ e

```



```

      using ‹0 ≤ B› ‹0 < e› by (simp add: field_split_simps)
    finally show ?thesis .
  qed
qed (rule inta)
}
then show lintg: l contour_integrable_on γ
  unfolding contour_integrable_on by (metis (mono_tags, lifting) integrable_uniform_limit_real)
{ fix e::real
  define B' where B' = B + 1
  have B': B' > 0 B' > B using ‹0 ≤ B› by (auto simp: B'_def)
  assume 0 < e
  then have ev_no': ∀ F n in F. ∀ x ∈ path_image γ. 2 * cmod (f n x - l x) < e
    / B'
  using ul_f [unfolded uniform_limit_iff dist_norm, rule_format, of e / B'/2]
  B'
  by (simp add: field_simps)
  have ie: integral {0..1::real} (λx. e/2) < e using ‹0 < e› by simp
  have *: cmod (f x (γ t) * vector_derivative γ (at t) - l (γ t) * vector_derivative
    γ (at t)) ≤ e/2
    if t: t ∈ {0..1} and leB': 2 * cmod (f x (γ t) - l (γ t)) < e / B' for x t
  proof -
    have 2 * cmod (f x (γ t) - l (γ t)) * cmod (vector_derivative γ (at t)) ≤ e
    * (B / B')
    using mult_mono [OF less_imp_le [OF leB'] noleB] B' ‹0 < e› t by auto
    also have ... < e
    by (simp add: B' ‹0 < e› mult_imp_div_pos_less)
    finally have 2 * cmod (f x (γ t) - l (γ t)) * cmod (vector_derivative γ (at
    t)) < e .
  then show ?thesis
    by (simp add: left_diff_distrib [symmetric] norm_mult)
  qed
  have le_e: ∧ x. [∀ u ∈ {0..1}. 2 * cmod (f x (γ u) - l (γ u)) < e / B'; f x
    contour_integrable_on γ]
    ⇒ cmod (integral {0..1}
      (λu. f x (γ u) * vector_derivative γ (at u) - l (γ u) * vector_derivative
        γ (at u))) < e
    apply (rule le_less_trans [OF integral_norm_bound_integral ie])
    apply (simp add: lintg integrable_diff contour_integrable_on [symmetric])
    apply (blast intro: *)+
  done
  have ∀ F x in F. dist (contour_integral γ (f x)) (contour_integral γ l) < e
    apply (rule eventually_mono [OF eventually_conj [OF ev_no' ev_fint]])
    apply (simp add: dist_norm contour_integrable_on path_image_def con-
    tour_integral_integral)
    apply (simp add: lintg integral_diff [symmetric] contour_integrable_on
    [symmetric] le_e)
  done
}
then show ((λn. contour_integral γ (f n)) → contour_integral γ l) F

```

by (rule tendstoI)
qed

corollary *contour_integral_uniform_limit_circlepath*:

assumes $\forall_F n::'a \text{ in } F. (f\ n) \text{ contour_integrable_on } (\text{circlepath } z\ r)$
 and *uniform_limit* (sphere z r) f l F
 and $\neg \text{trivial_limit } F\ 0 < r$
 shows *l contour_integrable_on* (circlepath z r)
 $((\lambda n. \text{contour_integral } (\text{circlepath } z\ r) (f\ n)) \longrightarrow \text{contour_integral } (\text{circlepath } z\ r) l) F$
 using *assms* by (auto simp: vector_derivative_circlepath norm_mult intro!: contour_integral_uniform_limit)

lemma *has_contour_integral_linepath_same_Re_iff*:

assumes $\text{Re } z = c \text{ Re } z' = c \text{ Im } z = a \text{ Im } z' = b\ a < b$
 shows $(f \text{ has_contour_integral } I) (\text{linepath } z\ z') \longleftrightarrow$
 $((\lambda x. f (\text{Complex } c\ x)) \text{ has_integral } (-i * I)) \{a..b\}$

proof –

have $(f \text{ has_contour_integral } I) (\text{linepath } z\ z') \longleftrightarrow$
 $((\lambda x. f (\text{linepath } z\ z'\ x) * (z' - z)) \text{ has_integral } I) \{0..1\}$
 by (subst *has_contour_integral_linepath*) *simp_all*
 also have $\dots \longleftrightarrow ((\lambda x. f (c + (a + (b - a) * x) * i) * (i * (b - a))) \text{ has_integral } I) \{0..1\}$
 using *assms*
 by (intro *has_integral_cong* *arg_cong2*[of $_ _ _ (*)$] *arg_cong*[of $_ _ f$])
 (auto simp: *linepath_def* *complex_eq_iff* *algebra_simps*)
 also have $\{0..1\} = (\lambda x. x / (b - a)) ' \{0..b-a\}$
 using *assms* by *simp*
 also have $((\lambda x. f (c + (a + (b-a) * x) * i) * (i * (b-a))) \text{ has_integral } I) \dots$
 \longleftrightarrow
 $((\lambda x. f (c + (a + x) * i) * (i * (b-a))) \text{ has_integral } ((b-a) * i))$
 $\{0..b-a\}$
 by (subst *has_integral_stretch_real_iff*) (use *assms* in *simp_all*)
 also have $\dots \longleftrightarrow ((\lambda x. \text{of_real } (b-a) * i * (f (c + x * i))) \text{ has_integral } (b-a)$
 $* i) \{a..b\}$
 by (subst *has_integral_shift_real_ivl_iff*[**where** $c = -a$])
 (*simp_all* add: *scaleR_conv_of_real* *mult_ac*)
 also have $\dots \longleftrightarrow ((\lambda x. f (c + x * i) \text{ has_integral } (-i * I)) \{a..b\}$
 by (subst *has_integral_mult_right_iff*) (use *assms* in $\langle \text{auto simp: scaleR_conv_of_real} \rangle$)
 finally show ?thesis
 by (*simp* add: *scaleR_conv_of_real* *Complex_eq* *mult.commute*)
 qed

end

2 Complex Path Integrals and Cauchy's Integral Theorem

By John Harrison et al. Ported from HOL Light by L C Paulson (2015)

theory *Cauchy_Integral_Theorem*

imports

HOL-Analysis.Analysis

Contour_Integration

begin

lemma *leibniz_rule_holomorphic*:

fixes $f::\text{complex} \Rightarrow 'b::\text{euclidean_space} \Rightarrow \text{complex}$

assumes $\bigwedge x t. x \in U \implies t \in \text{cbox } a b \implies ((\lambda x. f x t) \text{ has_field_derivative } f x t) \text{ (at } x \text{ within } U)$

assumes $\bigwedge x. x \in U \implies (f x) \text{ integrable_on } \text{cbox } a b$

assumes *continuous_on* $(U \times (\text{cbox } a b)) (\lambda(x, t). f x t)$

assumes *convex* U

shows $(\lambda x. \text{integral } (\text{cbox } a b) (f x)) \text{ holomorphic_on } U$

using *leibniz_rule_field_differentiable* $[OF \text{ assms}(1-3) \text{ __ assms}(4)]$

by $(\text{auto simp: holomorphic_on_def})$

lemma *Ln_measurable* $[measurable]: Ln \in \text{measurable borel borel}$

proof $-$

have $*$: $Ln (-\text{of_real } x) = \text{of_real } (\ln x) + i * \pi \text{ if } x > 0 \text{ for } x$

using *that* **by** $(\text{subst } Ln_minus) (\text{auto simp: } Ln_of_real)$

have $**$: $Ln (\text{of_real } x) = \text{of_real } (\ln (-x)) + i * \pi \text{ if } x < 0 \text{ for } x$

using $*[of \text{ } -x]$ *that* **by** *simp*

have *cont*: $(\lambda x. \text{indicat_real } (-\mathbb{R}_{\leq 0}) x *_R Ln x) \in \text{borel_measurable borel}$

by $(\text{intro borel_measurable_continuous_on_indicator continuous_intros}) \text{ auto}$

have $(\lambda x. \text{if } x \in \mathbb{R}_{\leq 0} \text{ then } \ln (-\text{Re } x) + i * \pi \text{ else indicator } (-\mathbb{R}_{\leq 0}) x *_R Ln x) \in \text{borel} \rightarrow_M \text{borel}$

$(\text{is } ?f \in _)$ **by** $(\text{rule measurable_If_set}[OF \text{ __ cont}]) \text{ auto}$

hence $(\lambda x. \text{if } x = 0 \text{ then } Ln 0 \text{ else } ?f x) \in \text{borel} \rightarrow_M \text{borel}$ **by** *measurable*

also have $(\lambda x. \text{if } x = 0 \text{ then } Ln 0 \text{ else } ?f x) = Ln$

by $(\text{auto simp: fun_eq_iff ** nonpos_Reals_def})$

finally show *?thesis* .

qed

lemma *powr_complex_measurable* $[measurable]$:

assumes $[measurable]: f \in \text{measurable } M \text{ borel } g \in \text{measurable } M \text{ borel}$

shows $(\lambda x. f x \text{ powr } g x :: \text{complex}) \in \text{measurable } M \text{ borel}$

using *assms* **by** $(\text{simp add: powr_def})$

The special case of midpoints used in the main quadrisection

lemma *has_contour_integral_midpoint*:

assumes $(f \text{ has_contour_integral } i) (\text{linepath } a (\text{midpoint } a b))$

$(f \text{ has_contour_integral } j) (\text{linepath } (\text{midpoint } a b) b)$

shows $(f \text{ has_contour_integral } (i + j)) (\text{linepath } a b)$

```

proof (rule has_contour_integral_split)
  show midpoint a b - a = (1/2) *R (b - a)
  using assms by (auto simp: midpoint_def scaleR_conv_of_real)
qed (use assms in auto)

```

```

lemma contour_integral_midpoint:
  assumes continuous_on (closed_segment a b) f
  shows contour_integral (linepath a b) f =
    contour_integral (linepath a (midpoint a b)) f + contour_integral (linepath
(midpoint a b) b) f
proof (rule contour_integral_split)
  show midpoint a b - a = (1/2) *R (b - a)
  using assms by (auto simp: midpoint_def scaleR_conv_of_real)
qed (use assms in auto)

```

A couple of special case lemmas that are useful below

```

lemma triangle_linear_has_chain_integral:
  (( $\lambda x. m*x + d$ ) has_contour_integral 0) (linepath a b +++ linepath b c +++
linepath c a)
proof (rule Cauchy_theorem_primitive)
  show  $\bigwedge x. x \in \text{UNIV} \implies ((\lambda x. m / 2 * x^2 + d * x)$  has_field_derivative  $m * x + d$ ) (at x)
  by (auto intro!: derivative_eq_intros)
qed auto

```

```

lemma has_chain_integral_chain_integral3:
  assumes (f has_contour_integral i) (linepath a b +++ linepath b c +++ linepath
c d)
  (is (f has_contour_integral i) ?g)
  shows contour_integral (linepath a b) f + contour_integral (linepath b c) f +
contour_integral (linepath c d) f = i
  (is ?lhs = _)
proof -
  have f contour_integrable_on ?g
  using assms contour_integrable_on_def by blast
  then have ?lhs = contour_integral ?g f
  by (simp add: valid_path_join has_contour_integral_integrable)
  then show ?thesis
  using assms contour_integral_unique by blast
qed

```

```

lemma has_chain_integral_chain_integral4:
  assumes (f has_contour_integral i) (linepath a b +++ linepath b c +++ linepath
c d +++ linepath d e)
  (is (f has_contour_integral i) ?g)
  shows contour_integral (linepath a b) f + contour_integral (linepath b c) f +
contour_integral (linepath c d) f + contour_integral (linepath d e) f = i
  (is ?lhs = _)
proof -

```

```

have f contour_integrable_on ?g
  using assms contour_integrable_on_def by blast
then have ?lhs = contour_integral ?g f
  by (simp add: valid_path_join has_contour_integral_integrable)
then show ?thesis
  using assms contour_integral_unique by blast
qed

```

2.1 The key quadrisection step

```

lemma norm_sum_half:
  assumes norm(a + b) ≥ e
  shows norm a ≥ e/2 ∨ norm b ≥ e/2
proof -
  have e ≤ norm (- a - b)
  by (simp add: add_commute assms norm_minus_commute)
  thus ?thesis
  using norm_triangle_ineq4 order_trans by fastforce
qed

lemma norm_sum_lemma:
  assumes e ≤ norm (a + b + c + d)
  shows e / 4 ≤ norm a ∨ e / 4 ≤ norm b ∨ e / 4 ≤ norm c ∨ e / 4 ≤ norm d
proof -
  have e ≤ norm ((a + b) + (c + d)) using assms
  by (simp add: algebra_simps)
  then show ?thesis
  by (auto dest!: norm_sum_half)
qed

```

```

lemma Cauchy_theorem_quadrisection:
  assumes f: continuous_on (convex hull {a,b,c}) f
  and dist a b ≤ K dist b c ≤ K dist c a ≤ K
  and e: e * K^2 ≤
    norm (contour_integral(linepath a b) f + contour_integral(linepath b
c) f + contour_integral(linepath c a) f)
  shows ∃ a' b' c'.
    a' ∈ convex hull {a,b,c} ∧ b' ∈ convex hull {a,b,c} ∧ c' ∈ convex hull
{a,b,c} ∧
    dist a' b' ≤ K/2 ∧ dist b' c' ≤ K/2 ∧ dist c' a' ≤ K/2 ∧
    e * (K/2)^2 ≤ norm(contour_integral(linepath a' b') f + contour_integral(linepath
b' c') f + contour_integral(linepath c' a') f)
    (is ∃ x y z. ?Φ x y z)
proof -
  note divide_le_eq_numeral1 [simp del]
  define a' where a' = midpoint b c
  define b' where b' = midpoint c a
  define c' where c' = midpoint a b
  have fabc: continuous_on (closed_segment a b) f continuous_on (closed_segment

```

```

b c) f continuous_on (closed_segment c a) f
  using f continuous_on_subset segments_subset_convex_hull by metis+
have fcont': continuous_on (closed_segment c' b') f
  continuous_on (closed_segment a' c') f
  continuous_on (closed_segment b' a') f
  unfolding a'_def b'_def c'_def
  by (rule continuous_on_subset [OF f],
      metis midpoints_in_convex_hull convex_hull_subset hull_subset in-
sert_subset segment_convex_hull)+
define pathint where pathint x y  $\equiv$  contour_integral(linepath x y) f for x y
have *: pathint a b + pathint b c + pathint c a =
  (pathint a c' + pathint c' b' + pathint b' a) +
  (pathint a' c' + pathint c' b + pathint b a') +
  (pathint a' c + pathint c b' + pathint b' a') +
  (pathint a' b' + pathint b' c' + pathint c' a')
  unfolding pathint_def
  by (simp add: fcont' contour_integral_reverse_linepath) (simp add: a'_def
b'_def c'_def contour_integral_midpoint fabc)
have [simp]:  $\bigwedge x y. \text{cmod } (x * 2 - y * 2) = \text{cmod } (x - y) * 2$ 
  by (metis left_diff_distrib mult.commute norm_mult_numeral1)
have [simp]:  $\bigwedge x y. \text{cmod } (x - y) = \text{cmod } (y - x)$ 
  by (simp add: norm_minus_commute)
consider e * K2 / 4  $\leq$  cmod (pathint a c' + pathint c' b' + pathint b' a) |
  e * K2 / 4  $\leq$  cmod (pathint a' c' + pathint c' b + pathint b a') |
  e * K2 / 4  $\leq$  cmod (pathint a' c + pathint c b' + pathint b' a') |
  e * K2 / 4  $\leq$  cmod (pathint a' b' + pathint b' c' + pathint c' a')
  using assms by (metis * norm_sum_lemma pathint_def)
then show ?thesis
proof cases
case 1 then have ? $\Phi$  a c' b'
  using assms unfolding pathint_def [symmetric]
  apply (clarsimp simp: c'_def b'_def midpoints_in_convex_hull hull_subset
[THEN subsetD])
  apply (auto simp: midpoint_def dist_norm scaleR_conv_of_real field_split_simps)
  done
then show ?thesis by blast
next
case 2 then have ? $\Phi$  a' c' b
  using assms unfolding pathint_def [symmetric]
  apply (clarsimp simp: a'_def c'_def midpoints_in_convex_hull hull_subset
[THEN subsetD])
  apply (auto simp: midpoint_def dist_norm scaleR_conv_of_real field_split_simps)
  done
then show ?thesis by blast
next
case 3 then have ? $\Phi$  a' c b'
  using assms unfolding pathint_def [symmetric]
  apply (clarsimp simp: a'_def b'_def midpoints_in_convex_hull hull_subset
[THEN subsetD])

```

```

    apply (auto simp: midpoint_def dist_norm scaleR_conv_of_real field_split_simps)
    done
  then show ?thesis by blast
next
case 4 then have ?Φ a' b' c'
  using assms unfolding pathint_def [symmetric]
  apply (clarsimp simp: a'_def c'_def b'_def midpoints_in_convex_hull
hull_subset [THEN subsetD])
  apply (auto simp: midpoint_def dist_norm scaleR_conv_of_real field_split_simps)
  done
  then show ?thesis by blast
qed
qed

```

2.2 Cauchy's theorem for triangles

lemma triangle_points_closer:

```

fixes a::complex
shows  $\llbracket x \in \text{convex hull } \{a,b,c\}; y \in \text{convex hull } \{a,b,c\} \rrbracket$ 
 $\implies \text{norm}(x - y) \leq \text{norm}(a - b) \vee$ 
 $\text{norm}(x - y) \leq \text{norm}(b - c) \vee$ 
 $\text{norm}(x - y) \leq \text{norm}(c - a)$ 
using simplex_extremal_le [of  $\{a,b,c\}$ ]
by (auto simp: norm_minus_commute)

```

lemma holomorphic_point_small_triangle:

```

assumes x:  $x \in S$ 
and f: continuous_on S f
and cd: f field_differentiable (at x within S)
and e:  $0 < e$ 
shows  $\exists k > 0. \forall a b c. \text{dist } a b \leq k \wedge \text{dist } b c \leq k \wedge \text{dist } c a \leq k \wedge$ 
 $x \in \text{convex hull } \{a,b,c\} \wedge \text{convex hull } \{a,b,c\} \subseteq S$ 
 $\longrightarrow \text{norm}(\text{contour\_integral}(\text{linepath } a b) f + \text{contour\_integral}(\text{linepath}$ 
 $b c) f +$ 
 $\text{contour\_integral}(\text{linepath } c a) f)$ 
 $\leq e * (\text{dist } a b + \text{dist } b c + \text{dist } c a)^2$ 
(is  $\exists k > 0. \forall a b c. \_ \longrightarrow ?\text{normle } a b c$ )

```

proof -

```

have le_of_3:  $\bigwedge a x y z. \llbracket 0 \leq x*y; 0 \leq x*z; 0 \leq y*z; a \leq (e*(x + y + z))*x$ 
 $+ (e*(x + y + z))*y + (e*(x + y + z))*z \rrbracket$ 
 $\implies a \leq e*(x + y + z)^2$ 

```

by (simp add: algebra_simps power2_eq_square)

```

have disj_le:  $\llbracket x \leq a \vee x \leq b \vee x \leq c; 0 \leq a; 0 \leq b; 0 \leq c \rrbracket \implies x \leq a + b + c$ 
for x::real and a b c

```

by linarith

```

have fabc: f contour_integrable_on linepath a b f contour_integrable_on linepath
b c f contour_integrable_on linepath c a
if convex_hull {a, b, c}  $\subseteq S$  for a b c

```

```

using segments_subset_convex_hull that
by (metis continuous_on_subset f contour_integrable_continuous_linepath)+
note path_bound = has_contour_integral_bound_linepath [simplified norm_minus_commute,
OF has_contour_integral_integral]
{ fix f' a b c d
  assume d: 0 < d
    and f':  $\bigwedge y. \llbracket cmod (y - x) \leq d; y \in S \rrbracket \implies cmod (f y - f x - f' * (y - x))$ 
   $\leq e * cmod (y - x)$ 
    and le:  $cmod (a - b) \leq d \ cmod (b - c) \leq d \ cmod (c - a) \leq d$ 
    and xc:  $x \in convex\_hull \{a, b, c\}$ 
    and S:  $convex\_hull \{a, b, c\} \subseteq S$ 
  have pa:  $contour\_integral (linepath a b) f + contour\_integral (linepath b c) f$ 
+  $contour\_integral (linepath c a) f =$ 
     $contour\_integral (linepath a b) (\lambda y. f y - f x - f' * (y - x)) +$ 
     $contour\_integral (linepath b c) (\lambda y. f y - f x - f' * (y - x)) +$ 
     $contour\_integral (linepath c a) (\lambda y. f y - f x - f' * (y - x))$ 
  apply (simp add: contour_integral_diff contour_integral_lmul contour_integrable_lmul
contour_integrable_diff fabc [OF S])
  apply (simp add: field_simps)
  done
{ fix y
  assume yc:  $y \in convex\_hull \{a, b, c\}$ 
  have cmod (f y - f x - f' * (y - x))  $\leq e * norm (y - x)$ 
  proof (rule f')
    show  $cmod (y - x) \leq d$ 
    by (metis triangle_points_closer [OF xc yc] le norm_minus_commute
order_trans)
  qed (use S yc in blast)
  also have  $\dots \leq e * (cmod (a - b) + cmod (b - c) + cmod (c - a))$ 
    by (simp add: yc e xc disj_le [OF triangle_points_closer])
  finally have  $cmod (f y - f x - f' * (y - x)) \leq e * (cmod (a - b) + cmod$ 
(b - c) +  $cmod (c - a))$  .
  } note cm_le = this
  have ?normle a b c
    unfolding dist_norm pa
    using f' xc S e
    apply (intro le_of_3 norm_triangle_le add_mono path_bound)
    apply (simp_all add: contour_integral_diff contour_integral_lmul con-
tour_integrable_lmul contour_integrable_diff fabc)
    apply (blast intro: cm_le elim: dest: segments_subset_convex_hull [THEN
subsetD])+)
  done
} note * = this
show ?thesis
  using cd e
  apply (simp add: field_differentiable_def has_field_derivative_def has_derivative_within_alt
approachable_lt le2 Ball_def)
  apply (clarify dest!: spec mp)
  using * unfolding dist_norm

```



```

    apply blast
  done
qed

```

Hence the most basic theorem for a triangle.

```

locale Chain =
  fixes x0 At Follows
  assumes At0: At x0 0
    and AtSuc:  $\bigwedge x n. \text{At } x \ n \implies \exists x'. \text{At } x' \ (\text{Suc } n) \wedge \text{Follows } x' \ x$ 
begin
  primrec f where
    f 0 = x0
  | f (Suc n) = (SOME x. At x (Suc n)  $\wedge$  Follows x (f n))

  lemma At: At (f n) n
  proof (induct n)
    case 0 show ?case
    by (simp add: At0)
  next
    case (Suc n) show ?case
    by (metis (no_types, lifting) AtSuc [OF Suc] f.simps(2) someI_ex)
  qed

  lemma Follows: Follows (f (Suc n)) (f n)
    by (metis (no_types, lifting) AtSuc [OF At [of n]] f.simps(2) someI_ex)

  declare f.simps(2) [simp del]
end

lemma Chain3:
  assumes At0: At x0 y0 z0 0
    and AtSuc:  $\bigwedge x y z n. \text{At } x \ y \ z \ n \implies \exists x' y' z'. \text{At } x' \ y' \ z' \ (\text{Suc } n) \wedge \text{Follows } x' \ y' \ z' \ x \ y \ z$ 
  obtains f g h where
    f 0 = x0 g 0 = y0 h 0 = z0
     $\bigwedge n. \text{At } (f \ n) \ (g \ n) \ (h \ n) \ n$ 
     $\bigwedge n. \text{Follows } (f \ (\text{Suc } n)) \ (g \ (\text{Suc } n)) \ (h \ (\text{Suc } n)) \ (f \ n) \ (g \ n) \ (h \ n)$ 
  proof –
    interpret three: Chain (x0,y0,z0)  $\lambda(x,y,z). \text{At } x \ y \ z \ \lambda(x',y',z'). \lambda(x,y,z). \text{Follows } x' \ y' \ z' \ x \ y \ z$ 
    proof qed (use At0 AtSuc in auto)
    show ?thesis
  proof
    show  $\bigwedge n. \text{Follows } (fst \ (\text{three.f } (\text{Suc } n))) \ (fst \ (\text{snd } (\text{three.f } (\text{Suc } n))))$ 
       $(\text{snd } (\text{snd } (\text{three.f } (\text{Suc } n)))) \ (fst \ (\text{three.f } n))$ 
       $(fst \ (\text{snd } (\text{three.f } n))) \ (\text{snd } (\text{snd } (\text{three.f } n)))$ 
       $\bigwedge n. \text{At } (fst \ (\text{three.f } n)) \ (fst \ (\text{snd } (\text{three.f } n))) \ (\text{snd } (\text{snd } (\text{three.f } n))) \ n$ 
    using three.At three.Follows
    by (simp_all add: split_beta')
  qed

```

qed auto
qed

proposition *Cauchy_theorem_triangle:*

assumes f holomorphic_on (convex hull $\{a,b,c\}$)
shows $(f \text{ has_contour_integral } 0) \text{ (linepath } a \text{ } b \text{ } ++ \text{ linepath } b \text{ } c \text{ } ++ \text{ linepath } c \text{ } a)$

proof –

have contf : continuous_on (convex hull $\{a,b,c\}$) f
by (metis assms holomorphic_on_imp_continuous_on)
let $?pathint = \lambda x \ y. \text{contour_integral}(\text{linepath } x \ y) \ f$
{ fix $y::\text{complex}$
assume fy : $(f \text{ has_contour_integral } y) \text{ (linepath } a \text{ } b \text{ } ++ \text{ linepath } b \text{ } c \text{ } ++ \text{ linepath } c \text{ } a)$
and ynz : $y \neq 0$
define K where $K = 1 + \max (\text{dist } a \ b) (\max (\text{dist } b \ c) (\text{dist } c \ a))$
define e where $e = \text{norm } y / K^2$
have $K1$: $K \geq 1$ by (simp add: $K_def \text{ max.coboundedI1}$)
then have K : $K > 0$ by linarith
have $[iff]$: $\text{dist } a \ b \leq K \ \text{dist } b \ c \leq K \ \text{dist } c \ a \leq K$
by (simp_all add: K_def)
have e : $e > 0$
unfolding e_def using $\text{ynz } K1$ by simp
define At where $At \ x \ y \ z \ n \longleftrightarrow$
 $\text{convex hull } \{x,y,z\} \subseteq \text{convex hull } \{a,b,c\} \wedge$
 $\text{dist } x \ y \leq K/2^n \wedge \text{dist } y \ z \leq K/2^n \wedge \text{dist } z \ x \leq K/2^n \wedge$
 $\text{norm} (?pathint \ x \ y + ?pathint \ y \ z + ?pathint \ z \ x) \geq e * (K/2^n)^2$
for $x \ y \ z \ n$
have $At0$: $At \ a \ b \ c \ 0$
using fy
by (simp add: $At_def \ e_def \text{ has_chain_integral_chain_integral3}$)
{ fix $x \ y \ z \ n$
assume At : $At \ x \ y \ z \ n$
then have contf' : continuous_on (convex hull $\{x,y,z\}$) f
using $\text{contf } At_def \text{ continuous_on_subset}$ by metis
have $\exists x' \ y' \ z'. At \ x' \ y' \ z' \ (\text{Suc } n) \wedge \text{convex hull } \{x',y',z'\} \subseteq \text{convex hull } \{x,y,z\}$
using $At \text{ Cauchy_theorem_quadrisection } [OF \ \text{contf}', \text{ of } K/2^n \ e]$
apply (simp add: $At_def \text{ algebra_simps}$)
apply (meson convex_hull_subset empty_subsetI insert_subset subsetCE)
done
} note $AtSuc = this$
obtain $fa \ fb \ fc$
where $f0$ [simp]: $fa \ 0 = a \ fb \ 0 = b \ fc \ 0 = c$
and cosb : $\bigwedge n. \text{convex hull } \{fa \ n, fb \ n, fc \ n\} \subseteq \text{convex hull } \{a,b,c\}$
and dist : $\bigwedge n. \text{dist } (fa \ n) \ (fb \ n) \leq K/2^n$
 $\bigwedge n. \text{dist } (fb \ n) \ (fc \ n) \leq K/2^n$
 $\bigwedge n. \text{dist } (fc \ n) \ (fa \ n) \leq K/2^n$

```

    and no:  $\bigwedge n. \text{norm}(\text{?pathint } (fa\ n) (fb\ n) +$ 
       $\text{?pathint } (fb\ n) (fc\ n) +$ 
       $\text{?pathint } (fc\ n) (fa\ n)) \geq e * (K/2^n)^2$ 
    and conv_le:  $\bigwedge n. \text{convex\_hull } \{fa(Suc\ n), fb(Suc\ n), fc(Suc\ n)\} \subseteq \text{convex\_hull } \{fa\ n, fb\ n, fc\ n\}$ 
  by (rule Chain3 [of At, OF At0 AtSuc]) (auto simp: At_def)
  obtain x where x:  $\bigwedge n. x \in \text{convex\_hull } \{fa\ n, fb\ n, fc\ n\}$ 
  proof (rule bounded_closed_nest)
    show  $\bigwedge n. \text{closed } (\text{convex\_hull } \{fa\ n, fb\ n, fc\ n\})$ 
    by (simp add: compact_imp_closed finite_imp_compact convex_hull)
    show  $\bigwedge m\ n. m \leq n \implies \text{convex\_hull } \{fa\ n, fb\ n, fc\ n\} \subseteq \text{convex\_hull } \{fa\ m, fb\ m, fc\ m\}$ 
    by (erule transitive_stepwise_le) (auto simp: conv_le)
  qed (fastforce intro: finite_imp_bounded_convex_hull)
  then have xin:  $x \in \text{convex\_hull } \{a, b, c\}$ 
  using assms f0 by blast
  then have fx:  $f \text{ field\_differentiable at } x \text{ within } (\text{convex\_hull } \{a, b, c\})$ 
  using assms holomorphic_on_def by blast
  { fix k n
    assume k:  $0 < k$ 
    and le:
       $\bigwedge x' y' z'. \llbracket \text{dist } x' y' \leq k; \text{dist } y' z' \leq k; \text{dist } z' x' \leq k;$ 
       $x \in \text{convex\_hull } \{x', y', z'\};$ 
       $\text{convex\_hull } \{x', y', z'\} \subseteq \text{convex\_hull } \{a, b, c\} \rrbracket$ 
       $\implies$ 
       $\text{cmod } (\text{?pathint } x' y' + \text{?pathint } y' z' + \text{?pathint } z' x') * 10$ 
       $\leq e * (\text{dist } x' y' + \text{dist } y' z' + \text{dist } z' x')^2$ 
    and Kk:  $K / k < 2^n$ 
    have  $K / 2^n < k$  using Kk k
    by (auto simp: field_simps)
    then have DD:  $\text{dist } (fa\ n) (fb\ n) \leq k \text{ dist } (fb\ n) (fc\ n) \leq k \text{ dist } (fc\ n) (fa\ n)$ 
     $\leq k$ 
    using dist [of n] k
    by linarith+
    have dle:  $(\text{dist } (fa\ n) (fb\ n) + \text{dist } (fb\ n) (fc\ n) + \text{dist } (fc\ n) (fa\ n))^2$ 
       $\leq (3 * K / 2^n)^2$ 
    using dist [of n] e K
    by (simp add: abs_le_square_iff [symmetric])
    have less10:  $\bigwedge x y::\text{real}. 0 < x \implies y \leq 9*x \implies y < x*10$ 
    by linarith
    have  $e * (\text{dist } (fa\ n) (fb\ n) + \text{dist } (fb\ n) (fc\ n) + \text{dist } (fc\ n) (fa\ n))^2 \leq e * (3 * K / 2^n)^2$ 
    using ynz dle e mult_le_cancel_left_pos by blast
    also have ... <
       $\text{cmod } (\text{?pathint } (fa\ n) (fb\ n) + \text{?pathint } (fb\ n) (fc\ n) + \text{?pathint } (fc\ n) (fa\ n)) * 10$ 
    using no [of n] e K
    by (simp add: e_def field_simps) (simp only: zero_less_norm_iff [symmetric])
  }
```

```

    finally have False
    using le [OF DD x cosb] by auto
  } then
  have ?thesis
    using holomorphic_point_small_triangle [OF xin conf fx, of e/10] e
    apply clarsimp
    apply (rule_tac y1=K/k in exE [OF real_arch_pow[of 2]], force+)
    done
  }
  moreover have f contour_integrable_on (linepath a b +++ linepath b c +++
linepath c a)
    by simp (meson conf continuous_on_subset contour_integrable_continuous_linepath
segments_subset_convex_hull(1)
      segments_subset_convex_hull(3) segments_subset_convex_hull(5))
  ultimately show ?thesis
    using has_contour_integral_integral by fastforce
qed

```

2.3 Version needing function holomorphic in interior only

lemma Cauchy_theorem_flat_lemma:

```

  assumes f: continuous_on (convex_hull {a,b,c}) f
    and c: c - a = k *R (b - a)
    and k: 0 ≤ k
  shows contour_integral (linepath a b) f + contour_integral (linepath b c) f +
    contour_integral (linepath c a) f = 0
proof -
  have fabc: continuous_on (closed_segment a b) f continuous_on (closed_segment
b c) f continuous_on (closed_segment c a) f
    using f continuous_on_subset segments_subset_convex_hull by metis+
  show ?thesis
  proof (cases k ≤ 1)
    case True show ?thesis
      by (simp add: contour_integral_split [OF fabc(1) k True c] contour_integral_reverse_linepath
fabc)
    next
      case False
      show ?thesis
      proof (subst contour_integral_split [symmetric])
        show b - a = (1/k) *R (c - a)
        using False c by force
        show contour_integral (linepath a c) f + contour_integral (linepath c a) f =
0
        by (simp add: contour_integral_reverse_linepath fabc(3))
        show continuous_on (closed_segment a c) f
        by (metis closed_segment_commute fabc(3))
      qed (use False in auto)
    qed
  qed
qed

```

```

lemma Cauchy_theorem_flat:
  assumes f: continuous_on (convex hull {a,b,c}) f
    and c: c - a = k *R (b - a)
  shows contour_integral (linepath a b) f +
    contour_integral (linepath b c) f +
    contour_integral (linepath c a) f = 0
proof (cases 0 ≤ k)
  case True with assms show ?thesis
    by (blast intro: Cauchy_theorem_flat_lemma)
next
  case False
    have continuous_on (closed_segment a b) f continuous_on (closed_segment b
c) f continuous_on (closed_segment c a) f
      using f continuous_on_subset segments_subset_convex_hull by metis+
    moreover have contour_integral (linepath b a) f + contour_integral (linepath
a c) f +
      contour_integral (linepath c b) f = 0
    proof (rule Cauchy_theorem_flat_lemma [of b a c f 1-k])
      show continuous_on (convex hull {b, a, c}) f
        by (simp add: f insert_commute)
      show c - b = (1 - k) *R (a - b)
        using c by (auto simp: algebra_simps)
    qed (use False in auto)
    ultimately show ?thesis
      by (metis (no_types, lifting) contour_integral_reverse_linepath eq_neg_iff_add_eq_0
minus_add_cancel)
  qed

```

```

proposition Cauchy_theorem_triangle_interior:
  assumes conf: continuous_on (convex hull {a,b,c}) f
    and holf: f holomorphic_on interior (convex hull {a,b,c})
  shows (f has_contour_integral 0) (linepath a b +++ linepath b c +++ linepath
c a)
proof -
  define pathint where pathint ≡ λx y. contour_integral (linepath x y) f
  have fabc: continuous_on (closed_segment a b) f continuous_on (closed_segment
b c) f continuous_on (closed_segment c a) f
    using conf continuous_on_subset segments_subset_convex_hull by metis+
  have bounded (f ' (convex hull {a,b,c}))
    by (simp add: compact_continuous_image compact_convex_hull compact_imp_bounded
conf)
  then obtain B where 0 < B and Bnf: ∧x. x ∈ convex hull {a,b,c} ⇒ norm
(f x) ≤ B
    by (auto simp: dest!: bounded_pos [THEN iffD1])
  have bounded (convex hull {a,b,c})
    by (simp add: bounded_convex_hull)
  then obtain C where C: 0 < C and Cno: ∧y. y ∈ convex hull {a,b,c} ⇒

```

```

norm y < C
  using bounded_pos_less by blast
then have diff_2C: norm(x - y) ≤ 2*C
  if x: x ∈ convex hull {a, b, c} and y: y ∈ convex hull {a, b, c} for x y
proof -
  have cmod x ≤ C
  using x by (meson Cno not_le not_less_iff_gr_or_eq)
  hence cmod (x - y) ≤ C + C
  using y by (meson Cno add_mono_thms_linordered_field(4) less_eq_real_def
norm_triangle_ineq4 order_trans)
  thus cmod (x - y) ≤ 2 * C
  by (metis mult_2)
qed
have contf': continuous_on (convex hull {b,a,c}) f
  using contf by (simp add: insert_commute)
{ fix y::complex
  assume fy: (f has_contour_integral y) (linepath a b +++ linepath b c +++
linepath c a)
  and ynz: y ≠ 0
  have pi_eq_y: pathint a b + pathint b c + pathint c a = y
  unfolding pathint_def by (rule has_chain_integral_chain_integral3 [OF
fy])
  have ?thesis
  proof (cases c=a ∨ a=b ∨ b=c)
  case True then show ?thesis
    using Cauchy_theorem_flat [OF contf, of 0]
    using has_chain_integral_chain_integral3 [OF fy] ynz
    by (force simp: fabc contour_integral_reverse_linepath)
  next
  case False
  then have car3: card {a, b, c} = Suc (DIM(complex))
  by auto
  { assume interior(convex hull {a,b,c}) = {}
  then have collinear{a,b,c}
  using interior_convex_hull_eq_empty [OF car3]
  by (simp add: collinear_3_eq_affine_dependent)
  with False obtain d where c ≠ a a ≠ b b ≠ c c - b = d *R (a - b)
  by (auto simp: collinear_3 collinear_lemma)
  then have False
  using False Cauchy_theorem_flat [OF contf'] pi_eq_y ynz
  by (simp add: fabc add_eq_0_iff contour_integral_reverse_linepath
pathint_def)
  }
  then obtain d where d: d ∈ interior (convex hull {a, b, c})
  by blast
  { fix d1
  assume d1_pos: 0 < d1
  and d1: ⋀x x'. ‖x ∈ convex hull {a, b, c}; x' ∈ convex hull {a, b, c}; cmod
(x' - x) < d1‖

```

```

       $\implies \text{cmod } (f x' - f x) < \text{cmod } y / (24 * C)$ 
define e where e =  $\min 1 (\min (d1 / (4 * C)) ((\text{norm } y / 24 / C) / B))$ 
define shrink where shrink x =  $x - e *_R (x - d)$  for x
have e:  $0 < e \leq 1 \ e \leq d1 / (4 * C) \ e \leq \text{cmod } y / 24 / C / B$ 
  using d1_pos  $\langle C > 0 \rangle \ \langle B > 0 \rangle$  ynz by (simp_all add: e_def)
have e_le_d1:  $e * (4 * C) \leq d1$ 
  using e  $\langle C > 0 \rangle$  by (simp add: field_simps)
have shrink a  $\in \text{interior}(\text{convex hull } \{a, b, c\})$ 
  shrink b  $\in \text{interior}(\text{convex hull } \{a, b, c\})$ 
  shrink c  $\in \text{interior}(\text{convex hull } \{a, b, c\})$ 
using d e by (auto simp: hull_inc mem_interior_convex_shrink shrink_def)
then have fhp0: (f has_contour_integral 0)
  (linepath (shrink a) (shrink b) +++ linepath (shrink b) (shrink c)
  +++ linepath (shrink c) (shrink a))
  by (simp add: Cauchy_theorem_triangle holomorphic_on_subset [OF holf]
hull_minimal)
then have f_0_shrink: pathint (shrink a) (shrink b) + pathint (shrink b)
(shrink c) + pathint (shrink c) (shrink a) = 0
  by (simp add: has_chain_integral_chain_integral3 pathint_def)
have fpi_abc: f contour_integrable_on linepath (shrink a) (shrink b)
  f contour_integrable_on linepath (shrink b) (shrink c)
  f contour_integrable_on linepath (shrink c) (shrink a)
using fhp0 by (auto simp: valid_path_join dest: has_contour_integral_integrable)
have cmod_shr:  $\bigwedge x y. \text{cmod } (\text{shrink } y - \text{shrink } x - (y - x)) = e * \text{cmod}$ 
(x - y)
  using e by (simp add: shrink_def real_vector.scale_right_diff_distrib
[symmetric])
have sh_eq:  $\bigwedge a b d::\text{complex}. (b - e *_R (b - d)) - (a - e *_R (a - d)) -$ 
(b - a) =  $e *_R (a - b)$ 
  by (simp add: algebra_simps)
have  $\text{cmod } y / (24 * C) \leq \text{cmod } y / \text{cmod } (b - a) / 12$ 
  using False  $\langle C > 0 \rangle$  diff_2C [of b a] ynz
  by (auto simp: field_split_simps hull_inc)
have less_C:  $x * \text{cmod } u < C$  if u  $\in \text{convex hull } \{a, b, c\}$   $0 \leq x \leq 1$  for
x u
proof (cases x=0)
  case False
    with that show ?thesis
      using Cno [of u] mult_left_le_one_le [of cmod u x] le_less_trans
norm_ge_zero by blast
    qed (simp add:  $\langle 0 < C \rangle$ )
  { fix u v
    assume uv: u  $\in \text{convex hull } \{a, b, c\}$  v  $\in \text{convex hull } \{a, b, c\}$  u  $\neq v$ 
    and fpi_uv: f contour_integrable_on linepath (shrink u) (shrink v)
    have shr_uv: shrink u  $\in \text{interior}(\text{convex hull } \{a, b, c\})$ 
      shrink v  $\in \text{interior}(\text{convex hull } \{a, b, c\})$ 
    using d e uv
    by (auto simp: hull_inc mem_interior_convex_shrink shrink_def)
    have cmod_fw:  $\bigwedge x. 0 \leq x \implies x \leq 1 \implies \text{cmod } (f (\text{linepath } (\text{shrink } u)$ 

```

```

(shrink v) x)) ≤ B
  using shr_uv by (blast intro: Bnflinepath_in_convex_hull interior_subset
[THEN subsetD])
  { fix x::real assume x: 0 ≤ x ≤ 1
    have |1 - x| * cmod u < C |x| * cmod v < C
      using uv x by (auto intro!: less_C)
    moreover have |x| * cmod d < C |1 - x| * cmod d < C
      using x d interior_subset by (auto intro!: less_C)
    ultimately
      have cmod_less_4C: cmod ((1 - x) *R u - (1 - x) *R d) + cmod (x
*_R v - x *_R d) < (C+C) + (C+C)
        by (metis add_strict_mono le_less_trans norm_scaleR norm_triangle_ineq4)
      have ll: linepath (shrink u) (shrink v) x - linepath u v x = -e * ((1 -
x) *_R (u - d) + x *_R (v - d))
        by (simp add: linepath_def shrink_def algebra_simps scaleR_conv_of_real)
      have cmod_less_dt: cmod (linepath (shrink u) (shrink v) x - linepath u
v x) < d1
        unfolding ll norm_mult scaleR_diff_right
        using ‹e>0› cmod_less_4C by (force intro: norm_triangle_lt
less_le_trans [OF _ e_le_d1])
      have cmod (f (linepath (shrink u) (shrink v) x)) * cmod (shrink v -
shrink u - (v - u)) +
        cmod (v - u) * cmod (f (linepath (shrink u) (shrink v) x) -
f (linepath u v x))
        ≤ B * (cmod y / 24 / C / B * 2 * C) + 2 * C * (cmod y /
24 / C)
      proof (intro add_mono [OF mult_mono])
        show cmod (f (linepath (shrink u) (shrink v) x)) ≤ B
          using cmod_fuv x by blast
        have B * (12 * (e * cmod (u - v))) ≤ 24 * e * C * B
          using e ‹B>0› diff_2C [of u v] uv by (auto simp: field_simps)
        also have ... ≤ cmod y
          using ‹C>0› ‹B>0› e by (simp add: field_simps)
        finally show cmod (shrink v - shrink u - (v - u)) ≤ cmod y / 24 /
C / B * 2 * C
          using ‹0 < B› ‹0 < C› by (simp add: cmod_shr mult_ac divide_simps)
        have cmod (f (linepath (shrink u) (shrink v) x) - f (linepath u v x))
< cmod y / (24 * C)
          using x uv shr_uv cmod_less_dt
          by (auto simp: hull_inc intro: d1 interior_subset [THEN subsetD]
linepath_in_convex_hull)
        also have ... ≤ cmod y / cmod (v - u) / 12
          using False uv ‹C>0› diff_2C [of v u] ynz
          by (auto simp: field_split_simps hull_inc)
        finally have cmod (f (linepath (shrink u) (shrink v) x) - f (linepath
u v x)) ≤ cmod y / cmod (v - u) / 12
          by simp
        then show cmod (v - u) * cmod (f (linepath (shrink u) (shrink v) x)
- f (linepath u v x))

```



```

      ≤ 2 * C * (cmod y / 24 / C)
    using uv C by (simp add: field_simps)
  qed (use ‹0 < B› in auto)
  also have ... ≤ cmod y / 6
    by simp
  finally have cmod (f (linepath (shrink u) (shrink v) x)) * cmod (shrink
v - shrink u - (v - u)) +
      cmod (v - u) * cmod (f (linepath (shrink u) (shrink v) x) -
f (linepath u v x))
      ≤ cmod y / 6 .
} note cmod_diff_le = this
have f_uv: continuous_on (closed_segment u v) f
by (blast intro: uv continuous_on_subset [OF contf closed_segment_subset convex_hull])
have **: ‹f' x' f x::complex. f'*x' - f*x = f'*(x' - x) + x*(f' - f)›
  by (simp add: algebra_simps)
have norm (pathint (shrink u) (shrink v) - pathint u v)
  ≤ (B*(norm y / 24 / C / B)*2*C + (2*C)*(norm y / 24 / C)) * measure
lborel (cbox 0 (1::real))
  apply (rule has_integral_bound
    [of_ ‹λx. f(linepath (shrink u) (shrink v) x) * (shrink v - shrink
u) - f(linepath u v x)*(v - u)
      - 0 1›])
  using ynz ‹0 < B› ‹0 < C›
  apply (simp_all add: pathint_def has_integral_diff has_contour_integral_linepath
[symmetric] has_contour_integral_integral
  fpi_uv f_uv contour_integrable_continuous_linepath del: le_divide_eq_numeral1)
  apply (auto simp: ** norm_triangle_le norm_mult cmod_diff_le simp
del: le_divide_eq_numeral1)
done
also have ... ≤ norm y / 6
  by simp
finally have norm (pathint (shrink u) (shrink v) - pathint u v) ≤ norm
y / 6 .
} note * = this
have norm (pathint (shrink a) (shrink b) - pathint a b) ≤ norm y / 6
  using False fpi_abc by (rule_tac *) (auto simp: hull_inc)
moreover
have norm (pathint (shrink b) (shrink c) - pathint b c) ≤ norm y / 6
  using False fpi_abc by (rule_tac *) (auto simp: hull_inc)
moreover
have norm (pathint (shrink c) (shrink a) - pathint c a) ≤ norm y / 6
  using False fpi_abc by (rule_tac *) (auto simp: hull_inc)
ultimately
have norm((pathint (shrink a) (shrink b) - pathint a b) +
  (pathint (shrink b) (shrink c) - pathint b c) + (pathint (shrink c)
(shrink a) - pathint c a))
  ≤ norm y / 6 + norm y / 6 + norm y / 6
  by (metis norm_triangle_le add_mono)
also have ... = norm y / 2

```

```

      by simp
      finally have norm((pathint (shrink a) (shrink b) + pathint (shrink b)
(shrink c) + pathint (shrink c) (shrink a)) -
      (pathint a b + pathint b c + pathint c a))
      ≤ norm y / 2
      by (simp add: algebra_simps)
    then
      have norm(pathint a b + pathint b c + pathint c a) ≤ norm y / 2
      by (simp add: f_0_shrink) (metis (mono_tags) add.commute mi-
nus_add_distrib norm_minus_cancel uminus_add_conv_diff)
      then have False
      using pi_eq_y ynz by auto
    }
    note * = this
    have uniformly_continuous_on (convex hull {a,b,c}) f
    by (simp add: contf compact_convex_hull_compact_uniformly_continuous)
    moreover have norm y / (24 * C) > 0
    using ynz ⟨C > 0⟩ by auto
    ultimately obtain δ where δ > 0 and
    ∀ x ∈ convex hull {a, b, c}. ∀ x' ∈ convex hull {a, b, c}.
    dist x' x < δ ⟶ dist (f x') (f x) < cmod y / (24 * C)
    using ⟨C > 0⟩ ynz unfolding uniformly_continuous_on_def dist_norm
  by blast
    hence False using *[of δ] by (auto simp: dist_norm)
    then show ?thesis ..
  qed
}
moreover have f contour_integrable_on (linepath a b +++ linepath b c +++
linepath c a)
  using fabc contour_integrable_continuous_linepath by auto
ultimately show ?thesis
  using has_contour_integral_integral by fastforce
qed

```

2.4 Version allowing finite number of exceptional points

proposition *Cauchy_theorem_triangle_cofinite:*

```

  assumes continuous_on (convex hull {a,b,c}) f
  and finite S
  and (⋀ x. x ∈ interior(convex hull {a,b,c}) - S ⟹ f field_differentiable (at
x))
  shows (f has_contour_integral 0) (linepath a b +++ linepath b c +++ linepath
c a)
using assms
proof (induction card S arbitrary: a b c S rule: less_induct)
  case (less S a b c)
  show ?case
  proof (cases S={})
    case True with less show ?thesis

```

```

    by (fastforce simp: holomorphic_on_def field_differentiable_at_within Cauchy_theorem_triangle_interior)
  next
    case False
    then obtain d S' where d: S = insert d S' d ∉ S'
      by (meson Set.set_insert all_not_in_conv)
    then show ?thesis
    proof (cases d ∈ convex_hull {a, b, c})
      case False
      show (f has_contour_integral 0) (linepath a b +++ linepath b c +++ linepath
c a)
      proof (rule less.hyps)
        show  $\bigwedge x. x \in \text{interior} (\text{convex\_hull } \{a, b, c\}) - S' \implies f \text{ field\_differentiable at } x$ 
        using False d interior_subset by (auto intro!: less.prem)
      qed (use d less.prem in auto)
    next
      case True
      have *: convex_hull {a, b, d} ⊆ convex_hull {a, b, c}
        by (meson True hull_subset insert_subset convex_hull_subset)
      have abd: (f has_contour_integral 0) (linepath a b +++ linepath b d +++
linepath d a)
      proof (rule less.hyps)
        show  $\bigwedge x. x \in \text{interior} (\text{convex\_hull } \{a, b, d\}) - S' \implies f \text{ field\_differentiable at } x$ 
        using d not_in_interior_convex_hull_3
        by (clarsimp intro!: less.prem) (metis * insert_absorb insert_subset interior_mono)
      qed (use d continuous_on_subset [OF _ *] less.prem in auto)
      have *: convex_hull {b, c, d} ⊆ convex_hull {a, b, c}
        by (meson True hull_subset insert_subset convex_hull_subset)
      have bcd: (f has_contour_integral 0) (linepath b c +++ linepath c d +++
linepath d b)
      proof (rule less.hyps)
        show  $\bigwedge x. x \in \text{interior} (\text{convex\_hull } \{b, c, d\}) - S' \implies f \text{ field\_differentiable at } x$ 
        using d not_in_interior_convex_hull_3
        by (clarsimp intro!: less.prem) (metis * insert_absorb insert_subset interior_mono)
      qed (use d continuous_on_subset [OF _ *] less.prem in auto)
      have *: convex_hull {c, a, d} ⊆ convex_hull {a, b, c}
        by (meson True hull_subset insert_subset convex_hull_subset)
      have cad: (f has_contour_integral 0) (linepath c a +++ linepath a d +++
linepath d c)
      proof (rule less.hyps)
        show  $\bigwedge x. x \in \text{interior} (\text{convex\_hull } \{c, a, d\}) - S' \implies f \text{ field\_differentiable at } x$ 
        using d not_in_interior_convex_hull_3
        by (clarsimp intro!: less.prem) (metis * insert_absorb insert_subset interior_mono)

```

```

qed (use d continuous_on_subset [OF _ *] less.prem in auto)
have f contour_integrable_on linepath a b
  using less.prem abd contour_integrable_joinD1 contour_integrable_on_def
by blast
moreover have f contour_integrable_on linepath b c
  using less.prem bcd contour_integrable_joinD1 contour_integrable_on_def
by blast
moreover have f contour_integrable_on linepath c a
  using less.prem cad contour_integrable_joinD1 contour_integrable_on_def
by blast
ultimately have fpi: f contour_integrable_on (linepath a b +++ linepath b
c +++ linepath c a)
  by auto
{ fix y::complex
  assume fy: (f has_contour_integral y) (linepath a b +++ linepath b c +++
linepath c a)
  and ynz: y ≠ 0
  have cont_ad: continuous_on (closed_segment a d) f
  by (meson * continuous_on_subset less.prem(1) segments_subset_convex_hull(3))
  have cont_bd: continuous_on (closed_segment b d) f
  by (meson True closed_segment_subset_convex_hull continuous_on_subset
hull_subset insert_subset less.prem(1))
  have cont_cd: continuous_on (closed_segment c d) f
  by (meson * continuous_on_subset less.prem(1) segments_subset_convex_hull(2))
  have contour_integral (linepath a b) f = - (contour_integral (linepath b
d) f + (contour_integral (linepath d a) f))
  contour_integral (linepath b c) f = - (contour_integral (linepath c d)
f + (contour_integral (linepath d b) f))
  contour_integral (linepath c a) f = - (contour_integral (linepath a d)
f + contour_integral (linepath d c) f)
  using has_chain_integral_chain_integral3 [OF abd]
  has_chain_integral_chain_integral3 [OF bcd]
  has_chain_integral_chain_integral3 [OF cad]
  by (simp_all add: algebra_simps add_eq_0_iff)
  then have ?thesis
  using cont_ad cont_bd cont_cd fy has_chain_integral_chain_integral3
contour_integral_reverse_linepath by fastforce
}
then show ?thesis
  using fpi contour_integrable_on_def by blast
qed
qed
qed

```

2.5 Cauchy's theorem for an open starlike set

lemma starlike_convex_subset:

assumes S : $a \in S$ closed_segment $b c \subseteq S$ and subs: $\bigwedge x. x \in S \implies \text{closed_segment } a x \subseteq S$

```

  shows convex hull {a,b,c}  $\subseteq$  S
proof -
  have convex hull {b, c}  $\subseteq$  S
    using assms(2) segment_convex_hull by auto
  then have  $\bigwedge u v d. [0 \leq u; 0 \leq v; u + v = 1; d \in \text{convex hull } \{b, c\}] \implies u$ 
     $\cdot_R a + v \cdot_R d \in S$ 
    by (meson subs convexD convex_closed_segment ends_in_segment subsetCE)
  then show ?thesis
    by (auto simp add: convex_hull_insert [of {b,c} a])
qed

lemma triangle_contour_integrals_starlike_primitive:
  assumes contf: continuous_on S f
    and S: a  $\in$  S open S
    and x: x  $\in$  S
    and subs:  $\bigwedge y. y \in S \implies \text{closed\_segment } a y \subseteq S$ 
    and zer:  $\bigwedge b c. \text{closed\_segment } b c \subseteq S$ 
       $\implies \text{contour\_integral (linepath a b) } f + \text{contour\_integral (linepath$ 
b c) } f +
       $\text{contour\_integral (linepath c a) } f = 0$ 
  shows (( $\lambda x. \text{contour\_integral (linepath a x) } f$ ) has_field_derivative f x) (at x)
proof -
  let ?pathint =  $\lambda x y. \text{contour\_integral (linepath x y) } f$ 
  { fix e y
    assume e:  $0 < e$  and bxe: ball x e  $\subseteq$  S and close: cmod (y - x) < e
    have y: y  $\in$  S
      using bxe close by (force simp: dist_norm norm_minus_commute)
    have cont_ayf: continuous_on (closed_segment a y) f
      using contf continuous_on_subset subs y by blast
    have xys: closed_segment x y  $\subseteq$  S
      by (metis bxe centre_in_ball close closed_segment_subset convex_ball dist_norm
dual_order.trans e mem_ball norm_minus_commute)
    have ?pathint a y - ?pathint a x = ?pathint x y
      using zer [OF xys] contour_integral_reverse_linepath [OF cont_ayf] add_eq_0_iff
    by force
  } note [simp] = this
  { fix e::real
    assume e:  $0 < e$ 
    have cont_atx: continuous (at x) f
      using x S contf continuous_on_eq_continuous_at by blast
    then obtain d1 where d1: d1 > 0 and d1_less:  $\bigwedge y. \text{cmod } (y - x) < d1 \implies$ 
cmod (f y - f x) < e/2
      unfolding continuous_at Lim_at dist_norm using e
    by (drule_tac x=e/2 in spec) force
    obtain d2 where d2: d2 > 0 ball x d2  $\subseteq$  S using ‹open S› x
      by (auto simp: open_contains_ball)
    have dpos: min d1 d2 > 0 using d1 d2 by simp
    { fix y
      assume yx: y  $\neq$  x and close: cmod (y - x) < min d1 d2

```

```

have  $y: y \in S$ 
  using  $d2$  close by (force simp: dist_norm norm_minus_commute)
have  $\text{closed\_segment } x \ y \subseteq S$ 
  using  $\text{close } d2$  by (auto simp: dist_norm norm_minus_commute dest!:
segment_bound(1))
  then have  $fxy: f \text{ contour\_integrable\_on } \text{linepath } x \ y$ 
    by (metis contour_integrable_continuous_linepath continuous_on_subset
[OF contf])
  then obtain  $i$  where  $i: (f \text{ has\_contour\_integral } i) (\text{linepath } x \ y)$ 
    by (auto simp: contour_integrable_on_def)
  then have  $((\lambda w. f \ w - f \ x) \text{ has\_contour\_integral } (i - f \ x * (y - x)))$ 
    (linepath  $x \ y$ )
    by (rule has_contour_integral_diff [OF has_contour_integral_const_linepath])
  then have  $cmod \ (i - f \ x * (y - x)) \leq e / 2 * cmod \ (y - x)$ 
  proof (rule has_contour_integral_bound_linepath)
    show  $\bigwedge u. u \in \text{closed\_segment } x \ y \implies cmod \ (f \ u - f \ x) \leq e / 2$ 
    by (meson close  $d1\_less$  le_less_trans less_imp_le min.strict_boundedE
segment_bound1)
  qed (use  $e$  in simp)
  also have  $\dots < e * cmod \ (y - x)$ 
    by (simp add:  $e \ yx$ )
  finally have  $cmod \ (?pathint \ x \ y - f \ x * (y - x)) / cmod \ (y - x) < e$ 
    using  $i \ yx$  by (simp add: contour_integral_unique divide_less_eq)
}
then have  $\exists d > 0. \forall y. y \neq x \wedge cmod \ (y - x) < d \implies cmod \ (?pathint \ x \ y - f$ 
 $x * (y - x)) / cmod \ (y - x) < e$ 
  using  $dpos$  by blast
}
then have  $(\lambda y. (?pathint \ x \ y - f \ x * (y - x)) /_R cmod \ (y - x)) -x \rightarrow 0$ 
  by (simp add: Lim_at dist_norm inverse_eq_divide)
then have  $(\lambda y. (1 / cmod \ (y - x)) *_R (?pathint \ a \ y - (?pathint \ a \ x + f \ x * (y$ 
 $- x)))) -x \rightarrow 0$ 
  using  $\langle \text{open } S \rangle x$ 
  by (force simp: dist_norm open_contains_ball inverse_eq_divide [symmetric]
eventually_at intro: Lim_transform [OF tendsto_eventually])
  then show ?thesis
    by (simp add: has_field_derivative_def has_derivative_at2 bounded_linear_mult_right)
qed

```

Existence of a primitive

lemma holomorphic_starlike_primitive:

```

fixes  $f :: \text{complex} \Rightarrow \text{complex}$ 
assumes  $\text{contf: continuous\_on } S \ f$ 
  and  $S: \text{starlike } S$  and  $os: \text{open } S$ 
  and  $k: \text{finite } k$ 
  and  $\text{fcd: } \bigwedge x. x \in S - k \implies f \text{ field\_differentiable at } x$ 
shows  $\exists g. \forall x \in S. (g \text{ has\_field\_derivative } f \ x) \text{ (at } x)$ 

```

proof –

```

obtain  $a$  where  $a: a \in S$  and  $a\_cs: \bigwedge x. x \in S \implies \text{closed\_segment } a \ x \subseteq S$ 

```

```

    using S by (auto simp: starlike_def)
  { fix x b c
    assume x ∈ S closed_segment b c ⊆ S
    then have abcs: convex hull {a, b, c} ⊆ S
      by (simp add: a a_cs starlike_convex_subset)
    then have continuous_on (convex hull {a, b, c}) f
      by (simp add: continuous_on_subset [OF conf])
    then have (f has_contour_integral 0) (linepath a b +++ linepath b c +++
linepath c a)
      using abcs interior_subset by (force intro: fcd Cauchy_theorem_triangle_cofinite
[OF _ k])
    } note 0 = this
    show ?thesis
    proof (intro exI ballI)
      show  $\bigwedge x. x \in S \implies ((\lambda x. \text{contour\_integral} (\text{linepath } a \ x) f) \text{ has\_field\_derivative } f \ x) \text{ (at } x)$ 
        using 0 a a_cs conf has_chain_integral_chain_integral3 os triangle_contour_integrals_starlike_primitive
      by force
    qed
  qed

```

lemma *Cauchy_theorem_starlike:*

```

[[open S; starlike S; finite k; continuous_on S f;
 $\bigwedge x. x \in S - k \implies f \text{ field\_differentiable at } x;$ 
valid_path g; path_image g ⊆ S; pathfinish g = pathstart g]]
 $\implies (f \text{ has\_contour\_integral } 0) \ g$ 
by (metis holomorphic_starlike_primitive Cauchy_theorem_primitive at_within_open)

```

lemma *Cauchy_theorem_starlike_simple:*

```

[[open S; starlike S; f holomorphic_on S; valid_path g; path_image g ⊆ S; pathfinish g = pathstart g]]
 $\implies (f \text{ has\_contour\_integral } 0) \ g$ 
using Cauchy_theorem_starlike [OF _ _ finite.emptyI]
by (simp add: holomorphic_on_imp_continuous_on holomorphic_on_imp_differentiable_at)

```

2.6 Cauchy's theorem for a convex set

For a convex set we can avoid assuming openness and boundary analyticity

lemma *triangle_contour_integrals_convex_primitive:*

```

assumes conf: continuous_on S f
and S: a ∈ S convex S
and x: x ∈ S
and zer:  $\bigwedge b \ c. [b \in S; c \in S]$ 
 $\implies \text{contour\_integral} (\text{linepath } a \ b) f + \text{contour\_integral} (\text{linepath } b \ c) f +$ 
 $\text{contour\_integral} (\text{linepath } c \ a) f = 0$ 
shows (( $\lambda x. \text{contour\_integral} (\text{linepath } a \ x) f$ ) has_field_derivative f x) (at x within S)
proof -

```

```

let ?pathint =  $\lambda x y. \text{contour\_integral}(\text{linepath } x y) f$ 
{ fix y
  assume y:  $y \in S$ 
  have cont_ayf: continuous_on (closed_segment a y) f
    using S y by (meson contf continuous_on_subset convex_contains_segment)
  have xys: closed_segment x y  $\subseteq S$ 
    using convex_contains_segment S x y by auto
  have ?pathint a y - ?pathint a x = ?pathint x y
    using zer [OF x y] contour_integral_reverse_linepath [OF cont_ayf] add_eq_0_iff
  by force
} note [simp] = this
{ fix e::real
  assume e:  $0 < e$ 
  have cont_atx: continuous (at x within S) f
    using x S contf by (simp add: continuous_on_eq_continuous_within)
  then obtain d1 where d1:  $d1 > 0$  and d1_less:  $\bigwedge y. \llbracket y \in S; \text{cmod } (y - x) < d1 \rrbracket \implies \text{cmod } (f y - f x) < e/2$ 
    unfolding continuous_within Lim_within dist_norm using e
    by (drule_tac x=e/2 in spec) force
  { fix y
    assume yx:  $y \neq x$  and close:  $\text{cmod } (y - x) < d1$  and y:  $y \in S$ 
    have fxy: f contour_integrable_on linepath x y
      using convex_contains_segment S x y
    by (blast intro!: contour_integrable_continuous_linepath continuous_on_subset
[OF contf])
    then obtain i where i: (f has_contour_integral i) (linepath x y)
      by (auto simp: contour_integrable_on_def)
    then have (( $\lambda w. f w - f x$ ) has_contour_integral (i - f x * (y - x)))
      (linepath x y)
    by (rule has_contour_integral_diff [OF _ has_contour_integral_const_linepath])
    then have  $\text{cmod } (i - f x * (y - x)) \leq e / 2 * \text{cmod } (y - x)$ 
    proof (rule has_contour_integral_bound_linepath)
      show  $\bigwedge u. u \in \text{closed\_segment } x y \implies \text{cmod } (f u - f x) \leq e / 2$ 
        by (meson assms(3) close convex_contains_segment d1_less le_less_trans
less_imp_le segment_bound1 subset_iff x y)
    qed (use e in simp)
    also have  $\dots < e * \text{cmod } (y - x)$ 
    by (simp add: e yx)
    finally have  $\text{cmod } (?pathint x y - f x * (y - x)) / \text{cmod } (y - x) < e$ 
      using i yx by (simp add: contour_integral_unique divide_less_eq)
    }
  then have  $\exists d > 0. \forall y \in S. y \neq x \wedge \text{cmod } (y - x) < d \longrightarrow \text{cmod } (?pathint x y - f x * (y - x)) / \text{cmod } (y - x) < e$ 
    using d1 by blast
  }
  then have (( $\lambda y. (?pathint x y - f x * (y - x)) /_R \text{cmod } (y - x)$ )  $\longrightarrow 0$ ) (at
x within S)
    by (simp add: Lim_within dist_norm inverse_eq_divide)
  then have (( $\lambda y. (1 / \text{cmod } (y - x)) *_R (?pathint a y - (?pathint a x + f x *$ 

```



```

(y - x)))  $\longrightarrow$  0)
  (at x within S)
  using linordered_field_no_ub
  by (force simp: inverse_eq_divide [symmetric] eventually_at intro: Lim_transform
    [OF tendsto_eventually])
  then show ?thesis
  by (simp add: has_field_derivative_def has_derivative_within bounded_linear_mult_right)
qed

```

lemma *contour_integral_convex_primitive*:

```

  assumes convex S continuous_on S f
     $\wedge a\ b\ c. [a \in S; b \in S; c \in S] \implies (f \text{ has\_contour\_integral } 0) \text{ (linepath } a$ 
 $b \text{ +++ linepath } b\ c \text{ +++ linepath } c\ a)$ 
  obtains g where  $\wedge x. x \in S \implies (g \text{ has\_field\_derivative } f\ x) \text{ (at } x \text{ within } S)$ 
proof (cases S={})
  case False
  with asms that show ?thesis
  by (blast intro: triangle_contour_integrals_convex_primitive has_chain_integral_chain_integral3)
qed auto

```

lemma *holomorphic_convex_primitive*:

```

  fixes f :: complex  $\Rightarrow$  complex
  assumes convex S finite K and contf: continuous_on S f
    and fd:  $\wedge x. x \in \text{interior } S - K \implies f \text{ field\_differentiable at } x$ 
  obtains g where  $\wedge x. x \in S \implies (g \text{ has\_field\_derivative } f\ x) \text{ (at } x \text{ within } S)$ 
proof (rule contour_integral_convex_primitive [OF convex contf Cauchy_theorem_triangle_cofinite])
  have *: convex hull {a, b, c}  $\subseteq S$  if a  $\in S$  b  $\in S$  c  $\in S$  for a b c
  by (simp add: convex_S hull_minimal that)
  show continuous_on (convex hull {a, b, c}) f if a  $\in S$  b  $\in S$  c  $\in S$  for a b c
  by (meson * contf continuous_on_subset that)
  show f field_differentiable at x if a  $\in S$  b  $\in S$  c  $\in S$  x  $\in \text{interior (convex hull } \{a, b, c\}) - K$  for a b c x
  by (metis * DiffD1 DiffD2 DiffI fd interior_mono subsetCE that)
qed (use asms in force+)

```

lemma *holomorphic_convex_primitive'*:

```

  fixes f :: complex  $\Rightarrow$  complex
  assumes convex S and open S and f holomorphic_on S
  obtains g where  $\wedge x. x \in S \implies (g \text{ has\_field\_derivative } f\ x) \text{ (at } x \text{ within } S)$ 
proof (rule holomorphic_convex_primitive)
  fix x assume x  $\in \text{interior } S - \{ \}$ 
  with asms show f field_differentiable at x
  by (auto intro!: holomorphic_on_imp_differentiable_at simp: interior_open)
qed (use asms in auto intro: holomorphic_on_imp_continuous_on)

```

corollary *Cauchy_theorem_convex*:

```

   $[ \text{continuous\_on } S\ f; \text{convex } S; \text{finite } K;$ 
 $\wedge x. x \in \text{interior } S - K \implies f \text{ field\_differentiable at } x;$ 
 $\text{valid\_path } g; \text{path\_image } g \subseteq S; \text{pathfinish } g = \text{pathstart } g ]$ 

```

```

     $\implies (f \text{ has\_contour\_integral } 0) \ g$ 
  by (metis holomorphic_convex_primitive Cauchy_theorem_primitive)

corollary Cauchy_theorem_convex_simple:
  assumes holf:  $f \text{ holomorphic\_on } S$ 
    and convex  $S \text{ valid\_path } g \text{ path\_image } g \subseteq S \text{ pathfinish } g = \text{pathstart } g$ 
  shows  $(f \text{ has\_contour\_integral } 0) \ g$ 
proof -
  have  $f \text{ holomorphic\_on interior } S$ 
    by (meson holf holomorphic_on_subset interior_subset)
  with Cauchy_theorem_convex [where  $K = \{\}$ ] show ?thesis
    using assms
    by (metis Diff_empty finite.emptyI holomorphic_on_imp_continuous_on holomorphic_on_imp_differentiable_at open_interior)
qed

```

In particular for a disc

```

corollary Cauchy_theorem_disc:
   $\llbracket \text{finite } K; \text{continuous\_on } (\text{cball } a \ e) \ f; \$ 
   $\bigwedge x. x \in \text{ball } a \ e - K \implies f \text{ field\_differentiable at } x;$ 
   $\text{valid\_path } g; \text{path\_image } g \subseteq \text{cball } a \ e;$ 
   $\text{pathfinish } g = \text{pathstart } g \rrbracket \implies (f \text{ has\_contour\_integral } 0) \ g$ 
  by (auto intro: Cauchy_theorem_convex)

```

```

corollary Cauchy_theorem_disc_simple:
   $\llbracket f \text{ holomorphic\_on } (\text{ball } a \ e); \text{valid\_path } g; \text{path\_image } g \subseteq \text{ball } a \ e;$ 
   $\text{pathfinish } g = \text{pathstart } g \rrbracket \implies (f \text{ has\_contour\_integral } 0) \ g$ 
  by (simp add: Cauchy_theorem_convex_simple)

```

2.7 Generalize integrability to local primitives

```

lemma contour_integral_local_primitive_lemma:
  fixes  $f :: \text{complex} \Rightarrow \text{complex}$ 
  assumes gpd:  $g \text{ piecewise\_differentiable\_on } \{a..b\}$ 
    and dh:  $\bigwedge x. x \in S \implies (f \text{ has\_field\_derivative } f' \ x) \text{ (at } x \text{ within } S)$ 
    and gs:  $\bigwedge x. x \in \{a..b\} \implies g \ x \in S$ 
  shows  $(\lambda x. f' (g \ x) * \text{vector\_derivative } g \text{ (at } x \text{ within } \{a..b\})) \text{ integrable\_on } \{a..b\}$ 
proof (cases cbox  $a \ b = \{\}$ )
  case False
  then show ?thesis
    unfolding integrable_on_def by (auto intro: assms contour_integral_primitive_lemma)
qed auto

```

```

lemma contour_integral_local_primitive_any:
  fixes  $f :: \text{complex} \Rightarrow \text{complex}$ 
  assumes gpd:  $g \text{ piecewise\_differentiable\_on } \{a..b\}$ 
    and dh:  $\bigwedge x. x \in S$ 
     $\implies \exists d \ h. 0 < d \wedge$ 

```

```

      (∀ y. norm(y - x) < d ⟶ (h has_field_derivative f y) (at y
within S))
    and gs: ⋀ x. x ∈ {a..b} ⟹ g x ∈ S
    shows (λ x. f(g x) * vector_derivative g (at x)) integrable_on {a..b}
  proof -
    { fix x
      assume x: a ≤ x x ≤ b
      obtain d h where d: 0 < d
        and h: (⋀ y. norm(y - g x) < d ⟹ (h has_field_derivative f y) (at
y within S))
      using x gs dh by (metis atLeastAtMost_iff)
      have continuous_on {a..b} g using gpd piecewise_differentiable_on_def by
blast
      then obtain e where e: e > 0 and lessd: ⋀ x'. x' ∈ {a..b} ⟹ |x' - x| < e
      ⟹ cmod (g x' - g x) < d
      using x d by (fastforce simp: dist_norm continuous_on_iff)
      have ∃ e > 0. ∀ u v. u ≤ x ∧ x ≤ v ∧ {u..v} ⊆ ball x e ∧ (u ≤ v ⟶ a ≤ u ∧
v ≤ b) ⟶
        (λ x. f (g x) * vector_derivative g (at x)) integrable_on {u..v}
      proof -
        have (λ x. f (g x) * vector_derivative g (at x within {u..v})) integrable_on
{u..v}
        if u ≤ x x ≤ v and ball: {u..v} ⊆ ball x e and auvb: u ≤ v ⟹ a ≤ u ∧ v
≤ b
        for u v
        proof (rule contour_integral_local_primitive_lemma)
          show g piecewise_differentiable_on {u..v}
          by (metis atLeastAtMost_subset_iff gpd piecewise_differentiable_on_subset
auvb)
          show ⋀ x. x ∈ g ' {u..v} ⟹ (h has_field_derivative f x) (at x within g '
{u..v})
          using that by (force simp: ball_def dist_norm intro: lessd gs DERIV_subset
[OF h])
          qed auto
          then show ?thesis
          using e integrable_on_localized_vector_derivative by blast
          qed
        } then
        show ?thesis
        by (force simp: intro!: integrable_on_little_subintervals [of a b, simplified])
      qed
    }

  lemma contour_integral_local_primitive:
    fixes f :: complex ⇒ complex
    assumes g: valid_path g path_image g ⊆ S
    and dh: ⋀ x. x ∈ S
      ⟹ ∃ d h. 0 < d ∧
      (∀ y. norm(y - x) < d ⟶ (h has_field_derivative f y) (at y
within S))

```

```

  shows  $f$  contour_integrable_on  $g$ 
proof -
  have  $(\lambda x. f (g x) * \text{vector\_derivative } g \text{ (at } x)) \text{ integrable\_on } \{0..1\}$ 
  using contour_integral_local_primitive_any [OF dh]  $g$ 
  unfolding path_image_def valid_path_def
  by (metis (no_types, lifting) image_subset_iff piecewise_C1_imp_differentiable)
  then show ?thesis
  using contour_integrable_on by presburger
qed

```

In particular if a function is holomorphic

lemma *contour_integrable_holomorphic*:

```

  assumes contf: continuous_on  $S$   $f$ 
    and os: open  $S$ 
    and k: finite  $k$ 
    and g: valid_path  $g$  path_image  $g \subseteq S$ 
    and fed:  $\bigwedge x. x \in S - k \implies f \text{ field\_differentiable at } x$ 
  shows  $f$  contour_integrable_on  $g$ 
proof -
  { fix  $z$ 
    assume  $z: z \in S$ 
    obtain  $d$  where  $d > 0$  and  $d: \text{ball } z \ d \subseteq S$  using  $\langle \text{open } S \rangle z$ 
    by (auto simp: open_contains_ball)
    then have contfb: continuous_on  $(\text{ball } z \ d)$   $f$ 
    using contf continuous_on_subset by blast
    obtain  $h$  where  $\forall y \in \text{ball } z \ d. (h \text{ has\_field\_derivative } f \ y) \text{ (at } y \text{ within ball } z \ d)$ 
    by (metis holomorphic_convex_primitive [OF convex_ball  $k$  contfb fed] d
    interior_subset Diff_iff subsetD)
    then have  $\forall y \in \text{ball } z \ d. (h \text{ has\_field\_derivative } f \ y) \text{ (at } y \text{ within } S)$ 
    by (metis open_ball at_within_open  $d$  os subsetCE)
    then have  $\exists h. (\forall y. \text{cmod } (y - z) < d \longrightarrow (h \text{ has\_field\_derivative } f \ y) \text{ (at } y \text{ within } S))$ 
    by (force simp: dist_norm norm_minus_commute)
    then have  $\exists d \ h. 0 < d \wedge (\forall y. \text{cmod } (y - z) < d \longrightarrow (h \text{ has\_field\_derivative } f \ y) \text{ (at } y \text{ within } S))$ 
    using  $\langle 0 < d \rangle$  by blast
  }
  then show ?thesis
  by (rule contour_integral_local_primitive [OF  $g$ ])
qed

```

lemma *contour_integrable_holomorphic_simple*:

```

  assumes fh:  $f$  holomorphic_on  $S$ 
    and os: open  $S$ 
    and g: valid_path  $g$  path_image  $g \subseteq S$ 
  shows  $f$  contour_integrable_on  $g$ 
proof -
  have  $\bigwedge x. x \in S \implies f \text{ field\_differentiable at } x$ 
  using fh holomorphic_on_imp_differentiable_at os by blast

```

```

moreover have continuous_on S f
  by (simp add: fh holomorphic_on_imp_continuous_on)
ultimately show ?thesis
  by (metis Diff_empty contour_integrable_holomorphic_finite.emptyI g os)
qed

```

```

lemma analytic_imp_contour_integrable:
  assumes f analytic_on path_image p valid_path p
  shows f contour_integrable_on p
  by (meson analytic_on_holomorphic assms contour_integrable_holomorphic_simple)

```

```

lemma continuous_on_inversediff:
  fixes z:: 'a::real_normed_field shows  $z \notin S \implies \text{continuous\_on } S (\lambda w. 1 / (w - z))$ 
  by (rule continuous_intros | force)+

```

```

lemma contour_integrable_inversediff:
  assumes g: valid_path g
  and notin:  $z \notin \text{path\_image } g$ 
  shows  $(\lambda w. 1 / (w - z)) \text{ contour\_integrable\_on } g$ 
proof (rule contour_integrable_holomorphic_simple)
  show  $(\lambda w. 1 / (w - z)) \text{ holomorphic\_on } \text{UNIV} - \{z\}$ 
  by (auto simp: holomorphic_on_open open_delete intro!: derivative_eq_intros)
qed (use assms in auto)

```

Key fact that path integral is the same for a "nearby" path. This is the main lemma for the homotopy form of Cauchy's theorem and is also useful if we want "without loss of generality" to assume some nice properties of a path (e.g. smoothness). It can also be used to define the integrals of analytic functions over arbitrary continuous paths. This is just done for winding numbers now.

A technical definition to avoid duplication of similar proofs, for paths joined at the ends versus looping paths

```

definition linked_paths :: bool  $\Rightarrow$  (real  $\Rightarrow$  'a)  $\Rightarrow$  (real  $\Rightarrow$  'a::topological_space)  $\Rightarrow$  bool
  where linked_paths attends g h ==
    (if attends then pathstart h = pathstart g  $\wedge$  pathfinish h = pathfinish g
     else pathfinish g = pathstart g  $\wedge$  pathfinish h = pathstart h)

```

This formulation covers two cases: g and h share their start and end points; g and h both loop upon themselves.

```

lemma contour_integral_nearby:
  assumes os: open S and p: path p path_image p  $\subseteq$  S
  shows  $\exists d. 0 < d \wedge$ 
     $(\forall g h. \text{valid\_path } g \wedge \text{valid\_path } h \wedge$ 
       $(\forall t \in \{0..1\}. \text{norm}(g\ t - p\ t) < d \wedge \text{norm}(h\ t - p\ t) < d) \wedge$ 
      linked_paths attends g h
    )

```

```

     $\longrightarrow \text{path\_image } g \subseteq S \wedge \text{path\_image } h \subseteq S \wedge$ 
     $(\forall f. f \text{ holomorphic\_on } S \longrightarrow \text{contour\_integral } h f = \text{contour\_integral } g f))$ 
  proof -
    have  $\forall z. \exists e. z \in \text{path\_image } p \longrightarrow 0 < e \wedge \text{ball } z e \subseteq S$ 
      using open_contains_ball os p(2) by blast
    then obtain ee where ee:  $\bigwedge z. z \in \text{path\_image } p \implies 0 < ee z \wedge \text{ball } z (ee z) \subseteq S$ 
      by metis
    define cover where cover =  $(\lambda z. \text{ball } z (ee z / 3)) \text{ ` } (\text{path\_image } p)$ 
    have compact (path_image p)
      by (metis p(1) compact_path_image)
    moreover have  $\text{path\_image } p \subseteq \bigcup c \in \text{path\_image } p. \text{ball } c (ee c / 3)$ 
      using ee by auto
    ultimately have  $\exists D \subseteq \text{cover}. \text{finite } D \wedge \text{path\_image } p \subseteq \bigcup D$ 
      by (simp add: compact_eq_Heine_Borel cover_def)
    then obtain D where D:  $D \subseteq \text{cover finite } D \text{ path\_image } p \subseteq \bigcup D$ 
      by blast
    then obtain k where k:  $k \subseteq \{0..1\} \text{ finite } k \text{ and } D_{\text{eq}}: D = ((\lambda z. \text{ball } z (ee z / 3)) \circ p) \text{ ` } k$ 
      unfolding cover_def path_image_def image_comp
      by (meson finite_subset_image)
    then have kne:  $k \neq \{\}$ 
      using D by auto
    have pi:  $\bigwedge i. i \in k \implies p i \in \text{path\_image } p$ 
      using k by (auto simp: path_image_def)
    then have eepi:  $\bigwedge i. i \in k \implies 0 < ee(p i)$ 
      by (metis ee)
    define e where  $e = \text{Min}((ee \circ p) \text{ ` } k)$ 
    have fin_eep:  $\text{finite } ((ee \circ p) \text{ ` } k)$ 
      using k by blast
    have  $0 < e$ 
      using ee k by (simp add: kne e_def Min_gr_iff [OF fin_eep] eepi)
    have uniformly_continuous_on  $\{0..1\} p$ 
      using p by (simp add: path_def compact_uniformly_continuous)
    then obtain d::real where d:  $d > 0$ 
      and de:  $\bigwedge x x'. |x' - x| < d \implies x \in \{0..1\} \implies x' \in \{0..1\} \implies \text{cmod } (p x' - p x) < e / 3$ 
      unfolding uniformly_continuous_on_def dist_norm real_norm_def
      by (metis divide_pos_pos  $\langle 0 < e \rangle$  zero_less_numeral)
    then obtain N::nat where N:  $N > 0 \text{ inverse } N < d$ 
      using real_arch_inverse [of d] by auto
    show ?thesis
    proof (intro exI conjI allI; clarify?)
      show  $e / 3 > 0$ 
        using  $\langle 0 < e \rangle$  by simp
      fix g h
      assume g:  $\text{valid\_path } g \text{ and } ghp: \forall t \in \{0..1\}. \text{cmod } (g t - p t) < e / 3 \wedge \text{cmod } (h t - p t) < e / 3$ 

```

```

    and h: valid_path h
    and joins: linked_paths attends g h
  { fix t::real
    assume t: 0 ≤ t t ≤ 1
    then obtain u where u: u ∈ k and ptu: p t ∈ ball(p u) (ee(p u) / 3)
      using ‹path_image p ⊆ ⋃ D› D_eq by (force simp: path_image_def)
    then have ele: e ≤ ee (p u) using fin_eep
      by (simp add: e_def)
    have cmod (g t - p t) < e / 3 cmod (h t - p t) < e / 3
      using ghp t by auto
    with ele have cmod (g t - p t) < ee (p u) / 3
      cmod (h t - p t) < ee (p u) / 3
      by linarith+
    then have g t ∈ ball(p u) (ee(p u)) h t ∈ ball(p u) (ee(p u))
      using norm_diff_triangle_ineq [of g t p t p t p u]
      norm_diff_triangle_ineq [of h t p t p t p u] ptu eepi u
      by (force simp: dist_norm ball_def norm_minus_commute)+
    then have g t ∈ S h t ∈ S using ee u k
      by (auto simp: path_image_def ball_def)
  }
  then have ghs: path_image g ⊆ S path_image h ⊆ S
    by (auto simp: path_image_def)
  moreover
  { fix f
    assume fhols: f holomorphic_on S
    then have fpa: f contour_integrable_on g f contour_integrable_on h
      using g ghs h holomorphic_on_imp_continuous_on os contour_integrable_holomorphic_simple
      by blast+
    have conf: continuous_on S f
      by (simp add: fhols holomorphic_on_imp_continuous_on)
    { fix z
      assume z: z ∈ path_image p
      have f holomorphic_on ball z (ee z)
        using fhols ee z holomorphic_on_subset by blast
      then have ∃ ff. (∀ w ∈ ball z (ee z). (ff has_field_derivative f w) (at w))
        using holomorphic_convex_primitive [of ball z (ee z) {} f, simplified]
        by (metis open_ball at_within_open holomorphic_on_def holomorphic_on_imp_continuous_on mem_ball)
    }
    then obtain ff where ff:
      ⋀ z w. [z ∈ path_image p; w ∈ ball z (ee z)] ⟹ (ff z has_field_derivative
      f w) (at w)
      by metis
    { fix n
      assume n: n ≤ N
      then have contour_integral(subpath 0 (n/N) h) f - contour_integral(subpath
      0 (n/N) g) f =
        contour_integral(linepath (g(n/N)) (h(n/N))) f - con-
        tour_integral(linepath (g 0) (h 0)) f

```

```

proof (induct n)
  case 0 show ?case by simp
next
  case (Suc n)
  obtain t where t:  $t \in k$  and  $p (n/N) \in \text{ball}(p\ t) (ee(p\ t) / 3)$ 
    using  $\langle \text{path\_image } p \subseteq \bigcup D \rangle$  [THEN subsetD, where  $c=p (n/N)$ ] D_eq
N Suc.premis
  by (force simp: path_image_def)
  then have ptu:  $\text{cmod } (p\ t - p (n/N)) < ee (p\ t) / 3$ 
    by (simp add: dist_norm)
  have e3le:  $e/3 \leq ee (p\ t) / 3$  using fin_ee p t
    by (simp add: e_def)
  { fix x
    assume x:  $n/N \leq x \leq (1 + n)/N$ 
    then have nN01:  $0 \leq n/N (1 + n)/N \leq 1$ 
      using Suc.premis by auto
    then have x01:  $0 \leq x \leq 1$ 
      using x by linarith+
    have cmod (p t - p x) <  $ee (p\ t) / 3 + e/3$ 
    proof (rule norm_diff_triangle_less [OF ptu de])
      show  $|\text{real } n / \text{real } N - x| < d$ 
      using x N by (auto simp: field_simps)
    qed (use x01 Suc.premis in auto)
    then have ptx:  $\text{cmod } (p\ t - p\ x) < 2*ee (p\ t)/3$ 
      using e3le ee pi [OF t] by simp
    have cmod (p t - g x) <  $2*ee (p\ t)/3 + e/3$ 
      using gh p x01
    by (force simp add: norm_minus_commute intro!: norm_diff_triangle_less
[OF ptx])
    also have ...  $\leq ee (p\ t)$ 
      using e3le ee pi [OF t] by simp
    finally have gg:  $\text{cmod } (p\ t - g\ x) < ee (p\ t)$  .
    have cmod (p t - h x) <  $2*ee (p\ t)/3 + e/3$ 
      using gh p x01
    by (force simp add: norm_minus_commute intro!: norm_diff_triangle_less
[OF ptx])
    also have ...  $\leq ee (p\ t)$ 
      using e3le ee pi [OF t] by simp
    finally have cmod (p t - g x) <  $ee (p\ t)$  cmod (p t - h x) <  $ee (p\ t)$ 
      using gg by auto
  } note ptgh_ee = this
  have closed_segment (g (n/N)) (h (n/N)) = path_image (linepath (h
(n/N)) (g (n/N)))
    by (simp add: closed_segment_commute)
  also have pi_hgn:  $\dots \subseteq \text{ball } (p\ t) (ee (p\ t))$ 
    using ptgh_ee [of n/N] Suc.premis
  by (auto simp: field_simps dist_norm dest: segment_furthest_le [where
 $y=p\ t$ ])
  finally have gh_ns:  $\text{closed\_segment } (g (n/N)) (h (n/N)) \subseteq S$ 

```



```

      using ee pi t by blast
    have pi_ghn': path_image (linepath (g ((1 + n) / N)) (h ((1 + n) / N)))
  ⊆ ball (p t) (ee (p t))
      using ptgh_ee [of (1+n)/N] Suc.premss
    by (auto simp: field_simps dist_norm dest: segment_furthest_le [where
y=p t])
  then have gh_n's: closed_segment (g ((1 + n) / N)) (h ((1 + n) / N))
  ⊆ S
      using ⟨N>0⟩ Suc.premss ee pi t
    by (auto simp: Path_Connected.path_image_join field_simps)
  have pi_subset_ball:
    path_image (subpath (n/N) ((1+n) / N) g +++ linepath (g ((1+n)
/ N)) (h ((1+n) / N)) +++)
      subpath ((1+n) / N) (n/N) h +++ linepath (h (n/N)) (g
(n/N)))
    ⊆ ball (p t) (ee (p t))
  proof (intro subset_path_image_join pi_hgn pi_ghn')
    show path_image (subpath (n/N) ((1+n) / N) g) ⊆ ball (p t) (ee (p t))
    path_image (subpath ((1+n) / N) (n/N) h) ⊆ ball (p t) (ee (p t))
    using ⟨N>0⟩ Suc.premss
    by (auto simp: path_image_subpath dist_norm field_simps ptgh_ee)
  qed
  have pi0: (f has_contour_integral 0)
    (subpath (n/N) ((Suc n)/N) g +++ linepath(g ((Suc n) / N))
(h((Suc n) / N)) +++)
    subpath ((Suc n) / N) (n/N) h +++ linepath(h (n/N)) (g
(n/N)))
  proof (rule Cauchy_theorem_primitive)
    show ∧x. x ∈ ball (p t) (ee (p t))
      ⇒ (ff (p t) has_field_derivative f x) (at x within ball (p t) (ee
(p t)))
    by (metis ff_open_ball at_within_open pi t)
  qed (use Suc.premss pi_subset_ball in ⟨simp_all add: valid_path_subpath
g h⟩)
  have fpa1: f contour_integrable_on subpath (n/N) (real (Suc n) / real N)
g
    using Suc.premss by (simp add: contour_integrable_subpath g fpa)
  have fpa2: f contour_integrable_on linepath (g (real (Suc n) / real N)) (h
(real (Suc n) / real N))
    using gh_n's
    by (auto intro!: contour_integrable_continuous_linepath continu-
ous_on_subset [OF conf])
  have fpa3: f contour_integrable_on linepath (h (n/N)) (g (n/N))
    using gh_ns
    by (auto simp: closed_segment_commute intro!: contour_integrable_continuous_linepath
continuous_on_subset [OF conf])
  have eq0: contour_integral (subpath (n/N) ((Suc n) / real N) g) f +
    contour_integral (linepath (g ((Suc n) / N)) (h ((Suc n) / N))) f
+

```

```

      contour_integral (subpath ((Suc n) / N) (n/N) h) f +
      contour_integral (linepath (h (n/N)) (g (n/N))) f = 0
    using contour_integral_unique [OF pi0] Suc.premis
  by (simp add: g h fpa valid_path_subpath contour_integrable_subpath
      fpa1 fpa2 fpa3 algebra_simps del: of_nat_Suc)
  have *:  $\bigwedge hn\ he\ hn'\ gn\ gd\ gn'\ hgn\ ghn\ gh0\ ghn'.$ 
     $\llbracket hn - gn = ghn - gh0;$ 
     $gd + ghn' + he + hgn = (0::complex);$ 
     $hn - he = hn'; gn + gd = gn'; hgn = -ghn \rrbracket \implies hn' - gn' =$ 
     $ghn' - gh0$ 
  by (auto simp: algebra_simps)
  have contour_integral (subpath 0 (n/N) h) f - contour_integral (subpath
    ((Suc n) / N) (n/N) h) f =
    contour_integral (subpath 0 (n/N) h) f + contour_integral (subpath
    (n/N) ((Suc n) / N) h) f
  unfolding reversepath_subpath [symmetric, of ((Suc n) / N)]
  using Suc.premis by (simp add: h fpa contour_integral_reversepath
    valid_path_subpath contour_integrable_subpath)
  also have ... = contour_integral (subpath 0 ((Suc n) / N) h) f
  using Suc.premis by (simp add: contour_integral_subpath_combine h
    fpa)
  finally have pi0_eq:
    contour_integral (subpath 0 (n/N) h) f - contour_integral (subpath
    ((Suc n) / N) (n/N) h) f =
    contour_integral (subpath 0 ((Suc n) / N) h) f .
  show ?case
  proof (rule * [OF Suc.hyps eq0 pi0_eq])
    show contour_integral (subpath 0 (n/N) g) f +
      contour_integral (subpath (n/N) ((Suc n) / N) g) f =
      contour_integral (subpath 0 ((Suc n) / N) g) f
    using Suc.premis contour_integral_subpath_combine fpa(1) g by auto
    show contour_integral (linepath (h (n/N)) (g (n/N))) f = - con-
    tour_integral (linepath (g (n/N)) (h (n/N))) f
    by (metis contour_integral_unique fpa3 has_contour_integral_integral
      has_contour_integral_reverse_linepath)
    qed (use Suc.premis in auto)
  qed
  } note ind = this
  have contour_integral h f = contour_integral g f
  using ind [OF order_refl] N joins
  by (simp add: linked_paths_def pathstart_def pathfinish_def split: if_split_asm)
  }
  ultimately
  show path_image g  $\subseteq S \wedge$  path_image h  $\subseteq S \wedge (\forall f. f \text{ holomorphic\_on } S \longrightarrow$ 
  contour_integral h f = contour_integral g f)
  by metis
  qed
  qed
  qed

```

lemma

assumes $\text{open } S \text{ path } p \text{ path_image } p \subseteq S$
shows $\text{contour_integral_nearby_ends}$:
 $\exists d. 0 < d \wedge$
 $(\forall g \ h. \text{valid_path } g \wedge \text{valid_path } h \wedge$
 $(\forall t \in \{0..1\}. \text{norm}(g \ t - p \ t) < d \wedge \text{norm}(h \ t - p \ t) < d) \wedge$
 $\text{pathstart } h = \text{pathstart } g \wedge \text{pathfinish } h = \text{pathfinish } g$
 $\longrightarrow \text{path_image } g \subseteq S \wedge$
 $\text{path_image } h \subseteq S \wedge$
 $(\forall f. f \text{ holomorphic_on } S$
 $\longrightarrow \text{contour_integral } h \ f = \text{contour_integral } g \ f))$
and $\text{contour_integral_nearby_loops}$:
 $\exists d. 0 < d \wedge$
 $(\forall g \ h. \text{valid_path } g \wedge \text{valid_path } h \wedge$
 $(\forall t \in \{0..1\}. \text{norm}(g \ t - p \ t) < d \wedge \text{norm}(h \ t - p \ t) < d) \wedge$
 $\text{pathfinish } g = \text{pathstart } g \wedge \text{pathfinish } h = \text{pathstart } h$
 $\longrightarrow \text{path_image } g \subseteq S \wedge$
 $\text{path_image } h \subseteq S \wedge$
 $(\forall f. f \text{ holomorphic_on } S$
 $\longrightarrow \text{contour_integral } h \ f = \text{contour_integral } g \ f))$
using $\text{contour_integral_nearby}$ [OF assms, **where** $\text{atends}=\text{True}$]
using $\text{contour_integral_nearby}$ [OF assms, **where** $\text{atends}=\text{False}$]
unfolding linked_paths_def **by** simp_all

lemma $\text{contour_integral_bound_exists}$:

assumes S : $\text{open } S$
and g : $\text{valid_path } g$
and pag : $\text{path_image } g \subseteq S$
shows $\exists L. 0 < L \wedge$
 $(\forall f \ B. f \text{ holomorphic_on } S \wedge (\forall z \in S. \text{norm}(f \ z) \leq B)$
 $\longrightarrow \text{norm}(\text{contour_integral } g \ f) \leq L*B)$

proof –

have $\text{path } g$ **using** g
by $(\text{simp add: valid_path_imp_path})$
then obtain $d::\text{real}$ **and** p
where d : $0 < d$
and p : $\text{polynomial_function } p \text{ path_image } p \subseteq S$
and pi : $\bigwedge f. f \text{ holomorphic_on } S \implies \text{contour_integral } g \ f = \text{contour_integral}$

$p \ f$

using $\text{contour_integral_nearby_ends}$ [OF $S \langle \text{path } g \rangle \text{ pag}$]
by $(\text{metis cancel_comm_monoid_add_class.diff_cancel } g \ \text{norm_zero } \text{path_approx_polynomial_function}$
 $\text{valid_path_polynomial_function})$
then obtain p' **where** p' : $\text{polynomial_function } p'$
 $\bigwedge x. (p \text{ has_vector_derivative } (p' \ x)) \text{ (at } x)$
by $(\text{blast intro: has_vector_derivative_polynomial_function that})$
then have $\text{bounded}(p' \ ' \ \{0..1\})$
using $\text{continuous_on_polynomial_function}$
by $(\text{force simp: intro!: compact_imp_bounded compact_continuous_image})$

```

    then obtain L where L: L>0 and nop':  $\bigwedge x. \llbracket 0 \leq x; x \leq 1 \rrbracket \implies \text{norm } (p' x) \leq L$ 
    by (force simp: bounded_pos)
  { fix f B
    assume f: f holomorphic_on S and B:  $\bigwedge z. z \in S \implies \text{cmod } (f z) \leq B$ 
    then have f contour_integrable_on p  $\wedge$  valid_path p
      using p S
    by (blast intro: valid_path_polynomial_function contour_integrable_holomorphic_simple
        holomorphic_on_imp_continuous_on)
    moreover have cmod (vector_derivative p (at x)) * cmod (f (p x))  $\leq L * B$ 
  if  $0 \leq x \leq 1$  for x
    proof (rule mult_mono)
      show cmod (vector_derivative p (at x))  $\leq L$ 
        by (metis nop' p'(2) that vector_derivative_at)
      show cmod (f (p x))  $\leq B$ 
        by (metis B atLeastAtMost_iff imageI p(2) path_defs(4) subset_eq that)
    qed (use <L>0 in auto)
    ultimately
    have cmod (integral {0..1} ( $\lambda x. f (p x) * \text{vector\_derivative } p \text{ (at } x)$ )))  $\leq L * B$ 
      by (intro order_trans [OF integral_norm_bound_integral])
      (auto simp: mult.commute norm_mult contour_integrable_on)
    then have cmod (contour_integral g f)  $\leq L * B$ 
      using contour_integral_integral f pi by presburger
  } then
  show ?thesis using <L > 0>
    by (intro exI[of _ L]) auto
qed

```

2.8 Homotopy forms of Cauchy's theorem

lemma Cauchy_theorem_homotopic:

```

  assumes hom: if attends then homotopic_paths S g h else homotopic_loops S g h
  and open S and f: f holomorphic_on S
  and vpg: valid_path g and vph: valid_path h
  shows contour_integral g f = contour_integral h f
proof -
  have pathsf: linked_paths attends g h
    using hom by (auto simp: linked_paths_def homotopic_paths_imp_pathstart
        homotopic_paths_imp_pathfinish homotopic_loops_imp_loop)
  obtain k :: real  $\times$  real  $\Rightarrow$  complex
    where contk: continuous_on ({0..1}  $\times$  {0..1}) k
    and ks: k ' ({0..1}  $\times$  {0..1})  $\subseteq S$ 
    and k [simp]:  $\forall x. k (0, x) = g x \ \forall x. k (1, x) = h x$ 
    and ksf:  $\forall t \in \{0..1\}. \text{linked\_paths attends } g (\lambda x. k (t, x))$ 
    using hom pathsf by (auto simp: linked_paths_def homotopic_paths_def
        homotopic_loops_def homotopic_with_def split: if_split_asm)
  have ucontk: uniformly_continuous_on ({0..1}  $\times$  {0..1}) k
    by (blast intro: compact_Times compact_uniformly_continuous [OF contk])

```

```

{ fix t::real assume t: t ∈ {0..1}
  have Pair t ' {0..1} ⊆ {0..1} × {0..1}
    using t by force
  then have pak: path (k ∘ (λu. (t, u)))
    unfolding path_def
    by (intro continuous_intros continuous_on_subset [OF contk])
  have pik: path_image (k ∘ Pair t) ⊆ S
    using ks t by (auto simp: path_image_def)
  obtain e where e>0 and e:
    ∧g h. [valid_path g; valid_path h;
      ∀u∈{0..1}. cmod (g u - (k ∘ Pair t) u) < e ∧ cmod (h u - (k ∘
Pair t) u) < e;
      linked_paths attends g h]
    ⇒ contour_integral h f = contour_integral g f
    using contour_integral_nearby [OF ‹open S› pak pik, of attends] f by metis
  obtain d where d>0 and d:
    ∧x x'. [x ∈ {0..1} × {0..1}; x' ∈ {0..1} × {0..1}; norm (x'-x) < d] ⇒
norm (k x' - k x) < e/4
    by (rule uniformly_continuous_onE [OF ucontk, of e/4]) (auto simp: dist_norm
‹e>0)
    { fix t1 t2
      assume t1: 0 ≤ t1 t1 ≤ 1 and t2: 0 ≤ t2 t2 ≤ 1 and ltd: |t1 - t| < d |t2
- t| < d
      have no2: norm(g1 - kt) < e if norm(g1 - k1) < e/4 norm(k1 - kt) <
e/4 for g1 k1 kt :: complex
      proof (rule norm_triangle_half_l)
        show cmod (g1 - k1) < e/2 cmod (kt - k1) < e/2
        using ‹e > 0› that by (auto simp: norm_minus_commute intro: or-
der_less_trans)
      qed
      have ∃ d>0. ∀ g1 g2. valid_path g1 ∧ valid_path g2 ∧
        (∀ u∈{0..1}. cmod (g1 u - k (t1, u)) < d ∧ cmod (g2 u - k
(t2, u)) < d) ∧
        linked_paths attends g1 g2 →
        contour_integral g2 f = contour_integral g1 f
      using t t1 t2 ltd ‹e > 0›
      by (rule_tac x=e/4 in exI) (auto intro!: e simp: d no2 simp del: less_divide_eq_numeral1)
    }
  then have ∃ e. 0 < e ∧
    (∀ t1 t2. t1 ∈ {0..1} ∧ t2 ∈ {0..1} ∧ |t1 - t| < e ∧ |t2 - t| < e
    → (∃ d. 0 < d ∧
      (∀ g1 g2. valid_path g1 ∧ valid_path g2 ∧
        (∀ u ∈ {0..1}.
          norm(g1 u - k((t1,u))) < d ∧ norm(g2 u - k((t2,u))) < d) ∧
          linked_paths attends g1 g2
          → contour_integral g2 f = contour_integral g1 f)))
    by (rule_tac x=d in exI) (simp add: ‹d > 0›)
  }
then obtain ee where ee:

```

```


$$\bigwedge t. t \in \{0..1\} \implies ee\ t > 0 \wedge$$


$$(\forall t1\ t2. t1 \in \{0..1\} \longrightarrow t2 \in \{0..1\} \longrightarrow |t1 - t| < ee\ t \longrightarrow |t2 - t| <$$


$$ee\ t$$


$$\longrightarrow (\exists d. 0 < d \wedge$$


$$(\forall g1\ g2. \text{valid\_path } g1 \wedge \text{valid\_path } g2 \wedge$$


$$(\forall u \in \{0..1\}. \text{norm}(g1\ u - k((t1, u))) < d \wedge \text{norm}(g2\ u - k((t2, u))) < d) \wedge$$


$$\text{linked\_paths\_atends } g1\ g2$$


$$\longrightarrow \text{contour\_integral } g2\ f = \text{contour\_integral } g1\ f)))$$

by metis
note ee_rule = ee [THEN conjunct2, rule_format, of 0 0 0]
define C where C = ( $\lambda t. \text{ball } t\ (ee\ t / 3)$ ) ‘ {0..1}
obtain C' where C': C'  $\subseteq C$  finite C' and C'01: {0..1}  $\subseteq \bigcup C'$ 
proof (rule_compactE [OF compact_interval])
  show {0..1}  $\subseteq \bigcup C$ 
    using ee [THEN conjunct1] by (auto simp: C_def dist_norm)
qed (use C_def in auto)
define kk where kk = {t  $\in$  {0..1}. ball t (ee t / 3)  $\in C'$ }
have kk01: kk  $\subseteq \{0..1\}$  by (auto simp: kk_def)
define e where e = Min (ee ‘ kk)
have C'_eq: C' = ( $\lambda t. \text{ball } t\ (ee\ t / 3)$ ) ‘ kk
  using C' by (auto simp: kk_def C_def)
have ee_pos[simp]:  $\bigwedge t. t \in \{0..1\} \implies ee\ t > 0$ 
  by (simp add: kk_def ee)
moreover have finite_kk
  using ‘finite C'’ kk01 by (force simp: C'_eq inj_on_def ball_eq_ball_iff dest: ee_pos finite_imageD)
moreover have kk  $\neq \{\}$  using ‘{0..1}  $\subseteq \bigcup C'$ ’ C'_eq by force
ultimately have e > 0
  using finite_less_Inf_iff [of ee ‘ kk 0] kk01 by (force simp: e_def)
then obtain N::nat where N > 0 and N: 1/N < e/3
  by (meson divide_pos_pos nat_approx_posE zero_less_Suc zero_less_numeral)
have e_le_ee:  $\bigwedge i. i \in kk \implies e \leq ee\ i$ 
  using ‘finite kk’ by (simp add: e_def Min_le_iff [of ee ‘ kk])
have plus:  $\exists t \in kk. x \in \text{ball } t\ (ee\ t / 3)$  if x  $\in \{0..1\}$  for x
  using C' subsetD [OF C'01 that] unfolding C'_eq by blast
have [OF order_refl]:
  
$$\exists d. 0 < d \wedge (\forall j. \text{valid\_path } j \wedge (\forall u \in \{0..1\}. \text{norm}(j\ u - k\ (n/N, u)) <$$


$$d) \wedge \text{linked\_paths\_atends } g\ j$$


$$\longrightarrow \text{contour\_integral } j\ f = \text{contour\_integral } g\ f)$$

  if n  $\leq N$  for n
using that
proof (induct n)
  case 0 show ?case
    using ee_rule
    by clarsimp (metis diff_self norm_eq_zero vpg)
next
  case (Suc n)
  then have N01: n/N  $\in \{0..1\}$  (Suc n)/N  $\in \{0..1\}$  by auto

```

```

then obtain t where t: t ∈ kk n/N ∈ ball t (ee t / 3)
  using plus [of n/N] by blast
then have nN_less: |n/N - t| < ee t
  by (simp add: dist_norm del: less_divide_eq_numeral1)
have n'N_less: |real (Suc n) / real N - t| < ee t
  using t N ⟨N > 0⟩ e_le_ee [of t]
  by (simp add: dist_norm add_divide_distrib abs_diff_less_iff del: less_divide_eq_numeral1)
(simp add: field_simps)
have t01: t ∈ {0..1} using ⟨kk ⊆ {0..1}⟩ ⟨t ∈ kk⟩ by blast
obtain d1 where d1 > 0 and d1:
  ∧g1 g2. [valid_path g1; valid_path g2;
    ∀u∈{0..1}. cmod (g1 u - k (n/N, u)) < d1 ∧ cmod (g2 u - k
((Suc n) / N, u)) < d1;
    linked_paths_atends g1 g2]
    ⇒ contour_integral g2 f = contour_integral g1 f
  using ee [THEN conjunct2, rule_format, OF t01 N01 nN_less n'N_less] by
fastforce
have n ≤ N using Suc.prem by auto
with Suc.hyps
obtain d2 where d2 > 0
  and d2: ∧j. [valid_path j; ∀u∈{0..1}. cmod (j u - k (n/N, u)) < d2;
    linked_paths_atends g j]
    ⇒ contour_integral j f = contour_integral g f
  by auto
have Pair (n/N) ' {0..1} ⊆ {0..1} × {0..1}
  using N01 by auto
then have continuous_on {0..1} (k ∘ (λu. (n/N, u)))
  by (intro continuous_intros continuous_on_subset [OF contk])
then have pkn: path (λu. k (n/N, u))
  by (simp add: path_def)
have min12: min d1 d2 > 0 by (simp add: ⟨0 < d1⟩ ⟨0 < d2⟩)
obtain p where polynomial_function p
  and psf: pathstart p = pathstart (λu. k (n/N, u))
    pathfinish p = pathfinish (λu. k (n/N, u))
  and pk_le: ∧t. t∈{0..1} ⇒ cmod (p t - k (n/N, t)) < min d1 d2
  using path_approx_polynomial_function [OF pkn min12] by blast
then have vpp: valid_path p using valid_path_polynomial_function by blast
have lpa: linked_paths_atends g p
  by (metis (mono_tags, lifting) N01(1) ksf linked_paths_def pathfinish_def
pathstart_def psf)
show ?case
proof (intro exI; safe)
fix j
assume valid_path j linked_paths_atends g j
  and ∀u∈{0..1}. cmod (j u - k (real (Suc n) / real N, u)) < min d1 d2
then have contour_integral j f = contour_integral p f
  using pk_le N01(1) ksf by (force intro!: vpp d1 simp add: linked_paths_def
psf)
also have ... = contour_integral g f

```

```

    using pk_le by (force intro!: vpp d2 lpa)
    finally show contour_integral j f = contour_integral g f .
  qed (simp add: <0 < d1> <0 < d2>)
  qed
  then obtain d where 0 < d
     $\bigwedge j. \text{valid\_path } j \wedge (\forall u \in \{0..1\}. \text{norm}(j\ u - k\ (1,u)) < d) \wedge$ 
    linked_paths attends g j
     $\implies \text{contour\_integral } j\ f = \text{contour\_integral } g\ f$ 
    using <N>0> by auto
    then have linked_paths attends g h  $\implies \text{contour\_integral } h\ f = \text{contour\_integral } g\ f$ 
    using <N>0> vph by fastforce
    then show ?thesis
      by (simp add: pathsf)
  qed

```

proposition *Cauchy_theorem_homotopic_paths:*

```

  assumes hom: homotopic_paths S g h
    and open S and f: f holomorphic_on S
    and vpg: valid_path g and vph: valid_path h
  shows contour_integral g f = contour_integral h f
  using Cauchy_theorem_homotopic [of True S g h] assms by simp

```

proposition *Cauchy_theorem_homotopic_loops:*

```

  assumes hom: homotopic_loops S g h
    and open S and f: f holomorphic_on S
    and vpg: valid_path g and vph: valid_path h
  shows contour_integral g f = contour_integral h f
  using Cauchy_theorem_homotopic [of False S g h] assms by simp

```

lemma *has_contour_integral_newpath:*

```

   $\llbracket (f \text{ has\_contour\_integral } y) \ h; f \text{ contour\_integrable\_on } g; \text{contour\_integral } g\ f$ 
   $= \text{contour\_integral } h\ f \rrbracket$ 
   $\implies (f \text{ has\_contour\_integral } y) \ g$ 
  using has_contour_integral_integral contour_integral_unique by auto

```

lemma *Cauchy_theorem_null_homotopic:*

```

   $\llbracket f \text{ holomorphic\_on } S; \text{open } S; \text{valid\_path } g; \text{homotopic\_loops } S\ g\ (\text{linepath } a$ 
   $a) \rrbracket$ 
   $\implies (f \text{ has\_contour\_integral } 0) \ g$ 
  by (metis Cauchy_theorem_homotopic_loops contour_integrable_holomorphic_simple
    valid_path_linepath
    contour_integral_trivial has_contour_integral_integral homotopic_loops_imp_subset)

```

end

3 Winding numbers

theory *Winding_Numbers*


```

imports Cauchy_Integral_Theorem
begin

```

3.1 Definition

definition *winding_number_prop* :: $[real \Rightarrow complex, complex, real, real \Rightarrow complex, complex] \Rightarrow bool$ **where**

```

  winding_number_prop  $\gamma$   $z$   $e$   $p$   $n$   $\equiv$ 
    valid_path  $p \wedge z \notin \text{path\_image } p \wedge$ 
    pathstart  $p = \text{pathstart } \gamma \wedge$ 
    pathfinish  $p = \text{pathfinish } \gamma \wedge$ 
     $(\forall t \in \{0..1\}. \text{norm}(\gamma t - p t) < e) \wedge$ 
    contour_integral  $p (\lambda w. 1/(w - z)) = 2 * \pi * i * n$ 

```

definition *winding_number* :: $[real \Rightarrow complex, complex] \Rightarrow complex$ **where**

```

  winding_number  $\gamma$   $z \equiv \text{SOME } n. \forall e > 0. \exists p. \text{winding\_number\_prop } \gamma z e p n$ 

```

lemma *winding_number*:

```

  assumes path  $\gamma$   $z \notin \text{path\_image } \gamma$   $0 < e$ 

```

```

  shows  $\exists p. \text{winding\_number\_prop } \gamma z e p (\text{winding\_number } \gamma z)$ 

```

proof –

```

  have path_image  $\gamma \subseteq \text{UNIV} - \{z\}$ 

```

```

  using assms by blast

```

```

  then obtain  $d$ 

```

```

    where  $d > 0$ 

```

```

    and  $pi\_eq: \bigwedge h1 h2. \text{valid\_path } h1 \wedge \text{valid\_path } h2 \wedge$ 

```

```

       $(\forall t \in \{0..1\}. \text{cmod } (h1 t - \gamma t) < d \wedge \text{cmod } (h2 t - \gamma t) < d) \wedge$ 

```

```

       $\text{pathstart } h2 = \text{pathstart } h1 \wedge \text{pathfinish } h2 = \text{pathfinish } h1 \longrightarrow$ 

```

```

       $\text{path\_image } h1 \subseteq \text{UNIV} - \{z\} \wedge \text{path\_image } h2 \subseteq \text{UNIV} -$ 

```

```

       $\{z\} \wedge$ 

```

```

       $(\forall f. f \text{ holomorphic\_on } \text{UNIV} - \{z\} \longrightarrow \text{contour\_integral } h2 f$ 

```

```

       $= \text{contour\_integral } h1 f)$ 

```

```

    using contour_integral_nearby_ends [of  $\text{UNIV} - \{z\}$   $\gamma$ ] assms by (auto simp:
open_delete)

```

```

    then obtain  $h$  where  $h: \text{polynomial\_function } h \wedge \text{pathstart } h = \text{pathstart } \gamma \wedge$ 
       $\text{pathfinish } h = \text{pathfinish } \gamma \wedge$ 

```

```

       $(\forall t \in \{0..1\}. \text{norm}(h t - \gamma t) < d/2)$ 

```

```

    using path_approx_polynomial_function [OF  $\langle \text{path } \gamma \rangle$ , of  $d/2$ ]  $d$  by (metis
half_gt_zero_iff)

```

```

    define  $nn$  where  $nn = 1/(2 * \pi * i) * \text{contour\_integral } h (\lambda w. 1/(w - z))$ 

```

```

    have  $\exists n. \forall e > 0. \exists p. \text{winding\_number\_prop } \gamma z e p n$ 

```

```

    proof (rule_tac  $x=nn$  in exI, clarify)

```

```

      fix  $e::real$ 

```

```

      assume  $e > 0$ 

```

```

      obtain  $p$  where  $p: \text{polynomial\_function } p \wedge$ 

```

```

         $\text{pathstart } p = \text{pathstart } \gamma \wedge \text{pathfinish } p = \text{pathfinish } \gamma \wedge (\forall t \in \{0..1\}. \text{cmod } (p t - \gamma t) < \min e (d/2))$ 

```

```

       $\text{cmod } (p t - \gamma t) < \min e (d/2))$ 

```

```

      using path_approx_polynomial_function [OF  $\langle \text{path } \gamma \rangle$ , of  $\min e (d/2)$ ]  $d$ 

```

```

<0<e>
  by (metis min_less_iff_conj zero_less_divide_iff zero_less_numeral)
  have ( $\lambda w. 1 / (w - z)$ ) holomorphic_on UNIV - {z}
  by (auto simp: intro!: holomorphic_intros)
  then have winding_number_prop  $\gamma z e p nn$ 
  using pi_eq [of h p] h p d
  by (auto simp: valid_path_polynomial_function norm_minus_commute
nn_def winding_number_prop_def)
  then show  $\exists p. \text{winding\_number\_prop } \gamma z e p nn$ 
  by metis
qed
then show ?thesis
  unfolding winding_number_def by (rule someI2_ex) (blast intro: <0<e>)
qed

```

lemma winding_number_unique:

```

  assumes  $\gamma: \text{path } \gamma z \notin \text{path\_image } \gamma$ 
  and pi:  $\bigwedge e. e > 0 \implies \exists p. \text{winding\_number\_prop } \gamma z e p n$ 
  shows winding_number  $\gamma z = n$ 
proof -
  have path_image  $\gamma \subseteq \text{UNIV} - \{z\}$ 
  using assms by blast
  then obtain e
  where e:  $e > 0$ 
  and pi_eq:  $\bigwedge h1 h2 f. [\text{valid\_path } h1; \text{valid\_path } h2; \\ (\forall t \in \{0..1\}. \text{cmod } (h1\ t - \gamma\ t) < e \wedge \text{cmod } (h2\ t - \gamma\ t) < e); \\ \text{pathstart } h2 = \text{pathstart } h1; \text{pathfinish } h2 = \text{pathfinish } h1; f \\ \text{holomorphic\_on } \text{UNIV} - \{z\}] \implies \\ \text{contour\_integral } h2\ f = \text{contour\_integral } h1\ f$ 
  using contour_integral_nearby_ends [of UNIV - {z}  $\gamma$ ] assms by (auto simp:
open_delete)
  obtain p where p: winding_number_prop  $\gamma z e p n$ 
  using pi [OF e] by blast
  obtain q where q: winding_number_prop  $\gamma z e q (\text{winding\_number } \gamma z)$ 
  using winding_number [OF  $\gamma e$ ] by blast
  have  $2 * \text{complex\_of\_real } pi * i * n = \text{contour\_integral } p (\lambda w. 1 / (w - z))$ 
  using p by (auto simp: winding_number_prop_def)
  also have  $\dots = \text{contour\_integral } q (\lambda w. 1 / (w - z))$ 
  proof (rule pi_eq)
    show  $(\lambda w. 1 / (w - z)) \text{ holomorphic\_on } \text{UNIV} - \{z\}$ 
    by (auto intro!: holomorphic_intros)
  qed (use p q in <auto simp: winding_number_prop_def norm_minus_commute>)
  also have  $\dots = 2 * \text{complex\_of\_real } pi * i * \text{winding\_number } \gamma z$ 
  using q by (auto simp: winding_number_prop_def)
  finally have  $2 * \text{complex\_of\_real } pi * i * n = 2 * \text{complex\_of\_real } pi * i * \\ \text{winding\_number } \gamma z .$ 
  then show ?thesis
  by simp
qed

```

```

lemma winding_number_prop_reversepath:
  assumes winding_number_prop  $\gamma$   $z \in p$   $n$ 
  shows winding_number_prop (reversepath  $\gamma$ )  $z \in$  (reversepath  $p$ )  $(-n)$ 
proof -
  have  $p$ : valid_path  $p$   $z \notin$  path_image  $p$  pathstart  $p$  = pathstart  $\gamma$ 
    pathfinish  $p$  = pathfinish  $\gamma \wedge t. t \in \{0..1\} \implies \text{norm } (\gamma t - p t) < e$ 
    contour_integral  $p$  ( $\lambda w. 1 / (w - z)$ ) =  $2 * \text{complex\_of\_real } \pi * i * n$ 
  using assms by (auto simp: winding_number_prop_def)
  show ?thesis
  unfolding winding_number_prop_def
proof (intro conjI strip)
  show norm (reversepath  $\gamma$   $t$  - reversepath  $p$   $t$ )  $< e$  if  $t \in \{0..1\}$  for  $t$ 
    unfolding reversepath_def using  $p(5)[\text{of } 1 - t]$  that by auto
  show contour_integral (reversepath  $p$ ) ( $\lambda w. 1 / (w - z)$ ) =
    complex_of_real ( $2 * \pi$ ) *  $i * -n$ 
    using  $p$  by (subst contour_integral_reversepath) auto
qed (use  $p$  in auto)
qed

lemma winding_number_prop_reversepath_iff:
  winding_number_prop (reversepath  $\gamma$ )  $z \in p$   $n \longleftrightarrow$  winding_number_prop  $\gamma$   $z \in$ 
  (reversepath  $p$ )  $(-n)$ 
  using winding_number_prop_reversepath[of reversepath  $\gamma$   $z \in p$   $n$ ]
    winding_number_prop_reversepath[of  $\gamma$   $z \in$  reversepath  $p$   $-n$ ] by auto

lemma winding_number_unique_loop:
  assumes  $\gamma$ : path  $\gamma$   $z \notin$  path_image  $\gamma$ 
    and loop: pathfinish  $\gamma$  = pathstart  $\gamma$ 
    and  $\pi$ :
       $\wedge e. e > 0 \implies \exists p. \text{valid\_path } p \wedge z \notin \text{path\_image } p \wedge$ 
        pathfinish  $p$  = pathstart  $p \wedge$ 
        ( $\forall t \in \{0..1\}. \text{norm } (\gamma t - p t) < e$ )  $\wedge$ 
        contour_integral  $p$  ( $\lambda w. 1 / (w - z)$ ) =  $2 * \pi * i * n$ 
  shows winding_number  $\gamma$   $z$  =  $n$ 
proof -
  have path_image  $\gamma \subseteq \text{UNIV} - \{z\}$ 
    using assms by blast
  then obtain  $e$ 
    where  $e$ :  $e > 0$ 
    and  $\pi$ _eq:  $\wedge h1 h2 f. [\text{valid\_path } h1; \text{valid\_path } h2;$ 
      ( $\forall t \in \{0..1\}. \text{cmod } (h1 t - \gamma t) < e \wedge \text{cmod } (h2 t - \gamma t) < e$ );
      pathfinish  $h1$  = pathstart  $h1$ ; pathfinish  $h2$  = pathstart  $h2$ ;  $f$ 
    holomorphic_on  $\text{UNIV} - \{z\}] \implies$ 
      contour_integral  $h2$   $f$  = contour_integral  $h1$   $f$ 
    using contour_integral_nearby_loops [of  $\text{UNIV} - \{z\}$   $\gamma$ ] assms by (auto simp:
    open_delete)
  obtain  $p$  where  $p$ :

```

```

    valid_path p ∧ z ∉ path_image p ∧ pathfinish p = pathstart p ∧
    (∀ t ∈ {0..1}. norm (γ t - p t) < e) ∧
    contour_integral p (λw. 1/(w - z)) = 2 * pi * i * n
  using pi [OF e] by blast
  obtain q where q: winding_number_prop γ z e q (winding_number γ z)
  using winding_number [OF γ e] by blast
  have 2 * complex_of_real pi * i * n = contour_integral p (λw. 1 / (w - z))
  using p by auto
  also have ... = contour_integral q (λw. 1 / (w - z))
  proof (rule pi_eq)
    show (λw. 1 / (w - z)) holomorphic_on UNIV - {z}
    by (auto intro!: holomorphic_intros)
  qed (use p q loop in ⟨auto simp: winding_number_prop_def norm_minus_commute⟩)
  also have ... = 2 * complex_of_real pi * i * winding_number γ z
  using q by (auto simp: winding_number_prop_def)
  finally have 2 * complex_of_real pi * i * n = 2 * complex_of_real pi * i *
winding_number γ z .
  then show ?thesis
  by simp
qed

```

proposition *winding_number_valid_path*:

```

  assumes valid_path γ z ∉ path_image γ
  shows winding_number γ z = 1/(2*pi*i) * contour_integral γ (λw. 1/(w - z))
  by (rule winding_number_unique)
  (use assms in ⟨auto simp: valid_path_imp_path winding_number_prop_def⟩)

```

proposition *has_contour_integral_winding_number*:

```

  assumes γ: valid_path γ z ∉ path_image γ
  shows ((λw. 1/(w - z)) has_contour_integral (2*pi*i*winding_number γ z))
  γ
  by (simp add: winding_number_valid_path has_contour_integral_integral con-
tour_integrable_inversediff assms)

```

lemma *winding_number_trivial* [simp]: $z \neq a \implies \text{winding_number}(\text{linepath } a \ a) \ z = 0$

```

  by (simp add: winding_number_valid_path)

```

lemma *winding_number_subpath_trivial* [simp]: $z \neq g \ x \implies \text{winding_number}(\text{subpath } x \ x \ g) \ z = 0$

```

  by (simp add: path_image_subpath winding_number_valid_path)

```

lemma *winding_number_join*:

```

  assumes γ1: path γ1 z ∉ path_image γ1
  and γ2: path γ2 z ∉ path_image γ2
  and pathfinish γ1 = pathstart γ2
  shows winding_number(γ1 +++ γ2) z = winding_number γ1 z + wind-
ing_number γ2 z
  proof (rule winding_number_unique)

```

```

show  $\exists p. \text{winding\_number\_prop } (\gamma 1 +++ \gamma 2) \ z \ e \ p$ 
      ( $\text{winding\_number } \gamma 1 \ z + \text{winding\_number } \gamma 2 \ z$ ) if  $e > 0$  for  $e$ 
proof -
  obtain  $p1$  where  $\text{winding\_number\_prop } \gamma 1 \ z \ e \ p1$  ( $\text{winding\_number } \gamma 1 \ z$ )
    using  $\langle 0 < e \rangle \ \gamma 1 \ \text{winding\_number}$  by blast
  moreover
  obtain  $p2$  where  $\text{winding\_number\_prop } \gamma 2 \ z \ e \ p2$  ( $\text{winding\_number } \gamma 2 \ z$ )
    using  $\langle 0 < e \rangle \ \gamma 2 \ \text{winding\_number}$  by blast
  ultimately
  have  $\text{winding\_number\_prop } (\gamma 1 +++ \gamma 2) \ z \ e \ (p1 +++ p2)$  ( $\text{winding\_number } \gamma 1 \ z + \text{winding\_number } \gamma 2 \ z$ )
    using assms
  apply (simp add: winding_number_prop_def not_in_path_image_join contour_integrable_inversediff algebra_simps)
  apply (auto simp: joinpaths_def)
  done
  then show ?thesis
    by blast
qed
qed (use assms in  $\langle \text{auto simp: not\_in\_path\_image\_join} \rangle$ )

```

```

lemma winding_number_reversepath:
  assumes  $\text{path } \gamma \ z \notin \text{path\_image } \gamma$ 
  shows  $\text{winding\_number}(\text{reversepath } \gamma) \ z = - (\text{winding\_number } \gamma \ z)$ 
proof (rule winding_number_unique)
  show  $\exists p. \text{winding\_number\_prop } (\text{reversepath } \gamma) \ z \ e \ p$  ( $-\text{winding\_number } \gamma \ z$ )
  if  $e > 0$  for  $e$ 
  proof -
    obtain  $p$  where  $\text{winding\_number\_prop } \gamma \ z \ e \ p$  ( $\text{winding\_number } \gamma \ z$ )
      using  $\langle 0 < e \rangle \ \text{assms winding\_number}$  by blast
    then have  $\text{winding\_number\_prop } (\text{reversepath } \gamma) \ z \ e \ (\text{reversepath } p)$  ( $-\text{winding\_number } \gamma \ z$ )
      using assms unfolding winding_number_prop_def
    apply (simp add: contour_integral_reversepath contour_integrable_inversediff valid_path_imp_reverse)
    apply (auto simp: reversepath_def)
    done
    then show ?thesis
      by blast
  qed
qed (use assms in auto)

```

```

lemma winding_number_shiftpath:
  assumes  $\gamma: \text{path } \gamma \ z \notin \text{path\_image } \gamma$ 
  and  $\text{pathfinish } \gamma = \text{pathstart } \gamma \ a \in \{0..1\}$ 
  shows  $\text{winding\_number}(\text{shiftpath } a \ \gamma) \ z = \text{winding\_number } \gamma \ z$ 
proof (rule winding_number_unique_loop)
  show  $\exists p. \text{valid\_path } p \wedge z \notin \text{path\_image } p \wedge \text{pathfinish } p = \text{pathstart } p \wedge$ 
    ( $\forall t \in \{0..1\}. \text{cmod } (\text{shiftpath } a \ \gamma \ t - p \ t) < e$ )  $\wedge$ 

```

```

      contour_integral p ( $\lambda w. 1 / (w - z)$ ) =
      2 * pi * i * winding_number  $\gamma$  z
    if  $e > 0$  for  $e$ 
  proof -
    obtain p where winding_number_prop  $\gamma$  z e p (winding_number  $\gamma$  z)
    using  $\langle 0 < e \rangle$  assms winding_number by blast
    then show ?thesis
    apply (rule_tac x=shiftpath a p in exI)
    using assms that
    apply (auto simp: winding_number_prop_def path_image_shiftpath pathfinish_shiftpath pathstart_shiftpath contour_integral_shiftpath)
    apply (simp add: shiftpath_def)
    done
  qed
qed (use assms in  $\langle$ auto simp: path_shiftpath path_image_shiftpath pathfinish_shiftpath pathstart_shiftpath $\rangle$ )

```

```

lemma winding_number_split_linepath:
  assumes  $c \in \text{closed\_segment } a \ b$   $z \notin \text{closed\_segment } a \ b$ 
  shows winding_number(linepath a b) z = winding_number(linepath a c) z +
  winding_number(linepath c b) z
proof -
  have  $z \notin \text{closed\_segment } a \ c$   $z \notin \text{closed\_segment } c \ b$ 
  using assms by (meson convex_contains_segment convex_segment_ends_in_segment subsetCE)+
  then show ?thesis
  using assms
  by (simp add: winding_number_valid_path contour_integral_split_linepath [symmetric] continuous_on_inversediff field_simps)
qed

```

```

lemma winding_number_cong:
  ( $\bigwedge t. [0 \leq t; t \leq 1] \implies p \ t = q \ t$ )  $\implies$  winding_number p z = winding_number q z
  by (simp add: winding_number_def winding_number_prop_def pathstart_def pathfinish_def)

```

```

lemma winding_number_constI:
  assumes  $c \neq z$  and  $g: \bigwedge t. [0 \leq t; t \leq 1] \implies g \ t = c$ 
  shows winding_number g z = 0
proof -
  have winding_number g z = winding_number (linepath c c) z
  using g winding_number_cong by fastforce
  moreover have winding_number (linepath c c) z = 0
  using  $\langle c \neq z \rangle$  by auto
  ultimately show ?thesis by auto
qed

```

```

lemma winding_number_offset: winding_number p z = winding_number ( $\lambda w. p$ 

```

```

 $w - z) \ 0$ 
  unfolding winding_number_def
proof (intro ext arg_cong [where f = Eps] arg_cong [where f = All] imp_cong
  refl, safe)
  fix n e g
  assume  $0 < e$  and  $g$ : winding_number_prop  $p \ z \ e \ g \ n$ 
  then show  $\exists r. \text{winding\_number\_prop } (\lambda w. p \ w - z) \ 0 \ e \ r \ n$ 
    by (rule_tac  $x = \lambda t. g \ t - z$  in exI)
      (force simp: winding_number_prop_def contour_integral_integral_valid_path_def
  path_defs
      vector_derivative_def has_vector_derivative_diff_const piece-
wise_C1_differentiable_diff C1_differentiable_imp_piecewise)
  next
  fix n e g
  assume  $0 < e$  and  $g$ : winding_number_prop  $(\lambda w. p \ w - z) \ 0 \ e \ g \ n$ 
  then have winding_number_prop  $p \ z \ e \ (\lambda t. g \ t + z) \ n$ 
    apply (simp add: winding_number_prop_def contour_integral_integral_valid_path_def
  path_defs
      piecewise_C1_differentiable_add vector_derivative_def has_vector_derivative_add_const
  C1_differentiable_imp_piecewise)
    apply (force simp: algebra_simps)
    done
  then show  $\exists r. \text{winding\_number\_prop } p \ z \ e \ r \ n$ 
    by metis
qed

lemma winding_number_offset_NO_MATCH:
  NO_MATCH  $0 \ z \implies \text{winding\_number } p \ z = \text{winding\_number } (\lambda w. p \ w - z) \ 0$ 
  using winding_number_offset by metis

lemma winding_number_negatepath:
  assumes  $\gamma$ : valid_path  $\gamma$  and  $0$ :  $0 \notin \text{path\_image } \gamma$ 
  shows winding_number( $\text{uminus} \circ \gamma$ )  $0 = \text{winding\_number } \gamma \ 0$ 
proof -
  have  $(/) \ 1$  contour_integrable_on  $\gamma$ 
    using  $0 \ \gamma$  contour_integrable_inversediff by fastforce
  then have  $((\lambda z. 1/z) \text{ has\_contour\_integral } \text{contour\_integral } \gamma \ ((/) \ 1)) \ \gamma$ 
    by (rule has_contour_integral_integral)
  then have  $((\lambda z. 1 / - z) \text{ has\_contour\_integral } - \text{contour\_integral } \gamma \ ((/) \ 1))$ 
 $\gamma$ 
    using has_contour_integral_neg by auto
  then have contour_integral ( $\text{uminus} \circ \gamma$ )  $((/) \ 1) =$ 
    contour_integral  $\gamma \ ((/) \ 1)$ 
    using  $\gamma$  by (simp add: contour_integral_unique has_contour_integral_negatepath)
  then show ?thesis
    using assms by (simp add: winding_number_valid_path valid_path_negatepath
  image_def path_defs)
qed

```

```

lemma winding_number_cnj:
  assumes path  $\gamma$   $z \notin \text{path\_image } \gamma$ 
  shows winding_number (cnj  $\circ \gamma$ ) (cnj  $z$ ) = -cnj (winding_number  $\gamma$   $z$ )
proof (rule winding_number_unique)
  show  $\exists p. \text{winding\_number\_prop } (\text{cnj} \circ \gamma) (\text{cnj } z) \text{ e } p \text{ } (-\text{cnj } (\text{winding\_number } \gamma \text{ } z))$ 
  if  $e > 0$  for  $e$ 
  proof -
    from winding_number[OF assms(1,2)  $\langle e > 0 \rangle$ ]
    obtain  $p$  where winding_number_prop  $\gamma$   $z$   $e$   $p$  (winding_number  $\gamma$   $z$ )
    by blast
    then have  $p: \text{valid\_path } p \text{ } z \notin \text{path\_image } p$ 
      pathstart  $p = \text{pathstart } \gamma$ 
      pathfinish  $p = \text{pathfinish } \gamma$ 
       $\bigwedge t. t \in \{0..1\} \implies \text{cmod } (\gamma \text{ } t - p \text{ } t) < e$  and
       $p\_cont: \text{contour\_integral } p \text{ } (\lambda w. 1 / (w - z)) =$ 
        complex_of_real (2 * pi) * i * winding_number  $\gamma$   $z$ 
    unfolding winding_number_prop_def by auto

    have valid_path (cnj  $\circ p$ )
      using  $p(1)$  by (subst valid_path_cnj) auto
    moreover have  $\text{cnj } z \notin \text{path\_image } (\text{cnj} \circ p)$ 
      using  $p(2)$  by (auto simp: path_image_def)
    moreover have pathstart (cnj  $\circ p$ ) = pathstart (cnj  $\circ \gamma$ )
      using  $p(3)$  by (simp add: pathstart_compose)
    moreover have pathfinish (cnj  $\circ p$ ) = pathfinish (cnj  $\circ \gamma$ )
      using  $p(4)$  by (simp add: pathfinish_compose)
    moreover have  $\text{cmod } ((\text{cnj} \circ \gamma) \text{ } t - (\text{cnj} \circ p) \text{ } t) < e$ 
      if  $t \in \{0..1\}$  for  $t$ 
    proof -
      have  $(\text{cnj} \circ \gamma) \text{ } t - (\text{cnj} \circ p) \text{ } t = \text{cnj } (\gamma \text{ } t - p \text{ } t)$ 
        by simp
      also have  $\text{norm } \dots = \text{norm } (\gamma \text{ } t - p \text{ } t)$ 
        by (subst complex_mod_cnj) auto
      also have  $\dots < e$ 
        using  $p(5)$ [OF  $t$ ] by simp
      finally show ?thesis .
    qed

    moreover have  $\text{contour\_integral } (\text{cnj} \circ p) (\lambda w. 1 / (w - \text{cnj } z)) =$ 
       $\text{cnj } (\text{complex\_of\_real } (2 * \text{pi}) * \text{i} * \text{winding\_number } \gamma \text{ } z) \text{ (is ?L=?R)}$ 
    proof -
      have ?L = contour_integral (cnj  $\circ p$ ) (cnj  $\circ (\lambda w. 1 / (\text{cnj } w - z)))$ 
        by (simp add: o_def)
      also have  $\dots = \text{cnj } (\text{contour\_integral } p (\lambda x. 1 / (x - z)))$ 
        using  $p(1)$  by (subst contour_integral_cnj) (auto simp: o_def)
      also have  $\dots = ?R$ 
        using  $p\_cont$  by simp
      finally show ?thesis .
    qed

```



```

ultimately show ?thesis
  by (intro exI[of _ cnj ∘ γ]) (auto simp: winding_number_prop_def)
qed
show path (cnj ∘ γ)
  by (intro path_continuous_image continuous_intros) (use assms in auto)
show cnj z ∉ path_image (cnj ∘ γ)
  using ⟨z ∉ path_image γ⟩ unfolding path_image_def by auto
qed

```

A combined theorem deducing several things piecewise.

lemma *winding_number_join_pos_combined*:

```

  [[valid_path γ1; z ∉ path_image γ1; 0 < Re(winding_number γ1 z);
   valid_path γ2; z ∉ path_image γ2; 0 < Re(winding_number γ2 z); pathfinish
  γ1 = pathstart γ2]]
  ⇒ valid_path(γ1 +++ γ2) ∧ z ∉ path_image(γ1 +++ γ2) ∧ 0 <
  Re(winding_number(γ1 +++ γ2) z)
  by (simp add: valid_path_join path_image_join winding_number_join valid_path_imp_path)

```

3.1.1 Useful sufficient conditions for the winding number to be positive

lemma *Re_winding_number*:

```

  [[valid_path γ; z ∉ path_image γ]]
  ⇒ Re(winding_number γ z) = Im(contour_integral γ (λw. 1/(w - z))) /
  (2*πi)
  by (simp add: winding_number_valid_path field_simps Re_divide power2_eq_square)

```

lemma *winding_number_pos_le*:

```

  assumes γ: valid_path γ z ∉ path_image γ
  and ge: ∧x. [[0 < x; x < 1]] ⇒ 0 ≤ Im (vector_derivative γ (at x) * cnj(γ
  x - z))
  shows 0 ≤ Re(winding_number γ z)

```

proof –

```

  have ge0: 0 ≤ Im (vector_derivative γ (at x) / (γ x - z)) if x: 0 < x < 1
  for x

```

```

  using ge by (simp add: Complex.Im_divide algebra_simps x)

```

```

  let ?vd = λx. 1 / (γ x - z) * vector_derivative γ (at x)

```

```

  let ?int = λz. contour_integral γ (λw. 1 / (w - z))

```

```

  have 0 ≤ Im (?int z)

```

```

  proof (rule has_integral_component_nonneg [of i, simplified])

```

```

    show ∧x. x ∈ cbox 0 1 ⇒ 0 ≤ Im (if 0 < x ∧ x < 1 then ?vd x else 0)

```

```

    by (force simp: ge0)

```

```

  have ((λa. 1 / (a - z)) has_contour_integral contour_integral γ (λw. 1 / (w
  - z))) γ

```

```

  using γ by (simp flip: add: contour_integrable_inversediff has_contour_integral_integral)

```

```

  then have hi: (?vd has_integral ?int z) (cbox 0 1)

```

```

  using has_contour_integral by auto

```

```

  show ((λx. if 0 < x ∧ x < 1 then ?vd x else 0) has_integral ?int z) (cbox 0 1)

```

```

  by (rule has_integral_spike_interior [OF hi]) simp

```

```

qed
then show ?thesis
  by (simp add: Re_winding_number [OF  $\gamma$ ] field_simps)
qed

lemma winding_number_pos_lt_lemma:
  assumes  $\gamma$ : valid_path  $\gamma$   $z \notin \text{path\_image } \gamma$ 
    and  $e$ :  $0 < e$ 
    and  $ge$ :  $\bigwedge x. [0 < x; x < 1] \implies e \leq \text{Im } (\text{vector\_derivative } \gamma \text{ (at } x) / (\gamma x - z))$ 
  shows  $0 < \text{Re}(\text{winding\_number } \gamma z)$ 
proof -
  let ?vd =  $\lambda x. 1 / (\gamma x - z) * \text{vector\_derivative } \gamma \text{ (at } x)$ 
  let ?int =  $\lambda z. \text{contour\_integral } \gamma (\lambda w. 1 / (w - z))$ 
  have  $e \leq \text{Im } (\text{contour\_integral } \gamma (\lambda w. 1 / (w - z)))$ 
proof (rule has_integral_component_le [of i  $\lambda x. i * e$   $i * e \{0..1\}$ , simplified])
  have (( $\lambda a. 1 / (a - z)$ ) has_contour_integral contour_integral  $\gamma (\lambda w. 1 / (w - z))) \gamma$ 
  - z)))  $\gamma$ 
  thm has_integral_component_le [of i  $\lambda x. i * e$   $i * e \{0..1\}$ , simplified]
  using  $\gamma$  by (simp add: contour_integrable_inversediff has_contour_integral_integral)
  then have hi: (?vd has_integral ?int  $z$ ) (cbox 0 1)
  using has_contour_integral by auto
  show (( $\lambda x. \text{if } 0 < x \wedge x < 1 \text{ then ?vd } x \text{ else } i * e$ ) has_integral ?int  $z$ )  $\{0..1\}$ 
  by (rule has_integral_spike_interior [OF hi, simplified box_real]) (use  $e$  in simp)
  show  $\bigwedge x. 0 \leq x \wedge x \leq 1 \implies e \leq \text{Im } (\text{if } 0 < x \wedge x < 1 \text{ then ?vd } x \text{ else } i * e)$ 
  by (simp add: ge)
qed (use has_integral_const_real [of _ 0 1] in auto)
with  $e$  show ?thesis
  by (simp add: Re_winding_number [OF  $\gamma$ ] field_simps)
qed

lemma winding_number_pos_lt:
  assumes  $\gamma$ : valid_path  $\gamma$   $z \notin \text{path\_image } \gamma$ 
    and  $e$ :  $0 < e$ 
    and  $ge$ :  $\bigwedge x. [0 < x; x < 1] \implies e \leq \text{Im } (\text{vector\_derivative } \gamma \text{ (at } x) * \text{cnj}(\gamma x - z))$ 
  shows  $0 < \text{Re } (\text{winding\_number } \gamma z)$ 
proof -
  have bm: bounded (( $\lambda w. w - z$ ) ' ( $\text{path\_image } \gamma$ ))
  using bounded_translation [of _  $-z$ ]  $\gamma$  by (simp add: bounded_valid_path_image)
  then obtain  $B$  where  $B$ :  $B > 0$  and Bno:  $\bigwedge x. x \in (\lambda w. w - z) ' (\text{path\_image } \gamma) \implies \text{norm } x \leq B$ 
  using bounded_pos [THEN iffD1, OF bm] by blast
  { fix  $x::\text{real}$  assume  $x$ :  $0 < x < 1$ 
  then have B2:  $\text{cmod } (\gamma x - z)^2 \leq B^2$  using Bno [of  $\gamma x - z$ ]
  by (simp add: path_image_def power2_eq_square mult_mono')
  with  $x$  have  $\gamma x \neq z$  using  $\gamma$ 
  using path_image_def by fastforce

```

```

    then have  $e / B^2 \leq e / (\text{cmod } (\gamma x - z))^2$ 
      using  $B B^2 e$  by (auto simp: divide_left_mono)
    also have  $\dots \leq \text{Im } (\text{vector\_derivative } \gamma (at x) * \text{cnj } (\gamma x - z)) / (\text{cmod } (\gamma x - z))^2$ 
      using  $ge [OF x]$  by (auto simp: divide_right_mono)
    finally have  $e / B^2 \leq \text{Im } (\text{vector\_derivative } \gamma (at x) * \text{cnj } (\gamma x - z)) / (\text{cmod } (\gamma x - z))^2$  .
    then have  $e / B^2 \leq \text{Im } (\text{vector\_derivative } \gamma (at x) / (\gamma x - z))$ 
      by (simp add: complex_div_cnj [of  $\_ \gamma x - z$  for  $x$ ] del: complex_cnj_diff times_complex.sel)
  } note * = this
  show ?thesis
    using  $e B$  by (simp add: * winding_number_pos_lt_lemma [OF  $\gamma$ , of  $e/B^2$ ])
qed

```

3.2 The winding number is an integer

Proof from the book Complex Analysis by Lars V. Ahlfors, Chapter 4, section 2.1, Also on page 134 of Serge Lang's book with the name title, etc.

lemma *exp_fg*:

```

  fixes  $z::\text{complex}$ 
  assumes  $g: (g \text{ has\_vector\_derivative } g') (at x \text{ within } s)$ 
    and  $f: (f \text{ has\_vector\_derivative } (g' / (g x - z))) (at x \text{ within } s)$ 
    and  $z: g x \neq z$ 
  shows  $((\lambda x. \text{exp}(-f x) * (g x - z)) \text{ has\_vector\_derivative } 0) (at x \text{ within } s)$ 
proof -
  have *:  $(\text{exp} \circ (\lambda x. (-f x))) \text{ has\_vector\_derivative } - (g' / (g x - z)) * \text{exp}(-f x)$ 
    (at  $x$  within  $s$ )
    using assms unfolding has_vector_derivative_def scaleR_conv_of_real
    by (auto intro!: derivative_eq_intros)
  show ?thesis
    using  $z$  by (auto intro!: derivative_eq_intros * [unfolded o_def]  $g$ )
qed

```

lemma *winding_number_exp_integral*:

```

  fixes  $z::\text{complex}$ 
  assumes  $\gamma: \gamma \text{ piecewise\_C1\_differentiable\_on } \{a..b\}$ 
    and  $ab: a \leq b$ 
    and  $z: z \notin \gamma ' \{a..b\}$ 
  shows  $(\lambda x. \text{vector\_derivative } \gamma (at x) / (\gamma x - z)) \text{ integrable\_on } \{a..b\}$ 
    (is ?thesis1)
     $\text{exp}(-(\text{integral } \{a..b\} (\lambda x. \text{vector\_derivative } \gamma (at x) / (\gamma x - z)))) * (\gamma b - z) = \gamma a - z$ 
    (is ?thesis2)
proof -
  let  $?D\gamma = \lambda x. \text{vector\_derivative } \gamma (at x)$ 
  have [simp]:  $\bigwedge x. a \leq x \implies x \leq b \implies \gamma x \neq z$ 
    using  $z$  by force
  have  $con\_g: \text{continuous\_on } \{a..b\} \gamma$ 

```

```

    using  $\gamma$  by (simp add: piecewise_C1_differentiable_on_def)
  obtain  $k$  where  $fink$ : finite  $k$  and  $g\_C1\_diff$ :  $\gamma$  C1_differentiable_on  $\{a..b\} - k$ 
  using  $\gamma$  by (force simp: piecewise_C1_differentiable_on_def)
  have  $\circ$ : open  $\{a <..< b\} - k$ 
  using  $\langle finite\ k \rangle$  by (simp add: finite_imp_closed open_Diff)
  moreover have  $\{a <..< b\} - k \subseteq \{a..b\} - k$ 
  by force
  ultimately have  $g\_diff\_at$ :  $\bigwedge x. \llbracket x \notin k; x \in \{a <..< b\} \rrbracket \implies \gamma$  differentiable at  $x$ 
  by (metis Diff_iff differentiable_on_subset C1_diff_imp_diff [OF  $g\_C1\_diff$ ]
differentiable_on_def at_within_open)
  { fix  $w$ 
    assume  $w \neq z$ 
    have continuous_on (ball  $w$  (cmod  $(w - z)$ ))  $(\lambda w. 1 / (w - z))$ 
    by (auto simp: dist_norm intro!: continuous_intros)
    moreover have  $\bigwedge x. cmod\ (w - x) < cmod\ (w - z) \implies \exists f'. ((\lambda w. 1 / (w - z)))$  has_field_derivative  $f'$  (at  $x$ )
    by (auto simp: intro!: derivative_eq_intros)
    ultimately have  $\exists h. \forall y. norm(y - w) < norm(w - z) \longrightarrow (h$  has_field_derivative  $1 / (y - z))$  (at  $y$ )
    using holomorphic_convex_primitive [of ball  $w$  (norm  $(w - z)$ )  $\{\}$   $\lambda w. 1 / (w - z)$ ]
    by (force simp: field_differentiable_def Ball_def dist_norm at_within_open NO_MATCH norm_minus_commute)
  }
  then obtain  $h$  where  $h$ :  $\bigwedge w\ y. w \neq z \implies norm(y - w) < norm(w - z) \implies (h$  has_field_derivative  $1 / (y - z))$  (at  $y$ )
  by meson
  have  $exy$ :  $\exists y. ((\lambda x. inverse\ (\gamma\ x - z) * ?D\gamma\ x)$  has_integral  $y$ )  $\{a..b\}$ 
  unfolding integrable_on_def [symmetric]
  proof (rule contour_integral_local_primitive_any [OF piecewise_C1_imp_differentiable [OF  $\gamma$ ]])
    show  $\exists d\ h. 0 < d \wedge (\forall y. cmod\ (y - w) < d \longrightarrow (h$  has_field_derivative  $inverse\ (y - z))$  (at  $y$  within  $-\{z\}$ ))
    if  $w \in -\{z\}$  for  $w$ 
    using that inverse_eq_divide has_field_derivative_at_within  $h$ 
    by (metis Compl_insert DiffD2 insertCI right_minus_eq zero_less_norm_iff)
  qed simp
  have  $vg\_int$ :  $(\lambda x. ?D\gamma\ x / (\gamma\ x - z))$  integrable_on  $\{a..b\}$ 
  unfolding box_real [symmetric] divide_inverse_commute
  by (auto intro!: exy integrable_subinterval simp add: integrable_on_def ab)
  with  $ab$  show ?thesis1
  by (simp add: divide_inverse_commute integral_def integrable_on_def)
  { fix  $t$ 
    assume  $t$ :  $t \in \{a..b\}$ 
    have cball: continuous_on (ball  $(\gamma\ t)$  (dist  $(\gamma\ t)\ z$ ))  $(\lambda x. inverse\ (x - z))$ 
    using  $z$  by (auto intro!: continuous_intros simp: dist_norm)
  }

```

```

have icd:  $\bigwedge x. \text{cmod } (\gamma \ t - x) < \text{cmod } (\gamma \ t - z) \implies (\lambda w. \text{inverse } (w - z))$ 
field_differentiable at x
unfolding field_differentiable_def by (force simp: intro!: derivative_eq_intros)
obtain h where h:  $\bigwedge x. \text{cmod } (\gamma \ t - x) < \text{cmod } (\gamma \ t - z) \implies$ 
 $(h \text{ has\_field\_derivative } \text{inverse } (x - z)) \text{ (at } x \text{ within } \{y. \text{cmod } (\gamma \ t - y) < \text{cmod } (\gamma \ t - z)\})$ 
using holomorphic_convex_primitive [where f =  $\lambda w. \text{inverse}(w - z)$ , OF
convex_ball finite.emptyI cball icd]
by simp (auto simp: ball_def dist_norm that)
have exp  $(- (\text{integral } \{a..t\} (\lambda x. ?D\gamma \ x / (\gamma \ x - z)))) * (\gamma \ t - z) = \gamma \ a - z$ 
proof (rule has_derivative_zero_unique_strong_interval [of  $\{a..b\} \cup k \ a \ b$ ])
show continuous_on  $\{a..b\} (\lambda b. \text{exp } (- \text{integral } \{a..b\} (\lambda x. ?D\gamma \ x / (\gamma \ x - z)))) * (\gamma \ b - z)$ 
by (auto intro!: continuous_intros con_g indefinite_integral_continuous_1
[OF vg_int])
show  $((\lambda b. \text{exp } (- \text{integral } \{a..b\} (\lambda x. ?D\gamma \ x / (\gamma \ x - z)))) * (\gamma \ b - z))$ 
has_derivative  $(\lambda h. 0)$ 
 $(\text{at } x \text{ within } \{a..b\})$ 
if  $x \in \{a..b\} - (\{a, b\} \cup k)$  for x
proof -
have x:  $x \notin k \ a < x < b$ 
using that by auto
then have  $x \in \text{interior } (\{a..b\} - k)$ 
using open_subset_interior [OF o] by fastforce
then have con: isCont  $?D\gamma \ x$ 
using g_C1_diff x by (auto simp: C1_differentiable_on_eq intro: continuous_on_interior)
then have con_vd: continuous  $(\text{at } x \text{ within } \{a..b\}) (\lambda x. ?D\gamma \ x)$ 
by (rule continuous_at_imp_continuous_within)
have gdx:  $\gamma$  differentiable at x
using x by (simp add: g_diff_at)
then obtain d where d:  $(\gamma \text{ has\_derivative } (\lambda x. x *_R d)) \text{ (at } x)$ 
by (auto simp add: differentiable_iff_scaleR)
show  $((\lambda c. \text{exp } (- \text{integral } \{a..c\} (\lambda x. ?D\gamma \ x / (\gamma \ x - z)))) * (\gamma \ c - z))$ 
has_derivative  $(\lambda h. 0)$ 
 $(\text{at } x \text{ within } \{a..b\})$ 
proof (rule exp_fg [unfolded has_vector_derivative_def, simplified])
show  $(\gamma \text{ has\_derivative } (\lambda c. c *_R d)) \text{ (at } x \text{ within } \{a..b\})$ 
using d by (blast intro: has_derivative_at_withinI)
have  $((\lambda x. \text{integral } \{a..x\} (\lambda x. ?D\gamma \ x / (\gamma \ x - z))) \text{ has\_vector\_derivative } d / (\gamma \ x - z))$ 
 $(\text{at } x \text{ within } \{a..b\})$ 
proof (rule has_vector_derivative_eq_rhs [OF integral_has_vector_derivative_continuous_at
[where S = {}, simplified]])
show continuous  $(\text{at } x \text{ within } \{a..b\}) (\lambda x. \text{vector\_derivative } \gamma \text{ (at } x) / (\gamma \ x - z))$ 
using continuous_at_imp_continuous_at_within differentiable_imp_continuous_within
gdx x
by (intro con_vd continuous_intros) (force+)

```

```

      show vector_derivative  $\gamma$  (at  $x$ ) / ( $\gamma x - z$ ) =  $d$  / ( $\gamma x - z$ )
      using d vector_derivative_at
      by (simp add: vector_derivative_at has_vector_derivative_def)
    qed (use x vg_int in auto)
  then show (( $\lambda x$ . integral {a..x} ( $\lambda x$ .  $?D\gamma x$  / ( $\gamma x - z$ ))) has_derivative
( $\lambda c$ .  $c *_R (d / (\gamma x - z))$ ))
    (at  $x$  within {a..b})
    by (auto simp: has_vector_derivative_def)
  qed (use x in auto)
qed
qed
}
with ab show ?thesis2
  by (simp add: divide_inverse_commute integral_def)
qed

```

lemma winding_number_exp_2pi:

```

  [[path p;  $z \notin \text{path\_image } p$ ]]
   $\implies \text{pathfinish } p - z = \exp (2 * \pi * i * \text{winding\_number } p z) * (\text{pathstart } p - z)$ 
using winding_number [of p z 1] unfolding valid_path_def path_image_def path-
start_def pathfinish_def winding_number_prop_def
  by (force dest: winding_number_exp_integral(2) [of _ 0 1 z] simp: field_simps
contour_integral_integral_exp_minus)

```

lemma integer_winding_number_eq:

```

  assumes  $\gamma$ : path  $\gamma$  and  $z$ :  $z \notin \text{path\_image } \gamma$ 
  shows winding_number  $\gamma z \in \mathbb{Z} \iff \text{pathfinish } \gamma = \text{pathstart } \gamma$ 
proof -
  obtain p where p: valid_path p  $z \notin \text{path\_image } p$ 
    pathstart p = pathstart  $\gamma$  pathfinish p = pathfinish  $\gamma$ 
    and eq: contour_integral p ( $\lambda w$ .  $1 / (w - z)$ ) = complex_of_real (2 *
 $\pi i$ ) * i * winding_number  $\gamma z$ 
  using winding_number [OF assms, of 1] unfolding winding_number_prop_def
  by auto
  then have wneg: winding_number  $\gamma z = \text{winding\_number } p z$ 
    using eq winding_number_valid_path by force
  have iff: (winding_number  $\gamma z \in \mathbb{Z}$ )  $\iff (\exp (\text{contour\_integral } p (\lambda w$ .  $1 / (w - z))) = 1)$ 
    using eq by (simp add: exp_eq_1 complex_is_Int_iff)
  have  $\gamma 0 \neq z$ 
    by (metis pathstart_def pathstart_in_path_image z)
  then have exp (contour_integral p ( $\lambda w$ .  $1 / (w - z)$ )) = ( $\gamma 1 - z$ ) / ( $\gamma 0 - z$ )
    using p winding_number_exp_integral(2) [of p 0 1 z]
    by (simp add: valid_path_def path_defs contour_integral_integral_exp_minus
field_split_simps)
  then have winding_number p  $z \in \mathbb{Z} \iff \text{pathfinish } p = \text{pathstart } p$ 
    using p wneg iff by (auto simp: path_defs)
  then show ?thesis using p eq

```

by (auto simp: winding_number_valid_path)
qed

theorem integer_winding_number:

$\llbracket \text{path } \gamma; \text{pathfinish } \gamma = \text{pathstart } \gamma; z \notin \text{path_image } \gamma \rrbracket \implies \text{winding_number } \gamma$
 $z \in \mathbb{Z}$

by (metis integer_winding_number_eq)

If the winding number's magnitude is at least one, then the path must contain points in every direction.*) We can thus bound the winding number of a path that doesn't intersect a given ray.

lemma winding_number_pos_meets:

fixes $z::\text{complex}$

assumes $\gamma: \text{valid_path } \gamma$ and $z: z \notin \text{path_image } \gamma$ and $1: \text{Re } (\text{winding_number } \gamma z) \geq 1$

and $w: w \neq z$

shows $\exists a::\text{real}. 0 < a \wedge z + \text{of_real } a * (w - z) \in \text{path_image } \gamma$

proof –

have [simp]: $\bigwedge x. 0 \leq x \implies x \leq 1 \implies \gamma x \neq z$

using z by (auto simp: path_image_def)

have [simp]: $z \notin \gamma \text{ ‘ } \{0..1\}$

using path_image_def z by auto

have gpd: $\gamma \text{ piecewise_C1_differentiable_on } \{0..1\}$

using $\gamma \text{ valid_path_def}$ by blast

define r where $r = (w - z) / (\gamma 0 - z)$

have [simp]: $r \neq 0$

using $w z$ by (auto simp: r_def)

have cont: continuous_on $\{0..1\}$

$(\lambda x. \text{Im } (\text{integral } \{0..x\} (\lambda x. \text{vector_derivative } \gamma (at x) / (\gamma x - z))))$

by (intro continuous_intros indefinite_integral_continuous_1 winding_number_exp_integral [OF gpd]; simp)

have $\text{Arg2pi } r \leq 2*\pi$

by (simp add: Arg2pi_less_eq_real_def)

also have $\dots \leq \text{Im } (\text{integral } \{0..1\} (\lambda x. \text{vector_derivative } \gamma (at x) / (\gamma x - z)))$

using 1

by (simp add: winding_number_valid_path [OF γz] contour_integral_integral Complex.Re_divide field_simps power2_eq_square)

finally have $\text{Arg2pi } r \leq \text{Im } (\text{integral } \{0..1\} (\lambda x. \text{vector_derivative } \gamma (at x) / (\gamma x - z)))$.

then have $\exists t. t \in \{0..1\} \wedge \text{Im}(\text{integral } \{0..t\} (\lambda x. \text{vector_derivative } \gamma (at x) / (\gamma x - z))) = \text{Arg2pi } r$

by (simp add: Arg2pi_ge_0 cont IVT')

then obtain t where $t: t \in \{0..1\}$

and eqArg: $\text{Im } (\text{integral } \{0..t\} (\lambda x. \text{vector_derivative } \gamma (at x) / (\gamma x - z))) = \text{Arg2pi } r$

by blast

define i where $i = \text{integral } \{0..t\} (\lambda x. \text{vector_derivative } \gamma (at x) / (\gamma x - z))$

have gpd: $\gamma \text{ piecewise_C1_differentiable_on } \{0..t\}$

```

    by (metis atLeastAtMost_iff atLeastatMost_subset_iff order_refl piecewise_C1_differentiable_on_subset
    gpd t)
    have  $\exp(-i) * (\gamma t - z) = \gamma 0 - z$ 
    unfolding i_def
    proof (rule winding_number_exp_integral [OF gpd])
      show  $z \notin \gamma \cdot \{0..t\}$ 
      using t z unfolding path_image_def by force
    qed (use t in auto)
    then have  $*: \gamma t - z = \exp i * (\gamma 0 - z)$ 
    by (simp add: exp_minus field_simps)
    then have  $(w - z) = r * (\gamma 0 - z)$ 
    by (simp add: r_def)
    moreover have  $z + \exp(\operatorname{Re} i) * (\exp(i * \operatorname{Im} i) * (\gamma 0 - z)) = \gamma t$ 
    using * by (simp add: exp_eq_polar field_simps)
    moreover have  $\operatorname{Arg} 2\pi r = \operatorname{Im} i$ 
    using eqArg by (simp add: i_def)
    ultimately have  $z + \operatorname{complex\_of\_real}(\exp(\operatorname{Re} i)) * (w - z) / \operatorname{complex\_of\_real}(cmod r) = \gamma t$ 
    using Complex_Transcendental.Arg2pi_eq [of r]  $\langle r \neq 0 \rangle$ 
    by (metis mult.left_commute nonzero_mult_div_cancel_left norm_eq_zero
    of_real_0 of_real_eq_iff times_divide_eq_left)
    with t show ?thesis
    by (rule_tac  $x = \exp(\operatorname{Re} i) / \operatorname{norm} r$  in exI) (auto simp: path_image_def)
  qed

lemma winding_number_big_meets:
  fixes  $z::\text{complex}$ 
  assumes  $\gamma$ : valid_path  $\gamma$  and  $z$ :  $z \notin \text{path\_image } \gamma$  and  $|\operatorname{Re}(\text{winding\_number } \gamma z)| \geq 1$ 
  and  $w$ :  $w \neq z$ 
  shows  $\exists a::\text{real}. 0 < a \wedge z + \operatorname{of\_real} a * (w - z) \in \text{path\_image } \gamma$ 
proof -
  { assume  $\operatorname{Re}(\text{winding\_number } \gamma z) \leq -1$ 
    then have  $\operatorname{Re}(\text{winding\_number } (\text{reversepath } \gamma) z) \geq 1$ 
    by (simp add:  $\gamma$  valid_path_imp_path winding_number_reversepath z)
    moreover have valid_path (reversepath  $\gamma$ )
    using  $\gamma$  valid_path_imp_reverse by auto
    moreover have  $z \notin \text{path\_image } (\text{reversepath } \gamma)$ 
    by (simp add: z)
    ultimately have  $\exists a::\text{real}. 0 < a \wedge z + \operatorname{of\_real} a * (w - z) \in \text{path\_image } (\text{reversepath } \gamma)$ 
    using winding_number_pos_meets w by blast
    then have ?thesis
    by simp
  }
  then show ?thesis
  using assms
  by (simp add: abs_if winding_number_pos_meets split: if_split_asm)
qed

```



```

lemma winding_number_less_1:
  fixes z::complex
  shows
     $\llbracket \text{valid\_path } \gamma; z \notin \text{path\_image } \gamma; w \neq z; \\
    \bigwedge a::\text{real}. 0 < a \implies z + \text{of\_real } a * (w - z) \notin \text{path\_image } \gamma \rrbracket \\
    \implies \text{Re}(\text{winding\_number } \gamma z) < 1$ 
  by (auto simp: not_less dest: winding_number_big_meets)

```

One way of proving that WN=1 for a loop.

```

lemma winding_number_eq_1:
  fixes z::complex
  assumes  $\gamma$ : valid_path  $\gamma$  and z:  $z \notin \text{path\_image } \gamma$  and loop: pathfinish  $\gamma$  =
    pathstart  $\gamma$ 
  and 0:  $0 < \text{Re}(\text{winding\_number } \gamma z)$  and 2:  $\text{Re}(\text{winding\_number } \gamma z) < 2$ 
  shows winding_number  $\gamma z = 1$ 
proof -
  have winding_number  $\gamma z \in \text{Ints}$ 
  by (simp add:  $\gamma$  integer_winding_number loop valid_path_imp_path z)
  then show ?thesis
  using 0 2 by (auto simp: Ints_def)
qed

```

3.3 Continuity of winding number and invariance on connected sets

```

theorem continuous_at_winding_number:
  fixes z::complex
  assumes  $\gamma$ : path  $\gamma$  and z:  $z \notin \text{path\_image } \gamma$ 
  shows continuous (at z) (winding_number  $\gamma$ )
proof -
  obtain e where e>0 and cbg: cball z e  $\subseteq$  - path_image  $\gamma$ 
  using open_contains_cball [of - path_image  $\gamma$ ] z
  by (force simp: closed_def [symmetric] closed_path_image [OF  $\gamma$ ])
  then have ppag: path_image  $\gamma \subseteq$  - cball z (e/2)
  by (force simp: cball_def dist_norm)
  have oc: open (- cball z (e/2))
  by (simp add: closed_def [symmetric])
  obtain d where d>0 and pi_eq:
     $\bigwedge h1 h2. \llbracket \text{valid\_path } h1; \text{valid\_path } h2; \\
    (\forall t \in \{0..1\}. \text{cmod } (h1 t - \gamma t) < d \wedge \text{cmod } (h2 t - \gamma t) < d); \\
    \text{pathstart } h2 = \text{pathstart } h1; \text{pathfinish } h2 = \text{pathfinish } h1 \rrbracket \\
    \implies \\
    \text{path\_image } h1 \subseteq - \text{cball } z (e/2) \wedge \\
    \text{path\_image } h2 \subseteq - \text{cball } z (e/2) \wedge \\
    (\forall f. f \text{ holomorphic\_on } - \text{cball } z (e/2) \longrightarrow \text{contour\_integral } h2 f = \\
    \text{contour\_integral } h1 f)$ 
  using contour_integral_nearby_ends [OF oc  $\gamma$  ppag] by metis
  obtain p where valid_path p  $z \notin \text{path\_image } p$ 

```

```

    and p: pathstart p = pathstart  $\gamma$  pathfinish p = pathfinish  $\gamma$ 
    and pg:  $\bigwedge t. t \in \{0..1\} \implies \text{cmod } (\gamma t - p t) < \min d \ e/2$ 
    and pi:  $\text{contour\_integral } p (\lambda x. 1 / (x - z)) = \text{complex\_of\_real } (2 * \pi i) * i * \text{winding\_number } \gamma z$ 
    using winding_number [OF  $\gamma z$ , of  $\min d \ e/2$ ]  $\langle d > 0 \rangle \langle e > 0 \rangle$  by (auto simp:
winding_number_prop_def)
  { fix w
    assume d2:  $\text{cmod } (w - z) < d/2$  and e2:  $\text{cmod } (w - z) < e/2$ 
    have wnotp:  $w \notin \text{path\_image } p$ 
    proof (clarsimp simp add: path_image_def)
      show False if w:  $w = p x$  and  $0 \leq x \leq 1$  for x
      proof -
        have cmod  $(\gamma x - p x) < \min d \ e/2$ 
        using pg that by auto
        then have  $\text{cmod } (z - \gamma x) < e$ 
        by (metis e2 less_divide_eq_numeral1(1) min_less_iff_conj norm_minus_commute
norm_triangle_half_l w)
        then show ?thesis
        using cbg that by (auto simp add: path_image_def cball_def dist_norm
less_eq_real_def)
      qed
    qed
    have wnotg:  $w \notin \text{path\_image } \gamma$ 
    using cbg e2  $\langle e > 0 \rangle$  by (force simp: dist_norm norm_minus_commute)
    { fix k::real
      assume k:  $k > 0$ 
      then obtain q where q: valid_path q  $w \notin \text{path\_image } q$ 
        pathstart q = pathstart  $\gamma \wedge$  pathfinish q = pathfinish  $\gamma$ 
        and qg:  $\bigwedge t. t \in \{0..1\} \implies \text{cmod } (\gamma t - q t) < \min k (\min d \ e)$ 
      / 2
        and qi:  $\text{contour\_integral } q (\lambda u. 1 / (u - w)) = \text{complex\_of\_real } (2 * \pi i) * i * \text{winding\_number } \gamma w$ 
      using winding_number [OF  $\gamma$  wnotg, of  $\min k (\min d \ e) / 2$ ]  $\langle d > 0 \rangle \langle e > 0 \rangle$ 
      k
        by (force simp: min_divide_distrib_right winding_number_prop_def)
      moreover have  $\bigwedge t. t \in \{0..1\} \implies \text{cmod } (q t - \gamma t) < d \wedge \text{cmod } (p t - \gamma t) < d$ 
      using pg qg  $\langle 0 < d \rangle$  by (fastforce simp add: norm_minus_commute)
      moreover have  $(\lambda u. 1 / (u - w)) \text{ holomorphic\_on } - \text{cball } z (e/2)$ 
      using e2 by (auto simp: dist_norm norm_minus_commute intro!: holomorphic_intros)
      ultimately have  $\text{contour\_integral } p (\lambda u. 1 / (u - w)) = \text{contour\_integral } q (\lambda u. 1 / (u - w))$ 
      by (metis p <valid_path p> pi_eq)
      then have  $\text{contour\_integral } p (\lambda x. 1 / (x - w)) = \text{complex\_of\_real } (2 * \pi i) * i * \text{winding\_number } \gamma w$ 
      by (simp add: pi qi)
    } note pip = this
    have path p

```

```

    by (simp add: ⟨valid_path p⟩ valid_path_imp_path)
  moreover have  $\bigwedge e. e > 0 \implies \text{winding\_number\_prop } p \ w \ e \ p \ (\text{winding\_number } \gamma \ w)$ 
  by (simp add: ⟨valid_path p⟩ pip_winding_number_prop_def wnotp)
  ultimately have  $\text{winding\_number } p \ w = \text{winding\_number } \gamma \ w$ 
  using winding_number_unique wnotp by blast
} note wnw = this
obtain pe where  $pe > 0$  and cbp:  $\text{cball } z \ (3 / 4 * pe) \subseteq - \text{path\_image } p$ 
using ⟨valid_path p⟩ ⟨ $z \notin \text{path\_image } p$ ⟩ open_contains_cball [of - path_image p]
by (force simp: closed_def [symmetric] closed_path_image [OF valid_path_imp_path])
obtain L
where  $L > 0$ 
and L:  $\bigwedge f B. \llbracket f \text{ holomorphic\_on } - \text{cball } z \ (3 / 4 * pe);$ 
 $\forall z \in - \text{cball } z \ (3 / 4 * pe). \text{cmod } (f \ z) \leq B \rrbracket \implies$ 
 $\text{cmod } (\text{contour\_integral } p \ f) \leq L * B$ 
using contour_integral_bound_exists [of - cball z (3/4*pe) p] cbp ⟨valid_path p⟩ by blast
{ fix e::real and w::complex
  assume e:  $0 < e$  and w:  $\text{cmod } (w - z) < pe/4$   $\text{cmod } (w - z) < e * pe^2 / (8 * L)$ 
  then have [simp]:  $w \notin \text{path\_image } p$ 
  using cbp p(2) ⟨ $0 < pe$ ⟩
  by (force simp: dist_norm norm_minus_commute path_image_def cball_def)
  have [simp]:  $\text{contour\_integral } p \ (\lambda x. 1/(x - w)) - \text{contour\_integral } p \ (\lambda x. 1/(x - z)) =$ 
 $\text{contour\_integral } p \ (\lambda x. 1/(x - w) - 1/(x - z))$ 
  by (simp add: ⟨valid_path p⟩ ⟨ $z \notin \text{path\_image } p$ ⟩ contour_integrable_inversediff contour_integral_diff)
  { fix x
    assume pe:  $3/4 * pe < \text{cmod } (z - x)$ 
    have  $\text{cmod } (w - x) < pe/4 + \text{cmod } (z - x)$ 
    by (meson add_less_cancel_right norm_diff_triangle_le order_refl order_trans_rules(21) w(1))
    then have wx:  $\text{cmod } (w - x) < 4/3 * \text{cmod } (z - x)$  using pe by simp
    have  $\text{cmod } (z - x) \leq \text{cmod } (z - w) + \text{cmod } (w - x)$ 
    using norm_diff_triangle_le by blast
    also have  $\dots < pe/4 + \text{cmod } (w - x)$ 
    using w by (simp add: norm_minus_commute)
    finally have  $pe/2 < \text{cmod } (w - x)$ 
    using pe by auto
    then have pe_less:  $(pe/2)^2 < \text{cmod } (w - x)^2$ 
    by (simp add: ⟨ $0 < pe$ ⟩ less_eq_real_def power_strict_mono)
    then have pe2:  $pe^2 < 4 * \text{cmod } (w - x)^2$ 
    by (simp add: power_divide)
    have  $8 * L * \text{cmod } (w - z) < e * pe^2$ 
    using w ⟨ $L > 0$ ⟩ by (simp add: field_simps)
    also have  $\dots < e * 4 * \text{cmod } (w - x) * \text{cmod } (w - x)$ 
    using pe2 ⟨ $e > 0$ ⟩ by (simp add: power2_eq_square)
  }

```

```

also have ... < e * 4 * cmod (w - x) * (4/3 * cmod (z - x))
  using <0 < pe> pe_less e less_eq_real_def wx by fastforce
finally have L * cmod (w - z) < 2/3 * e * cmod (w - x) * cmod (z - x)
  by simp
also have ... ≤ e * cmod (w - x) * cmod (z - x)
  using e by simp
finally have Lwz: L * cmod (w - z) < e * cmod (w - x) * cmod (z - x) .
have L * cmod (1 / (x - w) - 1 / (x - z)) ≤ e
proof (cases x=z ∨ x=w)
  case True
  with pe <pe>0> w <L>0>
  show ?thesis
    by (force simp: norm_minus_commute)
next
  case False
  with wx w(2) <L>0> pe pe2 Lwz
  show ?thesis
    by (auto simp: divide_simps mult_less_0_iff norm_minus_commute
norm_divide norm_mult power2_eq_square)
  qed
} note L_cmod_le = this
let ?f = (λx. 1 / (x - w) - 1 / (x - z))
have cmod (contour_integral p ?f) ≤ L * (e * pe2 / L / 4 * (inverse (pe /
2))2)
proof (rule L)
  show ?f holomorphic_on - cball z (3 / 4 * pe)
    using <pe>0> w
  by (force simp: dist_norm norm_minus_commute intro!: holomorphic_intros)
  show ∀ u ∈ - cball z (3 / 4 * pe). cmod (?f u) ≤ e * pe2 / L / 4 * (inverse
(pe / 2))2
    using <pe>0> w <L>0>
    by (auto simp: cball_def dist_norm field_simps L_cmod_le simp del:
less_divide_eq_numeral1 le_divide_eq_numeral1)
  qed
also have ... < 2*e
  using <L>0> e by (force simp: field_simps)
finally have cmod (winding_number p w - winding_number p z) < e
  using pi_ge_two e
  by (force simp: winding_number_valid_path <valid_path p> <z ∉ path_image
p> field_simps norm_divide norm_mult intro: less_le_trans)
} note cmod_wn_diff = this
have isCont (winding_number p) z
proof (clarsimp simp add: continuous_at_eps_delta)
  fix e::real assume e>0
  show ∃ d>0. ∀ x'. dist x' z < d ⟶ dist (winding_number p x') (winding_number
p z) < e
    using <pe>0> <L>0> <e>0>
    by (rule_tac x=min (pe/4) (e/2*pe2/L/4) in exI) (simp add: dist_norm
cmod_wn_diff)

```

```

qed
then show ?thesis
  apply (rule continuous_transform_within [where  $\delta = \min d \ e/2$ ])
  apply (auto simp:  $\langle d > 0 \rangle \langle e > 0 \rangle \text{dist\_norm wnw}$ )
  done
qed

corollary continuous_on_winding_number:
  path  $\gamma \implies \text{continuous\_on } (- \text{path\_image } \gamma) (\lambda w. \text{winding\_number } \gamma \ w)$ 
  by (simp add: continuous_at_imp_continuous_on continuous_at_winding_number)

```

3.4 The winding number is constant on a connected region

```

lemma winding_number_constant:
  assumes  $\gamma$ : path  $\gamma$  and loop: pathfinish  $\gamma = \text{pathstart } \gamma$  and cs: connected  $S$ 
  and sg:  $S \cap \text{path\_image } \gamma = \{\}$ 
  shows winding_number  $\gamma$  constant_on  $S$ 
proof -
  have *:  $1 \leq \text{cmod } (\text{winding\_number } \gamma \ y - \text{winding\_number } \gamma \ z)$ 
    if ne: winding_number  $\gamma \ y \neq \text{winding\_number } \gamma \ z$  and  $y \in S \ z \in S$  for  $y \ z$ 
  proof -
    have winding_number  $\gamma \ y \in \mathbb{Z}$  winding_number  $\gamma \ z \in \mathbb{Z}$ 
      using that integer_winding_number [OF  $\gamma$  loop] sg  $\langle y \in S \rangle$  by auto
    with ne show ?thesis
      by (auto simp: Ints_def simp flip: of_int_diff)
  qed
  have cont: continuous_on  $S$  ( $\lambda w. \text{winding\_number } \gamma \ w$ )
    using continuous_on_winding_number [OF  $\gamma$ ] sg
  by (meson continuous_on_subset disjoint_eq_subset_Compl)
  show ?thesis
    using * zero_less_one
    by (blast intro: continuous_discrete_range_constant [OF cs cont])
qed

```

```

lemma winding_number_eq:
   $\llbracket \text{path } \gamma; \text{pathfinish } \gamma = \text{pathstart } \gamma; w \in S; z \in S; \text{connected } S; S \cap \text{path\_image } \gamma = \{\} \rrbracket$ 
 $\implies \text{winding\_number } \gamma \ w = \text{winding\_number } \gamma \ z$ 
  using winding_number_constant by (metis constant_on_def)

```

```

lemma open_winding_number_levelsets:
  assumes  $\gamma$ : path  $\gamma$  and loop: pathfinish  $\gamma = \text{pathstart } \gamma$ 
  shows open  $\{z. z \notin \text{path\_image } \gamma \wedge \text{winding\_number } \gamma \ z = k\}$ 
proof (clarsimp simp: open_dist)
  fix  $z$  assume  $z: z \notin \text{path\_image } \gamma$  and  $k: k = \text{winding\_number } \gamma \ z$ 
  have open  $(- \text{path\_image } \gamma)$ 
    by (simp add: closed_path_image  $\gamma$  open_Compl)
  then obtain  $e$  where  $e > 0$  ball  $z \ e \subseteq - \text{path\_image } \gamma$ 
    using open_contains_ball [of  $- \text{path\_image } \gamma$ ]  $z$  by blast

```

```

then show  $\exists e > 0. \forall y. \text{dist } y \ z < e \longrightarrow y \notin \text{path\_image } \gamma \wedge \text{winding\_number}$ 
 $\gamma \ y = \text{winding\_number } \gamma \ z$ 
using  $\langle e > 0 \rangle$  by (force simp: norm_minus_commute dist_norm intro: winding_number_eq [OF assms, where  $S = \text{ball } z \ e$ ])
qed

```

3.5 Winding number is zero "outside" a curve

proposition *winding_number_zero_in_outside:*

assumes γ : path γ **and** loop: pathfinish $\gamma = \text{pathstart } \gamma$ **and** z : $z \in \text{outside}$ (path_image γ)

shows $\text{winding_number } \gamma \ z = 0$

proof –

obtain $B::\text{real}$ **where** $0 < B$ **and** B : path_image $\gamma \subseteq \text{ball } 0 \ B$

using bounded_subset_ballD [OF bounded_path_image [OF γ]] **by** auto

obtain $w::\text{complex}$ **where** w : $w \notin \text{ball } 0 \ (B + 1)$

by (metis abs_of_nonneg le_less less_irrefl mem_ball_0 norm_of_real)

have $-\text{ball } 0 \ (B + 1) \subseteq \text{outside} \ (\text{path_image } \gamma)$

using B subset_ball **by** (intro outside_subset_convex) auto

then have $w \text{out}$: $w \in \text{outside} \ (\text{path_image } \gamma)$

using w **by** blast

moreover have $\text{winding_number } \gamma \ \text{constant_on } \text{outside} \ (\text{path_image } \gamma)$

using winding_number_constant [OF γ loop, of outside(path_image γ)] connected_outside

by (metis DIM_complex bounded_path_image dual_order.refl γ outside_no_overlap)

ultimately have $\text{winding_number } \gamma \ z = \text{winding_number } \gamma \ w$

by (metis (no_types, opaque_lifting) constant_on_def z)

also have $\dots = 0$

proof –

have $w \text{not}$: $w \notin \text{path_image } \gamma$ **using** $w \text{out}$ **by** (simp add: outside_def)

{ fix $e::\text{real}$ **assume** $0 < e$

obtain p **where** p : polynomial_function p pathstart $p = \text{pathstart } \gamma$ pathfinish $p = \text{pathfinish } \gamma$

and $pg1$: $(\bigwedge t. \llbracket 0 \leq t; t \leq 1 \rrbracket \Longrightarrow \text{cmod } (p \ t - \gamma \ t) < 1)$

and pge : $(\bigwedge t. \llbracket 0 \leq t; t \leq 1 \rrbracket \Longrightarrow \text{cmod } (p \ t - \gamma \ t) < e)$

using path_approx_polynomial_function [OF γ , of min 1 e] $\langle e > 0 \rangle$

by (metis atLeastAtMost_iff min_less_iff_conj zero_less_one)

have $\exists p. \text{valid_path } p \wedge w \notin \text{path_image } p \wedge$

$\text{pathstart } p = \text{pathstart } \gamma \wedge \text{pathfinish } p = \text{pathfinish } \gamma \wedge$

$(\forall t \in \{0..1\}. \text{cmod } (\gamma \ t - p \ t) < e) \wedge \text{contour_integral } p \ (\lambda wa. 1$

$/ (wa - w)) = 0$

proof (intro exI conjI)

have $\bigwedge x. \llbracket 0 \leq x; x \leq 1 \rrbracket \Longrightarrow \text{cmod } (p \ x) < B + 1$

using B unfolding image_subset_iff path_image_def

by (meson add_strict_mono atLeastAtMost_iff le_less_trans mem_ball_0 norm_triangle_sub $pg1$)

then have pip : path_image $p \subseteq \text{ball } 0 \ (B + 1)$

by (auto simp add: path_image_def dist_norm ball_def)

then show $w \notin \text{path_image } p$ **using** w **by** blast

```

    show vap: valid_path p
      by (simp add: p(1) valid_path_polynomial_function)
    show  $\forall t \in \{0..1\}. \text{cmod } (\gamma t - p t) < e$ 
      by (metis atLeastAtMost_iff norm_minus_commute pge)
    show contour_integral p ( $\lambda w a. 1 / (w a - w)$ ) = 0
    proof (rule contour_integral_unique [OF Cauchy_theorem_convex_simple
[OF _ convex_ball [of 0 B+1]]])
      have  $\bigwedge z. \text{cmod } z < B + 1 \implies z \neq w$ 
        using mem_ball_0 w by blast
      then show ( $\lambda z. 1 / (z - w)$ ) holomorphic_on ball 0 (B + 1)
        by (intro holomorphic_intros; simp add: dist_norm)
      qed (use p vap pip loop in auto)
    qed (use p in auto)
  }
  then show ?thesis
    by (auto intro: winding_number_unique [OF  $\gamma$ ] simp add: winding_number_prop_def
wnot)
  qed
  finally show ?thesis .
qed

```

corollary winding_number_zero_const: $a \neq z \implies \text{winding_number } (\lambda t. a) z = 0$

```

  by (rule winding_number_zero_in_outside)
    (auto simp: pathfinish_def pathstart_def path_polynomial_function)

```

corollary winding_number_zero_outside:

```

 $\llbracket \text{path } \gamma; \text{convex } s; \text{pathfinish } \gamma = \text{pathstart } \gamma; z \notin s; \text{path\_image } \gamma \subseteq s \rrbracket \implies$ 
winding_number  $\gamma z = 0$ 
  by (meson convex_in_outside outside_mono subsetCE winding_number_zero_in_outside)

```

lemma winding_number_zero_at_infinity:

```

  assumes  $\gamma: \text{path } \gamma$  and loop:  $\text{pathfinish } \gamma = \text{pathstart } \gamma$ 
  shows  $\exists B. \forall z. B \leq \text{norm } z \longrightarrow \text{winding\_number } \gamma z = 0$ 
proof -
  obtain B::real where  $0 < B$  and B:  $\text{path\_image } \gamma \subseteq \text{ball } 0 B$ 
    using bounded_subset_ballD [OF bounded_path_image [OF  $\gamma$ ]] by auto
  have winding_number  $\gamma z = 0$  if  $B + 1 \leq \text{cmod } z$  for  $z$ 
  proof (rule winding_number_zero_outside [OF  $\gamma$  convex_cball loop])
    show  $z \notin \text{cball } 0 B$ 
      using that by auto
    show  $\text{path\_image } \gamma \subseteq \text{cball } 0 B$ 
      using B order.trans by blast
  qed
  then show ?thesis
    by metis
qed

```

lemma bounded_winding_number_nz:

```

assumes path g pathfinish g = pathstart g
shows   bounded {z. winding_number g z  $\neq$  0}
proof -
  obtain B where  $\bigwedge x. \text{norm } x \geq B \implies \text{winding\_number } g \ x = 0$ 
  using winding_number_zero_at_infinity[OF assms] by auto
  thus ?thesis
  unfolding bounded_iff by (intro exI[of _ B + 1]) force
qed

```

```

lemma winding_number_zero_point:
   $\llbracket \text{path } \gamma; \text{convex } S; \text{pathfinish } \gamma = \text{pathstart } \gamma; \text{open } S; \text{path\_image } \gamma \subseteq S \rrbracket$ 
   $\implies \exists z. z \in S \wedge \text{winding\_number } \gamma \ z = 0$ 
  using outside_compact_in_open [of path_image  $\gamma$  S] path_image_nonempty
  winding_number_zero_in_outside
  by (fastforce simp add: compact_path_image)

```

If a path winds round a set, it winds rounds its inside.

```

lemma winding_number_around_inside:
  assumes  $\gamma$ : path  $\gamma$  and loop: pathfinish  $\gamma$  = pathstart  $\gamma$ 
  and cls: closed S and cos: connected S and S_disj:  $S \cap \text{path\_image } \gamma = \{\}$ 
  and z:  $z \in S$  and wn_nz: winding_number  $\gamma$  z  $\neq$  0 and w:  $w \in S \cup \text{inside } S$ 
  shows winding_number  $\gamma$  w = winding_number  $\gamma$  z
proof -
  have ssb:  $S \subseteq \text{inside}(\text{path\_image } \gamma)$ 
  proof
    fix x :: complex
    assume x  $\in$  S
    hence x  $\notin$  path_image  $\gamma$ 
    by (meson disjoint_iff_not_equal S_disj)
    thus x  $\in$  inside (path_image  $\gamma$ )
    by (metis Compl_iff S_disj UnE  $\gamma$   $\langle x \in S \rangle$  cos inside_outside loop winding_number_eq winding_number_zero_in_outside wn_nz z)
  qed
  show ?thesis
proof (rule winding_number_eq [OF  $\gamma$  loop w])
  show z  $\in$  S  $\cup$  inside S
  using z by blast
  show connected (S  $\cup$  inside S)
  by (simp add: cls connected_with_inside cos)
  show (S  $\cup$  inside S)  $\cap$  path_image  $\gamma$  =  $\{\}$ 
  unfolding disjoint_iff Un_iff
  by (metis ComplD UnI1  $\gamma$  cls compact_path_image connected_path_image inside_Un_outside inside_inside_compact_connected ssb subsetD)
qed
qed

```

Bounding a WN by 1/2 for a path and point in opposite halfspaces.

```

lemma winding_number_subpath_continuous:

```



```

assumes  $\gamma$ : valid_path  $\gamma$  and  $z$ :  $z \notin \text{path\_image } \gamma$ 
shows continuous_on  $\{0..1\}$  ( $\lambda x. \text{winding\_number}(\text{subpath } 0 \ x \ \gamma) \ z$ )
proof (rule continuous_on_eq)
  let  $?f = \lambda x. \text{integral } \{0..x\} (\lambda t. \text{vector\_derivative } \gamma \ (\text{at } t) / (\gamma \ t - z))$ 
show continuous_on  $\{0..1\}$  ( $\lambda x. 1 / (2 * \pi * i) * ?f \ x$ )
  proof (intro indefinite_integral_continuous_1 winding_number_exp_integral
    continuous_intros)
    show  $\gamma$  piecewise_C1_differentiable_on  $\{0..1\}$ 
    using  $\gamma$  valid_path_def by blast
  qed (use path_image_def  $z$  in auto)
show  $1 / (2 * \pi * i) * ?f \ x = \text{winding\_number}(\text{subpath } 0 \ x \ \gamma) \ z$ 
  if  $x \in \{0..1\}$  for  $x$ 
proof -
  have  $1 / (2 * \pi * i) * ?f \ x = 1 / (2 * \pi * i) * \text{contour\_integral}(\text{subpath } 0 \ x \ \gamma)$ 
  ( $\lambda w. 1 / (w - z)$ )
  using assms  $x$ 
  by (simp add: contour_integral_subcontour_integral [OF contour_integrable_inversediff])
  also have  $\dots = \text{winding\_number}(\text{subpath } 0 \ x \ \gamma) \ z$ 
proof (subst winding_number_valid_path)
  show  $z \notin \text{path\_image}(\text{subpath } 0 \ x \ \gamma)$ 
  using assms  $x$  atLeastAtMost_iff path_image_subpath_subset by force
qed (use assms  $x$  valid_path_subpath in <force+>)
finally show ?thesis .
qed
qed

```

lemma winding_number_ivt_pos:

```

assumes  $\gamma$ : valid_path  $\gamma$  and  $z$ :  $z \notin \text{path\_image } \gamma$  and  $0 \leq w \wedge w \leq \text{Re}(\text{winding\_number}$ 
 $\gamma \ z)$ 
shows  $\exists t \in \{0..1\}. \text{Re}(\text{winding\_number}(\text{subpath } 0 \ t \ \gamma) \ z) = w$ 
proof -
  have continuous_on  $\{0..1\}$  ( $\lambda x. \text{winding\_number}(\text{subpath } 0 \ x \ \gamma) \ z$ )
  using  $\gamma$  winding_number_subpath_continuous  $z$  by blast
  moreover have  $\text{Re}(\text{winding\_number}(\text{subpath } 0 \ 0 \ \gamma) \ z) \leq w \wedge w \leq \text{Re}(\text{winding\_number}$ 
  ( $\text{subpath } 0 \ 1 \ \gamma) \ z)$ 
  using assms by (auto simp: path_image_def image_def)
  ultimately show ?thesis
  using ivt_increasing_component_on_1 [of 0 1, where  $?k = 1$ ] by force
qed

```

lemma winding_number_ivt_neg:

```

assumes  $\gamma$ : valid_path  $\gamma$  and  $z$ :  $z \notin \text{path\_image } \gamma$  and  $\text{Re}(\text{winding\_number}$ 
 $\gamma \ z) \leq w \wedge w \leq 0$ 
shows  $\exists t \in \{0..1\}. \text{Re}(\text{winding\_number}(\text{subpath } 0 \ t \ \gamma) \ z) = w$ 
proof -
  have continuous_on  $\{0..1\}$  ( $\lambda x. \text{winding\_number}(\text{subpath } 0 \ x \ \gamma) \ z$ )
  using  $\gamma$  winding_number_subpath_continuous  $z$  by blast
  moreover have  $\text{Re}(\text{winding\_number}(\text{subpath } 0 \ 0 \ \gamma) \ z) \geq w \wedge w \geq \text{Re}(\text{winding\_number}$ 
  ( $\text{subpath } 0 \ 1 \ \gamma) \ z)$ 

```

```

    using assms by (auto simp: path_image_def image_def)
  ultimately show ?thesis
    using ivt_decreasing_component_on_1 [of 0 1, where ?k = 1] by force
qed

```

```

lemma winding_number_ivt_abs:
  assumes  $\gamma$ : valid_path  $\gamma$  and  $z$ :  $z \notin \text{path\_image } \gamma$  and  $0 \leq w$   $w \leq |\text{Re}(\text{winding\_number } \gamma \ z)|$ 
  shows  $\exists t \in \{0..1\}. |\text{Re}(\text{winding\_number } (\text{subpath } 0 \ t \ \gamma) \ z)| = w$ 
  using assms winding_number_ivt_pos [of  $\gamma \ z \ w$ ] winding_number_ivt_neg [of  $\gamma \ z \ -w$ ]
  by force

```

```

lemma winding_number_lt_half_lemma:
  assumes  $\gamma$ : valid_path  $\gamma$  and  $z$ :  $z \notin \text{path\_image } \gamma$  and  $az$ :  $a \cdot z \leq b$  and  $pag$ :
     $\text{path\_image } \gamma \subseteq \{w. a \cdot w > b\}$ 
  shows  $\text{Re}(\text{winding\_number } \gamma \ z) < 1/2$ 
proof -
  { assume  $\text{Re}(\text{winding\_number } \gamma \ z) \geq 1/2$ 
    then obtain  $t::\text{real}$  where  $t$ :  $0 \leq t \ t \leq 1$  and  $sub12$ :  $\text{Re}(\text{winding\_number } (\text{subpath } 0 \ t \ \gamma) \ z) = 1/2$ 
    using winding_number_ivt_pos [OF  $\gamma \ z$ , of  $1/2$ ] by auto
    have  $gt$ :  $\gamma \ t - z = -(\text{of\_real } (\exp(-(2 * \pi * \text{Im}(\text{winding\_number } (\text{subpath } 0 \ t \ \gamma) \ z)))) * (\gamma \ 0 - z))$ 
    using winding_number_exp_2pi [of  $\text{subpath } 0 \ t \ \gamma \ z$ ]
    apply (simp add:  $t \ \gamma$  valid_path_imp_path)
    using closed_segment_eq_real_ivl path_image_def  $t \ z$  by (fastforce simp: path_image_subpath Euler sub12)
    have  $b < a \cdot \gamma \ 0$ 
    proof -
      have  $\gamma \ 0 \in \{c. b < a \cdot c\}$ 
      by (metis (no_types)  $pag$  atLeastAtMost_iff image_subset_iff order_refl path_image_def zero_le_one)
      thus ?thesis
      by blast
    qed
    moreover have  $b < a \cdot \gamma \ t$ 
    by (metis atLeastAtMost_iff image_eqI mem_Collect_eq  $pag$  path_image_def subset_iff  $t$ )
    ultimately have  $0 < a \cdot (\gamma \ 0 - z) \ 0 < a \cdot (\gamma \ t - z)$  using  $az$ 
    by (simp add: inner_diff_right)+
    then have False
    by (simp add:  $gt$  inner_mult_right mult_less_0_iff)
  }
  then show ?thesis by force
qed

```

```

lemma winding_number_lt_half:
  assumes valid_path  $\gamma$   $a \cdot z \leq b$   $\text{path\_image } \gamma \subseteq \{w. a \cdot w > b\}$ 

```

```

    shows  $|Re (winding\_number \ \gamma \ z)| < 1/2$ 
  proof -
    have  $z \notin path\_image \ \gamma$  using assms by auto
    with assms have  $Re (winding\_number \ \gamma \ z) < 0 \implies - Re (winding\_number \ \gamma \ z) < 1/2$ 
    by (metis complex_inner_1_right winding_number_lt_half_lemma [OF valid_path_imp_reverse, of  $\gamma \ z \ a \ b$ ])
    winding_number_reversepath valid_path_imp_path inner_minus_left path_image_reversepath
    with assms show ?thesis
    using  $\langle z \notin path\_image \ \gamma \rangle$  winding_number_lt_half_lemma by fastforce
  qed

```

lemma *winding_number_le_half*:

```

  assumes  $\gamma$ : valid_path  $\gamma$  and  $z$ :  $z \notin path\_image \ \gamma$ 
    and anz:  $a \neq 0$  and azb:  $a \cdot z \leq b$  and pag:  $path\_image \ \gamma \subseteq \{w. a \cdot w \geq b\}$ 
  shows  $|Re (winding\_number \ \gamma \ z)| \leq 1/2$ 
  proof -
    { assume wnz_12:  $|Re (winding\_number \ \gamma \ z)| > 1/2$ 
      have isCont (winding_number  $\gamma$ )  $z$ 
        by (metis continuous_at_winding_number valid_path_imp_path  $\gamma \ z$ )
      then obtain d where  $d > 0$  and d:  $\bigwedge x'. dist \ x' \ z < d \implies dist (winding\_number \ \gamma \ x') (winding\_number \ \gamma \ z) < |Re (winding\_number \ \gamma \ z)| - 1/2$ 
        using continuous_at_eps_delta wnz_12 diff_gt_0_iff_gt by blast
      define  $z'$  where  $z' = z - (d / (2 * cmod \ a)) *_R \ a$ 
      have  $a \cdot z * 6 \leq d * cmod \ a + b * 6$ 
        by (metis  $\langle 0 < d \rangle$  add_increasing azb less_eq_real_def mult_nonneg_nonneg mult_right_mono norm_ge_zero norm_numeral)
      with anz have  $a \cdot z' \leq b - d / 3 * cmod \ a$ 
        unfolding  $z'_def$  inner_mult_right' divide_inverse
      by (simp add: field_split_simps algebra_simps dot_square_norm power2_eq_square)
      have  $cmod (winding\_number \ \gamma \ z') - winding\_number \ \gamma \ z < |Re (winding\_number \ \gamma \ z)| - 1/2$ 
        using d [of  $z'$ ] anz  $\langle d > 0 \rangle$  by (simp add: dist_norm  $z'_def$ )
      then have  $1/2 < |Re (winding\_number \ \gamma \ z)| - cmod (winding\_number \ \gamma \ z') - winding\_number \ \gamma \ z$ 
        by simp
      then have  $1/2 < |Re (winding\_number \ \gamma \ z)| - |Re (winding\_number \ \gamma \ z') - Re (winding\_number \ \gamma \ z)|$ 
        using abs_Re_le_cmod [of  $winding\_number \ \gamma \ z' - winding\_number \ \gamma \ z$ ]
      by simp
    }
    then have wnz_12':  $|Re (winding\_number \ \gamma \ z')| > 1/2$ 
      by linarith
    moreover have  $|Re (winding\_number \ \gamma \ z')| < 1/2$ 
      proof (rule winding_number_lt_half [OF  $\gamma \ *$ ])
        show  $path\_image \ \gamma \subseteq \{w. b - d / 3 * cmod \ a < a \cdot w\}$ 
          using azb  $\langle d > 0 \rangle$  pag by (auto simp: add_strict_increasing anz field_split_simps dest!: subsetD)
      qed
    ultimately have False

```

```

    by simp
  }
  then show ?thesis by force
qed

lemma winding_number_lt_half_linepath:
  assumes  $z \notin \text{closed\_segment } a \ b$  shows  $|\text{Re } (\text{winding\_number } (\text{linepath } a \ b) \ z)| < 1/2$ 
proof -
  obtain  $u \ v$  where  $u \cdot z \leq v$  and  $uv: \bigwedge x. x \in \text{closed\_segment } a \ b \implies \text{inner } u \ x > v$ 
  using separating_hyperplane_closed_point assms closed_segment convex_closed_segment less_eq_real_def by metis
  moreover have  $\text{path\_image } (\text{linepath } a \ b) \subseteq \{w. v < u \cdot w\}$ 
  using in_segment(1) uv by auto
  ultimately show ?thesis
  using winding_number_lt_half by auto
qed

```

Positivity of WN for a linepath.

```

lemma winding_number_linepath_pos_lt:
  assumes  $0 < \text{Im } ((b - a) * \text{cnj } (b - z))$ 
  shows  $0 < \text{Re}(\text{winding\_number}(\text{linepath } a \ b) \ z)$ 
proof -
  have  $z: z \notin \text{path\_image } (\text{linepath } a \ b)$ 
  using assms
  by (simp add: closed_segment_def) (force simp: algebra_simps)
  show ?thesis
  by (intro winding_number_pos_lt [OF valid_path_linepath z assms]) (simp add: linepath_def algebra_simps)
qed

```

```

lemma winding_number_linepath_neg_lt:
  assumes  $0 < \text{Im } ((a - b) * \text{cnj } (a - z))$ 
  shows  $\text{Re}(\text{winding\_number}(\text{linepath } a \ b) \ z) < 0$ 
proof -
  have  $z: z \notin \text{path\_image } (\text{linepath } a \ b)$ 
  using assms
  by (simp add: closed_segment_def) (force simp: algebra_simps)
  have  $\text{Re}(\text{winding\_number}(\text{linepath } a \ b) \ z) =$ 
     $-\text{Re}(\text{winding\_number}(\text{reversepath } (\text{linepath } a \ b)) \ z)$ 
  by (subst winding_number_reversepath) (use z in auto)
  also have  $\dots = -\text{Re}(\text{winding\_number}(\text{linepath } b \ a) \ z)$ 
  by simp
  finally have  $\text{Re } (\text{winding\_number } (\text{linepath } a \ b) \ z)$ 
     $= -\text{Re } (\text{winding\_number } (\text{linepath } b \ a) \ z) .$ 
  moreover have  $0 < \text{Re } (\text{winding\_number } (\text{linepath } b \ a) \ z)$ 
  using winding_number_linepath_pos_lt[OF assms] .
  ultimately show ?thesis by auto

```

qed

3.6 More winding number properties

including the fact that it's ± 1 inside a simple closed curve.

lemma *winding_number_homotopic_paths*:

assumes *homotopic_paths* $(-\{z\})$ *g h*

shows *winding_number* *g z* = *winding_number* *h z*

proof –

have *path g path h* **using** *homotopic_paths_imp_path* [*OF* *assms*] **by** *auto*

moreover have *pag*: $z \notin \text{path_image } g$ **and** *pah*: $z \notin \text{path_image } h$

using *homotopic_paths_imp_subset* [*OF* *assms*] **by** *auto*

ultimately obtain *d e* **where** $d > 0$ $e > 0$

and *d*: $\bigwedge p. \llbracket \text{path } p; \text{pathstart } p = \text{pathstart } g; \text{pathfinish } p = \text{pathfinish } g; \forall t \in \{0..1\}. \text{norm } (p \ t - g \ t) < d \rrbracket$

$\implies \text{homotopic_paths } (-\{z\}) \ g \ p$

and *e*: $\bigwedge q. \llbracket \text{path } q; \text{pathstart } q = \text{pathstart } h; \text{pathfinish } q = \text{pathfinish } h; \forall t \in \{0..1\}. \text{norm } (q \ t - h \ t) < e \rrbracket$

$\implies \text{homotopic_paths } (-\{z\}) \ h \ q$

using *homotopic_nearby_paths* [*of g* $-\{z\}$] *homotopic_nearby_paths* [*of h* $-\{z\}$] **by** *force*

obtain *p* **where** *p*:

valid_path p z $z \notin \text{path_image } p$

pathstart p = *pathstart g* *pathfinish p* = *pathfinish g*

and *gp_less*: $\forall t \in \{0..1\}. \text{cmod } (g \ t - p \ t) < d$

and *pap*: $\text{contour_integral } p \ (\lambda w. 1 / (w - z)) = \text{complex_of_real } (2 * \pi i)$

* i * *winding_number g z*

using *winding_number* [*OF* $\langle \text{path } g \rangle \text{ pag } \langle 0 < d \rangle$] **unfolding** *winding_number_prop_def* **by** *blast*

obtain *q* **where** *q*:

valid_path q z $z \notin \text{path_image } q$

pathstart q = *pathstart h* *pathfinish q* = *pathfinish h*

and *hq_less*: $\forall t \in \{0..1\}. \text{cmod } (h \ t - q \ t) < e$

and *paq*: $\text{contour_integral } q \ (\lambda w. 1 / (w - z)) = \text{complex_of_real } (2 * \pi i)$

* i * *winding_number h z*

using *winding_number* [*OF* $\langle \text{path } h \rangle \text{ pah } \langle 0 < e \rangle$] **unfolding** *winding_number_prop_def* **by** *blast*

have *homotopic_paths* $(-\{z\}) \ g \ p$

by (*simp add*: *d p valid_path_imp_path norm_minus_commute gp_less*)

moreover have *homotopic_paths* $(-\{z\}) \ h \ q$

by (*simp add*: *e q valid_path_imp_path norm_minus_commute hq_less*)

ultimately have *homotopic_paths* $(-\{z\}) \ p \ q$

by (*blast intro*: *homotopic_paths_trans homotopic_paths_sym assms*)

then have $\text{contour_integral } p \ (\lambda w. 1 / (w - z)) = \text{contour_integral } q \ (\lambda w. 1 / (w - z))$

by (*rule Cauchy_theorem_homotopic_paths*) (*auto intro*!: *holomorphic_intros simp*: *p q*)

then show *?thesis*

by (*simp add*: *pap paq*)

qed

lemma *winding_number_homotopic_loops*:

assumes *homotopic_loops* ($-\{z\}$) *g h*

shows *winding_number* *g z* = *winding_number* *h z*

proof –

have *path* *g* *path* *h* **using** *homotopic_loops_imp_path* [*OF* *assms*] **by** *auto*

moreover **have** *pag*: $z \notin \text{path_image } g$ **and** *pah*: $z \notin \text{path_image } h$

using *homotopic_loops_imp_subset* [*OF* *assms*] **by** *auto*

moreover **have** *gloop*: *pathfinish* *g* = *pathstart* *g* **and** *hloop*: *pathfinish* *h* = *pathstart* *h*

using *homotopic_loops_imp_loop* [*OF* *assms*] **by** *auto*

ultimately obtain *d e* **where** $d > 0$ $e > 0$

and *d*: $\bigwedge p. \llbracket \text{path } p; \text{pathfinish } p = \text{pathstart } p; \forall t \in \{0..1\}. \text{norm } (p \ t - g \ t) < d \rrbracket$

$\implies \text{homotopic_loops } (-\{z\}) \ g \ p$

and *e*: $\bigwedge q. \llbracket \text{path } q; \text{pathfinish } q = \text{pathstart } q; \forall t \in \{0..1\}. \text{norm } (q \ t - h \ t) < e \rrbracket$

$\implies \text{homotopic_loops } (-\{z\}) \ h \ q$

using *homotopic_nearby_loops* [*of* *g* $-\{z\}$] *homotopic_nearby_loops* [*of* *h* $-\{z\}$] **by** *force*

obtain *p* **where** *p*:

valid_path *p* $z \notin \text{path_image } p$

pathstart *p* = *pathstart* *g* *pathfinish* *p* = *pathfinish* *g*

and *gp_less*: $\forall t \in \{0..1\}. \text{cmod } (g \ t - p \ t) < d$

and *pap*: *contour_integral* *p* $(\lambda w. 1 / (w - z)) = \text{complex_of_real } (2 * \pi i)$

* i * *winding_number* *g z*

using *winding_number* [*OF* $\langle \text{path } g \rangle \text{ pag } \langle 0 < d \rangle$] **unfolding** *winding_number_prop_def*

by *blast*

obtain *q* **where** *q*:

valid_path *q* $z \notin \text{path_image } q$

pathstart *q* = *pathstart* *h* *pathfinish* *q* = *pathfinish* *h*

and *hq_less*: $\forall t \in \{0..1\}. \text{cmod } (h \ t - q \ t) < e$

and *paq*: *contour_integral* *q* $(\lambda w. 1 / (w - z)) = \text{complex_of_real } (2 * \pi i)$

* i * *winding_number* *h z*

using *winding_number* [*OF* $\langle \text{path } h \rangle \text{ pah } \langle 0 < e \rangle$] **unfolding** *winding_number_prop_def*

by *blast*

have *gp*: *homotopic_loops* ($-\{z\}$) *g p*

by (*simp* *add*: *gloop* *d gp_less* *norm_minus_commute* *p* *valid_path_imp_path*)

have *hq*: *homotopic_loops* ($-\{z\}$) *h q*

by (*simp* *add*: *e hloop* *hq_less* *norm_minus_commute* *q* *valid_path_imp_path*)

have *contour_integral* *p* $(\lambda w. 1 / (w - z)) = \text{contour_integral } q \ (\lambda w. 1 / (w - z))$

proof (*rule* *Cauchy_theorem_homotopic_loops*)

show *homotopic_loops* ($-\{z\}$) *p q*

by (*blast* *intro*: *homotopic_loops_trans* *homotopic_loops_sym* *gp* *hq* *assms*)

qed (*auto* *intro*!: *holomorphic_intros* *simp*: *p q*)

then show *?thesis*

by (*simp* *add*: *pap* *paq*)

qed

lemma *winding_number_paths_linear_eq*:
 $\llbracket \text{path } g; \text{ path } h; \text{ pathstart } h = \text{pathstart } g; \text{ pathfinish } h = \text{pathfinish } g; \bigwedge t. t \in \{0..1\} \implies z \notin \text{closed_segment } (g \ t) \ (h \ t) \rrbracket$
 $\implies \text{winding_number } h \ z = \text{winding_number } g \ z$
by (blast intro: sym homotopic_paths_linear winding_number_homotopic_paths)

lemma *winding_number_loops_linear_eq*:
 $\llbracket \text{path } g; \text{ path } h; \text{ pathfinish } g = \text{pathstart } g; \text{ pathfinish } h = \text{pathstart } h; \bigwedge t. t \in \{0..1\} \implies z \notin \text{closed_segment } (g \ t) \ (h \ t) \rrbracket$
 $\implies \text{winding_number } h \ z = \text{winding_number } g \ z$
by (blast intro: sym homotopic_loops_linear winding_number_homotopic_loops)

lemma *winding_number_nearby_paths_eq*:
 $\llbracket \text{path } g; \text{ path } h; \text{ pathstart } h = \text{pathstart } g; \text{ pathfinish } h = \text{pathfinish } g; \bigwedge t. t \in \{0..1\} \implies \text{norm}(h \ t - g \ t) < \text{norm}(g \ t - z) \rrbracket$
 $\implies \text{winding_number } h \ z = \text{winding_number } g \ z$
by (metis segment_bound(2) norm_minus_commute not_le winding_number_paths_linear_eq)

lemma *winding_number_nearby_loops_eq*:
 $\llbracket \text{path } g; \text{ path } h; \text{ pathfinish } g = \text{pathstart } g; \text{ pathfinish } h = \text{pathstart } h; \bigwedge t. t \in \{0..1\} \implies \text{norm}(h \ t - g \ t) < \text{norm}(g \ t - z) \rrbracket$
 $\implies \text{winding_number } h \ z = \text{winding_number } g \ z$
by (metis segment_bound(2) norm_minus_commute not_le winding_number_loops_linear_eq)

lemma *winding_number_subpath_combine*:
assumes *path g and notin*: $z \notin \text{path_image } g$ **and** $u \in \{0..1\} \ v \in \{0..1\} \ w \in \{0..1\}$
shows $\text{winding_number } (\text{subpath } u \ v \ g) \ z + \text{winding_number } (\text{subpath } v \ w \ g) \ z =$
 $\text{winding_number } (\text{subpath } u \ w \ g) \ z$ (**is** ?lhs = ?rhs)
proof –
have ?lhs = $\text{winding_number } (\text{subpath } u \ v \ g \ ++ \ \text{subpath } v \ w \ g) \ z$
using *assms*
by (metis (no_types, lifting) path_image_subpath_subset path_subpath pathfinish_subpath pathstart_subpath subsetD winding_number_join)
also have ... = ?rhs
by (meson *assms* homotopic_join_subpaths subset_Compl_singleton winding_number_homotopic_paths)
finally show ?thesis .
qed

Winding numbers of circular contours

proposition *winding_number_part_circlepath_pos_less*:
assumes $s < t$ **and** *no*: $\text{norm}(w - z) < r$
shows $0 < \text{Re } (\text{winding_number}(\text{part_circlepath } z \ r \ s \ t) \ w)$
proof –
have $0 < r$ **by** (meson *no* norm_not_less_zero not_le order.strict_trans2)

```

note valid_path_part_circlepath
moreover have  $w \notin \text{path\_image } (\text{part\_circlepath } z \ r \ s \ t)$ 
  using assms by (auto simp: path_image_def image_def part_circlepath_def
norm_mult linepath_def)
moreover have  $0 < r * (t - s) * (r - \text{cmod } (w - z))$ 
  using assms by (metis  $\langle 0 < r \rangle$  diff_gt_0_iff_gt mult_pos_pos)
ultimately show ?thesis
  apply (rule winding_number_pos_lt [where  $e = r * (t - s) * (r - \text{norm}(w - z))$ ])
  apply (simp add: vector_derivative_part_circlepath right_diff_distrib [symmetric]
mult_ac mult_le_cancel_left_pos assms  $\langle 0 < r \rangle$ )
  using Re_Im_le_cmod [of  $w - z$  linepath  $s \ t \ x$  for  $x$ ]
  by (simp add: exp_Euler cos_of_real sin_of_real part_circlepath_def algebra_simps
cos_squared_eq [unfolded power2_eq_square])
qed

```

```

lemma winding_number_circlepath_centre:  $0 < r \implies \text{winding\_number } (\text{circlepath } z \ r) \ z = 1$ 
  apply (rule winding_number_unique_loop)
  apply (simp_all add: sphere_def valid_path_imp_path)
  apply (rule_tac  $x = \text{circlepath } z \ r$  in exI)
  apply (simp add: sphere_def contour_integral_circlepath)
done

```

proposition winding_number_circlepath:

assumes $\text{norm}(w - z) < r$ **shows** $\text{winding_number}(\text{circlepath } z \ r) \ w = 1$

proof (cases $w = z$)

case True **then show** ?thesis

using *assms* winding_number_circlepath_centre **by** auto

next

case False

have [simp]: $r > 0$

using *assms* le_less_trans norm_ge_zero **by** blast

define r' **where** $r' = \text{norm}(w - z)$

have $r' < r$

by (simp add: *assms* r' _def)

have $\text{disjo: } \text{cball } z \ r' \cap \text{sphere } z \ r = \{\}$

using $\langle r' < r \rangle$ **by** (force simp: cball_def sphere_def)

have $\text{winding_number}(\text{circlepath } z \ r) \ w = \text{winding_number}(\text{circlepath } z \ r) \ z$

proof (rule winding_number_around_inside [where $S = \text{cball } z \ r'$])

show $\text{winding_number } (\text{circlepath } z \ r) \ z \neq 0$

by (simp add: winding_number_circlepath_centre)

show $\text{cball } z \ r' \cap \text{path_image } (\text{circlepath } z \ r) = \{\}$

by (simp add: disjo less_eq_real_def)

qed (auto simp: r' _def dist_norm norm_minus_commute)

also have $\dots = 1$

by (simp add: winding_number_circlepath_centre)

finally show ?thesis .

qed


```

lemma no_bounded_connected_component_imp_winding_number_zero:
  assumes g: path g path_image g  $\subseteq$  S pathfinish g = pathstart g  $z \notin S$ 
    and nb:  $\bigwedge z. \text{bounded} (\text{connected\_component\_set } (- S) z) \implies z \in S$ 
  shows winding_number g z = 0
proof -
  have z  $\in$  outside (path_image g)
    by (metis nb [of z]  $\langle \text{path\_image } g \subseteq S \rangle \langle z \notin S \rangle \text{subsetD mem\_Collect\_eq}$ 
    outside outside_mono)
  then show ?thesis
    by (simp add: g winding_number_zero_in_outside)
qed

lemma no_bounded_path_component_imp_winding_number_zero:
  assumes g: path g path_image g  $\subseteq$  S pathfinish g = pathstart g  $z \notin S$ 
    and nb:  $\bigwedge z. \text{bounded} (\text{path\_component\_set } (- S) z) \implies z \in S$ 
  shows winding_number g z = 0
by (simp add: bounded_subset nb path_component_subset_connected_component

    no_bounded_connected_component_imp_winding_number_zero [OF
g])

```

3.7 Winding number for a triangle

```

lemma wn_triangle1:
  assumes 0  $\in$  interior(convex hull {a,b,c})
  shows  $\neg (\text{Im}(a/b) \leq 0 \wedge 0 \leq \text{Im}(b/c))$ 
proof -
  { assume 0:  $\text{Im}(a/b) \leq 0 \wedge 0 \leq \text{Im}(b/c)$ 
    have 0  $\notin$  interior (convex hull {a,b,c})
    proof (cases a=0  $\vee$  b=0  $\vee$  c=0)
      case True then show ?thesis
        by (auto simp: not_in_interior_convex_hull_3)
    next
      case False
      then have b  $\neq$  0 by blast
      { fix x y::complex and u::real
        assume eq_f':  $\text{Im } x * \text{Re } b \leq \text{Im } b * \text{Re } x \wedge \text{Im } y * \text{Re } b \leq \text{Im } b * \text{Re } y \wedge 0 \leq$ 
u u  $\leq$  1
        then have  $((1 - u) * \text{Im } x) * \text{Re } b \leq \text{Im } b * ((1 - u) * \text{Re } x)$ 
          by (simp add: mult_left_mono mult.assoc mult.left_commute [of Im b])
        then have  $((1 - u) * \text{Im } x + u * \text{Im } y) * \text{Re } b \leq \text{Im } b * ((1 - u) * \text{Re } x$ 
+ u * Re y)
          using eq_f' ordered_comm_semiring_class.comm_mult_left_mono
          by (fastforce simp add: algebra_simps)
        }
      then have convex {z.  $\text{Im } z * \text{Re } b \leq \text{Im } b * \text{Re } z$ }
        by (auto simp: algebra_simps convex_alt)
      with False 0 have convex hull {a,b,c}  $\leq$  {z.  $\text{Im } z * \text{Re } b \leq \text{Im } b * \text{Re } z$ }

```

```

    by (simp add: subset_hull mult.commute Complex.Im_divide divide_simps
complex_neq_0 [symmetric])
  moreover have  $0 \notin \text{interior}(\{z. \text{Im } z * \text{Re } b \leq \text{Im } b * \text{Re } z\})$ 
  proof
    assume  $0 \in \text{interior} \{z. \text{Im } z * \text{Re } b \leq \text{Im } b * \text{Re } z\}$ 
    then obtain  $e$  where  $e > 0$  and  $e: \text{ball } 0 \ e \subseteq \{z. \text{Im } z * \text{Re } b \leq \text{Im } b * \text{Re } z\}$ 
  z}
    by (meson mem_interior)
    define  $z$  where  $z \equiv - \text{sgn } (\text{Im } b) * (e/3) + \text{sgn } (\text{Re } b) * (e/3) * i$ 
    have  $\text{cmod } z = \text{cmod } (e/3) * \text{cmod } (- \text{sgn } (\text{Im } b) + \text{sgn } (\text{Re } b) * i)$ 
    unfolding  $z\_def$  norm_mult [symmetric] by (simp add: algebra_simps)
    also have  $\dots < e$ 
    using  $\langle e > 0 \rangle$  by (auto simp: norm_mult intro: le_less_trans [OF
norm_triangle_ineq4])
    finally have  $z \in \text{ball } 0 \ e$ 
    using  $\langle e > 0 \rangle$  by simp
    then have  $\text{Im } z * \text{Re } b \leq \text{Im } b * \text{Re } z$ 
    using  $e$  by blast
    then have  $b: 0 < \text{Re } b \ 0 < \text{Im } b$  and  $\text{disj}: e * \text{Re } b < - (\text{Im } b * e) \vee e * \text{Re } b = - (\text{Im } b * e)$ 
    using  $\langle e > 0 \rangle \ \langle b \neq 0 \rangle$ 
    by (auto simp add:  $z\_def$  dist_norm sgn_if_less_eq_real_def mult_less_0_iff
complex.expand split: if_split_asm)
    show False — or just one smt line
    using disj
    proof
      assume  $e * \text{Re } b < - (\text{Im } b * e)$ 
      with  $b \ \langle 0 < e \rangle$  less_trans [of _ 0] show False
    by (metis (no_types) mult_pos_pos neg_less_0_iff_less not_less_iff_gr_or_eq)
    next
      assume  $e * \text{Re } b = - (\text{Im } b * e)$ 
      with  $b \ \langle 0 < e \rangle$  show False
    by (metis mult_pos_pos neg_less_0_iff_less not_less_iff_gr_or_eq)
    qed
  qed
  ultimately show ?thesis
  using interior_mono by blast
qed
} with assms show ?thesis by blast
qed

```

lemma $wn_triangle2_0$:

assumes $0 \in \text{interior}(\text{convex hull } \{a, b, c\})$

shows

$$\begin{aligned}
 & 0 < \text{Im}((b - a) * \text{cnj } (b)) \wedge \\
 & 0 < \text{Im}((c - b) * \text{cnj } (c)) \wedge \\
 & 0 < \text{Im}((a - c) * \text{cnj } (a)) \\
 & \vee \\
 & \text{Im}((b - a) * \text{cnj } (b)) < 0 \wedge
 \end{aligned}$$

```

      0 < Im((b - c) * cnj (b)) ∧
      0 < Im((a - b) * cnj (a)) ∧
      0 < Im((c - a) * cnj (c))
proof -
  have [simp]: {b,c,a} = {a,b,c} {c,a,b} = {a,b,c} by auto
  show ?thesis
    using wn_triangle1 [OF assms] wn_triangle1 [of b c a] wn_triangle1 [of c a
b] assms
    by (auto simp: algebra_simps Im_complex_div_gt_0 Im_complex_div_lt_0
not_le not_less)
qed

```

```

lemma wn_triangle2:
  assumes z ∈ interior(convex hull {a,b,c})
  shows 0 < Im((b - a) * cnj (b - z)) ∧
        0 < Im((c - b) * cnj (c - z)) ∧
        0 < Im((a - c) * cnj (a - z))
        ∨
        Im((b - a) * cnj (b - z)) < 0 ∧
        0 < Im((b - c) * cnj (b - z)) ∧
        0 < Im((a - b) * cnj (a - z)) ∧
        0 < Im((c - a) * cnj (c - z))
proof -
  have 0: 0 ∈ interior(convex hull {a-z, b-z, c-z})
  using assms convex_hull_translation [of -z {a,b,c}]
    interior_translation [of -z]
  by (simp cong: image_cong_simp)
  show ?thesis using wn_triangle2_0 [OF 0]
  by simp
qed

```

```

lemma wn_triangle3:
  assumes z: z ∈ interior(convex hull {a,b,c})
  and 0 < Im((b-a) * cnj (b-z))
        0 < Im((c-b) * cnj (c-z))
        0 < Im((a-c) * cnj (a-z))
  shows winding_number (linepath a b +++ linepath b c +++ linepath c a) z
    = 1
proof -
  have znot[simp]: z ∉ closed_segment a b z ∉ closed_segment b c z ∉ closed_segment
c a
  using z interior_of_triangle [of a b c]
  by (auto simp: closed_segment_def)
  have gt0: 0 < Re (winding_number (linepath a b +++ linepath b c +++ linepath
c a) z)
  using assms
  by (simp add: winding_number_linepath_pos_lt path_image_join winding_number_join_pos_combined)
  have lt2: Re (winding_number (linepath a b +++ linepath b c +++ linepath c
a) z) < 2

```

```

    using winding_number_lt_half_linepath [of _ a b]
    using winding_number_lt_half_linepath [of _ b c]
    using winding_number_lt_half_linepath [of _ c a] znot
    by (fastforce simp add: winding_number_join path_image_join)
  show ?thesis
    by (rule winding_number_eq_1) (simp_all add: path_image_join gt0 lt2)
qed

proposition winding_number_triangle:
  assumes z:  $z \in \text{interior}(\text{convex hull } \{a, b, c\})$ 
  shows  $\text{winding\_number}(\text{linepath } a \text{ } ++ \text{linepath } b \text{ } ++ \text{linepath } c \text{ } a) \text{ } z =$ 
     $(\text{if } 0 < \text{Im}((b - a) * \text{cnj } (b - z)) \text{ then } 1 \text{ else } -1)$ 
proof -
  have [simp]:  $\{a, c, b\} = \{a, b, c\}$  by auto
  have znot[simp]:  $z \notin \text{closed\_segment } a \text{ } b \text{ } z \notin \text{closed\_segment } b \text{ } c \text{ } z \notin \text{closed\_segment } c \text{ } a$ 
  using z interior_of_triangle [of a b c]
  by (auto simp: closed_segment_def)
  then have [simp]:  $z \notin \text{closed\_segment } b \text{ } a \text{ } z \notin \text{closed\_segment } c \text{ } b \text{ } z \notin \text{closed\_segment } a \text{ } c$ 
  using closed_segment_commute by blast+
  have *:  $\text{winding\_number}(\text{linepath } a \text{ } ++ \text{linepath } b \text{ } ++ \text{linepath } c \text{ } a) \text{ } z =$ 
     $\text{winding\_number}(\text{reversepath}(\text{linepath } a \text{ } ++ \text{linepath } c \text{ } b \text{ } ++ \text{linepath } b \text{ } a)) \text{ } z$ 
  by (simp add: reversepath_joinpaths winding_number_join not_in_path_image_join)
  show ?thesis
    apply (rule disjE [OF wn_triangle2 [OF z]])
    subgoal
      by (simp add: wn_triangle3 z)
    subgoal
      by (simp add: path_image_join winding_number_reversepath * wn_triangle3 z)
    done
qed

```

3.8 Winding numbers for simple closed paths

```

lemma winding_number_from_innerpath:
  assumes simple_path c1 and c1:  $\text{pathstart } c1 = a \text{ } \text{pathfinish } c1 = b$ 
    and simple_path c2 and c2:  $\text{pathstart } c2 = a \text{ } \text{pathfinish } c2 = b$ 
    and simple_path c and c:  $\text{pathstart } c = a \text{ } \text{pathfinish } c = b$ 
    and c1c2:  $\text{path\_image } c1 \cap \text{path\_image } c2 = \{a, b\}$ 
    and c1c:  $\text{path\_image } c1 \cap \text{path\_image } c = \{a, b\}$ 
    and c2c:  $\text{path\_image } c2 \cap \text{path\_image } c = \{a, b\}$ 
    and ne_12:  $\text{path\_image } c \cap \text{inside}(\text{path\_image } c1 \cup \text{path\_image } c2) \neq \{\}$ 
    and z:  $z \in \text{inside}(\text{path\_image } c1 \cup \text{path\_image } c)$ 
    and wn_d:  $\text{winding\_number}(c1 \text{ } ++ \text{reversepath } c) \text{ } z = d$ 
    and a ≠ b d ≠ 0
  obtains  $z \in \text{inside}(\text{path\_image } c1 \cup \text{path\_image } c2) \text{ } \text{winding\_number}(c1 \text{ } ++$ 

```

$\text{reversepath } c2) \ z = d$

proof –

obtain 0 : $\text{inside}(\text{path_image } c1 \cup \text{path_image } c) \cap \text{inside}(\text{path_image } c2 \cup \text{path_image } c) = \{\}$

and 1 : $\text{inside}(\text{path_image } c1 \cup \text{path_image } c) \cup \text{inside}(\text{path_image } c2 \cup \text{path_image } c) \cup$

$(\text{path_image } c - \{a, b\}) = \text{inside}(\text{path_image } c1 \cup \text{path_image } c2)$

by (rule *split_inside_simple_closed_curve*

$[OF \langle \text{simple_path } c1 \rangle c1 \langle \text{simple_path } c2 \rangle c2 \langle \text{simple_path } c \rangle c \langle a \neq b \rangle c1c2 c1c c2c \text{ ne_12}]$)

have znot : $z \notin \text{path_image } c \ z \notin \text{path_image } c1 \ z \notin \text{path_image } c2$

using *union_with_outside* z **1** **by** *auto*

then have zout : $z \in \text{outside}(\text{path_image } c \cup \text{path_image } c2)$

by (*metis* 0 *CompII UnE disjoint_iff_not_equal sup commute union_with_inside* z)

have wn_cc2 : $\text{winding_number } (c +++ \text{reversepath } c2) \ z = 0$

by (rule *winding_number_zero_in_outside*; *simp add*: $\text{zout} \langle \text{simple_path } c2 \rangle$ $c \ c2 \langle \text{simple_path } c \rangle \text{simple_path_imp_path_path_image_join}$)

show *?thesis*

proof

show $z \in \text{inside}(\text{path_image } c1 \cup \text{path_image } c2)$

using $1 \ z$ **by** *blast*

have $\text{winding_number } c1 \ z - \text{winding_number } c \ z = d$

using *assms znot*

by (*metis* $\text{wn_d winding_number_join simple_path_imp_path winding_number_reversepath add commute path_image_reversepath path_reversepath pathstart_reversepath minus_add_conv_diff}$)

then show $\text{winding_number } (c1 +++ \text{reversepath } c2) \ z = d$

using wn_cc2 **by** (*simp add*: $\text{winding_number_join simple_path_imp_path assms znot winding_number_reversepath}$)

qed

qed

lemma *simple_closed_path_wn1*:

fixes $a::\text{complex}$ **and** $e::\text{real}$

assumes $0 < e$

and sp_pl : $\text{simple_path}(p +++ \text{linepath } (a - e) (a + e))$ (*is simple_path ?pae*)

and psp : $\text{pathstart } p = a + e$

and pfp : $\text{pathfinish } p = a - e$

and disj : $\text{ball } a \ e \cap \text{path_image } p = \{\}$

obtains z **where** $z \in \text{inside}(\text{path_image } ?pae) \ \text{cmod } (\text{winding_number } ?pae \ z) = 1$

proof –

have $\text{arc } p$ **and** arc_lp : $\text{arc } (\text{linepath } (a - e) (a + e))$

and pap : $\text{path_image } p \cap \text{path_image } (\text{linepath } (a - e) (a + e)) \subseteq \{\text{pathstart } p, a - e\}$

using *simple_path_join_loop_eq* [*of* $\text{linepath } (a - e) (a + e) \ p$] *assms* **by** *auto*

have $\text{mid_eq } a$: $\text{midpoint } (a - e) (a + e) = a$

```

    by (simp add: midpoint_def)
  with assms have a ∈ path_image ?pae
    by (simp add: assms path_image_join) (metis midpoint_in_closed_segment)
  then have a ∈ frontier (inside (path_image ?pae))
    using assms by (simp add: Jordan_inside_outside)
  with ‹0 < e› obtain c where c: c ∈ inside (path_image ?pae)
    and dac: dist a c < e
    by (auto simp: frontier_straddle)
  then have c ∉ path_image ?pae
    using inside_no_overlap by blast
  then have c ∉ path_image p c ∉ closed_segment (a - e) (a + e)
    by (simp_all add: assms path_image_join)
  with ‹0 < e› dac have c ∉ affine_hull {a - e, a + e}
    by (simp add: segment_as_ball not_le)
  with ‹0 < e› have *: ¬ collinear {a - e, c, a + e}
    using collinear_3_affine_hull [of a-e a+e] by (auto simp: insert_commute)
  have 13: 1/3 + 1/3 + 1/3 = (1::real) by simp
  have (1/3) *R (a - of_real e) + (1/3) *R c + (1/3) *R (a + of_real e) ∈
    interior (convex_hull {a - e, c, a + e})
    using interior_convex_hull_3_minimal [OF * DIM_complex]
    by clarsimp (metis 13 zero_less_divide_1 iff_zero_less_numeral)
  then obtain z where z: z ∈ interior (convex_hull {a - e, c, a + e}) by force
  have [simp]: z ∉ closed_segment (a - e) c
    by (metis DIM_complex Diff_iff IntD2 inf_sup_absorb interior_of_triangle
    z)
  have [simp]: z ∉ closed_segment (a + e) (a - e)
    by (metis DIM_complex DiffD2 Un_iff interior_of_triangle z)
  have [simp]: z ∉ closed_segment c (a + e)
    by (metis (no_types, lifting) DIM_complex DiffD2 Un_insert_right inf_sup_aci(5)
    insertCI interior_of_triangle mk_disjoint_insert z)
  show thesis
proof
  have norm (winding_number (linepath (a - e) c +++ linepath c (a + e) +++
    linepath (a + e) (a - e)) z) = 1
    using winding_number_triangle [OF z] by simp
  have zin: z ∈ inside (path_image (linepath (a + e) (a - e)) ∪ path_image p)
    and zeq: winding_number (linepath (a + e) (a - e) +++ reversepath p) z =
      winding_number (linepath (a - e) c +++ linepath c (a + e) +++
    linepath (a + e) (a - e)) z
  proof (rule winding_number_from_innerpath
    [of linepath (a + e) (a - e) a+e a-e p
      linepath (a + e) c +++ linepath c (a - e) z
      winding_number (linepath (a - e) c +++ linepath c (a + e) +++
    linepath (a + e) (a - e)) z])
    show sp_aec: simple_path (linepath (a + e) c +++ linepath c (a - e))
  proof (rule arc_imp_simple_path [OF arc_join])
    show arc (linepath (a + e) c)
    by (metis ‹c ∉ path_image p› arc_linepath pathstart_in_path_image psp)
    show arc (linepath c (a - e))

```

```

    by (metis ‹ $c \notin \text{path\_image } p$ › arc_linepath pathfinish_in_path_image pfp)
    show path_image (linepath (a + e) c)  $\cap$  path_image (linepath c (a - e))
 $\subseteq \{\text{pathstart (linepath c (a - e))}\}$ 
    by clarsimp (metis * IntI Int_closed_segment closed_segment_commute
singleton_iff)
qed auto
show simple_path p
    using ‹arc p› arc_simple_path by blast
show sp_ae2: simple_path (linepath (a + e) (a - e))
    using ‹arc p› arc_distinct_ends pfp psp by fastforce
show pa: pathfinish (linepath (a + e) (a - e)) = a - e
    pathstart (linepath (a + e) c +++ linepath c (a - e)) = a + e
    pathfinish (linepath (a + e) c +++ linepath c (a - e)) = a - e
    pathstart p = a + e pathfinish p = a - e
    pathstart (linepath (a + e) (a - e)) = a + e
    by (simp_all add: assms)
show 1: path_image (linepath (a + e) (a - e))  $\cap$  path_image p = {a + e,
a - e}
proof
    show path_image (linepath (a + e) (a - e))  $\cap$  path_image p  $\subseteq \{a + e, a - e\}$ 
        using pap closed_segment_commute psp segment_convex_hull by fastforce
    show {a + e, a - e}  $\subseteq \text{path\_image (linepath (a + e) (a - e))} \cap \text{path\_image } p$ 
        using pap pathfinish_in_path_image pathstart_in_path_image pfp psp
by fastforce
qed
show 2: path_image (linepath (a + e) (a - e))  $\cap$  path_image (linepath (a +
e) c +++ linepath c (a - e)) =
    {a + e, a - e} (is ?lhs = ?rhs)
proof
    have  $\neg \text{collinear } \{c, a + e, a - e\}$ 
        using * by (simp add: insert_commute)
    then have convex_hull {a + e, a - e}  $\cap$  convex_hull {a + e, c} = {a + e}
        convex_hull {a + e, a - e}  $\cap$  convex_hull {c, a - e} = {a - e}
    by (metis (full_types) Int_closed_segment insert_commute segment_convex_hull)
    then show ?lhs  $\subseteq$  ?rhs
        by (metis Int_Un_distrib equalityD1 insert_is_Un path_image_join
path_image_linepath path_join_eq path_linepath segment_convex_hull simple_path_def
sp_aec)
    show ?rhs  $\subseteq$  ?lhs
        using segment_convex_hull by (simp add: path_image_join)
qed
have path_image p  $\cap$  path_image (linepath (a + e) c)  $\subseteq \{a + e\}$ 
proof (clarsimp simp: path_image_join)
    fix x
    assume  $x \in \text{path\_image } p$  and x_ac:  $x \in \text{closed\_segment (a + e) c}$ 
    then have dist x a  $\geq$  e
        by (metis IntI all_not_in_conv disj_dist_commute mem_ball not_less)

```

```

    with x_ac dac ‹e > 0› show x = a + e
  by (auto simp: norm_minus_commute dist_norm closed_segment_eq_open
dest: open_segment_furthest_le [where y=a])
qed
moreover
have path_image p ∩ path_image (linepath c (a - e)) ⊆ {a - e}
proof (clarsimp simp: path_image_join)
  fix x
  assume x ∈ path_image p and x_ac: x ∈ closed_segment c (a - e)
  then have dist x a ≥ e
    by (metis IntI all_not_in_conv disj dist_commute mem_ball not_less)
  with x_ac dac ‹e > 0› show x = a - e
  by (auto simp: norm_minus_commute dist_norm closed_segment_eq_open
dest: open_segment_furthest_le [where y=a])
qed
ultimately
have path_image p ∩ path_image (linepath (a + e) c +++ linepath c (a -
e)) ⊆ {a + e, a - e}
  by (force simp: path_image_join)
then show ∃: path_image p ∩ path_image (linepath (a + e) c +++ linepath
c (a - e)) = {a + e, a - e}
  using 1 2 by blast
show 4: path_image (linepath (a + e) c +++ linepath c (a - e)) ∩
  inside (path_image (linepath (a + e) (a - e)) ∪ path_image p) ≠ {}
apply (clarsimp simp: path_image_join segment_convex_hull disjoint_iff_not_equal)
by (metis (no_types, opaque_lifting) UnI1 Un_commute c closed_segment_commute
ends_in_segment(1) path_image_join
  path_image_linepath pathstart_linepath pfp segment_convex_hull)
show zin_inside: z ∈ inside (path_image (linepath (a + e) (a - e)) ∪
  path_image (linepath (a + e) c +++ linepath c (a -
e)))
proof (simp add: path_image_join)
  show z ∈ inside (closed_segment (a + e) (a - e) ∪ (closed_segment (a +
e) c ∪ closed_segment c (a - e)))
  by (metis z inside_of_triangle DIM_complex Un_commute closed_segment_commute)
qed
show 5: winding_number
  (linepath (a + e) (a - e) +++ reversepath (linepath (a + e) c +++
linepath c (a - e))) z =
  winding_number (linepath (a - e) c +++ linepath c (a + e) +++
linepath (a + e) (a - e)) z
  by (simp add: reversepath_joinpaths path_image_join winding_number_join)
show 6: winding_number (linepath (a - e) c +++ linepath c (a + e) +++
linepath (a + e) (a - e)) z ≠ 0
  by (simp add: winding_number_triangle z)
show winding_number (linepath (a + e) (a - e) +++ reversepath p) z =
  winding_number (linepath (a - e) c +++ linepath c (a + e) +++
linepath (a + e) (a - e)) z
  by (metis 1 2 3 4 5 6 pa sp_aec sp_ae2 ‹arc p› ‹simple_path p› arc_distinct_ends

```



```

winding_number_from_innerpath zin_inside)
qed (use assms ⟨e > 0⟩ in auto)
show z_inside: z ∈ inside (path_image ?pae)
using zin by (simp add: assms path_image_join Un_commute closed_segment_commute)
have cmod (winding_number ?pae z) = cmod ((winding_number(reversepath
?pae) z))
proof (subst winding_number_reversepath)
show path ?pae
using simple_path_imp_path sp_pl by blast
show z ∉ path_image ?pae
by (metis IntI emptyE inside_no_overlap z_inside)
qed (simp add: inside_def)
also have ... = cmod (winding_number(linepath (a + e) (a - e) +++ re-
versepath p) z)
by (simp add: pfp reversepath_joinpaths)
also have ... = cmod (winding_number (linepath (a - e) c +++ linepath c (a
+ e) +++ linepath (a + e) (a - e)) z)
by (simp add: zeq)
also have ... = 1
using z by (simp add: interior_of_triangle winding_number_triangle)
finally show cmod (winding_number ?pae z) = 1 .
qed
qed

```

lemma simple_closed_path_wn2:

```

fixes a::complex and d e::real
assumes 0 < d 0 < e
and sp_pl: simple_path(p +++ linepath (a - d) (a + e))
and psp: pathstart p = a + e
and pfp: pathfinish p = a - d
obtains z where z ∈ inside (path_image (p +++ linepath (a - d) (a + e)))
cmod (winding_number (p +++ linepath (a - d) (a + e)) z) = 1
proof -
have [simp]: a + of_real x ∈ closed_segment (a - α) (a - β) ⟷ x ∈
closed_segment (-α) (-β) for x α β::real
using closed_segment_translation_eq [of a]
by (metis (no_types, opaque_lifting) add_uminus_conv_diff of_real_minus
of_real_closed_segment)
have [simp]: a - of_real x ∈ closed_segment (a + α) (a + β) ⟷ -x ∈
closed_segment α β for x α β::real
by (metis closed_segment_translation_eq diff_conv_add_uminus of_real_closed_segment
of_real_minus)
have arc p and arc_lp: arc (linepath (a - d) (a + e)) and path p
and pap: path_image p ∩ closed_segment (a - d) (a + e) ⊆ {a+e, a-d}
using simple_path_join_loop_eq [of linepath (a - d) (a + e) p] assms
arc_imp_path by auto
have 0 ∈ closed_segment (-d) e
using ⟨0 < d⟩ ⟨0 < e⟩ closed_segment_eq_real_ivl by auto
then have a ∈ path_image (linepath (a - d) (a + e))

```

```

    using of_real_closed_segment [THEN iffD2]
    by (force dest: closed_segment_translation_eq [of a, THEN iffD2] simp del:
of_real_closed_segment)
  then have  $a \notin \text{path\_image } p$ 
    using  $\langle 0 < d \rangle \langle 0 < e \rangle$  pap by auto
  then obtain k where  $0 < k$  and  $k: \text{ball } a \ k \cap (\text{path\_image } p) = \{\}$ 
    using  $\langle 0 < e \rangle \langle \text{path } p \rangle \text{ not\_on\_path\_ball}$  by blast
  define kde where  $\text{kde} \equiv (\min k (\min d e)) / 2$ 
  have  $0 < \text{kde} \ \text{kde} < k \ \text{kde} < d \ \text{kde} < e$ 
    using  $\langle 0 < k \rangle \langle 0 < d \rangle \langle 0 < e \rangle$  by (auto simp: kde_def)
  let ?q =  $\text{linepath } (a + \text{kde}) \ (a + e) \ +++ \ p \ +++ \ \text{linepath } (a - d) \ (a - \text{kde})$ 
  have  $-\text{kde} \in \text{closed\_segment } (-d) \ e$ 
    using  $\langle 0 < \text{kde} \rangle \langle \text{kde} < d \rangle \langle \text{kde} < e \rangle$  closed_segment_eq_real_ivl by auto
  then have  $a\_diff\_kde: a - \text{kde} \in \text{closed\_segment } (a - d) \ (a + e)$ 
    using of_real_closed_segment [THEN iffD2]
    by (force dest: closed_segment_translation_eq [of a, THEN iffD2] simp del:
of_real_closed_segment)
  then have clsub2:  $\text{closed\_segment } (a - d) \ (a - \text{kde}) \subseteq \text{closed\_segment } (a - d) \ (a + e)$ 
    by (simp add: subset_closed_segment)
  then have  $\text{path\_image } p \cap \text{closed\_segment } (a - d) \ (a - \text{kde}) \subseteq \{a + e, a - d\}$ 
    using pap by force
  moreover
  have  $a + e \notin \text{path\_image } p \cap \text{closed\_segment } (a - d) \ (a - \text{kde})$ 
    using  $\langle 0 < \text{kde} \rangle \langle \text{kde} < d \rangle \langle 0 < e \rangle$  by (auto simp: closed_segment_eq_real_ivl)
  ultimately have sub_a_diff_d:  $\text{path\_image } p \cap \text{closed\_segment } (a - d) \ (a - \text{kde}) \subseteq \{a - d\}$ 
    by blast
  have  $\text{kde} \in \text{closed\_segment } (-d) \ e$ 
    using  $\langle 0 < \text{kde} \rangle \langle \text{kde} < d \rangle \langle \text{kde} < e \rangle$  closed_segment_eq_real_ivl by auto
  then have a_diff_kde:  $a + \text{kde} \in \text{closed\_segment } (a - d) \ (a + e)$ 
    using of_real_closed_segment [THEN iffD2]
    by (force dest: closed_segment_translation_eq [of a, THEN iffD2] simp del:
of_real_closed_segment)
  then have clsub1:  $\text{closed\_segment } (a + \text{kde}) \ (a + e) \subseteq \text{closed\_segment } (a - d) \ (a + e)$ 
    by (simp add: subset_closed_segment)
  then have  $\text{closed\_segment } (a + \text{kde}) \ (a + e) \cap \text{path\_image } p \subseteq \{a + e, a - d\}$ 
    using pap by force
  moreover
  have  $\text{closed\_segment } (a + \text{kde}) \ (a + e) \cap \text{closed\_segment } (a - d) \ (a - \text{kde}) = \{\}$ 
    proof (clarsimp intro!: equals0I)
      fix y
      assume y1:  $y \in \text{closed\_segment } (a + \text{kde}) \ (a + e)$ 
      and y2:  $y \in \text{closed\_segment } (a - d) \ (a - \text{kde})$ 
      obtain u::real where  $u: y = a + u$  and  $0 < u$ 

```

```

proof -
  obtain  $\xi$  where  $\xi$ :  $y = (1 - \xi) *_R (a + kde) + \xi *_R (a + e)$  and  $0 \leq \xi \leq 1$ 
    using  $y1$  by (auto simp: in_segment)
  show thesis
  proof
    show  $y = a + ((1 - \xi) * kde + \xi * e)$ 
      using  $\xi$  by (auto simp: scaleR_conv_of_real algebra_simps)
    have  $(1 - \xi) * kde + \xi * e \geq 0$ 
      using  $\langle 0 < kde \rangle \langle 0 \leq \xi \rangle \langle \xi \leq 1 \rangle \langle 0 < e \rangle$  by auto
    moreover have  $(1 - \xi) * kde + \xi * e \neq 0$ 
      using  $\langle 0 < kde \rangle \langle 0 \leq \xi \rangle \langle \xi \leq 1 \rangle \langle 0 < e \rangle$  by (auto simp: add_nonneg_eq_0_iff)
    ultimately show  $(1 - \xi) * kde + \xi * e > 0$  by simp
  qed
qed
moreover
  obtain  $v :: \text{real}$  where  $v$ :  $y = a + v$  and  $v \leq 0$ 
  proof -
    obtain  $\xi$  where  $\xi$ :  $y = (1 - \xi) *_R (a - d) + \xi *_R (a - kde)$  and  $0 \leq \xi \leq 1$ 
      using  $y2$  by (auto simp: in_segment)
    show thesis
    proof
      show  $y = a + (-((1 - \xi) * d + \xi * kde))$ 
        using  $\xi$  by (force simp: scaleR_conv_of_real algebra_simps)
      show  $-((1 - \xi) * d + \xi * kde) \leq 0$ 
        using  $\langle 0 < kde \rangle \langle 0 \leq \xi \rangle \langle \xi \leq 1 \rangle \langle 0 < d \rangle$ 
        by (metis add_left_neutral add_nonneg_nonneg le_diff_eq less_eq_real_def
          neg_le_0_iff_le split_mult_pos_le)
    qed
  qed
  ultimately show False
    by auto
qed
moreover have  $a - d \notin \text{closed\_segment } (a + kde) (a + e)$ 
  using  $\langle 0 < kde \rangle \langle kde < d \rangle \langle 0 < e \rangle$  by (auto simp: closed_segment_eq_real_ivl)
  ultimately have  $\text{sub\_a\_plus\_e}$ :
     $\text{closed\_segment } (a + kde) (a + e) \cap (\text{path\_image } p \cup \text{closed\_segment } (a - d) (a - kde)) \subseteq \{a + e\}$ 
    by auto
  have  $kde \in \text{closed\_segment } (-kde) e$ 
    using  $\langle 0 < kde \rangle \langle kde < d \rangle \langle kde < e \rangle$   $\text{closed\_segment\_eq\_real\_ivl}$  by auto
  then have  $a\_add\_kde$ :  $a + kde \in \text{closed\_segment } (a - kde) (a + e)$ 
    using  $\text{of\_real\_closed\_segment } [THEN \text{iffD2}]$ 
    by (force dest: closed_segment_translation_eq [of  $a$ , THEN iffD2] simp del:
      of_real_closed_segment)
  have  $\text{closed\_segment } (a - kde) (a + kde) \cap \text{closed\_segment } (a + kde) (a + e) = \{a + kde\}$ 

```

```

    by (metis a_add_kde Int_closed_segment)
  moreover
  have path_image p  $\cap$  closed_segment (a - kde) (a + kde) = {}
  proof (rule equals0I, clarify)
    fix y assume y  $\in$  path_image p y  $\in$  closed_segment (a - kde) (a + kde)
    with equals0D [OF k, of y]  $\langle 0 < kde \rangle \langle kde < k \rangle$  show False
    by (auto simp: dist_norm dest: dist_decreases_closed_segment [where c=a])
  qed
  moreover
  have -kde  $\in$  closed_segment (-d) kde
    using  $\langle 0 < kde \rangle \langle kde < d \rangle \langle kde < e \rangle$  closed_segment_eq_real_ivl by auto
  then have a_diff_kde': a - kde  $\in$  closed_segment (a - d) (a + kde)
    using of_real_closed_segment [THEN iffD2]
    by (force dest: closed_segment_translation_eq [of a, THEN iffD2] simp del:
of_real_closed_segment)
  then have closed_segment (a - d) (a - kde)  $\cap$  closed_segment (a - kde) (a +
kde) = {a - kde}
    by (metis Int_closed_segment)
  ultimately
  have pa_subset_pm_kde: path_image ?q  $\cap$  closed_segment (a - kde) (a + kde)
 $\subseteq$  {a - kde, a + kde}
    by (auto simp: path_image_join assms)
  have ge_kde1:  $\exists y. x = a + \text{of\_real } y \wedge y \geq kde$  if x: x  $\in$  closed_segment (a +
kde) (a + e) for x
  proof -
    obtain u where  $0 \leq u \wedge u \leq 1$  and u:  $x = (1 - u) *_R (a + kde) + u *_R (a + e)$ 
    using x by (auto simp: in_segment)
    then have kde  $\leq (1 - u) * kde + u * e$ 
      using  $\langle kde < e \rangle$  segment_bound_lemma by auto
    moreover have  $x = a + ((1 - u) * kde + u * e)$ 
      by (fastforce simp: u_algebra_simps scaleR_conv_of_real)
    ultimately
    show ?thesis by blast
  qed
  have ge_kde2:  $\exists y. x = a + \text{of\_real } y \wedge y \leq -kde$  if x: x  $\in$  closed_segment (a -
d) (a - kde) for x
  proof -
    obtain u where  $0 \leq u \wedge u \leq 1$  and u:  $x = (1 - u) *_R (a - d) + u *_R (a - kde)$ 
    using x by (auto simp: in_segment)
    then have kde  $\leq ((1 - u) * d + u * kde)$ 
      using  $\langle kde < d \rangle$  segment_bound_lemma by auto
    moreover have  $x = a - ((1 - u) * d + u * kde)$ 
      by (fastforce simp: u_algebra_simps scaleR_conv_of_real)
    ultimately show ?thesis
      by (metis add_uminus_conv_diff neg_le_iff_le of_real_minus)
  qed
  have notin_paq: x  $\notin$  path_image ?q if dist a x < kde for x

```

```

proof -
  have  $x \notin \text{path\_image } p$ 
    using  $k \langle kde < k \rangle$  that by force
  then show ?thesis
    using that assms by (auto simp: path_image_join dist_norm dest!: ge_kde1
ge_kde2)
  qed
  obtain  $z$  where  $z \in \text{inside } (\text{path\_image } (?q +++ \text{linepath } (a - kde) (a +
kde)))$ 
    and  $z1: \text{cmod } (\text{winding\_number } (?q +++ \text{linepath } (a - kde) (a + kde))
z) = 1$ 
  proof (rule simple_closed_path_wn1 [of kde ?q a])
    show simple_path (?q +++ linepath (a - kde) (a + kde))
    proof (intro simple_path_join_loop conjI)
      show arc ?q
      proof (rule arc_join)
        show arc (linepath (a + kde) (a + e))
          using  $\langle kde < e \rangle \langle \text{arc } p \rangle$  by (force simp: pfp)
        show arc (p +++ linepath (a - d) (a - kde))
          using  $\langle kde < d \rangle \langle kde < e \rangle \langle \text{arc } p \rangle \text{sub\_a\_diff\_d}$  by (force simp: pfp intro:
arc_join)
        qed (auto simp: psp pfp path_image_join sub_a_plus_e)
        show arc (linepath (a - kde) (a + kde))
          using  $\langle 0 < kde \rangle$  by auto
      qed (use pa_subset_pm_kde in auto)
      qed (use  $\langle 0 < kde \rangle$  notin_paq in auto)
      have  $\text{eq}: \text{path\_image } (?q +++ \text{linepath } (a - kde) (a + kde)) = \text{path\_image } (p
+++ \text{linepath } (a - d) (a + e))$ 
        (is ?lhs = ?rhs)
      proof
        show ?lhs  $\subseteq$  ?rhs
          using clsub1 clsub2 apply (auto simp: path_image_join assms)
          by (meson subsetCE subset_closed_segment)
        show ?rhs  $\subseteq$  ?lhs
          apply (simp add: path_image_join assms Un_ac)
          by (metis Un_closed_segment Un_assoc a_diff_kde a_diff_kde' le_supI2
subset_refl)
        qed
      show thesis
      proof
        show  $z \in \text{inside } (\text{path\_image } (p +++ \text{linepath } (a - d) (a + e)))$ 
          by (metis eq zin)
        then have  $z \notin \text{path\_image } p$ 
          by (metis ComplD Un_iff inside_Un_outside path_image_join pathfin-
ish_linepath pathstart_reversepath pfp reversepath_linepath)
        have  $z \notin \text{closed\_segment } (a - d) (a + kde)$ 
          by (metis ComplD Un_iff Un_closed_segment a_diff_kde inside_Un_outside
path_image_join path_image_linepath pathstart_linepath pfp zzin)
        have  $z \notin \text{closed\_segment } (a - d) (a + e)$ 

```

```

    by (metis IntI UnCI emptyE inside_no_overlap path_image_join path_image_linepath
    pathstart_linepath pfp zzin)
    have znotin_kde_e:  $z \notin \text{closed\_segment } (a + kde) (a + e)$  and znotin_d_kde':
 $z \notin \text{closed\_segment } (a - d) (a - kde)$ 
    using clsub1 clsub2 zzin
    by (metis (no_types, opaque_lifting) IntI Un_iff emptyE inside_no_overlap
    path_image_join path_image_linepath pathstart_linepath pfp subsetD)+
    have winding_number (linepath (a - d) (a + e)) z =
      winding_number (linepath (a - d) (a + kde)) z + winding_number
    (linepath (a + kde) (a + e)) z
    proof (rule winding_number_split_linepath)
      show  $a + \text{complex\_of\_real } kde \in \text{closed\_segment } (a - d) (a + e)$ 
      by (simp add: a_diff_kde)
      show  $z \notin \text{closed\_segment } (a - d) (a + e)$ 
      by (metis ComplD Un_iff inside_Un_outside path_image_join path_image_linepath
    pathstart_linepath pfp zzin)
    qed
    also have ... = winding_number (linepath (a + kde) (a + e)) z +
      (winding_number (linepath (a - d) (a - kde)) z + winding_number
    (linepath (a - kde) (a + kde)) z)
    by (simp add: winding_number_split_linepath [of a-kde, symmetric] znotin_d_kde
    a_diff_kde')
    finally have winding_number (p +++ linepath (a - d) (a + e)) z =
      winding_number p z + winding_number (linepath (a + kde) (a +
    e)) z +
      (winding_number (linepath (a - d) (a - kde)) z +
      winding_number (linepath (a - kde) (a + kde)) z)
    by (metis (no_types, lifting) ComplD Un_iff ⟨arc p⟩ add.assoc arc_imp_path
    eq path_image_join path_join_path_ends path_linepath simple_path_imp_path
    sp_pl union_with_outside winding_number_join zin)
    also have ... = winding_number (linepath (a + kde) (a + e)) z
      + winding_number (p +++ linepath (a - d) (a - kde)) z
      + winding_number (linepath (a - kde) (a + kde)) z
    using ⟨path p⟩ znotin assms
    by simp (metis Un_iff Un_closed_segment a_diff_kde' path_image_linepath
    path_linepath pathstart_linepath winding_number_join znotin_d_kde)
    also have ... = winding_number ?q z + winding_number (linepath (a - kde)
    (a + kde)) z
    using ⟨path p⟩ znotin assms by (simp add: path_image_join winding_number_join
    znotin_kde_e znotin_d_kde')
    also have ... = winding_number (?q +++ linepath (a - kde) (a + kde)) z
    by (metis (mono_tags, lifting) ComplD UnCI ⟨path p⟩ eq inside_outside
    path_image_join path_join_eq path_linepath pathfinish_join pathfinish_linepath
    pathstart_join pathstart_linepath pfp psp winding_number_join zzin)
    finally have winding_number (p +++ linepath (a - d) (a + e)) z =
      winding_number (?q +++ linepath (a - kde) (a + kde)) z .
    then show cmod (winding_number (p +++ linepath (a - d) (a + e)) z) = 1
    by (simp add: z1)
  qed

```

qed

lemma *simple_closed_path_wn3*:

fixes $p :: \text{real} \Rightarrow \text{complex}$
assumes *simple_path* p **and** *loop*: $\text{pathfinish } p = \text{pathstart } p$
obtains z **where** $z \in \text{inside } (\text{path_image } p)$ **cmod** $(\text{winding_number } p \ z) = 1$
proof –
have *ins*: $\text{inside}(\text{path_image } p) \neq \{\}$ *open*($\text{inside}(\text{path_image } p)$)
 $\text{connected}(\text{inside}(\text{path_image } p))$
and *out*: $\text{outside}(\text{path_image } p) \neq \{\}$ *open*($\text{outside}(\text{path_image } p)$)
 $\text{connected}(\text{outside}(\text{path_image } p))$
and *bo*: $\text{bounded}(\text{inside}(\text{path_image } p)) \neg \text{bounded}(\text{outside}(\text{path_image } p))$
and *ins_out*: $\text{inside}(\text{path_image } p) \cap \text{outside}(\text{path_image } p) = \{\}$
 $\text{inside}(\text{path_image } p) \cup \text{outside}(\text{path_image } p) = - \text{path_image } p$
and *fro*: $\text{frontier}(\text{inside}(\text{path_image } p)) = \text{path_image } p$
 $\text{frontier}(\text{outside}(\text{path_image } p)) = \text{path_image } p$
using *Jordan_inside_outside* [*OF* *assms*] **by** *auto*
obtain a **where** $a \in \text{inside}(\text{path_image } p)$
using $\langle \text{inside } (\text{path_image } p) \neq \{\} \rangle$ **by** *blast*
obtain $d :: \text{real}$ **where** $0 < d$ **and** d_fro : $a - d \in \text{frontier } (\text{inside } (\text{path_image } p))$
and d_int : $\bigwedge \varepsilon. \llbracket 0 \leq \varepsilon; \varepsilon < d \rrbracket \implies (a - \varepsilon) \in \text{inside } (\text{path_image } p)$
using *ray_to_frontier* [*of* $\text{inside } (\text{path_image } p)$ $a - 1$] *bo* *ins* *a* *interior_eq*
by (*metis* *ab_group_add_class.ab_diff_conv_add_uminus* *of_real_def* *scale_minus_right* *zero_neq_neg_one*)
obtain $e :: \text{real}$ **where** $0 < e$ **and** e_fro : $a + e \in \text{frontier } (\text{inside } (\text{path_image } p))$
and e_int : $\bigwedge \varepsilon. \llbracket 0 \leq \varepsilon; \varepsilon < e \rrbracket \implies (a + \varepsilon) \in \text{inside } (\text{path_image } p)$
using *ray_to_frontier* [*of* $\text{inside } (\text{path_image } p)$ $a + 1$] *bo* *ins* *a* *interior_eq*
by (*metis* *of_real_def* *zero_neq_one*)
obtain $t0$ **where** $0 \leq t0$ $t0 \leq 1$ **and** pt : $p \ t0 = a - d$
using $a \ d_fro$ **by** (*auto* *simp*: *path_image_def*)
obtain q **where** *simple_path* q **and** q_ends : $\text{pathstart } q = a - d$ $\text{pathfinish } q = a - d$
and q_eq_p : $\text{path_image } q = \text{path_image } p$
and $wn_q_eq_wn_p$: $\bigwedge z. z \in \text{inside}(\text{path_image } p) \implies \text{winding_number } q \ z = \text{winding_number } p \ z$
proof
show *simple_path* ($\text{shiftpath } t0 \ p$)
by (*simp* *add*: *pathstart_shiftpath* *pathfinish_shiftpath* *simple_path_shiftpath* *path_image_shiftpath* $\langle 0 \leq t0 \rangle \langle t0 \leq 1 \rangle$ *assms*)
show $\text{pathstart } (\text{shiftpath } t0 \ p) = a - d$
using pt **by** (*simp* *add*: $\langle t0 \leq 1 \rangle$ *pathstart_shiftpath*)
show $\text{pathfinish } (\text{shiftpath } t0 \ p) = a - d$
by (*simp* *add*: $\langle 0 \leq t0 \rangle$ *loop* *pathfinish_shiftpath* pt)
show $\text{path_image } (\text{shiftpath } t0 \ p) = \text{path_image } p$
by (*simp* *add*: $\langle 0 \leq t0 \rangle \langle t0 \leq 1 \rangle$ *loop* *path_image_shiftpath*)
show $\text{winding_number } (\text{shiftpath } t0 \ p) \ z = \text{winding_number } p \ z$
if $z \in \text{inside } (\text{path_image } p)$ **for** z

```

    by (metis ComplD Un_iff ‹0 ≤ t0› ‹t0 ≤ 1› ‹simple_path p› atLeastAt-
Most_iff inside_Un_outside
        loop_simple_path_imp_path that winding_number_shiftpath)
  qed
  have ad_not_ae: a - d ≠ a + e
  by (metis ‹0 < d› ‹0 < e› add.left_inverse add_left_cancel add_uminus_conv_diff
        le_add_same_cancel2 less_eq_real_def not_less of_real_add of_real_def
of_real_eq_0_iff pt)
  have ad_ae_q: {a - d, a + e} ⊆ path_image q
  using ‹path_image q = path_image p› d_fro e_fro fro(1) by auto
  have ada: open_segment (a - d) a ⊆ inside (path_image p)
  proof (clarsimp simp: in_segment)
    fix u::real assume 0 < u < 1
    with d_int have a - (1 - u) * d ∈ inside (path_image p)
    by (metis ‹0 < d› add.commute diff_add_cancel left_diff_distrib' less_add_same_cancel2
less_eq_real_def mult.left_neutral zero_less_mult_iff)
    then show (1 - u) *R (a - d) + u *R a ∈ inside (path_image p)
    by (simp add: diff_add_eq of_real_def real_vector.scale_right_diff_distrib)
  qed
  have aae: open_segment a (a + e) ⊆ inside (path_image p)
  proof (clarsimp simp: in_segment)
    fix u::real assume 0 < u < 1
    with e_int have a + u * e ∈ inside (path_image p)
    by (meson ‹0 < e› less_eq_real_def mult_less_cancel_right2 not_less
zero_less_mult_iff)
    then show (1 - u) *R a + u *R (a + e) ∈ inside (path_image p)
    by (metis (mono_tags, lifting) add.assoc of_real_mult pth_6 scaleR_collapse
scaleR_conv_of_real)
  qed
  have complex_of_real (d * d + (e * e + d * (e + e))) ≠ 0
  using ad_not_ae
  by (metis ‹0 < d› ‹0 < e› add_strict_left_mono less_add_same_cancel1
not_sum_squares_lt_zero
of_real_eq_0_iff zero_less_double_add_iff zero_less_single_add zero_less_mult_iff)
  moreover have ∃ u>0. u < 1 ∧ d = u * d + u * e
  using ‹0 < d› ‹0 < e›
  by (rule_tac x=d / (d+e) in exI) (simp add: divide_simps scaleR_conv_of_real
flip: distrib_left)
  ultimately have a_in_de: a ∈ open_segment (a - d) (a + e)
  using ad_not_ae by (simp add: in_segment algebra_simps scaleR_conv_of_real
flip: of_real_add of_real_mult)
  then have open_segment (a - d) (a + e) ⊆ inside (path_image p)
  using ada aae Un_open_segment [of a a-d a+e] by blast
  then have path_image q ∩ open_segment (a - d) (a + e) = {}
  using inside_no_overlap by (fastforce simp: q_eq_p)
  with ad_ae_q have paq_Int_cs: path_image q ∩ closed_segment (a - d) (a +
e) = {a - d, a + e}
  by (simp add: closed_segment_eq_open)
  obtain t where 0 ≤ t ≤ 1 and qt: q t = a + e

```



```

    using a_e_fro fro ad_ae_q by (auto simp: path_defs)
  then have t ≠ 0
    by (metis ad_not_ae pathstart_def q_ends(1))
  then have t ≠ 1
    by (metis ad_not_ae pathfinish_def q_ends(2) qt)
  have q01: q 0 = a - d q 1 = a - d
    using q_ends by (auto simp: pathstart_def pathfinish_def)
  obtain z where zin: z ∈ inside (path_image (subpath t 0 q +++ linepath (a - d) (a + e)))
    and z1: cmod (winding_number (subpath t 0 q +++ linepath (a - d) (a + e)) z) = 1
  proof (rule simple_closed_path_wn2 [of d e subpath t 0 q a], simp_all add: q01)
    show simple_path (subpath t 0 q +++ linepath (a - d) (a + e))
  proof (rule simple_path_join_loop, simp_all add: qt q01)
    have inj_on q (closed_segment t 0)
      using ⟨0 ≤ t⟩ ⟨simple_path q⟩ ⟨t ≤ 1⟩ ⟨t ≠ 0⟩ ⟨t ≠ 1⟩
    by (fastforce simp: simple_path_def loop_free_def inj_on_def closed_segment_eq_real_ivl)
    then show arc (subpath t 0 q)
      using ⟨0 ≤ t⟩ ⟨simple_path q⟩ ⟨t ≤ 1⟩ ⟨t ≠ 0⟩
    by (simp add: arc_subpath_eq simple_path_imp_path)
    show arc (linepath (a - d) (a + e))
      by (simp add: ad_not_ae)
    show path_image (subpath t 0 q) ∩ closed_segment (a - d) (a + e) ⊆ {a + e, a - d}
      using qt paq_Int_cs ⟨simple_path q⟩ ⟨0 ≤ t⟩ ⟨t ≤ 1⟩
    by (force simp: dest: rev_subsetD [OF path_image_subpath_subset] intro: simple_path_imp_path)
  qed
  qed (auto simp: ⟨0 < d⟩ ⟨0 < e⟩ qt)
  have pa01_Un: path_image (subpath 0 t q) ∪ path_image (subpath 1 t q) = path_image q
    unfolding path_image_subpath
    using ⟨0 ≤ t⟩ ⟨t ≤ 1⟩ by (force simp: path_image_def image_iff)
  with paq_Int_cs have pa_01q:
    (path_image (subpath 0 t q) ∪ path_image (subpath 1 t q)) ∩ closed_segment (a - d) (a + e) = {a - d, a + e}
    by metis
  have z_notin_ed: z ∉ closed_segment (a + e) (a - d)
    using zin q01 by (simp add: path_image_join closed_segment_commute inside_def)
  have z_notin_0t: z ∉ path_image (subpath 0 t q)
    by (metis (no_types, opaque_lifting) IntI Un_upper1 subsetD empty_iff inside_no_overlap path_image_join
      path_image_subpath_commute pathfinish_subpath pathstart_def pathstart_linepath q_ends(1) qt subpath_trivial zin)
  have [simp]: - winding_number (subpath t 0 q) z = winding_number (subpath 0 t q) z
    by (metis ⟨0 ≤ t⟩ ⟨simple_path q⟩ ⟨t ≤ 1⟩ atLeastAtMost_iff zero_le_one path_image_subpath_commute path_subpath_real_eq_0_iff le_ge_0)

```

```

      reversepath_subpath simple_path_imp_path winding_number_reversepath
z_notin_0t)
  obtain z_in_q: z ∈ inside(path_image q)
  and wn_q: winding_number (subpath 0 t q +++ subpath t 1 q) z = -
winding_number (subpath t 0 q +++ linepath (a - d) (a + e)) z
  proof (rule winding_number_from_innerpath
    [of subpath 0 t q a-d a+e subpath 1 t q linepath (a - d) (a + e)
      z - winding_number (subpath t 0 q +++ linepath (a - d) (a + e)) z],
    simp_all add: q01 qt pa01_Un reversepath_subpath)
  show simple_path (subpath 0 t q) simple_path (subpath 1 t q)
  by (simp_all add: ⟨0 ≤ t⟩ ⟨simple_path q⟩ ⟨t ≤ 1⟩ ⟨t ≠ 0⟩ ⟨t ≠ 1⟩ sim-
ple_path_subpath)
  show simple_path (linepath (a - d) (a + e))
  using ad_not_ae by blast
  show path_image (subpath 0 t q) ∩ path_image (subpath 1 t q) = {a - d, a +
e} (is ?lhs = ?rhs)
  proof
    show ?lhs ⊆ ?rhs
    using ⟨0 ≤ t⟩ ⟨simple_path q⟩ ⟨t ≤ 1⟩ ⟨t ≠ 1⟩ q_ends qt q01
    by (force simp: pathfinish_def qt simple_path_def loop_free_def path_image_subpath)
    show ?rhs ⊆ ?lhs
    using ⟨0 ≤ t⟩ ⟨t ≤ 1⟩ q01 qt by (force simp: path_image_subpath)
  qed
  show path_image (subpath 0 t q) ∩ closed_segment (a - d) (a + e) = {a -
d, a + e} (is ?lhs = ?rhs)
  proof
    show ?lhs ⊆ ?rhs using paq_Int_cs pa01_Un by fastforce
    show ?rhs ⊆ ?lhs using ⟨0 ≤ t⟩ ⟨t ≤ 1⟩ q01 qt by (force simp: path_image_subpath)
  qed
  show path_image (subpath 1 t q) ∩ closed_segment (a - d) (a + e) = {a -
d, a + e} (is ?lhs = ?rhs)
  proof
    show ?lhs ⊆ ?rhs by (auto simp: pa_01q [symmetric])
    show ?rhs ⊆ ?lhs using ⟨0 ≤ t⟩ ⟨t ≤ 1⟩ q01 qt by (force simp: path_image_subpath)
  qed
  show closed_segment (a - d) (a + e) ∩ inside (path_image q) ≠ {}
  using a_a_in_de open_closed_segment pa01_Un q_eq_p by fastforce
  show z ∈ inside (path_image (subpath 0 t q) ∪ closed_segment (a - d) (a +
e))
  by (metis path_image_join path_image_linepath path_image_subpath_commute
pathfinish_subpath pathstart_linepath q01(1) zin)
  show winding_number (subpath 0 t q +++ linepath (a + e) (a - d)) z =
- winding_number (subpath t 0 q +++ linepath (a - d) (a + e)) z
  using z_notin_ed z_notin_0t ⟨0 ≤ t⟩ ⟨simple_path q⟩ ⟨t ≤ 1⟩
  by (simp add: simple_path_imp_path qt q01 path_image_subpath_commute
closed_segment_commute winding_number_join winding_number_reversepath [symmetric])
  show - d ≠ e
  using ad_not_ae by auto
  show winding_number (subpath t 0 q +++ linepath (a - d) (a + e)) z ≠ 0

```

```

    using z1 by auto
  qed
  show ?thesis
proof
  show  $z \in \text{inside}(\text{path\_image } p)$ 
    using  $q\_eq\_p \ z\_in\_q$  by auto
  then have [simp]:  $z \notin \text{path\_image } q$ 
    by (metis disjoint_iff_not_equal inside_no_overlap q_eq_p)
  have [simp]:  $z \notin \text{path\_image}(\text{subpath } 1 \ t \ q)$ 
    using inside_def pa01_Un z_in_q by fastforce
  have winding_number(subpath 0 t q +++ subpath t 1 q)  $z = \text{winding\_number}(\text{subpath } 0 \ 1 \ q) \ z$ 
    using  $z \notin \text{path\_image } q$  by auto
  by (simp add: simple_path_imp_path_image_subpath_commute winding_number_join winding_number_subpath_combine)
  with wn_q have winding_number(subpath t 0 q +++ linepath (a - d) (a + e))  $z = - \text{winding\_number } q \ z$ 
    by auto
  with z1 have cmod(winding_number q z) = 1
    by simp
  with z1 wn_q_eq_wn_p show cmod(winding_number p z) = 1
    using z1 wn_q_eq_wn_p by (simp add:  $\langle z \in \text{inside}(\text{path\_image } p) \rangle$ )
  qed
qed

```

proposition *simple_closed_path_winding_number_inside:*

```

  assumes simple_path  $\gamma$ 
  obtains  $\bigwedge z. z \in \text{inside}(\text{path\_image } \gamma) \implies \text{winding\_number } \gamma \ z = 1$ 
    |  $\bigwedge z. z \in \text{inside}(\text{path\_image } \gamma) \implies \text{winding\_number } \gamma \ z = -1$ 
proof (cases pathfinish  $\gamma = \text{pathstart } \gamma$ )
  case True
  have path  $\gamma$ 
    by (simp add: assms simple_path_imp_path)
  then have const: winding_number  $\gamma$  constant_on inside(path_image  $\gamma$ )
  proof (rule winding_number_constant)
    show connected (inside(path_image  $\gamma$ ))
      by (simp add: Jordan_inside_outside True assms)
  qed (use inside_no_overlap True in auto)
  obtain z where zin:  $z \in \text{inside}(\text{path\_image } \gamma)$  and z1: cmod(winding_number  $\gamma \ z) = 1$ 
    using simple_closed_path_wn3 [of  $\gamma$ ] True assms by blast
  have winding_number  $\gamma \ z \in \mathbb{Z}$ 
    using zin integer_winding_number [OF  $\langle \text{path } \gamma \rangle$  True] inside_def by blast
  moreover have real_of_int  $x = -1 \iff x = -1$  for  $x$ 
    by linarith
  ultimately consider winding_number  $\gamma \ z = 1$  | winding_number  $\gamma \ z = -1$ 
    using z1 by (auto simp: Ints_def abs_if_split: if_split_asm)
  with that const zin show ?thesis
    unfolding constant_on_def by metis

```

```

next
  case False
  then show ?thesis
    using inside_simple_curve_imp_closed assms that(2) by blast
qed

lemma simple_closed_path_abs_winding_number_inside:
  assumes simple_path  $\gamma$   $z \in \text{inside}(\text{path\_image } \gamma)$ 
  shows  $|\text{Re } (\text{winding\_number } \gamma \ z)| = 1$ 
  by (metis assms norm_minus_cancel norm_one one_complex.simps(1) real_norm_def
    simple_closed_path_winding_number_inside uminus_complex.simps(1))

lemma simple_closed_path_norm_winding_number_inside:
  assumes simple_path  $\gamma$   $z \in \text{inside}(\text{path\_image } \gamma)$ 
  shows  $\text{norm } (\text{winding\_number } \gamma \ z) = 1$ 
proof -
  have pathfinish  $\gamma = \text{pathstart } \gamma$ 
  using assms inside_simple_curve_imp_closed by blast
  with assms integer_winding_number have winding_number  $\gamma \ z \in \mathbb{Z}$ 
  by (simp add: inside_def simple_path_def)
  then show ?thesis
  by (metis assms norm_minus_cancel norm_one simple_closed_path_winding_number_inside)
qed

lemma simple_closed_path_winding_number_cases:
  assumes simple_path  $\gamma$  pathfinish  $\gamma = \text{pathstart } \gamma$   $z \notin \text{path\_image } \gamma$ 
  shows winding_number  $\gamma \ z \in \{-1, 0, 1\}$ 
proof -
  consider  $z \in \text{inside } (\text{path\_image } \gamma) \mid z \in \text{outside } (\text{path\_image } \gamma)$ 
  by (metis ComplI UnE assms(3) inside_Un_outside)
  then show ?thesis
  proof cases
    case 1
    then show ?thesis
    using assms simple_closed_path_winding_number_inside by auto
  next
    case 2
    then show ?thesis
    using assms simple_path_def winding_number_zero_in_outside by blast
  qed
qed

lemma simple_closed_path_winding_number_pos:
   $\llbracket \text{simple\_path } \gamma; \text{pathfinish } \gamma = \text{pathstart } \gamma; z \notin \text{path\_image } \gamma; 0 < \text{Re}(\text{winding\_number } \gamma \ z) \rrbracket$ 
 $\implies \text{winding\_number } \gamma \ z = 1$ 
using simple_closed_path_winding_number_cases
  by fastforce

```

lemma *simple_closed_path_winding_number_neg*:
 $\llbracket \text{simple_path } \gamma; \text{pathfinish } \gamma = \text{pathstart } \gamma; z \notin \text{path_image } \gamma; \text{Re } (\text{winding_number } \gamma \ z) < 0 \rrbracket$
 $\implies \text{winding_number } \gamma \ z = -1$
using *simple_closed_path_winding_number_cases* **by** *fastforce*

3.9 Winding number for rectangular paths

proposition *winding_number_rectpath*:
assumes $z \in \text{box } a1 \ a3$
shows $\text{winding_number } (\text{rectpath } a1 \ a3) \ z = 1$
proof –
from *assms* **have** *less*: $\text{Re } a1 < \text{Re } a3 \ \text{Im } a1 < \text{Im } a3$
by (*auto simp: in_box_complex_iff*)
define $a2 \ a4$ **where** $a2 = \text{Complex } (\text{Re } a3) (\text{Im } a1)$ **and** $a4 = \text{Complex } (\text{Re } a1) (\text{Im } a3)$
let $?l1 = \text{linepath } a1 \ a2$ **and** $?l2 = \text{linepath } a2 \ a3$
and $?l3 = \text{linepath } a3 \ a4$ **and** $?l4 = \text{linepath } a4 \ a1$
from *assms* **and** *less* **have** $z \notin \text{path_image } (\text{rectpath } a1 \ a3)$
by (*auto simp: path_image_rectpath_cbox_minus_box*)
also have $\text{path_image } (\text{rectpath } a1 \ a3) =$
 $\text{path_image } ?l1 \cup \text{path_image } ?l2 \cup \text{path_image } ?l3 \cup \text{path_image } ?l4$
by (*simp add: rectpath_def Let_def path_image_join Un_assoc a2_def a4_def*)
finally have $z \notin \dots$
moreover have $\forall l \in \{?l1, ?l2, ?l3, ?l4\}. \text{Re } (\text{winding_number } l \ z) > 0$
unfolding *ball_simps HOL.simp_thms a2_def a4_def*
by (*intro conjI; (rule winding_number_linepath_pos_lt;*
(insert assms, auto simp: a2_def a4_def in_box_complex_iff mult_neg_neg))+)
ultimately have $\text{Re } (\text{winding_number } (\text{rectpath } a1 \ a3) \ z) > 0$
by (*simp add: winding_number_join path_image_join rectpath_def Let_def a2_def a4_def*)
thus $\text{winding_number } (\text{rectpath } a1 \ a3) \ z = 1$ **using** *assms less*
by (*intro simple_closed_path_winding_number_pos simple_path_rectpath*)
(auto simp: path_image_rectpath_cbox_minus_box)
qed

proposition *winding_number_rectpath_outside*:
assumes $\text{Re } a1 \leq \text{Re } a3 \ \text{Im } a1 \leq \text{Im } a3$
assumes $z \notin \text{cbox } a1 \ a3$
shows $\text{winding_number } (\text{rectpath } a1 \ a3) \ z = 0$
using *assms* **by** (*intro winding_number_zero_outside[OF _ _ _ assms(3)]*
 $\text{path_image_rectpath_subset_cbox}$ *simp_all*)

A per-function version for continuous logs, a kind of monodromy

proposition *winding_number_compose_exp*:
assumes *path p*
shows $\text{winding_number } (\text{exp } \circ p) \ 0 = (\text{pathfinish } p - \text{pathstart } p) / (2 * \text{of_real } \pi * i)$

```

proof -
  obtain e where 0 < e and e:  $\bigwedge t. t \in \{0..1\} \implies e \leq \text{norm}(\text{exp}(p\ t))$ 
  proof
    have closed (path_image (exp  $\circ$  p))
      by (simp add: assms closed_path_image holomorphic_on_exp holomorphic_on_imp_continuous_on path_continuous_image)
    then show 0 < setdist {0} (path_image (exp  $\circ$  p))
      by (metis exp_not_eq_zero imageE image_comp infdist_eq_setdist infdist_pos_not_in_closed path_defs(4) path_image_nonempty)
  next
    fix t::real
    assume t  $\in$  {0..1}
    have setdist {0} (path_image (exp  $\circ$  p))  $\leq$  dist 0 (exp (p t))
    proof (rule setdist_le_dist)
      show exp (p t)  $\in$  path_image (exp  $\circ$  p)
        using  $\langle t \in \{0..1\} \rangle$  path_image_def by fastforce+
      qed auto
    then show setdist {0} (path_image (exp  $\circ$  p))  $\leq$  cmod (exp (p t))
      by simp
  qed
  have bounded (path_image p)
    by (simp add: assms bounded_path_image)
  then obtain B where 0 < B and B: path_image p  $\subseteq$  cball 0 B
    by (meson bounded_pos mem_cball_0 subsetI)
  let ?B = cball (0::complex) (B+1)
  have uniformly_continuous_on ?B exp
    using holomorphic_on_exp holomorphic_on_imp_continuous_on
    by (force intro: compact_uniformly_continuous)
  then obtain d where d > 0
    and d:  $\bigwedge x\ x'. \llbracket x \in ?B; x' \in ?B; \text{dist } x' x < d \rrbracket \implies \text{norm}(\text{exp } x' - \text{exp } x) < e$ 
    using  $\langle e > 0 \rangle$  by (auto simp: uniformly_continuous_on_def dist_norm)
  then have min 1 d > 0
    by force
  then obtain g where pfg: polynomial_function g and g 0 = p 0 g 1 = p 1
    and gless:  $\bigwedge t. t \in \{0..1\} \implies \text{norm}(g\ t - p\ t) < \min 1\ d$ 
    using path_approx_polynomial_function [OF  $\langle \text{path } p \rangle$ ]  $\langle d > 0 \rangle$   $\langle 0 < e \rangle$ 
    unfolding pathfinish_def pathstart_def by blast
  have winding_number (exp  $\circ$  p) 0 = winding_number (exp  $\circ$  g) 0
  proof (rule winding_number_nearby_paths_eq [symmetric])
    show path (exp  $\circ$  p) path (exp  $\circ$  g)
      by (simp_all add: pfg assms holomorphic_on_exp holomorphic_on_imp_continuous_on path_continuous_image path_polynomial_function)
  next
    fix t :: real
    assume t: t  $\in$  {0..1}
    with gless have norm(g t - p t) < 1
      using min_less_iff_conj by blast
    moreover have ptB: norm (p t)  $\leq$  B
      using B t by (force simp: path_image_def)

```

```

ultimately have  $\text{cmod } (g \ t) \leq B + 1$ 
  by (meson add_mono_thms_linordered_field(4) le_less_trans less_imp_le
norm_triangle_sub)
with ptB gless t have  $\text{cmod } ((\exp \circ g) \ t - (\exp \circ p) \ t) < e$ 
  by (auto simp: dist_norm d)
with e t show  $\text{cmod } ((\exp \circ g) \ t - (\exp \circ p) \ t) < \text{cmod } ((\exp \circ p) \ t - 0)$ 
  by fastforce
qed (use ⟨g 0 = p 0⟩ ⟨g 1 = p 1⟩ in ⟨auto simp: pathfinish_def pathstart_def⟩)
also have  $\dots = 1 / (\text{of\_real } (2 * \pi) * i) * \text{contour\_integral } (\exp \circ g) (\lambda w. 1 / (w - 0))$ 
  proof (rule winding_number_valid_path)
    have continuous_on (path_image g) (deriv exp)
      by (metis DERIV_exp DERIV_imp_deriv continuous_on_cong holomorphic_on_exp holomorphic_on_imp_continuous_on)
    then show valid_path (exp ∘ g)
      by (simp add: field_differentiable_within_exp pfg_valid_path_compose valid_path_polynomial_function)
    show  $0 \notin \text{path\_image } (\exp \circ g)$ 
      by (auto simp: path_image_def)
  qed
also have  $\dots = 1 / (\text{of\_real } (2 * \pi) * i) * \text{integral } \{0..1\} (\lambda x. \text{vector\_derivative } g \text{ (at } x))$ 
  proof (simp add: contour_integral_integral, rule integral_cong)
    fix t :: real
    assume t:  $t \in \{0..1\}$ 
    show  $\text{vector\_derivative } (\exp \circ g) \text{ (at } t) / \exp (g \ t) = \text{vector\_derivative } g \text{ (at } t)$ 
      proof -
        have  $(\exp \circ g) \text{ has\_vector\_derivative } \text{vector\_derivative } (\exp \circ g) \text{ (at } t) \text{ (at } t)$ 
          by (meson DERIV_exp differentiable_def field_vector_diff_chain_at has_vector_derivative_def
has_vector_derivative_polynomial_function pfg_vector_derivative_works)
        moreover have  $(\exp \circ g) \text{ has\_vector\_derivative } \text{vector\_derivative } g \text{ (at } t) * \exp (g \ t) \text{ (at } t)$ 
          by (metis DERIV_exp field_vector_diff_chain_at has_vector_derivative_polynomial_function pfg_vector_derivative_at)
        ultimately show ?thesis
          by (simp add: divide_simps, rule vector_derivative_unique_at)
      qed
  qed
also have  $\dots = (\text{pathfinish } p - \text{pathstart } p) / (2 * \text{of\_real } \pi * i)$ 
  proof -
    have  $(\lambda x. \text{vector\_derivative } g \text{ (at } x)) \text{ has\_integral } g \ 1 - g \ 0 \ \{0..1\}$ 
      by (meson differentiable_at_polynomial_function fundamental_theorem_of_calculus
has_vector_derivative_at_within pfg_vector_derivative_works
zero_le_one)
    then show ?thesis
      unfolding pathfinish_def pathstart_def
      using ⟨g 0 = p 0⟩ ⟨g 1 = p 1⟩ by auto
  qed

```

```

    finally show ?thesis .
qed

end

```

4 Cauchy's Integral Formula

```

theory Cauchy_Integral_Formula
  imports Winding_Numbers
begin

```

4.1 Proof

```

lemma Cauchy_integral_formula_weak:
  assumes S: convex S and finite k and conf: continuous_on S f
    and fcd: ( $\bigwedge x. x \in \text{interior } S - k \implies f \text{ field\_differentiable at } x$ )
    and z:  $z \in \text{interior } S - k$  and vpg: valid_path  $\gamma$ 
    and pasz: path_image  $\gamma \subseteq S - \{z\}$  and loop: pathfinish  $\gamma = \text{pathstart } \gamma$ 
  shows (( $\lambda w. f w / (w - z)$ ) has_contour_integral ( $2 * \pi * i * \text{winding\_number}$ 
 $\gamma z * f z$ ))  $\gamma$ 
proof -
  let ?fz =  $\lambda w. (f w - f z) / (w - z)$ 
  obtain f' where f': (f has_field_derivative f') (at z)
    using fcd [OF z] by (auto simp: field_differentiable_def)
  have pas: path_image  $\gamma \subseteq S$  and znotin:  $z \notin \text{path\_image } \gamma$  using pasz by
blast+
  have c: continuous (at x within S) ( $\lambda w. \text{if } w = z \text{ then } f' \text{ else } (f w - f z) / (w - z)$ )
if  $x \in S$  for x
  proof (cases  $x = z$ )
    case True then show ?thesis
      using LIM_equal [of z ?fz  $\lambda w. \text{if } w = z \text{ then } f' \text{ else } ?fz w$ ] has_field_derivativeD
[OF f']
      by (force simp add: continuous_within Lim_at_imp_Lim_at_within)
    next
      case False
      then have dxz: dist x z > 0 by auto
      have cf: continuous (at x within S) f
        using conf continuous_on_eq_continuous_within that by blast
      have continuous (at x within S) ( $\lambda w. (f w - f z) / (w - z)$ )
        by (rule cf continuous_intros | simp add: False)+
      then show ?thesis
        using continuous_transform_within [OF _ dxz that] by (force simp: dist_commute)
  qed
  have fink': finite (insert z k) using ⟨finite k⟩ by blast
  have *: (( $\lambda w. \text{if } w = z \text{ then } f' \text{ else } ?fz w$ ) has_contour_integral 0)  $\gamma$ 
proof (rule Cauchy_theorem_convex [OF _ S fink' _ vpg pas loop])
  show ( $\lambda w. \text{if } w = z \text{ then } f' \text{ else } ?fz w$ ) field_differentiable at w
    if  $w \in \text{interior } S - \text{insert } z k$  for w
  proof (rule field_differentiable_transform_within)

```



```

    show ( $\lambda w. ?fz\ w$ ) field_differentiable at w
    using that by (intro derivative_intros fcd; simp)
  qed (use that in <auto simp add: dist_pos_lt dist_commute>)
  qed (use c in <force simp: continuous_on_eq continuous_within>)
  note ** = has_contour_integral_add [OF has_contour_integral_lmul [OF has_contour_integral_winding_
[OF vpg znotin], of f z] *]
  show ?thesis
  apply (rule has_contour_integral_eq)
  using znotin ** apply (auto simp: ac_simps divide_simps)
  done
qed

```

```

theorem Cauchy_integral_formula_convex_simple:
  assumes convex S and holf: f holomorphic_on S and z  $\in$  interior S valid_path
   $\gamma$  path_image  $\gamma \subseteq S - \{z\}$ 
    pathfinish  $\gamma = \text{pathstart } \gamma$ 
  shows (( $\lambda w. f\ w / (w - z)$ ) has_contour_integral (2 * pi * i * winding_number
 $\gamma\ z * f\ z$ ))  $\gamma$ 
proof -
  have  $\bigwedge x. x \in \text{interior } S \implies f$  field_differentiable at x
  using holf at_within_interior holomorphic_onD interior_subset by fastforce
  then show ?thesis
  using assms
  by (intro Cauchy_integral_formula_weak [where k = {}]) (auto simp: holo-
morphic_on_imp_continuous_on)
qed

```

Hence the Cauchy formula for points inside a circle.

```

theorem Cauchy_integral_circlepath:
  assumes conf: continuous_on (cball z r) f and holf: f holomorphic_on (ball z
r) and wz: norm(w - z) < r
  shows (( $\lambda u. f\ u / (u - w)$ ) has_contour_integral (2 * of_real pi * i * f w))
    (circlepath z r)
proof -
  have r > 0
  using assms le_less_trans norm_ge_zero by blast
  have (( $\lambda u. f\ u / (u - w)$ ) has_contour_integral (2 * pi) * i * winding_number
(circlepath z r) w * f w)
    (circlepath z r)
  proof (rule Cauchy_integral_formula_weak [where S = cball z r and k = {}])
    show  $\bigwedge x. x \in \text{interior } (\text{cball } z\ r) - \{w\} \implies$ 
      f field_differentiable at x
    using holf holomorphic_on_imp_differentiable_at by auto
    have w  $\notin$  sphere z r
    by simp (metis dist_commute dist_norm not_le order_refl wz)
    then show path_image (circlepath z r)  $\subseteq \text{cball } z\ r - \{w\}$ 
    using <r > 0> by (auto simp add: cball_def sphere_def)
  qed (use wz in <simp_all add: dist_norm norm_minus_commute conf>)
  then show ?thesis

```

by (simp add: winding_number_circlepath assms)
qed

corollary *Cauchy_integral_circlepath_simple:*

assumes f holomorphic_on cball z r $\text{norm}(w-z) < r$
shows $((\lambda u. f\ u)/(u-w))$ has_contour_integral $(2 * \text{of_real } \pi * i * f\ w)$
(circlepath z r)
using assms by (force simp: holomorphic_on_imp_continuous_on holomorphic_on_subset
Cauchy_integral_circlepath)

4.2 General stepping result for derivative formulas

lemma *Cauchy_next_derivative:*

fixes $f' :: \text{complex} \Rightarrow \text{complex}$
defines $h \equiv \lambda k\ w\ u. f'\ u / (u-w) \wedge^k$
assumes continuous_on (path_image γ) f'
and $leB: \bigwedge t. t \in \{0..1\} \implies \text{norm}(\text{vector_derivative } \gamma\ (at\ t)) \leq B$
and $int: \bigwedge w. w \in S - \text{path_image } \gamma \implies (h\ k\ w\ \text{has_contour_integral } f\ w)\ \gamma$
and $k: k \neq 0$
and open S
and $\gamma: \text{valid_path } \gamma$
and $w: w \in S - \text{path_image } \gamma$
shows $h\ (\text{Suc } k)\ w$ contour_integrable_on γ
and $(f\ \text{has_field_derivative } (k * \text{contour_integral } \gamma\ (h\ (\text{Suc } k)\ w)))\ (at\ w)$
(is ?thes2)
proof –
have open $(S - \text{path_image } \gamma)$ using $\langle \text{open } S \rangle$ closed_valid_path_image γ by
blast
then obtain d where $d > 0$ and $d: \text{ball } w\ d \subseteq S - \text{path_image } \gamma$ using w
using open_contains_ball by blast
have [simp]: $\bigwedge n. \text{cmod } (1 + \text{of_nat } n) = 1 + \text{of_nat } n$
by (metis norm_of_nat of_nat_Suc)
have $cint: (\lambda z. (h\ k\ x\ z - h\ k\ w\ z) / (x * k - w * k))$ contour_integrable_on γ
if $x \neq w$ $\text{cmod } (x-w) < d$ for $x::\text{complex}$
proof –
have $x \in S - \text{path_image } \gamma$
by (metis d dist_commute dist_norm mem_ball subsetD that(2))
then show ?thesis
using contour_integrable_diff contour_integrable_div contour_integrable_on_def
 $int\ w$
by meson
qed
then have $1: \forall_F\ x\ \text{in } at\ w. (\lambda z. (h\ k\ x\ z - h\ k\ w\ z) / (x-w) / \text{of_nat } k)$
contour_integrable_on γ
unfolding eventually_at
by (force intro: exI [where $x=d$] simp add: $\langle d > 0 \rangle$ dist_norm field_simps)
have $bim_g: \text{bounded } (\text{image } f'\ (\text{path_image } \gamma))$
by (simp add: compact_imp_bounded compact_continuous_image compact_valid_path_image
assms)

```

then obtain  $C$  where  $C > 0$  and  $C: \bigwedge x. \llbracket 0 \leq x; x \leq 1 \rrbracket \implies \text{cmod } (f'(\gamma x)) \leq C$ 
by (force simp: bounded_pos path_image_def)
have twom:  $\forall_F n$  in at  $w$ .
       $\forall x \in \text{path\_image } \gamma.$ 
       $\text{cmod } ((\text{inverse } (x-n) \wedge^k - \text{inverse } (x-w) \wedge^k) / (n-w) / k -$ 
 $\text{inverse } (x-w) \wedge^{\text{Suc } k}) < e$ 
      if  $0 < e$  for  $e$ 
proof -
      have *:  $\text{cmod } ((\text{inverse } (x-u) \wedge^k - \text{inverse } (x-w) \wedge^k) / ((u-w) * k) -$ 
 $\text{inverse } (x-w) \wedge^{\text{Suc } k}) < e$ 
      if  $x: x \in \text{path\_image } \gamma$  and  $u \neq w$  and  $uwd: \text{cmod } (u-w) < d/2$ 
      and  $uw\_less: \text{cmod } (u-w) < e * (d/2) \wedge^{(k+2)} / (1 + \text{real } k)$ 
      for  $u x$ 
proof -
      define  $ff$  where [abs_def]:
       $ff\ n\ w =$ 
      (if  $n = 0$  then  $\text{inverse}(x-w) \wedge^k$ 
      else if  $n = 1$  then  $k / (x-w) \wedge^{\text{Suc } k}$ 
      else  $(k * \text{of\_real}(\text{Suc } k)) / (x-w) \wedge^{(k+2)})$ ) for  $n :: \text{nat}$  and  $w$ 
have km1:  $\bigwedge z :: \text{complex}. z \neq 0 \implies z \wedge^{(k - \text{Suc } 0)} = z \wedge^k / z$ 
by (simp add: field_simps) (metis Suc_pred <k ≠ 0> neq0_conv power_Suc)
have ff1: ( $ff\ i$  has field_derivative  $ff\ (\text{Suc } i)\ z$ ) (at  $z$  within ball  $w\ (d/2)$ )
      if  $z \in \text{ball } w\ (d/2)$   $i \leq 1$  for  $i z$ 
proof -
      have  $z \notin \text{path\_image } \gamma$ 
      using  $\langle x \in \text{path\_image } \gamma \rangle\ d$  that ball_divide_subset_numeral by blast
      then have  $xz[\text{simp}]: x \neq z$  using  $\langle x \in \text{path\_image } \gamma \rangle$  by blast
      then have  $\text{neg}: x * x + z * z \neq x * (z * 2)$ 
      by (blast intro: dest!: sum_sqs_eq)
      with  $xz$  have  $\bigwedge v. v \neq 0 \implies (x * x + z * z) * v \neq (x * (z * 2)) * v$  by
 $\text{auto}$ 
      then have  $\text{negq}: \bigwedge v. v \neq 0 \implies x * (x * v) + z * (z * v) \neq x * (z * (2 * v))$ 
      by (simp add: algebra_simps)
      show ?thesis using  $\langle i \leq 1 \rangle$ 
      apply (simp add: ff_def dist_norm Nat.le_Suc_eq, safe)
      apply (rule derivative_eq_intros | simp add: km1 | simp add: field_simps
 $\text{neg negq})+$ 
      done
qed
{ fix  $a :: \text{real}$  and  $b :: \text{real}$  assume  $ab: a > 0\ b > 0$ 
      then have  $k * (1 + \text{real } k) * (1 / a) \leq k * (1 + \text{real } k) * (4 / b) \longleftrightarrow b$ 
 $\leq 4 * a$ 
      by (subst mult_le_cancel_left_pos)
      (use  $\langle k \neq 0 \rangle$  in  $\langle \text{auto simp: divide_simps} \rangle$ )
      with  $ab$  have  $\text{real } k * (1 + \text{real } k) / a \leq (\text{real } k * 4 + \text{real } k * \text{real } k * 4)$ 
 $/ b \longleftrightarrow b \leq 4 * a$ 
      by (simp add: field_simps)

```

```

} note canc = this
have ff2:  $\text{cmod } (\text{ff } (\text{Suc } 1) \ v) \leq \text{real } (k * (k + 1)) / (d/2) ^{(k + 2)}$ 
  if  $v \in \text{ball } w \ (d/2)$  for  $v$ 
proof -
  have lessd:  $\bigwedge z. \text{cmod } (\gamma \ z - v) < d/2 \implies \text{cmod } (w - \gamma \ z) < d$ 
  by (metis that norm_minus_commute norm_triangle_half_r dist_norm
mem_ball)
  have  $d/2 \leq \text{cmod } (x-v)$  using  $d \ x$  that
  using lessd  $d \ x$  unfolding path_image_def
  by (smt (verit, best) dist_norm imageE insert_Diff mem_ball subset_Diff_insert)
  then have  $d \leq \text{cmod } (x-v) * 2$ 
  by (simp add: field_split_simps)
  then have dpow_le:  $d ^{(k+2)} \leq (\text{cmod } (x-v) * 2) ^{(k+2)}$ 
  using  $\langle 0 < d \rangle$  order_less_imp_le power_mono by blast
  have  $x \neq v$  using that
  using  $\langle x \in \text{path\_image } \gamma \rangle$  ball_divide_subset_numeral  $d$  by fastforce
  then show ?thesis
  using  $\langle d > 0 \rangle$  apply (simp add: ff_def norm_mult norm_divide norm_power
dist_norm canc)
  using dpow_le apply (simp add: field_split_simps)
  done
qed
have ub:  $u \in \text{ball } w \ (d/2)$ 
  using uwd by (simp add: dist_commute dist_norm)
  have  $\text{cmod } (\text{inverse } (x-u) ^k - (\text{inverse } (x-w) ^k + \text{of\_nat } k * (u-w) /$ 
 $((x-w) * (x-w) ^k)))$ 
 $\leq (\text{real } k * 4 + \text{real } k * \text{real } k * 4) * (\text{cmod } (u-w) * \text{cmod } (u-w))$ 
 $/ (d * (d * (d/2) ^k))$ 
  using complex_Taylor [OF _ ff1 ff2 _ ub, of  $w$ , simplified]
  by (simp add: ff_def  $\langle 0 < d \rangle$ )
  then have  $\text{cmod } (\text{inverse } (x-u) ^k - (\text{inverse } (x-w) ^k + \text{of\_nat } k *$ 
 $(u-w) / ((x-w) * (x-w) ^k)))$ 
 $\leq (\text{cmod } (u-w) * \text{real } k) * (1 + \text{real } k) * \text{cmod } (u-w) / (d/2) ^{(k+2)}$ 
  by (simp add: field_simps)
  then have  $\text{cmod } (\text{inverse } (x-u) ^k - (\text{inverse } (x-w) ^k + \text{of\_nat } k *$ 
 $(u-w) / ((x-w) * (x-w) ^k)))$ 
 $/ (\text{cmod } (u-w) * \text{real } k)$ 
 $\leq (1 + \text{real } k) * \text{cmod } (u-w) / (d/2) ^{(k+2)}$ 
  using  $\langle k \neq 0 \rangle \ \langle u \neq w \rangle$  by (simp add: mult_ac zero_less_mult_iff
pos_divide_le_eq)
  also have  $\dots < e$ 
  using uw_less  $\langle 0 < d \rangle$  by (simp add: mult_ac divide_simps)
  finally have  $e: \text{cmod } (\text{inverse } (x-u) ^k - (\text{inverse } (x-w) ^k + \text{of\_nat } k *$ 
 $(u-w) / ((x-w) * (x-w) ^k)))$ 
 $/ \text{cmod } ((u-w) * \text{real } k) < e$ 
  by (simp add: norm_mult)
  have  $x \neq u$ 

```

```

using uwd ⟨0 < d⟩ x d by (force simp: dist_norm ball_def norm_minus_commute)
show ?thesis
  using ⟨k ≠ 0⟩ ⟨x ≠ u⟩ ⟨u ≠ w⟩ le_less_trans [OF _ e]
  by (simp add: field_simps flip: norm_divide)
qed
show ?thesis
  unfolding eventually_at
  apply (rule_tac x = min (d/2) ((e*(d/2)^(k+2))/(Suc k)) in exI)
  apply (force simp: ⟨d > 0⟩ dist_norm that simp del: power_Suc intro: *)
  done
qed
have 2: uniform_limit (path_image γ) (λx z. (h k x z - h k w z) / (x-w) / k)
(h (Suc k) w) (at w)
  unfolding uniform_limit_iff dist_norm
proof clarify
  fix e::real
  assume 0 < e
  have *: cmod ((h k x (γ u) - h k w (γ u)) / ((x-w) * k) - h (Suc k) w (γ
u)) < e
    if ec: cmod ((inverse (γ u - x) ^ k - inverse (γ u - w) ^ k) / ((x-w) *
k) -
      inverse (γ u - w) * inverse (γ u - w) ^ k) < e / C
    and x: 0 ≤ u u ≤ 1
    for x u
  proof (cases (f' (γ u)) = 0)
    case True then show ?thesis by (simp add: ⟨0 < e⟩ h_def)
  next
    case False
    have cmod ((h k x (γ u) - h k w (γ u)) / ((x-w) * k) - h (Suc k) w (γ u))
=
      cmod (f' (γ u) * ((inverse (γ u - x) ^ k - inverse (γ u - w) ^ k) /
((x-w) * k) -
        inverse (γ u - w) * inverse (γ u - w) ^ k))
    by (simp add: h_def field_simps)
    also have ... = cmod (f' (γ u)) *
      cmod ((inverse (γ u - x) ^ k - inverse (γ u - w) ^ k) / ((x-w)
* k) -
        inverse (γ u - w) * inverse (γ u - w) ^ k)
    by (simp add: norm_mult)
    also have ... < cmod (f' (γ u)) * (e/C)
    using False mult_strict_left_mono [OF ec] by force
    also have ... ≤ e using C
    by (metis False ⟨0 < e⟩ frac_le less_eq_real_def mult.commute pos_le_divide_eq
x zero_less_norm_iff)
    finally show ?thesis .
  qed
show ∀_F u in at w.
  ∀ x∈path_image γ.
    cmod ((h k u x - h k w x) / (u-w) / of_nat k - h (Suc k) w x) < e

```

```

    using twom [OF divide_pos_pos [OF ‹0 < e› ‹C > 0›]] *
    unfolding path_image_def h_def
    by (force elim: eventually_mono)
qed
show h (Suc k) w contour_integrable_on γ
  using contour_integral_uniform_limit [OF 1 2 leB γ] by (simp add: h_def)
have *: (λu. contour_integral γ (λx. (h k u x - h k w x) / (u-w) / k))
  -w→ contour_integral γ (h (Suc k) w)
  by (rule contour_integral_uniform_limit [OF 1 2 leB γ]) auto
have **: contour_integral γ (λx. (h k u x - h k w x) / ((u-w) * k)) =
  (f u - f w) / (u-w) / k
  if dist u w < d for u
proof -
  have u ∈ S - path_image γ
    by (metis subsetD d dist_commute mem_ball that)
  then have (h k u has_contour_integral f u) γ (h k w has_contour_integral f
w) γ
    using w by (simp_all add: field_simps int)
  then show ?thesis
    using contour_integral_unique has_contour_integral_diff
    has_contour_integral_div by force
qed
show ?thes2
  unfolding has_field_derivative_iff
  by (simp add: ‹k ≠ 0› ** Lim_transform_within [OF tendsto_mult_left [OF
*] ‹0 < d›])
qed

lemma Cauchy_next_derivative_circlepath:
  assumes contf: continuous_on (path_image (circlepath z r)) f
  and int: ∧w. w ∈ ball z r ⇒ ((λu. f u / (u-w) ^ k) has_contour_integral g
w) (circlepath z r)
  and k: k ≠ 0
  and w: w ∈ ball z r
  shows (λu. f u / (u-w) ^ (Suc k)) contour_integrable_on (circlepath z r)
    (is ?thes1)
  and (g has_field_derivative (k * contour_integral (circlepath z r) (λu. f
u/(u-w) ^ (Suc k)))) (at w)
    (is ?thes2)
proof -
  have r > 0 using w
  using ball_eq_empty by fastforce
  have wim: w ∈ ball z r - path_image (circlepath z r)
  using w by (auto simp: dist_norm)
  show ?thes1 ?thes2
  by (rule Cauchy_next_derivative [OF contf int k open_ball valid_path_circlepath
wim, where B = 2 * pi * |r|];
    auto simp: vector_derivative_circlepath norm_mult)+
qed

```

In particular, the first derivative formula.

```

lemma Cauchy_derivative_integral_circlepath:
  assumes conf: continuous_on (cball z r) f
    and holf: f holomorphic_on ball z r
    and w: w ∈ ball z r
  shows (λu. f u / (u-w)^2) contour_integrable_on (circlepath z r)
    (is ?thes1)
    and (f has_field_derivative (1 / (2 * of_real pi * i) * contour_integral(circlepath
  z r) (λu. f u / (u-w)^2))) (at w)
    (is ?thes2)
proof -
  have [simp]: r ≥ 0 using w
    using ball_eq_empty by fastforce
  have f: continuous_on (path_image (circlepath z r)) f
    by (rule continuous_on_subset [OF conf]) (force simp: cball_def sphere_def)
  have int: ∧w. dist z w < r ⇒
    ((λu. f u / (u-w)) has_contour_integral (λx. 2 * of_real pi * i * f
  x) w) (circlepath z r)
    by (rule Cauchy_integral_circlepath [OF conf holf]) (simp add: dist_norm
  norm_minus_commute)
  show ?thes1
    unfolding power2_eq_square
    using int Cauchy_next_derivative_circlepath [OF f _ w, where k=1]
    by fastforce
  have ((λx. 2 * of_real pi * i * f x) has_field_derivative contour_integral
  (circlepath z r) (λu. f u / (u-w)^2)) (at w)
    unfolding power2_eq_square
    using int Cauchy_next_derivative_circlepath [OF f _ w, where k=1 and
  g = λx. 2 * of_real pi * i * f x]
    by fastforce
  then have fder: (f has_field_derivative contour_integral (circlepath z r) (λu. f
  u / (u-w)^2) / (2 * of_real pi * i)) (at w)
    by (rule DERIV_cdivide [where f = λx. 2 * of_real pi * i * f x and c = 2 *
  of_real pi * i, simplified])
  show ?thes2
    by simp (rule fder)
qed

```

4.3 Existence of all higher derivatives

```

proposition derivative_is_holomorphic:
  assumes open S
    and fder: ∧z. z ∈ S ⇒ (f has_field_derivative f' z) (at z)
  shows f' holomorphic_on S
proof -
  have *: ∃ h. (f' has_field_derivative h) (at z) if z ∈ S for z
proof -
  obtain r where r > 0 and r: cball z r ⊆ S
    using open_contains_cball ⟨z ∈ S⟩ ⟨open S⟩ by blast

```

```

then have holf_cball:  $f$  holomorphic_on cball  $z$   $r$ 
  unfolding holomorphic_on_def
  using field_differentiable_at_within field_differentiable_def fder by fastforce
then have continuous_on (path_image (circlepath  $z$   $r$ ))  $f$ 
  using  $\langle r > 0 \rangle$  by (force elim: holomorphic_on_subset [THEN holomorphic_on_imp_continuous_on])
then have contfpi: continuous_on (path_image (circlepath  $z$   $r$ ))  $(\lambda x. 1/(2 * \text{of\_real } \pi * i) * f x)$ 
  by (auto intro: continuous_intros)+
have contf_cball: continuous_on (cball  $z$   $r$ )  $f$  using holf_cball
by (simp add: holomorphic_on_imp_continuous_on holomorphic_on_subset)
have holf_ball:  $f$  holomorphic_on ball  $z$   $r$  using holf_cball
  using ball_subset_cball holomorphic_on_subset by blast
{ fix  $w$  assume  $w: w \in \text{ball } z \text{ } r$ 
  have intf:  $(\lambda u. f u / (u-w)^2)$  contour_integrable_on circlepath  $z$   $r$ 
    by (blast intro:  $w$  Cauchy_derivative_integral_circlepath [OF contf_cball holf_ball])
  have fder':  $(f \text{ has\_field\_derivative } 1 / (2 * \text{of\_real } \pi * i) * \text{contour\_integral (circlepath } z \text{ } r) (\lambda u. f u / (u-w)^2))$ 
     $(\text{at } w)$ 
    by (blast intro:  $w$  Cauchy_derivative_integral_circlepath [OF contf_cball holf_ball])
  have f'_eq:  $f' w = \text{contour\_integral (circlepath } z \text{ } r) (\lambda u. f u / (u-w)^2) / (2 * \text{of\_real } \pi * i)$ 
    using fder' ball_subset_cball  $r$   $w$  by (force intro: DERIV_unique [OF fder])
  have  $((\lambda u. f u / (u-w)^2) / (2 * \text{of\_real } \pi * i)) \text{ has\_contour\_integral contour\_integral (circlepath } z \text{ } r) (\lambda u. f u / (u-w)^2) / (2 * \text{of\_real } \pi * i)$ 
     $(\text{circlepath } z \text{ } r)$ 
    by (rule has_contour_integral_div [OF has_contour_integral_integral [OF intf]])
  then have  $((\lambda u. f u / (2 * \text{of\_real } \pi * i * (u-w)^2)) \text{ has\_contour\_integral contour\_integral (circlepath } z \text{ } r) (\lambda u. f u / (u-w)^2) / (2 * \text{of\_real } \pi * i))$ 
     $(\text{circlepath } z \text{ } r)$ 
    by (simp add: algebra_simps)
  then have  $((\lambda u. f u / (2 * \text{of\_real } \pi * i * (u-w)^2)) \text{ has\_contour\_integral } f' w) (\text{circlepath } z \text{ } r)$ 
    by (simp add: f'_eq)
} note  $*$  = this
show ?thesis
  using Cauchy_next_derivative_circlepath [OF contfpi, of 2  $f$ ]  $\langle 0 < r \rangle *$ 
  using centre_in_ball mem_ball by force
qed
show ?thesis
  by (simp add: holomorphic_on_open [OF  $\langle \text{open } S \rangle *$ ])
qed

```

lemma holomorphic_deriv [holomorphic_intros]:

$\llbracket f \text{ holomorphic_on } S; \text{ open } S \rrbracket \implies (\text{deriv } f) \text{ holomorphic_on } S$
by (metis DERIV_deriv_iff_field_differentiable_at_within_open derivative_is_holomorphic holomorphic_on_def)

lemma holomorphic_deriv_compose:

assumes $g: g \text{ holomorphic_on } B$ **and** $f: f \text{ holomorphic_on } A$ **and** $f' A \subseteq B$
 $\text{open } B$
shows $(\lambda x. \text{deriv } g (f x)) \text{ holomorphic_on } A$
using holomorphic_on_compose_gen [OF $f \text{ holomorphic_deriv}$ [OF g]] *assms*
by (auto simp: o_def)

lemma analytic_deriv [analytic_intros]: $f \text{ analytic_on } S \implies (\text{deriv } f) \text{ analytic_on } S$

using analytic_on_holomorphic holomorphic_deriv **by** auto

lemma holomorphic_higher_deriv [holomorphic_intros]: $\llbracket f \text{ holomorphic_on } S; \text{ open } S \rrbracket \implies (\text{deriv } \sim^n) f \text{ holomorphic_on } S$

by (induction n) (auto simp: holomorphic_deriv)

lemma analytic_higher_deriv [analytic_intros]: $f \text{ analytic_on } S \implies (\text{deriv } \sim^n) f \text{ analytic_on } S$

unfolding analytic_on_def **using** holomorphic_higher_deriv **by** blast

lemma has_field_derivative_higher_deriv:

$\llbracket f \text{ holomorphic_on } S; \text{ open } S; x \in S \rrbracket$
 $\implies ((\text{deriv } \sim^n) f \text{ has_field_derivative } (\text{deriv } \sim^n (\text{Suc } n)) f x) (at x)$
using holomorphic_derivI holomorphic_higher_deriv **by** fastforce

lemma higher_deriv_cmult:

assumes $f \text{ holomorphic_on } A$ $x \in A$ $\text{open } A$
shows $(\text{deriv } \sim^j) (\lambda x. c * f x) x = c * (\text{deriv } \sim^j) f x$
using *assms*
proof (induction j arbitrary: $f x$)
case (Suc j $f x$)
have $\text{deriv } ((\text{deriv } \sim^j) (\lambda x. c * f x)) x = \text{deriv } (\lambda x. c * (\text{deriv } \sim^j) f x) x$
using eventually_nhds_in_open [of A x] *assms*(2,3) *Suc.prem*s
by (intro deriv_cong_ev refl) (auto elim!: eventually_mono simp: *Suc.IH*)
also have $\dots = c * \text{deriv } ((\text{deriv } \sim^j) f) x$ **using** *Suc.prem*s *assms*(2,3)
by (intro deriv_cmult holomorphic_on_imp_differentiable_at holomorphic_higher_deriv)
auto
finally show ?case **by** simp
qed simp_all

lemma higher_deriv_cmult':

assumes $f \text{ analytic_on } \{x\}$
shows $(\text{deriv } \sim^j) (\lambda x. c * f x) x = c * (\text{deriv } \sim^j) f x$
using *assms* higher_deriv_cmult [of f x j c] *assms*
using analytic_at_two **by** blast

```

lemma deriv_cmult':
  assumes f analytic_on {x}
  shows deriv ( $\lambda x. c * f x$ ) x = c * deriv f x
  using higher_deriv_cmult'[OF assms, of 1 c] by simp

```

```

lemma analytic_derivI:
  assumes f analytic_on {z}
  shows (f has_field_derivative (deriv f z)) (at z within A)
  using assms holomorphic_derivI[of f _ z] analytic_at by blast

```

```

lemma deriv_compose_analytic:
  fixes f g :: complex  $\Rightarrow$  complex
  assumes f analytic_on {g z} g analytic_on {z}
  shows deriv ( $\lambda x. f (g x)$ ) z = deriv f (g z) * deriv g z
proof -
  have ((f  $\circ$  g) has_field_derivative (deriv f (g z) * deriv g z)) (at z)
    by (intro DERIV_chain analytic_derivI assms)
  thus ?thesis
    by (auto intro!: DERIV_imp_deriv simp add: o_def)
qed

```

```

lemma valid_path_compose_holomorphic:
  assumes valid_path g f holomorphic_on S and open S path_image g  $\subseteq$  S
  shows valid_path (f  $\circ$  g)
  by (meson assms holomorphic_deriv holomorphic_on_imp_continuous_on holomorphic_on_imp_differentiable_at
    holomorphic_on_subset subsetD valid_path_compose)

```

```

lemma valid_path_compose_analytic:
  assumes valid_path g and holo:f analytic_on S and path_image g  $\subseteq$  S
  shows valid_path (f  $\circ$  g)
proof (rule valid_path_compose[OF  $\langle$ valid_path g $\rangle$ ])
  fix x assume x  $\in$  path_image g
  then show f field_differentiable at x
    using analytic_on_imp_differentiable_at analytic_on_open assms holo by blast
next
  show continuous_on (path_image g) (deriv f)
    by (intro holomorphic_on_imp_continuous_on analytic_imp_holomorphic analytic_intros
      analytic_on_subset[OF holo] assms)
qed

```

```

lemma analytic_on_deriv [analytic_intros]:
  assumes f analytic_on g ' A
  assumes g analytic_on A
  shows ( $\lambda x. deriv f (g x)$ ) analytic_on A
proof -
  have (deriv f  $\circ$  g) analytic_on A

```

```

    by (rule analytic_on_compose_gen[OF assms(2) analytic_deriv[OF assms(1)]])
  auto
  thus ?thesis
    by (simp add: o_def)
qed

lemma contour_integral_comp_analyticW:
  assumes f analytic_on s valid_path  $\gamma$  path_image  $\gamma \subseteq s$ 
  shows contour_integral (f  $\circ \gamma$ ) g = contour_integral  $\gamma$  ( $\lambda w. \text{deriv } f w * g (f w)$ )
proof -
  obtain spikes where finite_spikes and  $\gamma\_diff: \gamma \text{ C1\_differentiable\_on } \{0..1\}$ 
  - spikes
  using  $\langle \text{valid\_path } \gamma \rangle$  unfolding valid_path_def piecewise_C1_differentiable_on_def
  by auto
  show contour_integral (f  $\circ \gamma$ ) g
    = contour_integral  $\gamma$  ( $\lambda w. \text{deriv } f w * g (f w)$ )
  unfolding contour_integral_integral
  proof (rule integral_spike[rule_format, OF negligible_finite[OF  $\langle \text{finite\_spikes} \rangle$ ]])
    fix t::real assume t:t  $\in \{0..1\}$  - spikes
    then have  $\gamma$  differentiable at t
      using  $\gamma\_diff$  unfolding C1_differentiable_on_eq by auto
    moreover have f field_differentiable at ( $\gamma t$ )
  proof -
    have  $\gamma t \in s$  using t assms unfolding path_image_def by auto
    thus ?thesis
      using  $\langle f \text{ analytic\_on } s \rangle$  analytic_on_imp_differentiable_at by blast
  qed
  ultimately show  $\text{deriv } f (\gamma t) * g (f (\gamma t)) * \text{vector\_derivative } \gamma (at t) =$ 
     $g ((f \circ \gamma) t) * \text{vector\_derivative } (f \circ \gamma) (at t)$ 
  by (subst vector_derivative_chain_at_general) (simp_all add: field_simps)
qed
qed

```

4.4 Morera's theorem

```

lemma Morera_local_triangle_ball:
  assumes  $\bigwedge z. z \in S$ 
     $\implies \exists e a. 0 < e \wedge z \in \text{ball } a e \wedge \text{continuous\_on } (\text{ball } a e) f \wedge$ 
       $(\forall b c. \text{closed\_segment } b c \subseteq \text{ball } a e$ 
         $\longrightarrow \text{contour\_integral } (\text{linepath } a b) f +$ 
         $\text{contour\_integral } (\text{linepath } b c) f +$ 
         $\text{contour\_integral } (\text{linepath } c a) f = 0)$ 
  shows f analytic_on S
proof -
  { fix z assume z  $\in S$ 
    with assms obtain e a where
      0 < e and z: z  $\in \text{ball } a e$  and contf: continuous_on (ball a e) f
    and 0:  $\bigwedge b c. \text{closed\_segment } b c \subseteq \text{ball } a e$ 
       $\implies \text{contour\_integral } (\text{linepath } a b) f +$ 

```

```

      contour_integral (linepath b c) f +
      contour_integral (linepath c a) f = 0
    by blast
  have az: dist a z < e using mem_ball z by blast
  have  $\exists e > 0. f \text{ holomorphic\_on ball } z \ e$ 
  proof (intro exI conjI)
    show  $f \text{ holomorphic\_on ball } z \ (e - \text{dist } a \ z)$ 
    proof (rule holomorphic_on_subset)
      show  $\text{ball } z \ (e - \text{dist } a \ z) \subseteq \text{ball } a \ e$ 
      by (simp add: dist_commute ball_subset_ball_iff)
      have sub_ball:  $\bigwedge y. \text{dist } a \ y < e \implies \text{closed\_segment } a \ y \subseteq \text{ball } a \ e$ 
      by (meson  $\langle 0 < e \rangle$  centre_in_ball convex_ball convex_contains_segment
        mem_ball)
      show  $f \text{ holomorphic\_on ball } a \ e$ 
      using triangle_contour_integrals_starlike_primitive [OF conf_t open_ball,
        of a]
      derivative_is_holomorphic [OF open_ball]
      by (force simp add:  $0 < 0 < e$  sub_ball)
    qed
  qed (simp add: az)
}
then show ?thesis
  by (simp add: analytic_on_def)
qed

```

lemma Morera_local_triangle:

```

  assumes  $\bigwedge z. z \in S$ 
     $\implies \exists t. \text{open } t \wedge z \in t \wedge \text{continuous\_on } t \ f \wedge$ 
       $(\forall a \ b \ c. \text{convex hull } \{a, b, c\} \subseteq t$ 
         $\longrightarrow \text{contour\_integral (linepath } a \ b) \ f +$ 
           $\text{contour\_integral (linepath } b \ c) \ f +$ 
           $\text{contour\_integral (linepath } c \ a) \ f = 0)$ 
      )
  shows  $f \text{ analytic\_on } S$ 
  proof -
    { fix z assume  $z \in S$ 
      with assms obtain t where
        open t and z:  $z \in t$  and conf_t:  $\text{continuous\_on } t \ f$ 
        and 0:  $\bigwedge a \ b \ c. \text{convex hull } \{a, b, c\} \subseteq t$ 
           $\implies \text{contour\_integral (linepath } a \ b) \ f +$ 
             $\text{contour\_integral (linepath } b \ c) \ f +$ 
             $\text{contour\_integral (linepath } c \ a) \ f = 0$ 

        by force
      then obtain e where  $e > 0$  and e:  $\text{ball } z \ e \subseteq t$ 
      using open_contains_ball by blast
      have [simp]:  $\text{continuous\_on (ball } z \ e) \ f$  using conf_t
      using continuous_on_subset e by blast
      have eq0:  $\bigwedge b \ c. \text{closed\_segment } b \ c \subseteq \text{ball } z \ e \implies$ 
         $\text{contour\_integral (linepath } z \ b) \ f +$ 
         $\text{contour\_integral (linepath } b \ c) \ f +$ 

```

```

      contour_integral (linepath c z) f = 0
    by (meson 0 z ⟨0 < e⟩ centre_in_ball closed_segment_subset convex_ball
dual_order.trans e starlike_convex_subset)
    have ∃ e a. 0 < e ∧ z ∈ ball a e ∧ continuous_on (ball a e) f ∧
      (∀ b c. closed_segment b c ⊆ ball a e ⟶
        contour_integral (linepath a b) f + contour_integral (linepath b
c) f + contour_integral (linepath c a) f = 0)
      using ⟨e > 0⟩ eq0 by force
  }
  then show ?thesis
    by (simp add: Morera_local_triangle_ball)
qed

```

proposition *Morera_triangle*:

```

  [[continuous_on S f; open S;
    ∧ a b c. convex hull {a,b,c} ⊆ S
    ⟶ contour_integral (linepath a b) f +
      contour_integral (linepath b c) f +
      contour_integral (linepath c a) f = 0]]
  ⟹ f analytic_on S
  using Morera_local_triangle by blast

```

4.5 Combining theorems for higher derivatives including Leibniz rule

lemma *higher_deriv_linear* [simp]:

```

  (deriv ~ n) (λw. c*w) = (λz. if n = 0 then c*z else if n = 1 then c else 0)
  by (induction n) auto

```

lemma *higher_deriv_const* [simp]: (deriv ~ n) (λw. c) = (λw. if n=0 then c else 0)

by (induction n) auto

lemma *higher_deriv_ident* [simp]:

```

  (deriv ~ n) (λw. w) z = (if n = 0 then z else if n = 1 then 1 else 0)

```

proof (induction n)

case (Suc n)

then show ?case by (metis higher_deriv_linear lambda_one)

qed auto

lemma *higher_deriv_id* [simp]:

```

  (deriv ~ n) id z = (if n = 0 then z else if n = 1 then 1 else 0)

```

by (simp add: id_def)

lemma *has_complex_derivative_funpow_1*:

```

  [(f has_field_derivative 1) (at z); f z = z] ⟹ (f ~ n has_field_derivative 1)
  (at z)

```

proof (induction n)

case 0

```

    then show ?case
      by (simp add: id_def)
  next
    case (Suc n)
    then show ?case
      by (metis DERIV_chain funpow_Suc_right mult.right_neutral)
  qed

lemma higher_deriv_uminus:
  assumes f holomorphic_on S open S and z: z ∈ S
  shows (deriv  $\hat{\sim}$  n) (λw. -(f w)) z = - ((deriv  $\hat{\sim}$  n) f z)
using z
proof (induction n arbitrary: z)
  case 0 then show ?case by simp
next
  case (Suc n z)
  have *: ((deriv  $\hat{\sim}$  n) f has_field_derivative deriv ((deriv  $\hat{\sim}$  n) f) z) (at z)
  using Suc.prem1 assms has_field_derivative_higher_deriv by auto
  have ∧x. x ∈ S ⇒ - (deriv  $\hat{\sim}$  n) f x = (deriv  $\hat{\sim}$  n) (λw. - f w) x
  by (auto simp add: Suc)
  then have ((deriv  $\hat{\sim}$  n) (λw. - f w) has_field_derivative - deriv ((deriv  $\hat{\sim}$  n)
f) z) (at z)
  using has_field_derivative_transform_within_open [of λw. -((deriv  $\hat{\sim}$  n) f
w)]
  using * DERIV_minus Suc.prem1 ⟨open S⟩ by blast
  then show ?case
    by (simp add: DERIV_imp_deriv)
qed

lemma higher_deriv_add:
  fixes z::complex
  assumes f holomorphic_on S g holomorphic_on S open S and z: z ∈ S
  shows (deriv  $\hat{\sim}$  n) (λw. f w + g w) z = (deriv  $\hat{\sim}$  n) f z + (deriv  $\hat{\sim}$  n) g z
using z
proof (induction n arbitrary: z)
  case 0 then show ?case by simp
next
  case (Suc n z)
  have *: ((deriv  $\hat{\sim}$  n) f has_field_derivative deriv ((deriv  $\hat{\sim}$  n) f) z) (at z)
    ((deriv  $\hat{\sim}$  n) g has_field_derivative deriv ((deriv  $\hat{\sim}$  n) g) z) (at z)
  using Suc.prem1 assms has_field_derivative_higher_deriv by auto
  have ∧x. x ∈ S ⇒ (deriv  $\hat{\sim}$  n) f x + (deriv  $\hat{\sim}$  n) g x = (deriv  $\hat{\sim}$  n) (λw. f
w + g w) x
  by (auto simp add: Suc)
  then have ((deriv  $\hat{\sim}$  n) (λw. f w + g w) has_field_derivative
deriv ((deriv  $\hat{\sim}$  n) f) z + deriv ((deriv  $\hat{\sim}$  n) g) z) (at z)
  using has_field_derivative_transform_within_open [of λw. (deriv  $\hat{\sim}$  n) f w
+ (deriv  $\hat{\sim}$  n) g w]
  using * Deriv.field_differentiable_add Suc.prem1 ⟨open S⟩ by blast

```

```

then show ?case
  by (simp add: DERIV_imp_deriv)
qed

```

```

lemma higher_deriv_diff:
  fixes z::complex
  assumes f holomorphic_on S g holomorphic_on S open S z ∈ S
  shows (deriv  $\hat{\sim}$  n) (λw. f w - g w) z = (deriv  $\hat{\sim}$  n) f z - (deriv  $\hat{\sim}$  n) g z
  unfolding diff_conv_add_uminus higher_deriv_add
  using assms higher_deriv_add higher_deriv_uminus holomorphic_on_minus
  by presburger

```

```

lemma Suc_choose: Suc n choose k = (n choose k) + (if k = 0 then 0 else (n
choose (k-1)))
  by (cases k) simp_all

```

```

lemma higher_deriv_mult:
  fixes z::complex
  assumes f holomorphic_on S g holomorphic_on S open S and z: z ∈ S
  shows (deriv  $\hat{\sim}$  n) (λw. f w * g w) z =
    (∑ i = 0..n. of_nat (n choose i) * (deriv  $\hat{\sim}$  i) f z * (deriv  $\hat{\sim}$  (n-i)) g z)
using z
proof (induction n arbitrary: z)
  case 0 then show ?case by simp
next
  case (Suc n z)
  have *: (∧ n. ((deriv  $\hat{\sim}$  n) f has_field_derivative deriv ((deriv  $\hat{\sim}$  n) f) z) (at z)
    ∧ n. ((deriv  $\hat{\sim}$  n) g has_field_derivative deriv ((deriv  $\hat{\sim}$  n) g) z) (at z)
  using Suc.prem1 assms has_field_derivative_higher_deriv by auto
  have sumeq: (∑ i = 0..n.
    of_nat (n choose i) * (deriv ((deriv  $\hat{\sim}$  i) f) z * (deriv  $\hat{\sim}$  (n-i)) g z
  + deriv ((deriv  $\hat{\sim}$  (n-i)) g) z * (deriv  $\hat{\sim}$  i) f z)) =
    g z * deriv ((deriv  $\hat{\sim}$  n) f) z + (∑ i = 0..n. (deriv  $\hat{\sim}$  i) f z * (of_nat
    (Suc n choose i) * (deriv  $\hat{\sim}$  (Suc n - i)) g z))
  apply (simp add: Suc_choose algebra_simps sum.distrib)
  apply (subst (4) sum_Suc_reindex)
  apply (auto simp: algebra_simps Suc_diff_le intro: sum.cong)
  done
  have ((deriv  $\hat{\sim}$  n) (λw. f w * g w) has_field_derivative
    (∑ i = 0..Suc n. (Suc n choose i) * (deriv  $\hat{\sim}$  i) f z * (deriv  $\hat{\sim}$  (Suc n -
    i)) g z))
    (at z)
  apply (rule has_field_derivative_transform_within_open
    [of λw. (∑ i = 0..n. of_nat (n choose i) * (deriv  $\hat{\sim}$  i) f w * (deriv  $\hat{\sim}$ 
    (n-i)) g w) _ _ S])
  apply (simp add: algebra_simps)
  apply (rule derivative_eq_intros | simp)+
  apply (auto intro: DERIV_mult * ⟨open S⟩ Suc.prem1 Suc.IH [symmetric])
  by (metis (no_types, lifting) mult.commute sum.cong sumeq)

```

```

then show ?case
  unfolding funpow.simps o_apply
  by (simp add: DERIV_imp_deriv)
qed

```

lemma *higher_deriv_transform_within_open*:

```

fixes z::complex
assumes f holomorphic_on S g holomorphic_on S open S and z: z ∈ S
  and fg:  $\bigwedge w. w \in S \implies f w = g w$ 
shows (deriv  $\hat{\sim}$  i) f z = (deriv  $\hat{\sim}$  i) g z
using z
by (induction i arbitrary: z)
  (auto simp: fg intro: complex_derivative_transform_within_open holomorphic_higher_deriv
  assms)

```

lemma *higher_deriv_compose_linear'*:

```

fixes z::complex
assumes f: f holomorphic_on T and S: open S and T: open T and z: z ∈ S
  and fg:  $\bigwedge w. w \in S \implies u * w + c \in T$ 
shows (deriv  $\hat{\sim}$  n) ( $\lambda w. f (u * w + c)$ ) z =  $u \hat{\sim} n * (deriv \hat{\sim} n) f (u * z + c)$ 
using z
proof (induction n arbitrary: z)
  case 0 then show ?case by simp
next
  case (Suc n z)
  have holo0: f holomorphic_on ( $\lambda w. u * w + c$ ) ' S
    by (meson fg f holomorphic_on_subset image_subset_iff)
  have holo2: (deriv  $\hat{\sim}$  n) f holomorphic_on ( $\lambda w. u * w + c$ ) ' S
    by (meson f fg holomorphic_higher_deriv holomorphic_on_subset image_subset_iff
    T)
  have holo3: ( $\lambda z. u \hat{\sim} n * (deriv \hat{\sim} n) f (u * z + c)$ ) holomorphic_on S
    by (intro holo2 holomorphic_on_compose [where g=(deriv  $\hat{\sim}$  n) f, unfolded
    o_def] holomorphic_intros)
  have ( $\lambda w. u * w + c$ ) holomorphic_on S f holomorphic_on ( $\lambda w. u * w + c$ ) ' S
    by (rule holo0 holomorphic_intros)+
  then have holo1: ( $\lambda w. f (u * w + c)$ ) holomorphic_on S
    by (rule holomorphic_on_compose [where g=f, unfolded o_def])
  have deriv ((deriv  $\hat{\sim}$  n) ( $\lambda w. f (u * w + c)$ )) z = deriv ( $\lambda z. u \hat{\sim} n * (deriv \hat{\sim} n)$ 
  f (u * z + c)) z
proof (rule complex_derivative_transform_within_open [OF _ holo3 S Suc.prem])
  show (deriv  $\hat{\sim}$  n) ( $\lambda w. f (u * w + c)$ ) holomorphic_on S
    by (rule holomorphic_higher_deriv [OF holo1 S])
qed (simp add: Suc.IH)
also have ... =  $u \hat{\sim} n * deriv (\lambda z. (deriv \hat{\sim} n) f (u * z + c)) z$ 
proof -
  have (deriv  $\hat{\sim}$  n) f analytic_on T
    by (simp add: analytic_on_open f holomorphic_higher_deriv T)
  then have ( $\lambda w. (deriv \hat{\sim} n) f (u * w + c)$ ) analytic_on S
    using holomorphic_on_compose[OF _ holo2] <( $\lambda w. u * w + c$ ) holomor-

```



```

phic_on S
  by (simp add: S analytic_on_open o_def)
  then show ?thesis
    by (intro deriv_cmult analytic_on_imp_differentiable_at [OF Suc.prem])
  qed
  also have ... = u * u ^ n * deriv ((deriv ^ n) f) (u * z + c)
  proof -
    have (deriv ^ n) f field_differentiable at (u * z + c)
    using Suc.prem T ffg holomorphic_higher_deriv holomorphic_on_imp_differentiable_at
  by blast
  then show ?thesis
    by (simp add: deriv_compose_linear')
  qed
  finally show ?case
    by simp
  qed

```

```

lemma higher_deriv_compose_linear:
  fixes z::complex
  assumes f: f holomorphic_on T and S: open S and T: open T and z: z ∈ S
    and fg:  $\bigwedge w. w \in S \implies u * w \in T$ 
  shows (deriv ^ n) ( $\lambda w. f (u * w)$ ) z =  $u^n * (deriv ^ n) f (u * z)$ 
  using higher_deriv_compose_linear' [where c=0] assms by simp

```

```

lemma higher_deriv_add_at:
  assumes f analytic_on {z} g analytic_on {z}
  shows (deriv ^ n) ( $\lambda w. f w + g w$ ) z = (deriv ^ n) f z + (deriv ^ n) g z
  using analytic_at_two assms higher_deriv_add by blast

```

```

lemma higher_deriv_diff_at:
  assumes f analytic_on {z} g analytic_on {z}
  shows (deriv ^ n) ( $\lambda w. f w - g w$ ) z = (deriv ^ n) f z - (deriv ^ n) g z
  using analytic_at_two assms higher_deriv_diff by blast

```

```

lemma higher_deriv_uminus_at:
  f analytic_on {z}  $\implies (deriv ^ n) (\lambda w. -(f w)) z = - ((deriv ^ n) f z)$ 
  using higher_deriv_uminus by (auto simp: analytic_at)

```

```

lemma higher_deriv_mult_at:
  assumes f analytic_on {z} g analytic_on {z}
  shows (deriv ^ n) ( $\lambda w. f w * g w$ ) z =
    ( $\sum i = 0..n. of\_nat (n \text{ choose } i) * (deriv ^ i) f z * (deriv ^ (n-i)) g z$ )
  using analytic_at_two assms higher_deriv_mult by blast

```

Nonexistence of isolated singularities and a stronger integral formula.

```

proposition no_isolated_singularity:
  fixes z::complex
  assumes f: continuous_on S f and holf: f holomorphic_on (S-K) and S: open
  S and K: finite K

```

```

    shows  $f$  holomorphic_on  $S$ 
  proof -
    { fix  $z$ 
      assume  $z \in S$  and  $\text{cdf}$ :  $\bigwedge x. x \in S - K \implies f \text{ field\_differentiable at } x$ 
      have  $f \text{ field\_differentiable at } z$ 
      proof (cases  $z \in K$ )
        case False then show ?thesis by (blast intro:  $\text{cdf } \langle z \in S \rangle$ )
      next
        case True
          with  $\text{finite\_set\_avoid } [OF\ K, \text{ of } z]$ 
          obtain  $d$  where  $d > 0$  and  $d$ :  $\bigwedge x. [x \in K; x \neq z] \implies d \leq \text{dist } z\ x$ 
            by blast
          obtain  $e$  where  $e > 0$  and  $e$ :  $\text{ball } z\ e \subseteq S$ 
            using  $S \langle z \in S \rangle$  by (force simp:  $\text{open\_contains\_ball}$ )
          have  $\text{fde}$ :  $\text{continuous\_on } (\text{ball } z\ (\min\ d\ e))\ f$ 
            by (metis  $\text{Int\_iff\_ball\_min\_Int continuous\_on\_subset } e\ f\ \text{subsetI}$ )
          have  $\text{cont}$ :  $\{a, b, c\} \subseteq \text{ball } z\ (\min\ d\ e) \implies \text{continuous\_on } (\text{convex hull } \{a, b, c\})\ f$ 
            for  $a\ b\ c$ 
            by (simp add:  $\text{hull\_minimal continuous\_on\_subset } [OF\ \text{fde}]$ )
          have  $\text{fd}$ :  $[\{a, b, c\} \subseteq \text{ball } z\ (\min\ d\ e); x \in \text{interior } (\text{convex hull } \{a, b, c\}) - K] \implies f \text{ field\_differentiable at } x$ 
            for  $a\ b\ c\ x$ 
            by (metis  $\text{cdf Diff\_iff\_Int\_iff\_ball\_min\_Int subsetD convex\_ball } e\ \text{interior\_mono interior\_subset subset\_hull}$ )
          obtain  $g$  where  $\bigwedge w. w \in \text{ball } z\ (\min\ d\ e) \implies (g \text{ has\_field\_derivative } f\ w)$ 
            (at  $w$  within  $\text{ball } z\ (\min\ d\ e)$ )
            apply (rule  $\text{contour\_integral\_convex\_primitive}$ 
                    [OF  $\text{convex\_ball fde Cauchy\_theorem\_triangle\_cofinite } [OF\ \_K]]$ )
            using  $\text{cont fd}$  by auto
          then have  $f \text{ holomorphic\_on } \text{ball } z\ (\min\ d\ e)$ 
            by (metis  $\text{open\_ball at\_within\_open derivative\_is\_holomorphic}$ )
          then show ?thesis
            unfolding  $\text{holomorphic\_on\_def}$ 
            by (metis  $\text{open\_ball } \langle 0 < d \rangle \langle 0 < e \rangle \text{ at\_within\_open centre\_in\_ball min\_less\_iff\_conj}$ )
        qed
      }
    with  $\text{holf } S\ K$  show ?thesis
      by (simp add:  $\text{holomorphic\_on\_open open\_Diff finite\_imp\_closed field\_differentiable\_def [symmetric]}$ )
    qed
  lemma  $\text{no\_isolated\_singularity'}$ :
    fixes  $z::\text{complex}$ 
    assumes  $f$ :  $\bigwedge z. z \in K \implies (f \longrightarrow f\ z)$  (at  $z$  within  $S$ )
      and  $\text{holf}$ :  $f \text{ holomorphic\_on } (S - K)$  and  $S$ :  $\text{open } S$  and  $K$ :  $\text{finite } K$ 
    shows  $f \text{ holomorphic\_on } S$ 
  proof (rule  $\text{no\_isolated\_singularity} [OF\ \_ \text{assms}(2-)]$ )

```

```

have continuous_on (S-K) f
  using holf holomorphic_on_imp_continuous_on by auto
then show continuous_on S f
  by (metis Diff_iff K S at_within_open continuous_on_eq_continuous_at
    continuous_within f finite_imp_closed open_Diff)
qed

```

proposition *Cauchy_integral_formula_convex:*

```

assumes S: convex S and K: finite K and conf: continuous_on S f
  and fcd: ( $\bigwedge x. x \in \text{interior } S - K \implies f \text{ field\_differentiable at } x$ )
  and z:  $z \in \text{interior } S$  and vpg: valid_path  $\gamma$ 
  and pasz: path_image  $\gamma \subseteq S - \{z\}$  and loop: pathfinish  $\gamma = \text{pathstart } \gamma$ 
shows (( $\lambda w. f w / (w-z)$ ) has_contour_integral ( $2\pi i * \text{winding\_number } \gamma$ 
 $z * f z$ ))  $\gamma$ 
proof -
  have *:  $\bigwedge x. x \in \text{interior } S \implies f \text{ field\_differentiable at } x$ 
  unfolding holomorphic_on_open [symmetric] field_differentiable_def
  using no_isolated_singularity [where  $S = \text{interior } S$ ]
  by (meson K conf continuous_on_subset fcd field_differentiable_def open_interior
    has_field_derivative_at_within holomorphic_derivI holomorphic_onI interior_subset)
  show ?thesis
    by (rule Cauchy_integral_formula_weak [OF S finite.emptyI conf]) (use *
    assms in auto)
qed

```

Formula for higher derivatives.

lemma *Cauchy_has_contour_integral_higher_derivative_circlepath:*

```

assumes conf: continuous_on (cball z r) f
  and holf: f holomorphic_on ball z r
  and w:  $w \in \text{ball } z r$ 
shows (( $\lambda u. f u / (u-w) \wedge (\text{Suc } k)$ ) has_contour_integral (( $2\pi i$ ) / (fact
 $k$ ) * (deriv  $\sim k$ ) f w))
  (circlepath z r)
using w
proof (induction k arbitrary: w)
  case 0 then show ?case
    using assms by (auto simp: Cauchy_integral_circlepath dist_commute dist_norm)
next
  case (Suc k)
  have [simp]:  $r > 0$  using w
  using ball_eq_empty by fastforce
  have f: continuous_on (path_image (circlepath z r)) f
    by (rule continuous_on_subset [OF conf]) (force simp: cball_def sphere_def
    less_imp_le)
  obtain X where X: (( $\lambda u. f u / (u-w) \wedge \text{Suc } (\text{Suc } k)$ ) has_contour_integral X)
    (circlepath z r)
  using Cauchy_next_derivative_circlepath(1) [OF f Suc.IH _ Suc.prem]
  by (auto simp: contour_integrable_on_def)

```

```

then have con: contour_integral (circlepath z r) (( $\lambda u. f u / (u-w) \wedge \text{Suc } (\text{Suc } k)$ )) = X
by (rule contour_integral_unique)
have  $\bigwedge n. ((\text{deriv } \wedge n) f \text{ has\_field\_derivative } \text{deriv } ((\text{deriv } \wedge n) f) w) \text{ (at } w)$ 
using Suc.prems assms has_field_derivative_higher_deriv by auto
then have dnf_diff:  $\bigwedge n. (\text{deriv } \wedge n) f \text{ field\_differentiable (at } w)$ 
by (force simp: field_differentiable_def)
have deriv ( $\lambda w. \text{complex\_of\_real } (2 * \pi i) * i / (\text{fact } k) * (\text{deriv } \wedge k) f w$ )  $w =$ 
 $\text{of\_nat } (\text{Suc } k) * \text{contour\_integral } (\text{circlepath } z r) (\lambda u. f u / (u-w) \wedge \text{Suc } (\text{Suc } k))$ 
by (force intro!: DERIV_imp_deriv Cauchy_next_derivative_circlepath [OF f Suc.IH _ Suc.prems])
also have ... = of_nat (Suc k) * X
by (simp only: con)
finally have deriv ( $\lambda w. ((2 * \pi i) * i / (\text{fact } k)) * (\text{deriv } \wedge k) f w$ )  $w = \text{of\_nat } (\text{Suc } k) * X$  .
then have  $((2 * \pi i) * i / (\text{fact } k)) * \text{deriv } (\lambda w. (\text{deriv } \wedge k) f w) w = \text{of\_nat } (\text{Suc } k) * X$ 
by (metis deriv_cmult dnf_diff)
then have deriv ( $\lambda w. (\text{deriv } \wedge k) f w$ )  $w = \text{of\_nat } (\text{Suc } k) * X / ((2 * \pi i) * i / (\text{fact } k))$ 
by (simp add: field_simps)
then show ?case
using of_nat_eq_0_iff X by fastforce
qed

```

lemma *Cauchy_higher_derivative_integral_circlepath*:

```

assumes contf: continuous_on (cball z r) f
and holf: f holomorphic_on ball z r
and w:  $w \in \text{ball } z r$ 
shows  $(\lambda u. f u / (u-w) \wedge (\text{Suc } k)) \text{ contour\_integrable\_on } (\text{circlepath } z r)$ 
(is ?thes1)
and  $(\text{deriv } \wedge k) f w = (\text{fact } k) / (2 * \pi i * i) * \text{contour\_integral}(\text{circlepath } z r) (\lambda u. f u / (u-w) \wedge (\text{Suc } k))$ 
(is ?thes2)
proof -
have *:  $((\lambda u. f u / (u-w) \wedge \text{Suc } k) \text{ has\_contour\_integral } (2 * \pi i) * i / (\text{fact } k) * (\text{deriv } \wedge k) f w)$ 
(circlepath z r)
using Cauchy_has_contour_integral_higher_derivative_circlepath [OF assms]
by simp
show ?thes1 using *
using contour_integrable_on_def by blast
show ?thes2
unfolding contour_integral_unique [OF *] by (simp add: field_split_simps)
qed

```

corollary *Cauchy_contour_integral_circlepath*:

assumes *continuous_on* (*cball* z r) *f* *f* *holomorphic_on* *ball* z r $w \in \text{ball } z r$

shows $\text{contour_integral}(\text{circlepath } z \ r) \ (\lambda u. f \ u / (u - w) \wedge (\text{Suc } k)) = (2 * \pi * i) * (\text{deriv } \wedge k) f \ w / (\text{fact } k)$
by (*simp add: Cauchy_higher_derivative_integral_circlepath [OF assms]*)

lemma *Cauchy_contour_integral_circlepath_2:*

assumes *continuous_on* (*cball* *z* *r*) *f* *f* *holomorphic_on* *ball* *z* *r* *w* \in *ball* *z* *r*
shows $\text{contour_integral}(\text{circlepath } z \ r) \ (\lambda u. f \ u / (u - w) \wedge 2) = (2 * \pi * i) * \text{deriv } f \ w$
using *Cauchy_contour_integral_circlepath [OF assms, of 1]*
by (*simp add: power2_eq_square*)

4.6 A holomorphic function is analytic, i.e. has local power series

theorem *holomorphic_power_series:*

assumes *holfc: f holomorphic_on ball z r*
and *w: w* \in *ball* *z* *r*
shows $((\lambda n. (\text{deriv } \wedge n) f \ z / (\text{fact } n) * (w - z) \wedge n) \text{ sums } f \ w)$
proof –
 — Replacing *r* and the original (weak) premises with stronger ones
obtain *r* **where** *r* > 0 **and** *holfc: f holomorphic_on cball z r* **and** *w: w* \in *ball* *z* *r*

proof
have *cball* *z* $((r + \text{dist } w \ z) / 2) \subseteq$ *ball* *z* *r*
using *w* **by** (*simp add: dist_commute field_sum_of_halves subset_eq*)
then show *f holomorphic_on cball z ((r + dist w z) / 2)*
by (*rule holomorphic_on_subset [OF holfc]*)
have *r* > 0
by (*metis w dist_norm mem_ball norm_ge_zero not_less_iff_gr_or_eq order_less_le_trans*)
then show $0 < (r + \text{dist } w \ z) / 2$
by *simp* (*use zero_le_dist [of w z] in linarith*)
qed (*use w in <auto simp: dist_commute>*)
then have *holfc: f holomorphic_on ball z r*
using *ball_subset_cball holomorphic_on_subset* **by** *blast*
have *confc: continuous_on (cball z r) f*
by (*simp add: holfc holomorphic_on_imp_continuous_on*)
have *cint: $\bigwedge k. (\lambda u. f \ u / (u - z) \wedge \text{Suc } k) \text{ contour_integrable_on circlepath } z \ r$*
by (*rule Cauchy_higher_derivative_integral_circlepath [OF confc holfc]*) (*simp add: <0 < r>*)
obtain *B* **where** $0 < B$ **and** *B: $\bigwedge u. u \in \text{cball } z \ r \implies \text{norm}(f \ u) \leq B$*
by (*metis (no_types) bounded_pos compact_cball compact_continuous_image compact_imp_bounded confc image_eqI*)
obtain *k* **where** $0 < k \ k \leq r$ **and** *wz_eq: $\text{norm}(w - z) = r - k$*
and *k1e: $\bigwedge u. \text{norm}(u - z) = r \implies k \leq \text{norm}(u - w)$*
proof
show $\bigwedge u. \text{cmod } (u - z) = r \implies r - \text{dist } z \ w \leq \text{cmod } (u - w)$
by (*metis add_diff_eq diff_add_cancel dist_norm norm_diff_ineq*)
qed (*use w in <auto simp: dist_norm norm_minus_commute>*)

```

have ul: uniform_limit (sphere z r) ( $\lambda n\ x. (\sum k < n. (w-z) ^ k * (f\ x / (x-z) ^ Suc\ k))$ ) ( $\lambda x. f\ x / (x-w)$ ) sequentially
unfolding uniform_limit_iff dist_norm
proof clarify
  fix e::real
  assume  $0 < e$ 
  have rr:  $0 \leq (r-k) / r \wedge (r-k) / r < 1$  using k by auto
  obtain n where n:  $((r-k) / r) ^ n < e / B * k$ 
  using real_arch_pow_inv [of  $e/B*k \ (r-k)/r$ ]  $\langle 0 < e \rangle \langle 0 < B \rangle k$  by force
  have norm  $((\sum k < N. (w-z) ^ k * f\ u / (u-z) ^ Suc\ k) - f\ u / (u-w)) < e$ 
    if  $n \leq N$  and r:  $r = dist\ z\ u$  for N u
  proof -
    have N:  $((r-k) / r) ^ N < e / B * k$ 
      using le_less_trans [OF power_decreasing n]
      using  $\langle n \leq N \rangle k$  by auto
    have u [simp]:  $(u \neq z) \wedge (u \neq w)$ 
      using  $\langle 0 < r \rangle r\ w$  by auto
    have wzu_not1:  $(w-z) / (u-z) \neq 1$ 
      by (metis (no_types) dist_norm divide_eq_1 iff less_irrefl mem_ball norm_minus_commute r w)
    have norm  $((\sum k < N. (w-z) ^ k * f\ u / (u-z) ^ Suc\ k) * (u-w) - f\ u)$ 
       $= norm\ ((\sum k < N. (((w-z) / (u-z)) ^ k)) * f\ u * (u-w) / (u-z) - f\ u)$ 
    unfolding sum_distrib_right sum_divide_distrib power_divide by (simp add: algebra_simps)
    also have  $\dots = norm\ (((w-z) / (u-z)) ^ N - 1) * (u-w) / (((w-z) / (u-z) - 1) * (u-z)) - 1) * norm\ (f\ u)$ 
      using  $\langle 0 < B \rangle$ 
      apply (simp add: geometric_sum [OF wzu_not1] flip: norm_mult)
      apply (simp add: field_simps)
      done
    also have  $\dots = norm\ ((u-z) ^ N * (w-u) - ((w-z) ^ N - (u-z) ^ N) * (u-w)) / (r ^ N * norm\ (u-w)) * norm\ (f\ u)$ 
      using  $\langle 0 < r \rangle r$  by (simp add: divide_simps norm_mult norm_divide norm_power dist_norm norm_minus_commute)
    also have  $\dots = norm\ ((w-z) ^ N * (w-u)) / (r ^ N * norm\ (u-w)) * norm\ (f\ u)$ 
      by (simp add: algebra_simps)
    also have  $\dots = norm\ (w-z) ^ N * norm\ (f\ u) / r ^ N$ 
      by (simp add: norm_mult norm_power norm_minus_commute)
    also have  $\dots \leq (((r-k)/r) ^ N) * B$ 
      using  $\langle 0 < r \rangle w\ k$ 
      by (simp add: B divide_simps mult_mono r wz_eq)
    also have  $\dots < e * k$ 
      using  $\langle 0 < B \rangle N$  by (simp add: divide_simps)
    also have  $\dots \leq e * norm\ (u-w)$ 
      using r k  $\langle 0 < e \rangle$  by (simp add: dist_commute dist_norm)
    finally show ?thesis
      by (simp add: field_split_simps norm_divide del: power_Suc)
  qed

```

```

with  $\langle 0 < r \rangle$  show  $\forall_F n$  in sequentially.  $\forall x \in \text{sphere } z \ r.$ 
  norm  $((\sum_{k < n}. (w-z) \wedge k * (f x / (x-z) \wedge \text{Suc } k)) - f x / (x-w))$ 
< e
  by (auto simp: mult_ac less_imp_le eventually_sequentially Ball_def)
qed
have  $\S: \bigwedge x \ k. k \in \{..<x\} \implies$ 
   $(\lambda u. (w-z) \wedge k * (f u / (u-z) \wedge \text{Suc } k)) \text{ contour\_integrable\_on circlepath }$ 
 $z \ r$ 
  using contour_integrable_lmul [OF cint, of  $(w-z) \wedge a$  for  $a$ ] by (simp add:
field_simps)
  have eq:  $\bigwedge n.$ 
   $(\sum_{k < n}. \text{contour\_integral } (\text{circlepath } z \ r) (\lambda u. f u / (u-z) \wedge \text{Suc } k) * (w-z) \wedge k) =$ 
   $\text{contour\_integral } (\text{circlepath } z \ r) (\lambda u. \sum_{k < n}. (w-z) \wedge k * (f u / (u-z) \wedge \text{Suc } k))$ 
  apply (subst contour_integral_sum)
  apply (simp_all only:  $\S$  finite_lessThan contour_integrable_lmul cint algebra_simps)
done
  have  $\bigwedge u \ k. k \in \{..<u\} \implies (\lambda x. f x / (x-z) \wedge \text{Suc } k) \text{ contour\_integrable\_on circlepath } z \ r$ 
  using  $\langle 0 < r \rangle$  by (force intro!: Cauchy_higher_derivative_integral_circlepath [OF contf_holf])
  then have  $\bigwedge u. (\lambda y. \sum_{k < u}. (w-z) \wedge k * (f y / (y-z) \wedge \text{Suc } k)) \text{ contour\_integrable\_on circlepath } z \ r$ 
  by (intro contour_integrable_sum contour_integrable_lmul, simp)
  then have  $(\lambda k. \text{contour\_integral } (\text{circlepath } z \ r) (\lambda u. f u / (u-z) \wedge (\text{Suc } k)) * (w-z) \wedge k)$ 
   $\text{sums contour\_integral } (\text{circlepath } z \ r) (\lambda u. f u / (u-w))$ 
  unfolding sums_def eq
  using  $\langle 0 < r \rangle$  contour_integral_uniform_limit_circlepath [OF eventuallyI ul]

  by fastforce
  then have  $(\lambda k. \text{contour\_integral } (\text{circlepath } z \ r) (\lambda u. f u / (u-z) \wedge (\text{Suc } k)) * (w-z) \wedge k)$ 
   $\text{sums } (2 * \text{of\_real } \pi * i * f w)$ 
  using  $w$  by (auto simp: dist_commute dist_norm contour_integral_unique [OF Cauchy_integral_circlepath_simple [OF holf]])
  then have  $(\lambda k. \text{contour\_integral } (\text{circlepath } z \ r) (\lambda u. f u / (u-z) \wedge \text{Suc } k) * (w-z) \wedge k / (i * (\text{of\_real } \pi * 2)))$ 
   $\text{sums } ((2 * \text{of\_real } \pi * i * f w) / (i * (\text{complex\_of\_real } \pi * 2)))$ 
  by (rule sums_divide)
  then have  $(\lambda n. (w-z) \wedge n * \text{contour\_integral } (\text{circlepath } z \ r) (\lambda u. f u / (u-z) \wedge \text{Suc } n) / (i * (\text{of\_real } \pi * 2)))$ 
   $\text{sums } f w$ 
  by (simp add: field_simps)
then show ?thesis
  by (simp add: field_simps  $\langle 0 < r \rangle$  Cauchy_higher_derivative_integral_circlepath [OF contf_holf])

```

qed

4.7 The Liouville theorem and the Fundamental Theorem of Algebra

These weak Liouville versions don't even need the derivative formula.

```

lemma Liouville_weak_0:
  assumes holf: f holomorphic_on UNIV and inf: (f  $\longrightarrow$  0) at_infinity
  shows f z = 0
proof (rule ccontr)
  assume fz: f z  $\neq$  0
  with inf [unfolded Lim_at_infinity, rule_format, of norm(f z)/2]
  obtain B where B:  $\bigwedge x. B \leq \text{cmod } x \implies \text{norm } (f x) * 2 < \text{cmod } (f z)$ 
    by (auto simp: dist_norm)
  define R where R = 1 + |B| + norm z
  have R > 0
    unfolding R_def by (smt (verit) norm_ge_zero)
  have *: (( $\lambda u. f u / (u - z)$ ) has_contour_integral 2 * complex_of_real  $\pi * i * f z$ )
    (circlepath z R)
    using continuous_on_subset holf holomorphic_on_subset  $\langle 0 < R \rangle$ 
    by (force intro: holomorphic_on_imp_continuous_on Cauchy_integral_circlepath)
  have cmod (x - z) = R  $\implies \text{cmod } (f x) * 2 < \text{cmod } (f z)$  for x
    unfolding R_def by (rule B) (use norm_triangle_ineq4 [of x z] in auto)
  with  $\langle R > 0 \rangle$  fz show False
    using has_contour_integral_bound_circlepath [OF *, of norm(f z)/2/R]
    by (auto simp: less_imp_le norm_mult norm_divide field_split_simps)
qed

```

```

proposition Liouville_weak:
  assumes f holomorphic_on UNIV and (f  $\longrightarrow$  l) at_infinity
  shows f z = l
  using Liouville_weak_0 [of  $\lambda z. f z - l$ ]
  by (simp add: assms holomorphic_on_diff LIM_zero)

```

```

proposition Liouville_weak_inverse:
  assumes f holomorphic_on UNIV and unbounded:  $\bigwedge B. \text{eventually } (\lambda x. \text{norm } (f x) \geq B) \text{ at\_infinity}$ 
  obtains z where f z = 0
proof –
  { assume f:  $\bigwedge z. f z \neq 0$ 
    have 1: ( $\lambda x. 1 / f x$ ) holomorphic_on UNIV
      by (simp add: holomorphic_on_divide assms f)
    have 2: (( $\lambda x. 1 / f x$ )  $\longrightarrow$  0) at_infinity
    proof (rule tendstoI [OF eventually_mono])
      fix e::real
      assume e > 0
      show eventually ( $\lambda x. 2/e \leq \text{cmod } (f x)$ ) at_infinity
        by (rule_tac B=2/e in unbounded)
    qed (simp add: dist_norm norm_divide field_split_simps)
  }

```



```

    have False
    using Liouville_weak_0 [OF 1 2] f by simp
  }
  then show ?thesis
    using that by blast
qed

```

In particular we get the Fundamental Theorem of Algebra.

```

theorem fundamental_theorem_of_algebra:
  fixes a :: nat  $\Rightarrow$  complex
  assumes a_0 = 0  $\vee$  ( $\exists i \in \{1..n\}. a\ i \neq 0$ )
  obtains z where ( $\sum_{i \leq n}. a\ i * z^i$ ) = 0
using assms
proof (elim disjE bexE)
  assume a_0 = 0 then show ?thesis
    by (auto simp: that [of 0])
next
  fix i
  assume i:  $i \in \{1..n\}$  and nz:  $a\ i \neq 0$ 
  have 1: ( $\lambda z. \sum_{i \leq n}. a\ i * z^i$ ) holomorphic_on UNIV
    by (rule holomorphic_intros)+
  show thesis
  proof (rule Liouville_weak_inverse [OF 1])
    show  $\forall_F x$  in at_infinity.  $B \leq cmod (\sum_{i \leq n}. a\ i * x^i)$  for B
      using i nz by (intro polyfun_extremal exI [of _ i]) auto
    qed (use that in auto)
  qed

```

4.8 Weierstrass convergence theorem

```

lemma holomorphic_uniform_limit:
  assumes cont: eventually ( $\lambda n. continuous\_on (cball\ z\ r) (f\ n) \wedge (f\ n)$  holomorphic_on ball z r) F
  and ulim: uniform_limit (cball z r) f g F
  and F:  $\neg$  trivial_limit F
  obtains continuous_on (cball z r) g g holomorphic_on ball z r
proof (cases r 0::real rule: linorder_cases)
  case less then show ?thesis by (force simp: ball_empty less_imp_le continuous_on_def holomorphic_on_def intro: that)
next
  case equal then show ?thesis
    by (force simp: holomorphic_on_def intro: that)
next
  case greater
  have contg: continuous_on (cball z r) g
    using cont uniform_limit_theorem [OF eventually_mono ulim F] by blast
  have path_image (circlepath z r)  $\subseteq$  cball z r
    using <0 < r> by auto
  then have 1: continuous_on (path_image (circlepath z r)) ( $\lambda x. 1 / (2 * com-$ 

```

```

plex_of_real pi * i) * g x)
  by (intro continuous_intros continuous_on_subset [OF contg])
  have 2: ((λu. 1 / (2 * of_real pi * i) * g u / (u-w) ^ 1) has_contour_integral
g w) (circlepath z r)
    if w: w ∈ ball z r for w
  proof -
    define d where d = (r - norm(w-z))
    have 0 < d d ≤ r using w by (auto simp: norm_minus_commute d_def
dist_norm)
    have dle: ∧u. cmod (z-u) = r ⟹ d ≤ cmod (u-w)
    unfolding d_def by (metis add_diff_eq diff_add_cancel norm_diff_ineq
norm_minus_commute)
    have ev_int: ∀F n in F. (λu. f n u / (u-w)) contour_integrable_on circlepath
z r
    using w
    by (auto intro: eventually_mono [OF cont] Cauchy_higher_derivative_integral_circlepath
[where k=0, simplified])
    have ∧e. [0 < r; 0 < d; 0 < e]
      ⟹ ∀F n in F.
        ∀x∈sphere z r.
          x ≠ w ⟶
            cmod (f n x - g x) < e * cmod (x-w)
    apply (rule_tac e1=e * d in eventually_mono [OF uniform_limitD [OF
ulim]])
    apply (force simp: dist_norm intro: dle mult_left_mono less_le_trans)+
    done
    then have ul_less: uniform_limit (sphere z r) (λn x. f n x / (x-w)) (λx. g x
/ (x-w)) F
    using greater ‹0 < d›
    by (auto simp add: uniform_limit_iff dist_norm norm_divide diff_divide_distrib
[symmetric] divide_simps)
    have g_cint: (λu. g u / (u-w)) contour_integrable_on circlepath z r
    by (rule contour_integral_uniform_limit_circlepath [OF ev_int ul_less F ‹0
< r›])
    have cif_tends_cig: ((λn. contour_integral(circlepath z r) (λu. f n u / (u-w)))
⟶ contour_integral(circlepath z r) (λu. g u / (u-w))) F
    by (rule contour_integral_uniform_limit_circlepath [OF ev_int ul_less F ‹0
< r›])
    have f_tends_cig: ((λn. 2 * of_real pi * i * f n w) ⟶ contour_integral
(circlepath z r) (λu. g u / (u-w))) F
    proof (rule Lim_transform_eventually)
      show ∀F x in F. contour_integral (circlepath z r) (λu. f x u / (u-w))
        = 2 * of_real pi * i * f x w
    using w ‹0 < d› d_def
    by (auto intro: eventually_mono [OF cont contour_integral_unique [OF
Cauchy_integral_circlepath]])
    qed (auto simp: cif_tends_cig)
    have ∧e. 0 < e ⟹ ∀F n in F. dist (f n w) (g w) < e
    by (rule eventually_mono [OF uniform_limitD [OF ulim]]) (use w in auto)

```

```

    then have  $((\lambda n. 2 * \text{of\_real } \pi * i * f\ n\ w) \longrightarrow 2 * \text{of\_real } \pi * i * g\ w)\ F$ 
      by (rule tendsto_mult_left [OF tendstoI])
    then have  $((\lambda u. g\ u / (u - w)) \text{ has\_contour\_integral } 2 * \text{of\_real } \pi * i * g\ w)$ 
      (circlepath z r)
      using has_contour_integral_integral [OF g_cint] tendsto_unique [OF F
f_tends_cig] w
      by fastforce
    then have  $((\lambda u. g\ u / (2 * \text{of\_real } \pi * i * (u - w))) \text{ has\_contour\_integral } g\ w)$ 
      (circlepath z r)
      using has_contour_integral_div [where c = 2 * of_real pi * i]
      by (force simp: field_simps)
    then show ?thesis
      by (simp add: dist_norm)
  qed
show ?thesis
  using Cauchy_next_derivative_circlepath(2) [OF 1 2, simplified]
  by (fastforce simp add: holomorphic_on_open contg intro: that)
qed

lemma higher_deriv_complex_uniform_limit:
  assumes ulim: uniform_limit A f g F
    and f_holo: eventually  $(\lambda n. f\ n \text{ holomorphic\_on } A)\ F$ 
    and F:  $F \neq \text{bot}$ 
    and A: open A z  $\in A$ 
  shows  $((\lambda n. (\text{deriv } \sim m)\ (f\ n)\ z) \longrightarrow (\text{deriv } \sim m)\ g\ z)\ F$ 
proof -
  obtain r where r:  $r > 0$  cball z r  $\subseteq A$ 
    using A by (meson open_contains_cball)
  have r': ball z r  $\subseteq A$ 
    using r by auto
  define h where  $h = (\lambda n\ z. f\ n\ z - g\ z)$ 
  define c where  $c = \text{of\_real } (2 * \pi) * i / \text{fact } m$ 
  have [simp]:  $c \neq 0$ 
    by (simp add: c_def)
  have g_holomorphic_on_ball z r  $\wedge$  continuous_on (cball z r) g
  proof (rule holomorphic_uniform_limit)
    show uniform_limit (cball z r) f g F
      by (rule uniform_limit_on_subset [OF ulim r(2)])
    show  $\forall_F\ n \text{ in } F. \text{continuous\_on } (cball\ z\ r)\ (f\ n) \wedge f\ n \text{ holomorphic\_on } ball\ z\ r$ 
  r using f_holo
    by eventually_elim
      (use holomorphic_on_subset [OF _ r(2)] holomorphic_on_subset [OF _ r'])

    in  $\langle \text{auto intro!}: \text{holomorphic\_on\_imp\_continuous\_on} \rangle$ 
  qed (use F in auto)
  hence g_holo: g holomorphic_on ball z r and g_cont: continuous_on (cball z r) g
  by blast+

```

```

have ulim': uniform_limit (sphere z r) ( $\lambda n x. h \ n \ x \ / \ (x - z) \wedge (Suc \ m)$ ) ( $\lambda \_.$ 
0) F
proof -
  have uniform_limit (sphere z r) ( $\lambda n x. f \ n \ x \ / \ (x - z) \wedge Suc \ m$ ) ( $\lambda x. g \ x \ / \ (x$ 
 $- z) \wedge Suc \ m$ ) F
  proof (intro uniform_lim_divide uniform_limit_intros uniform_limit_on_subset[OF
ulim])
    have compact (g ' sphere z r)
    by (intro compact_continuous_image continuous_on_subset[OF g_cont])
  auto
    thus bounded (g ' sphere z r)
    by (rule compact_imp_bounded)
    show  $r \wedge Suc \ m \leq norm \ ((x - z) \wedge Suc \ m)$  if  $x \in sphere \ z \ r$  for  $x$  unfolding
norm_power
    by (intro power_mono) (use that r(1) in  $\langle auto \ simp: dist\_norm \ norm\_minus\_commute \rangle$ )
    qed (use r in auto)
    hence uniform_limit (sphere z r) ( $\lambda n x. f \ n \ x \ / \ (x - z) \wedge Suc \ m - g \ x \ / \ (x$ 
 $- z) \wedge Suc \ m$ )
      ( $\lambda x. g \ x \ / \ (x - z) \wedge Suc \ m - g \ x \ / \ (x - z) \wedge Suc \ m$ ) F
    by (intro uniform_limit_intros)
    thus ?thesis
    by (simp add: h_def diff_divide_distrib)
  qed

have has_integral: eventually ( $\lambda n. ((\lambda u. h \ n \ u \ / \ (u - z) \wedge Suc \ m)$  has_contour_integral


$$c * (deriv \ \sim m) (h \ n) \ z) (circlepath \ z \ r)) F$$

using f_holo
proof eventually_elim
  case (elim n)
  show ?case
    unfolding c_def
  proof (rule Cauchy_has_contour_integral_higher_derivative_circlepath)
    show continuous_on (cball z r) (h n) unfolding h_def
    by (intro continuous_intros g_cont holomorphic_on_imp_continuous_on
holomorphic_on_subset[OF elim] r)
    show h n holomorphic_on ball z r
    unfolding h_def by (intro holomorphic_intros g_holo holomorphic_on_subset[OF
elim] r')
    qed (use r(1) in auto)
  qed

have ( $(\lambda n. contour\_integral \ (circlepath \ z \ r) \ (\lambda u. h \ n \ u \ / \ (u - z) \wedge Suc \ m))$ 
 $\longrightarrow$ 
 $contour\_integral \ (circlepath \ z \ r) \ (\lambda u. 0 \ / \ (u - z) \wedge Suc \ m)) F$ 
proof (rule contour_integral_uniform_limit_circlepath)
  show  $\forall_F \ n \ in \ F. (\lambda u. h \ n \ u \ / \ (u - z) \wedge Suc \ m) \ contour\_integrable\_on$ 
 $circlepath \ z \ r$ 
  using has_integral by eventually_elim (blast intro: has_contour_integral_integrable)

```

```

qed (use r(1) ⟨F ≠ bot⟩ ulim' in simp_all)
hence ((λn. contour_integral (circlepath z r) (λu. h n u / (u - z) ^ Suc m))
  ⟶ 0) F
  by simp
also have ?this ⟷ ((λn. c * (deriv ~ m) (h n) z) ⟶ 0) F
proof (rule tendsto_cong)
  show ∀ F x in F. contour_integral (circlepath z r) (λu. h x u / (u - z) ^ Suc
m) =
      c * (deriv ~ m) (h x) z
  using has_integral by eventually_elim (simp add: contour_integral_unique)
qed
finally have ((λn. (deriv ~ m) g z + c * (deriv ~ m) (h n) z / c) ⟶
  (deriv ~ m) g z + 0 / c) F
  by (intro tendsto_intros) auto
also have ?this ⟷ ((λn. (deriv ~ m) (f n) z) ⟶ (deriv ~ m) g z) F
proof (intro filterlim_cong)
  show ∀ F n in F. (deriv ~ m) g z + c * (deriv ~ m) (h n) z / c = (deriv ~
m) (f n) z
  using f_holo
proof eventually_elim
  case (elim n)
  have (deriv ~ m) (h n) z = (deriv ~ m) (f n) z - (deriv ~ m) g z
unfolding h_def
  by (rule higher_deriv_diff holomorphic_on_subset[OF elim r'] g_holo A)+
  (use r(1) in auto)
  thus ?case
  by simp
qed
qed auto
finally show ?thesis .
qed

```

lemma *deriv_complex_uniform_limit:*

```

assumes ulim: uniform_limit A f g F
  and f_holo: eventually (λn. f n holomorphic_on A) F
  and F: F ≠ bot
  and A: open A z ∈ A
  shows ((λn. deriv (f n) z) ⟶ deriv g z) F
using higher_deriv_complex_uniform_limit[OF assms, of 1] by simp

```

lemma *logderiv_producing_complex_uniform_limit:*

```

fixes f :: nat ⇒ complex ⇒ complex
assumes lim: uniform_limit A (λn x. ∏ k<n. f k x) P sequentially
assumes holo: ∧k. f k holomorphic_on A
assumes nz: P z ≠ 0
assumes A: open A z ∈ A
shows (λk. deriv (f k) z / f k z) sums (deriv P z / P z)
proof -
  define f' where f' = (λk. deriv (f k))

```

```

note [derivative_intros] = has_field_derivative_prod'
have [derivative_intros]:
  (f k has_field_derivative f' k z) (at z within B) if z ∈ A for B z k
  using that holomorphic_derivI[OF holo[of k], of z B] A unfolding f'_def by
auto
have lim': (λn. ∏ k<n. f k z) ⟶ P z
  using lim by (rule tendsto_uniform_limitI) fact+

have nz': f k z ≠ 0 for k
proof
  assume f k z = 0
  have eventually (λn. (∏ k<n. f k z) = 0) sequentially
    using eventually_gt_at_top[of k] by eventually_elim (use ⟨f k z = 0⟩ in
auto)
  hence (λn. (∏ k<n. f k z)) ⟶ 0
  by (rule tendsto_eventually)
with lim' have P z = 0
  using tendsto_unique sequentially_bot by blast
with nz show False
by simp
qed

from lim have (λn. deriv (λx. ∏ k<n. f k x) z) ⟶ deriv P z
  by (rule deriv_complex_uniform_limit)
  (use A in ⟨auto intro!: always_eventually_holomorphic_intros holo⟩)
also have (λn. deriv (λx. ∏ k<n. f k x) z) = (λn. (∏ k<n. f k z) * (∑ k<n. f'
k z / f k z))
  using ⟨z ∈ A⟩ by (auto intro!: ext DERIV_imp_deriv derivative_eq_intros
simp: nz')
  finally have (λn. (∏ k<n. f k z) * (∑ k<n. f' k z / f k z)) ⟶ deriv P z .
  hence (λn. (∏ k<n. f k z) * (∑ k<n. f' k z / f k z) / (∏ k<n. f k z)) ⟶
deriv P z / P z
  by (intro tendsto_intros) (use nz lim' in auto)
also have (λn. (∏ k<n. f k z) * (∑ k<n. f' k z / f k z) / (∏ k<n. f k z)) =
  (λn. (∑ k<n. f' k z / f k z))
  by (simp add: nz')
finally show (λk. f' k z / f k z) sums (deriv P z / P z)
  unfolding sums_def .
qed

```

Version showing that the limit is the limit of the derivatives.

```

proposition has_complex_derivative_uniform_limit:
  fixes z::complex
  assumes cont: eventually (λn. continuous_on (cball z r) (f n) ∧
    (∀ w ∈ ball z r. ((f n) has_field_derivative (f' n w)) (at
w))) F
  and ulim: uniform_limit (cball z r) f g F
  and F: ¬ trivial_limit F and 0 < r
  obtains g' where

```

```

      continuous_on (cball z r) g
       $\bigwedge w. w \in \text{ball } z \ r \implies (g \text{ has\_field\_derivative } (g' w)) \ (at \ w) \wedge ((\lambda n. f' n \ w) \longrightarrow g' w) \ F$ 
    proof -
      let ?conint = contour_integral (circlepath z r)
      have g: continuous_on (cball z r) g g holomorphic_on ball z r
      by (rule holomorphic_uniform_limit [OF eventually_mono [OF cont] ulim F];
          auto simp: holomorphic_on_open field_differentiable_def)+
      then obtain g' where g':  $\bigwedge x. x \in \text{ball } z \ r \implies (g \text{ has\_field\_derivative } g' x) \ (at \ x)$ 
      using DERIV_deriv_iff_has_field_derivative
      by (fastforce simp add: holomorphic_on_open)
      then have derg:  $\bigwedge x. x \in \text{ball } z \ r \implies \text{deriv } g \ x = g' x$ 
      by (simp add: DERIV_imp_deriv)
      have tends_f'n_g':  $((\lambda n. f' n \ w) \longrightarrow g' w) \ F$  if w:  $w \in \text{ball } z \ r$  for w
      proof -
        have eq_f': ?conint  $(\lambda x. f \ n \ x / (x-w)^2) - ?conint (\lambda x. g \ x / (x-w)^2) = (f' n \ w - g' w) * (2 * of\_real \pi * i)$ 
        if cont_fn: continuous_on (cball z r) (f n)
        and fnd:  $\bigwedge w. w \in \text{ball } z \ r \implies (f \ n \ \text{has\_field\_derivative } f' n \ w) \ (at \ w)$ 
      for n
      proof -
        have hol_fn: f n holomorphic_on ball z r
        using fnd by (force simp: holomorphic_on_open)
        have (f n has_field_derivative 1 / (2 * of_real pi * i) * ?conint  $(\lambda u. f \ n \ u / (u-w)^2)$ ) (at w)
        by (rule Cauchy_derivative_integral_circlepath [OF cont_fn hol_fn w])
        then have f':  $f' n \ w = 1 / (2 * of\_real \pi * i) * ?conint (\lambda u. f \ n \ u / (u-w)^2)$ 
        using DERIV_unique [OF fnd] w by blast
        show ?thesis
        by (simp add: f' Cauchy_contour_integral_circlepath_2 [OF g w] derg [OF w] field_split_simps)
      qed
      define d where  $d = (r - \text{norm}(w-z))^2$ 
      have d > 0
      using w by (simp add: dist_commute dist_norm d_def)
      have dle:  $d \leq cmod ((y-w)^2)$  if r = cmod (z-y) for y
      by (smt (verit, best) d_def diff_add_cancel diff_diff_eq2 dist_norm mem_ball norm_minus_commute norm_power norm_triangle_ineq2 power_mono that w)
      have 1:  $\forall_F \ n \ \text{in } F. (\lambda x. f \ n \ x / (x-w)^2) \text{ contour\_integrable\_on } \text{circlepath } z \ r$ 
      by (force simp: holomorphic_on_open intro: w Cauchy_derivative_integral_circlepath eventually_mono [OF cont])
      have 2: uniform_limit (sphere z r)  $(\lambda n \ x. f \ n \ x / (x-w)^2) (\lambda x. g \ x / (x-w)^2)$ 
      F
      unfolding uniform_limit_iff
    proof clarify
      fix e::real
      assume e > 0

```

```

with  $\langle r > 0 \rangle$ 
have  $\forall_F n \text{ in } F. \forall x. x \neq w \longrightarrow \text{cmod } (z-x) = r \longrightarrow \text{cmod } (f \ n \ x - g \ x) <$ 
 $e * \text{cmod } ((x-w)^2)$ 
by (force simp:  $\langle 0 < d \rangle$  dist_norm dle intro: less_le_trans eventually_mono
[OF uniform_limitD [OF ulim], of e*d])
with  $\langle r > 0 \rangle \langle e > 0 \rangle$ 
show  $\forall_F n \text{ in } F. \forall x \in \text{sphere } z \ r. \text{dist } (f \ n \ x / (x-w)^2) (g \ x / (x-w)^2) < e$ 
by (simp add: norm_divide field_split_simps sphere_def dist_norm)
qed
have  $((\lambda n. \text{contour\_integral } (\text{circlepath } z \ r) (\lambda x. f \ n \ x / (x-w)^2))$ 
 $\longrightarrow \text{contour\_integral } (\text{circlepath } z \ r) ((\lambda x. g \ x / (x-w)^2))) F$ 
by (rule contour_integral_uniform_limit_circlepath [OF 1 2 F  $\langle 0 < r \rangle$ ])
then have  $\text{tendsto\_0}: ((\lambda n. 1 / (2 * \text{of\_real } \pi * i) * (?conint (\lambda x. f \ n \ x /$ 
 $(x-w)^2) - ?conint (\lambda x. g \ x / (x-w)^2))) \longrightarrow 0) F$ 
using Lim_null by (force intro!: tendsto_mult_right_zero)
have  $((\lambda n. f' \ n \ w - g' \ w) \longrightarrow 0) F$ 
by (force simp: divide_simps
intro: eq_f' eventually_mono [OF cont] Lim_transform_eventually [OF
tendsto_0])
then show ?thesis using Lim_null by blast
qed
obtain  $g' \text{ where } \bigwedge w. w \in \text{ball } z \ r \implies (g \text{ has\_field\_derivative } (g' \ w)) \text{ (at } w) \wedge$ 
 $((\lambda n. f' \ n \ w) \longrightarrow g' \ w) F$ 
by (blast intro: tends_f'n_g' g')
then show ?thesis using g
using that by blast
qed

```

4.9 Some more simple/convenient versions for applications

lemma holomorphic_uniform_sequence:

```

assumes S: open S
and hol_fn:  $\bigwedge n. (f \ n) \text{ holomorphic\_on } S$ 
and ulim_g:  $\bigwedge x. x \in S \implies \exists d. 0 < d \wedge \text{cball } x \ d \subseteq S \wedge \text{uniform\_limit}$ 
 $(\text{cball } x \ d) \ f \ g \text{ sequentially}$ 
shows g holomorphic_on S
proof -
have  $\exists f'. (g \text{ has\_field\_derivative } f') \text{ (at } z) \text{ if } z \in S \text{ for } z$ 
proof -
obtain r where  $0 < r \text{ and } r: \text{cball } z \ r \subseteq S$ 
and ul: uniform_limit (cball z r) f g sequentially
using ulim_g [OF  $\langle z \in S \rangle$ ] by blast
have *:  $\forall_F n \text{ in sequentially. continuous\_on } (\text{cball } z \ r) (f \ n) \wedge f \ n \text{ holomor-}$ 
 $\text{phic\_on ball } z \ r$ 
by (smt (verit, best) ball_subset_cball hol_fn holomorphic_on_imp_continuous_on

holomorphic_on_subset not_eventuallyD r)
show ?thesis
using  $\langle 0 < r \rangle$  centre_in_ball ul

```



```

    by (auto simp: holomorphic_on_open intro: holomorphic_uniform_limit [OF
*])
  qed
  with S show ?thesis
    by (simp add: holomorphic_on_open)
qed

lemma has_complex_derivative_uniform_sequence:
  fixes S :: complex set
  assumes S: open S
    and hfd:  $\bigwedge n x. x \in S \implies ((f\ n) \text{ has\_field\_derivative } f'\ n\ x) \text{ (at } x)$ 
    and ulim_g:  $\bigwedge x. x \in S \implies \exists d. 0 < d \wedge \text{cball } x\ d \subseteq S \wedge \text{uniform\_limit } (\text{cball } x\ d) f\ g \text{ sequentially}$ 
  shows  $\exists g'. \forall x \in S. (g \text{ has\_field\_derivative } g'\ x) \text{ (at } x) \wedge ((\lambda n. f'\ n\ x) \longrightarrow g'\ x) \text{ sequentially}$ 
  proof -
    have y:  $\exists y. (g \text{ has\_field\_derivative } y) \text{ (at } z) \wedge (\lambda n. f'\ n\ z) \longrightarrow y \text{ if } z \in S$ 
  for z
  proof -
    obtain r where  $0 < r$  and  $r: \text{cball } z\ r \subseteq S$ 
      and ul:  $\text{uniform\_limit } (\text{cball } z\ r) f\ g \text{ sequentially}$ 
    using ulim_g [OF ‹z ∈ S›] by blast
    have *:  $\forall_F n \text{ in sequentially. continuous\_on } (\text{cball } z\ r) (f\ n) \wedge$ 
       $(\forall w \in \text{ball } z\ r. ((f\ n) \text{ has\_field\_derivative } (f'\ n\ w))$ 
    (at w))
    proof (intro eventuallyI conjI ballI)
      show  $\text{continuous\_on } (\text{cball } z\ r) (f\ x) \text{ for } x$ 
      by (meson S continuous_on_subset hfd holomorphic_on_imp_continuous_on
holomorphic_on_open r)
      show  $w \in \text{ball } z\ r \implies (f\ x \text{ has\_field\_derivative } f'\ x\ w) \text{ (at } w) \text{ for } w\ x$ 
      using ball_subset_cball hfd r by blast
    qed
    show ?thesis
      by (rule has_complex_derivative_uniform_limit [OF *, of g]) (use ‹0 < r›
ul in ‹force+›)
  qed
  show ?thesis
    by (rule bchoice) (blast intro: y)
qed

```

4.10 On analytic functions defined by a series

```

lemma series_and_derivative_comparison:
  fixes S :: complex set
  assumes S: open S
    and h: summable h
    and hfd:  $\bigwedge n x. x \in S \implies (f\ n \text{ has\_field\_derivative } f'\ n\ x) \text{ (at } x)$ 
    and to_g:  $\forall_F n \text{ in sequentially. } \forall x \in S. \text{norm } (f\ n\ x) \leq h\ n$ 
  obtains  $g\ g' \text{ where } \forall x \in S. ((\lambda n. f\ n\ x) \text{ sums } g\ x) \wedge ((\lambda n. f'\ n\ x) \text{ sums } g'\ x)$ 

```

```

 $\wedge (g \text{ has\_field\_derivative } g' x) (at x)$ 
proof –
  obtain  $g$  where  $g$ :  $\text{uniform\_limit } S (\lambda n x. \sum_{i < n} f i x) g$  sequentially
  using  $\text{Weierstrass\_m\_test\_ev } [OF \text{ to\_g } h]$  by force
  have  $*$ :  $\exists d > 0. \text{cball } x d \subseteq S \wedge \text{uniform\_limit } (\text{cball } x d) (\lambda n x. \sum_{i < n} f i x)$ 
g sequentially
  if  $x \in S$  for  $x$ 
    using  $\text{open\_contains\_cball } [of S] \langle x \in S \rangle S g \text{ uniform\_limit\_on\_subset}$  by
blast
    have  $\bigwedge x. x \in S \implies (\lambda n. \sum_{i < n} f i x) \longrightarrow g x$ 
    by  $(metis \text{tendsto\_uniform\_limitI } [OF g])$ 
    moreover have  $\exists g'. \forall x \in S. (g \text{ has\_field\_derivative } g' x) (at x) \wedge (\lambda n. \sum_{i < n} f' i x) \longrightarrow g' x$ 
    by  $(rule \text{has\_complex\_derivative\_uniform\_sequence } [OF S]) (auto \text{ intro: } * \text{ hfd } \text{DERIV\_sum})+$ 
    ultimately show ?thesis
    by  $(metis \text{sums\_def } that)$ 
qed

```

A version where we only have local uniform/comparative convergence.

lemma *series_and_derivative_comparison_local*:

```

fixes  $S :: \text{complex set}$ 
assumes  $S$ :  $\text{open } S$ 
  and  $\text{hfd}$ :  $\bigwedge n x. x \in S \implies (f n \text{ has\_field\_derivative } f' n x) (at x)$ 
  and  $\text{to\_g}$ :  $\bigwedge x. x \in S \implies \exists d h. 0 < d \wedge \text{summable } h \wedge (\forall_F n \text{ in sequentially. } \forall y \in \text{ball } x d \cap S. \text{norm } (f n y) \leq h n)$ 
shows  $\exists g g'. \forall x \in S. ((\lambda n. f n x) \text{ sums } g x) \wedge ((\lambda n. f' n x) \text{ sums } g' x) \wedge (g \text{ has\_field\_derivative } g' x) (at x)$ 
proof –
  have  $\exists y. (\lambda n. f n z) \text{ sums } (\sum n. f n z) \wedge (\lambda n. f' n z) \text{ sums } y \wedge ((\lambda x. \sum n. f n x) \text{ has\_field\_derivative } y) (at z)$ 
  if  $z \in S$  for  $z$ 
  proof –
    obtain  $d h$  where  $0 < d \text{ summable } h$  and  $\text{le\_h}$ :  $\forall_F n \text{ in sequentially. } \forall y \in \text{ball } z d \cap S. \text{norm } (f n y) \leq h n$ 
    using  $\text{to\_g } \langle z \in S \rangle$  by meson
    then obtain  $r$  where  $r > 0$  and  $r$ :  $\text{ball } z r \subseteq \text{ball } z d \cap S$  using  $\langle z \in S \rangle S$ 
    by  $(metis \text{Int\_iff\_open\_ball\_centre\_in\_ball } \text{open\_Int } \text{open\_contains\_ball\_eq})$ 
    have  $1$ :  $\text{open } (\text{ball } z d \cap S)$ 
    by  $(simp \text{ add: } \text{open\_Int } S)$ 
    have  $2$ :  $\bigwedge n x. x \in \text{ball } z d \cap S \implies (f n \text{ has\_field\_derivative } f' n x) (at x)$ 
    by  $(auto \text{ simp: } \text{hfd})$ 
    obtain  $g g'$  where  $gg'$ :  $\forall x \in \text{ball } z d \cap S. ((\lambda n. f n x) \text{ sums } g x) \wedge ((\lambda n. f' n x) \text{ sums } g' x) \wedge (g \text{ has\_field\_derivative } g' x) (at x)$ 
    by  $(auto \text{ intro: } \text{le\_h } \text{series\_and\_derivative\_comparison } [OF 1 \langle \text{summable } h \rangle \text{hfd}])$ 
    then have  $(\lambda n. f' n z) \text{ sums } g' z$ 
    by  $(meson \langle 0 < r \rangle \text{centre\_in\_ball } \text{contra\_subsetD } r)$ 
  qed

```

```

moreover have  $(\lambda n. f\ n\ z)\ \text{sums}\ (\sum n. f\ n\ z)$ 
  using summable_sums centre_in_ball  $\langle 0 < d \rangle \langle \text{summable } h \rangle \text{le\_}h$ 
  by (metis (full_types) Int_iff gg' summable_def that)
moreover have  $((\lambda x. \sum n. f\ n\ x)\ \text{has\_field\_derivative}\ g'\ z)\ (\text{at } z)$ 
by (metis (no_types, lifting) 1 r  $\langle 0 < r \rangle$  gg' has_field_derivative_transform_within_open

      open_contains_ball_eq sums_unique)
ultimately show ?thesis by auto
qed
then show ?thesis
  by meson
qed

```

Sometimes convenient to compare with a complex series of positive reals.
(?)

```

lemma series_and_derivative_comparison_complex:
  fixes S :: complex set
  assumes S: open S
    and hfd:  $\bigwedge n\ x. x \in S \implies (f\ n\ \text{has\_field\_derivative}\ f'\ n\ x)\ (\text{at } x)$ 
    and to_g:  $\bigwedge x. x \in S \implies \exists d\ h. 0 < d \wedge \text{summable } h \wedge \text{range } h \subseteq \mathbb{R}_{\geq 0} \wedge (\forall_F$ 
n in sequentially.  $\forall y \in \text{ball } x\ d \cap S. \text{cmod}(f\ n\ y) \leq \text{cmod}(h\ n)$ )
  shows  $\exists g\ g'. \forall x \in S. ((\lambda n. f\ n\ x)\ \text{sums } g\ x) \wedge ((\lambda n. f'\ n\ x)\ \text{sums } g'\ x) \wedge (g$ 
has_field_derivative g' x) (at x)
  apply (rule series_and_derivative_comparison_local [OF S hfd], assumption)
  apply (rule ex_forward [OF to_g], assumption)
  apply (erule exE)
  apply (rule_tac x=Re o h in exI)
  apply (force simp: summable_Re o_def nonneg_Reals_cmod_eq_Re image_subset_iff)
  done

```

Sometimes convenient to compare with a complex series of positive reals.
(?)

```

lemma series_differentiable_comparison_complex:
  fixes S :: complex set
  assumes S: open S
    and hfd:  $\bigwedge n\ x. x \in S \implies f\ n\ \text{field\_differentiable}\ (\text{at } x)$ 
    and to_g:  $\bigwedge x. x \in S \implies \exists d\ h. 0 < d \wedge \text{summable } h \wedge \text{range } h \subseteq \mathbb{R}_{\geq 0} \wedge (\forall_F$ 
n in sequentially.  $\forall y \in \text{ball } x\ d \cap S. \text{cmod}(f\ n\ y) \leq \text{cmod}(h\ n)$ )
  obtains g where  $\forall x \in S. ((\lambda n. f\ n\ x)\ \text{sums } g\ x) \wedge g\ \text{field\_differentiable}\ (\text{at } x)$ 
proof –
  have hfd':  $\bigwedge n\ x. x \in S \implies (f\ n\ \text{has\_field\_derivative}\ \text{deriv } (f\ n)\ x)\ (\text{at } x)$ 
    using hfd field_differentiable_derivI by blast
  show ?thesis
    by (metis field_differentiable_def that series_and_derivative_comparison_complex
      [OF S hfd' to_g])
qed

```

In particular, a power series is analytic inside circle of convergence.

```

lemma power_series_and_derivative_0:

```

```

fixes  $a :: \text{nat} \Rightarrow \text{complex}$  and  $r :: \text{real}$ 
assumes summable  $(\lambda n. a\ n * r^n)$ 
shows  $\exists g\ g'. \forall z. \text{cmod } z < r \longrightarrow$ 
 $((\lambda n. a\ n * z^n) \text{ sums } g\ z) \wedge ((\lambda n. \text{of\_nat } n * a\ n * z^{n-1}) \text{ sums } g'$ 
 $z) \wedge (g \text{ has\_field\_derivative } g' z) \text{ (at } z)$ 
proof (cases  $0 < r$ )
  case True
    have  $\text{der}: \bigwedge z. ((\lambda x. a\ x * x^n) \text{ has\_field\_derivative } \text{of\_nat } n * a\ n * x^{n-1}) \text{ (at } z)$ 
      by (rule derivative_eq_intros | simp) +
    have  $y\_le: \text{cmod } y \leq \text{cmod } (\text{of\_real } r + \text{of\_real } (\text{cmod } z)) / 2$ 
      if  $\text{cmod } (z - y) * 2 < r - \text{cmod } z$  for  $z\ y$ 
      by (smt (verit, best) field_sum_of_halves norm_minus_commute norm_of_real
norm_triangle_ineq2 of_real_add that)
    have summable  $(\lambda n. a\ n * \text{complex\_of\_real } r^n)$ 
      using assms  $\langle r > 0 \rangle$  by simp
    moreover have  $\bigwedge z. \text{cmod } z < r \implies \text{cmod } ((\text{of\_real } r + \text{of\_real } (\text{cmod } z)) / 2) < \text{cmod } (\text{of\_real } r)$ 
      using  $\langle r > 0 \rangle$ 
      by (simp flip: of_real_add)
    ultimately have  $\text{sum}: \bigwedge z. \text{cmod } z < r \implies \text{summable } (\lambda n. \text{of\_real } (\text{cmod } (a\ n)) * ((\text{of\_real } r + \text{complex\_of\_real } (\text{cmod } z)) / 2)^n)$ 
      by (rule power_series_conv_imp_absconv_weak)
    have  $\exists g\ g'. \forall z \in \text{ball } 0\ r. (\lambda n. (a\ n) * z^n) \text{ sums } g\ z \wedge$ 
 $(\lambda n. \text{of\_nat } n * (a\ n) * z^{n-1}) \text{ sums } g' z \wedge (g \text{ has\_field\_derivative } g' z) \text{ (at } z)$ 
      apply (rule series_and_derivative_comparison_complex [OF open_ball der])
      apply (rule_tac  $x = (r - \text{norm } z) / 2$  in exI)
      apply (rule_tac  $x = \lambda n. \text{of\_real}(\text{norm}(a\ n) * ((r + \text{norm } z) / 2)^n)$  in exI)
      using  $\langle r > 0 \rangle$ 
      apply (auto simp: sum_eventually_sequentially norm_mult norm_power
dist_norm intro!: mult_left_mono power_mono y_le)
    done
  then show ?thesis
    by (simp add: ball_def)
next
  case False then show ?thesis
    unfolding not_less using less_le_trans norm_not_less_zero by blast
qed

```

proposition *power_series_and_derivative*:

```

fixes  $a :: \text{nat} \Rightarrow \text{complex}$  and  $r :: \text{real}$ 
assumes summable  $(\lambda n. a\ n * r^n)$ 
obtains  $g\ g'$  where  $\forall z \in \text{ball } w\ r.$ 
 $((\lambda n. a\ n * (z - w)^n) \text{ sums } g\ z) \wedge ((\lambda n. \text{of\_nat } n * a\ n * (z - w)^{n-1}) \text{ sums } g' z) \wedge$ 
 $(g \text{ has\_field\_derivative } g' z) \text{ (at } z)$ 
using power_series_and_derivative_0 [OF assms]
apply clarify

```

```

apply (rule_tac g=( $\lambda z. g(z-w)$ ) in that)
using DERIV_shift [where  $z=-w$ ]
apply (auto simp: norm_minus_commute Ball_def dist_norm)
done

proposition power_series_holomorphic:
  assumes  $\bigwedge w. w \in \text{ball } z \ r \implies ((\lambda n. a \ n * (w-z)^{\wedge} n) \text{ sums } f \ w)$ 
  shows  $f \text{ holomorphic\_on ball } z \ r$ 
proof -
  have  $\exists f'. (f \text{ has\_field\_derivative } f') \text{ (at } w) \text{ if } w: \text{dist } z \ w < r \text{ for } w$ 
  proof -
    have  $wz: \text{cmod } (w-z) < r \text{ using } w$ 
    by (auto simp: field_split_simps dist_norm norm_minus_commute)
    then have  $0 \leq r$ 
    by (meson less_eq_real_def norm_ge_zero order_trans)
    have  $\text{inb}: z + \text{complex\_of\_real } ((\text{dist } z \ w + r) / 2) \in \text{ball } z \ r$ 
    using  $w$  by (simp add: dist_norm ‹ $0 \leq r$ › flip: of_real_add)
    have  $\text{sum}: \text{summable } (\lambda n. a \ n * \text{of\_real } (((\text{cmod } (z-w) + r) / 2)^{\wedge} n))$ 
    using  $\text{assms [OF inb]}$  by (force simp: summable_def dist_norm)
    obtain  $g \ g'$  where  $gg': \bigwedge u. u \in \text{ball } z \ ((\text{cmod } (z-w) + r) / 2) \implies$ 
       $(\lambda n. a \ n * (u-z)^{\wedge} n) \text{ sums } g \ u \wedge$ 
       $(\lambda n. \text{of\_nat } n * a \ n * (u-z)^{\wedge} (n-1)) \text{ sums } g' \ u \wedge (g$ 
 $\text{has\_field\_derivative } g' \ u) \text{ (at } u)$ 
    by (rule power_series_and_derivative [OF sum, of z]) fastforce
    have  $[\text{simp}]: g \ u = f \ u \text{ if } \text{cmod } (u-w) < (r - \text{cmod } (z-w)) / 2 \text{ for } u$ 
    proof -
      have  $\text{less}: \text{cmod } (z-u) * 2 < \text{cmod } (z-w) + r$ 
      using  $\text{that dist\_triangle2 [of } z \ u \ w]$ 
      by (simp add: dist_norm [symmetric] algebra_simps)
      have  $(\lambda n. a \ n * (u-z)^{\wedge} n) \text{ sums } g \ u \ (\lambda n. a \ n * (u-z)^{\wedge} n) \text{ sums } f \ u$ 
      using  $gg' [\text{of } u] \text{ less } w$  by (auto simp: assms dist_norm)
      then show  $?thesis$ 
      by (metis sums_unique2)
    qed
    have  $(f \text{ has\_field\_derivative } g' \ w) \text{ (at } w)$ 
  proof (rule has_field_derivative_transform_within [where  $d=(r - \text{norm}(z-w))/2$ ])
    qed (use  $w \ gg' [\text{of } w]$  in ‹(force simp: dist_norm)+›)
    then show  $?thesis ..$ 
  qed
  then show  $?thesis$  by (simp add: holomorphic_on_open)
qed

corollary holomorphic_iff_power_series:
   $f \text{ holomorphic\_on ball } z \ r \iff$ 
   $(\forall w \in \text{ball } z \ r. (\lambda n. (\text{deriv } \frown n) f \ z / (\text{fact } n) * (w-z)^{\wedge} n) \text{ sums } f \ w)$ 
using power_series_holomorphic [where  $a = \lambda n. (\text{deriv } \frown n) f \ z / (\text{fact } n)$ ]
  holomorphic_power_series
  by blast

```

$(\wedge w. w \in \text{ball } z \ r \implies (\lambda n. a \ n * (w - z) \hat{=} n) \text{ sums } f \ w) \implies f \text{ analytic_on_ball } z \ r$
by (force simp: analytic_on_open intro!: power_series_holomorphic)

$$\begin{aligned} & f \text{ analytic_on_ball } z \ r \longleftrightarrow \\ & (\forall w \in \text{ball } z \ r. (\lambda n. (\text{deriv } \frown n) f \ z / (\text{fact } n) * (w - z)^{\frown n}) \text{ sums } f \ w) \\ \mathbf{y} \text{ (simp add: analytic_on_open holomorphic_iff_power_series)} \end{aligned}$$

```

[[f holomorphic_on ball z r; g holomorphic_on ball z r;
  w ∈ ball z r;
  ∧n. (deriv  $\widehat{\sim}$  n) f z = (deriv  $\widehat{\sim}$  n) g z]]
 $\implies$  f w = g w
by (auto simp: holomorphic_iff_power_series_sums_unique2 [of λn. (deriv  $\widehat{\sim}$ 
) f z / (fact n) * (w-z) $\widehat{\sim}$ n])

```

$$\begin{aligned} & \llbracket f \text{ holomorphic_on_ball } z \ r; \ w \in \text{ball } z \ r; \\ & \quad \bigwedge^{n.} (\text{deriv } \widetilde{\sim} n) \ f \ z = 0 \rrbracket \\ & \implies f \ w = 0 \end{aligned}$$
using *holomorphic_fun_eq_on_ball* [where $g = \lambda z. 0$] **by** *simp*

assumes *holf*: *f* holomorphic_on *S* and open *S*
and *cons*: connected *S*
and *der*: $\bigwedge n. (\text{deriv } \widetilde{\sim} n) f z = 0$
and $z \in S \ w \in S$
shows $f w = 0$

$$\textbf{have } *: ball\ x\ e \subseteq (\bigcap n. \{w \in S. (deriv \widetilde{\sim} n)\ f\ w = 0\})$$

$$\textbf{if } \forall u. (deriv \widetilde{\sim} u)\ f\ x = 0\ ball\ x\ e \subseteq S\ \textbf{for } x\ e$$

have $(deriv \rightsquigarrow m) ((deriv \rightsquigarrow n) f) \ x = 0$ **for** $m \ n$
by $(metis funpow_add_o_apply \ that(1))$
then have $\bigwedge x' n. \ dist \ x \ x' < e \implies (deriv \rightsquigarrow n) \ f \ x' = 0$
using $\langle open \ S \rangle$

with *that* show *?thesis* by *auto*

obtain e where $e > 0$ and $e: \text{ball } w \text{ } e \subseteq S$ using $\text{openE} [OF \langle \text{open } S \rangle \langle w \in S \rangle]$

```
using holomorphic_on_subset by blast
have open (⋂ n. {w ∈ S. (deriv ~ n) f w = 0})
```

```

using ‹open S›
apply (simp add: open_contains_ball Ball_def image_iff)
by (metis (mono_tags) * mem_Collect_eq)
then have openin (top_of_set S) ( $\bigcap n. \{w \in S. (\text{deriv } \frown n) f w = 0\}$ )
by (force intro: open_subset)
moreover have closedin (top_of_set S) ( $\bigcap n. \{w \in S. (\text{deriv } \frown n) f w = 0\}$ )
using assms
by (auto intro: continuous_closedin_preimage_constant holomorphic_on_imp_continuous_on
holomorphic_higher_deriv)
moreover have ( $\bigcap n. \{w \in S. (\text{deriv } \frown n) f w = 0\} = S \implies f w = 0$ )
using ‹e>0› e by (force intro: holomorphic_fun_eq_0_on_ball [OF holfb])
ultimately show ?thesis
using cons der ‹z ∈ S›
by (auto simp add: connected_clopen)
qed

```

```

lemma holomorphic_fun_eq_on_connected:
  assumes f holomorphic_on S g holomorphic_on S and open S connected S
    and  $\bigwedge n. (\text{deriv } \frown n) f z = (\text{deriv } \frown n) g z$ 
    and  $z \in S \ w \in S$ 
  shows  $f w = g w$ 
proof (rule holomorphic_fun_eq_0_on_connected [of  $\lambda x. f x - g x$  S z, simplified])
  show  $(\lambda x. f x - g x)$  holomorphic_on S
    by (intro assms holomorphic_intros)
  show  $\bigwedge n. (\text{deriv } \frown n) (\lambda x. f x - g x) z = 0$ 
    using assms higher_deriv_diff by auto
qed (use assms in auto)

```

```

lemma holomorphic_fun_eq_const_on_connected:
  assumes holf: f holomorphic_on S and open S
    and cons: connected S
    and der:  $\bigwedge n. 0 < n \implies (\text{deriv } \frown n) f z = 0$ 
    and  $z \in S \ w \in S$ 
  shows  $f w = f z$ 
proof (rule holomorphic_fun_eq_0_on_connected [of  $\lambda w. f w - f z$  S z, simplified])
  show  $(\lambda w. f w - f z)$  holomorphic_on S
    by (intro assms holomorphic_intros)
  show  $\bigwedge n. (\text{deriv } \frown n) (\lambda w. f w - f z) z = 0$ 
    by (subst higher_deriv_diff) (use assms in ‹auto intro: holomorphic_intros›)
qed (use assms in auto)

```

4.12 Some basic lemmas about poles/singularities

```

lemma pole_lemma:
  assumes holf: f holomorphic_on S and a:  $a \in \text{interior } S$ 
  shows  $(\lambda z. \text{if } z = a \text{ then deriv } f a$ 
     $\text{else } (f z - f a) / (z - a))$  holomorphic_on S (is ?F holomorphic_on S)

```

```

proof -
  have *: ?F field_differentiable (at u within S) if  $u \in S$   $u \neq a$  for u
  proof -
    have fcd: f field_differentiable at u within S
      using hol_f holomorphic_on_def by (simp add: ⟨u ∈ S⟩)
    have cd: (λz. (f z - f a) / (z-a)) field_differentiable at u within S
      by (rule fcd derivative_intros | simp add: that)+
    have 0 < dist a u using that dist_nz by blast
    then show ?thesis
      by (rule field_differentiable_transform_within [OF _ _ _ cd]) (auto simp:
        ⟨u ∈ S⟩)
    qed
  moreover
    have ?F field_differentiable at a if 0 < e ball a e ⊆ S for e
    proof -
      have holfb: f holomorphic_on ball a e
        by (rule holomorphic_on_subset [OF hol_f ⟨ball a e ⊆ S⟩])
      have 2: ?F holomorphic_on ball a e - {a}
        using mem_ball that
      by (auto simp add: holomorphic_on_def simp flip: field_differentiable_def
        intro: * field_differentiable_within_subset)
      have isCont (λz. if z = a then deriv f a else (f z - f a) / (z-a)) x
        if dist a x < e for x
      proof (cases x=a)
        case True
          then have f field_differentiable at a
            using holfb ⟨0 < e⟩ holomorphic_on_imp_differentiable_at by auto
          with True show ?thesis
            by (smt (verit) DERIV_deriv_iff_field_differentiable LIM_equal continuous_at has_field_derivativeD)
        next
          case False with 2 that show ?thesis
            by (simp add: field_differentiable_imp_continuous_at holomorphic_on_imp_differentiable_at
              open_Diff)
      qed
      then have 1: continuous_on (ball a e) ?F
        by (clarsimp simp: continuous_on_eq_continuous_at)
      have ?F holomorphic_on ball a e
        by (auto intro: no_isolated_singularity [OF 1 2])
      with that show ?thesis
        by (simp add: holomorphic_on_imp_differentiable_at)
      qed
    ultimately show ?thesis
      by (metis (lifting) a at_within_interior holomorphic_onI mem_interior)
    qed

lemma pole_theorem:
  assumes holg: g holomorphic_on S and a: a ∈ interior S
  and eq:  $\bigwedge z. z \in S - \{a\} \implies g z = (z-a) * f z$ 

```



```

    shows ( $\lambda z. \text{if } z = a \text{ then deriv } g \ a$ 
            $\text{else } f \ z - g \ a / (z - a)$ ) holomorphic_on S
  using pole_lemma [OF holg a]
  by (rule holomorphic_transform) (simp add: eq_field_split_simps)

lemma pole_lemma_open:
  assumes f holomorphic_on S open S
  shows ( $\lambda z. \text{if } z = a \text{ then deriv } f \ a \text{ else } (f \ z - f \ a) / (z - a)$ ) holomorphic_on S
proof (cases a  $\in S$ )
  case True with assms interior_eq pole_lemma
    show ?thesis by fastforce
  next
  case False
  then have ( $\lambda z. (f \ z - f \ a) / (z - a)$ ) field_differentiable at x within S
    if x  $\in S$  for x
  using assms that unfolding holomorphic_on_def
  by (intro derivative_intros) auto
  with False show ?thesis
  using holomorphic_on_def holomorphic_transform by presburger
qed

lemma pole_theorem_open:
  assumes holg: g holomorphic_on S and S: open S
  and eq:  $\bigwedge z. z \in S - \{a\} \implies g \ z = (z - a) * f \ z$ 
  shows ( $\lambda z. \text{if } z = a \text{ then deriv } g \ a$ 
          $\text{else } f \ z - g \ a / (z - a)$ ) holomorphic_on S
  using pole_lemma_open [OF holg S]
  by (rule holomorphic_transform) (auto simp: eq_divide_simps)

lemma pole_theorem_0:
  assumes holg: g holomorphic_on S and a: a  $\in$  interior S
  and eq:  $\bigwedge z. z \in S - \{a\} \implies g \ z = (z - a) * f \ z$ 
  and [simp]: f a = deriv g a g a = 0
  shows f holomorphic_on S
  using pole_theorem [OF holg a eq]
  by (rule holomorphic_transform) (auto simp: eq_field_split_simps)

lemma pole_theorem_open_0:
  assumes holg: g holomorphic_on S and S: open S
  and eq:  $\bigwedge z. z \in S - \{a\} \implies g \ z = (z - a) * f \ z$ 
  and [simp]: f a = deriv g a g a = 0
  shows f holomorphic_on S
  using pole_theorem_open [OF holg S eq]
  by (rule holomorphic_transform) (auto simp: eq_field_split_simps)

lemma pole_theorem_analytic:
  assumes g: g analytic_on S
  and eq:  $\bigwedge z. z \in S$ 
            $\implies \exists d. 0 < d \wedge (\forall w \in \text{ball } z \ d - \{a\}. g \ w = (w - a) * f \ w)$ 

```

```

  shows  $(\lambda z. \text{if } z = a \text{ then deriv } g \ a \text{ else } f \ z - g \ a / (z - a)) \text{ analytic\_on } S$  (is ?F analytic_on S)
  unfolding analytic_on_def
proof
  fix x
  assume  $x \in S$ 
  with g obtain e where  $0 < e$  and  $e: g \text{ holomorphic\_on ball } x \ e$ 
  by (auto simp add: analytic_on_def)
  obtain d where  $0 < d$  and  $d: \bigwedge w. w \in \text{ball } x \ d - \{a\} \implies g \ w = (w - a) * f \ w$ 
  using  $\langle x \in S \rangle \text{ eq}$  by blast
  have ?F holomorphic_on ball x (min d e)
  using d e  $\langle x \in S \rangle$  by (fastforce simp: holomorphic_on_subset subset_ball intro!: pole_theorem_open)
  then show  $\exists e > 0. ?F \text{ holomorphic\_on ball } x \ e$ 
  using  $\langle 0 < d \rangle \langle 0 < e \rangle \text{ not\_le}$  by fastforce
qed

```

```

lemma pole_theorem_analytic_0:
  assumes  $g: g \text{ analytic\_on } S$ 
  and eq:  $\bigwedge z. z \in S \implies \exists d. 0 < d \wedge (\forall w \in \text{ball } z \ d - \{a\}. g \ w = (w - a) * f \ w)$ 
  and [simp]:  $f \ a = \text{deriv } g \ a \ g \ a = 0$ 
  shows  $f \text{ analytic\_on } S$ 
proof -
  have [simp]:  $(\lambda z. \text{if } z = a \text{ then deriv } g \ a \text{ else } f \ z - g \ a / (z - a)) = f$ 
  by auto
  show ?thesis
  using pole_theorem_analytic [OF g eq] by simp
qed

```

```

lemma pole_theorem_analytic_open_superset:
  assumes  $g: g \text{ analytic\_on } S$  and  $S \subseteq T \text{ open } T$ 
  and eq:  $\bigwedge z. z \in T - \{a\} \implies g \ z = (z - a) * f \ z$ 
  shows  $(\lambda z. \text{if } z = a \text{ then deriv } g \ a \text{ else } f \ z - g \ a / (z - a)) \text{ analytic\_on } S$ 
proof (rule pole_theorem_analytic [OF g])
  fix z
  assume  $z \in S$ 
  then obtain e where  $0 < e$  and  $e: \text{ball } z \ e \subseteq T$ 
  using assms openE by blast
  then show  $\exists d > 0. \forall w \in \text{ball } z \ d - \{a\}. g \ w = (w - a) * f \ w$ 
  using eq by auto
qed

```

```

lemma pole_theorem_analytic_open_superset_0:
  assumes  $g: g \text{ analytic\_on } S \ S \subseteq T \text{ open } T \bigwedge z. z \in T - \{a\} \implies g \ z = (z - a) * f \ z$ 
  and [simp]:  $f \ a = \text{deriv } g \ a \ g \ a = 0$ 
  shows  $f \text{ analytic\_on } S$ 
proof -

```

```

have [simp]: ( $\lambda z. \text{if } z = a \text{ then deriv } g \ a \text{ else } f \ z - g \ a / (z-a) = f$ )
  by auto
have ( $\lambda z. \text{if } z = a \text{ then deriv } g \ a \text{ else } f \ z - g \ a / (z-a)$ ) analytic_on S
  by (rule pole_theorem_analytic_open_superset [OF g])
then show ?thesis by simp
qed

```

4.13 General, homology form of Cauchy's theorem

Proof is based on Dixon's, as presented in Lang's "Complex Analysis" book (page 147).

lemma *contour_integral_continuous_on_linepath_2D:*

assumes *open U and cont_dw: $\bigwedge w. w \in U \implies F \ w \text{ contour_integrable_on } (\text{linepath } a \ b)$*

and *cond_uw: continuous_on $(U \times U) (\lambda(x,y). F \ x \ y)$*

and *abu: closed_segment a b $\subseteq U$*

shows *continuous_on U $(\lambda w. \text{contour_integral } (\text{linepath } a \ b) (F \ w))$*

proof –

have $\exists d > 0. \forall x' \in U. \text{dist } x' \ w < d \implies$
 $\text{dist } (\text{contour_integral } (\text{linepath } a \ b) (F \ x'))$
 $(\text{contour_integral } (\text{linepath } a \ b) (F \ w)) \leq \varepsilon$
if $w \in U \ 0 < \varepsilon \ a \neq b$ **for** $w \ \varepsilon$

proof –

obtain δ **where** $\delta > 0$ **and** δ : *cball w $\delta \subseteq U$ using open_contains_cball $\langle \text{open } U \rangle \langle w \in U \rangle$ by force*

let $?TZ = \text{cball } w \ \delta \times \text{closed_segment } a \ b$

have *uniformly_continuous_on ?TZ $(\lambda(x,y). F \ x \ y)$*

by (*metis Sigma_mono δ abu compact_Times compact_cball compact_segment compact_uniformly_continuous*

cond_uw continuous_on_subset)

then obtain η **where** $\eta > 0$

and η : $\bigwedge x \ x'. \llbracket x \in ?TZ; x' \in ?TZ; \text{dist } x' \ x < \eta \rrbracket \implies$
 $\text{dist } ((\lambda(x,y). F \ x \ y) \ x') ((\lambda(x,y). F \ x \ y) \ x) < \varepsilon / \text{norm}(b-a)$

using $\langle 0 < \varepsilon \rangle \langle a \neq b \rangle$

by (*auto elim: uniformly_continuous_onE [where $\varepsilon = \varepsilon / \text{norm}(b-a)$]*)

have η : $\llbracket \text{norm } (w - x1) \leq \delta; \ x2 \in \text{closed_segment } a \ b;$

$\text{norm } (w - x1') \leq \delta; \ x2' \in \text{closed_segment } a \ b; \text{norm } ((x1', x2') -$
 $(x1, x2)) < \eta \rrbracket$

$\implies \text{norm } (F \ x1' \ x2' - F \ x1 \ x2) \leq \varepsilon / \text{cmod } (b-a)$

for $x1 \ x2 \ x1' \ x2'$

using η [*of $(x1, x2) (x1', x2')$ by (force simp: dist_norm)*]

have le_ee : $\text{cmod } (\text{contour_integral } (\text{linepath } a \ b) (\lambda x. F \ x' \ x - F \ w \ x)) \leq \varepsilon$

if $x' \in U$ $\text{cmod } (x' - w) < \delta$ $\text{cmod } (x' - w) < \eta$ **for** x'

proof –

have $(\lambda x. F \ x' \ x - F \ w \ x)$ *contour_integrable_on linepath a b*

by (*simp add: $\langle w \in U \rangle \text{cont_dw contour_integrable_diff that}$*)

then have $\text{cmod } (\text{contour_integral } (\text{linepath } a \ b) (\lambda x. F \ x' \ x - F \ w \ x)) \leq$

$\varepsilon / \text{norm}(b-a) * \text{norm}(b-a)$

using *has_contour_integral_bound_linepath [OF has_contour_integral_integral*

```

  _  $\eta$ ]
    using  $\langle 0 < \varepsilon \rangle \langle 0 < \delta \rangle$  that by (force simp: norm_minus_commute)
    also have  $\dots = \varepsilon$  using  $\langle a \neq b \rangle$  by simp
    finally show ?thesis .
  qed
  show ?thesis
    using  $\langle 0 < \delta \rangle \langle 0 < \eta \rangle \langle w \in U \rangle$ 
    apply (intro exI[where  $x = \min \delta \eta$ ])
    by (auto simp: dist_norm contour_integral_diff [OF cont_dw cont_dw,
symmetric] intro: le_ee)
  qed
  then show ?thesis
    by (metis (no_types, lifting) continuous_onI continuous_on_iff
contour_integral_trivial dist_self)
qed

```

This version has *polynomial_function* γ as an additional assumption.

lemma *Cauchy_integral_formula_global_weak*:

```

  assumes open U and holf: f holomorphic_on U
    and z:  $z \in U$  and  $\gamma$ : polynomial_function  $\gamma$ 
    and pasz:  $\text{path\_image } \gamma \subseteq U - \{z\}$  and loop:  $\text{pathfinish } \gamma = \text{pathstart } \gamma$ 
    and zero:  $\bigwedge w. w \notin U \implies \text{winding\_number } \gamma w = 0$ 
  shows  $((\lambda w. f w / (w - z)) \text{ has\_contour\_integral } (2 * \pi * i * \text{winding\_number } \gamma z * f z)) \gamma$ 
proof -
  obtain  $\gamma'$  where pf $\gamma'$ : polynomial_function  $\gamma'$  and  $\gamma'$ :  $\bigwedge x. (\gamma \text{ has\_vector\_derivative } (\gamma' x)) \text{ (at } x)$ 
    using has_vector_derivative_polynomial_function [OF  $\gamma$ ] by blast
  then have bounded(path_image  $\gamma'$ )
    by (simp add: path_image_def compact_imp_bounded compact_continuous_image
continuous_on_polynomial_function)
  then obtain B where  $B > 0$  and B:  $\bigwedge x. x \in \text{path\_image } \gamma' \implies \text{norm } x \leq B$ 
    using bounded_pos by force
  define d where [abs_def]:  $d z w = (\text{if } w = z \text{ then deriv } f z \text{ else } (f w - f z) / (w - z))$ 
for z w
  define v where  $v = \{w. w \notin \text{path\_image } \gamma \wedge \text{winding\_number } \gamma w = 0\}$ 
  have path  $\gamma$  valid_path  $\gamma$  using  $\gamma$ 
    by (auto simp: path_polynomial_function valid_path_polynomial_function)
  then have ov: open v
    by (simp add: v_def open_winding_number_levelsets loop)
  have uv_Un:  $U \cup v = \text{UNIV}$ 
    using pasz zero by (auto simp: v_def)
  have conf: continuous_on U f
    by (metis holf holomorphic_on_imp_continuous_on)
  have holf_d:  $(d y)$  holomorphic_on U if  $y \in U$  for y
proof -
  have *:  $(\lambda c. \text{if } c = y \text{ then deriv } f y \text{ else } (f c - f y) / (c - y))$  holomorphic_on U
    by (simp add: holf pole_lemma_open (open U))
  then have isCont  $(\lambda x. \text{if } x = y \text{ then deriv } f y \text{ else } (f x - f y) / (x - y)) y$ 

```

```

      using at_within_open field_differentiable_imp_continuous_at holomor-
phic_on_def that ⟨open U⟩ by fastforce
    then have continuous_on U (d y)
      using * d_def holomorphic_on_imp_continuous_on by auto
    moreover have d y holomorphic_on U - {y}
    proof -
      have (λw. if w = y then deriv f y else (f w - f y) / (w - y)) field_differentiable
at w
      if w ∈ U - {y} for w
    proof (rule field_differentiable_transform_within)
      show (λw. (f w - f y) / (w - y)) field_differentiable at w
        using that ⟨open U⟩ holf
        by (auto intro!: holomorphic_on_imp_differentiable_at derivative_intros)
      show dist w y > 0
        using that by auto
    qed (auto simp: dist_commute)
    then show ?thesis
      unfolding field_differentiable_def by (simp add: d_def holomorphic_on_open
⟨open U⟩ open_delete)
    qed
    ultimately show ?thesis
      by (rule no_isolated_singularity) (auto simp: ⟨open U⟩)
    qed
    have cint_fxy: (λx. (f x - f y) / (x - y)) contour_integrable_on γ if y ∉
path_image γ for y
    proof (rule contour_integrable_holomorphic_simple [where S = U - {y}])
      show (λx. (f x - f y) / (x - y)) holomorphic_on U - {y}
        by (force intro: holomorphic_intros holomorphic_on_subset [OF holf])
      show path_image γ ⊆ U - {y}
        using pasz that by blast
    qed (auto simp: ⟨open U⟩ open_delete ⟨valid_path γ⟩)
    define h where
      h z = (if z ∈ U then contour_integral γ (d z) else contour_integral γ (λw. f
w / (w - z))) for z
    have U: ((d z) has_contour_integral h z) γ if z ∈ U for z
    proof -
      have d z holomorphic_on U
        by (simp add: hol_d that)
      with that show ?thesis
        by (metis Diff_subset ⟨valid_path γ⟩ ⟨open U⟩ contour_integrable_holomorphic_simple
h_def
has_contour_integral_integral pasz subset_trans)
    qed
    have V: ((λw. f w / (w - z)) has_contour_integral h z) γ if z: z ∈ v for z
    proof -
      have 0: 0 = (f z) * 2 * of_real (2 * pi) * i * winding_number γ z
        using v_def z by auto
      then have ((λx. 1 / (x - z)) has_contour_integral 0) γ
        using z v_def has_contour_integral_winding_number [OF ⟨valid_path γ⟩]

```

```

by fastforce
  then have  $((\lambda x. f z * (1 / (x - z))) \text{ has\_contour\_integral } 0) \gamma$ 
    using has_contour_integral_lmul by fastforce
  then have  $((\lambda x. f z / (x - z)) \text{ has\_contour\_integral } 0) \gamma$ 
    by (simp add: field_split_simps)
  moreover have  $((\lambda x. (f x - f z) / (x - z)) \text{ has\_contour\_integral } \text{contour\_integral } \gamma (d z)) \gamma$ 
    by (metis (no_types, lifting) z cint_fxy contour_integral_eq d_def has_contour_integral_integral
    mem_Collect_eq v_def)
  ultimately have *:  $((\lambda x. f z / (x - z) + (f x - f z) / (x - z)) \text{ has\_contour\_integral } (0 + \text{contour\_integral } \gamma (d z))) \gamma$ 
    by (rule has_contour_integral_add)
  have  $((\lambda w. f w / (w - z)) \text{ has\_contour\_integral } \text{contour\_integral } \gamma (d z)) \gamma$ 
    if  $z \in U$ 
    using * by (auto simp: divide_simps has_contour_integral_eq)
  moreover have  $((\lambda w. f w / (w - z)) \text{ has\_contour\_integral } \text{contour\_integral } \gamma (\lambda w. f w / (w - z))) \gamma$ 
    if  $z \notin U$ 
  proof (rule has_contour_integral_integral [OF contour_integrable_holomorphic_simple
  where  $S=U$ ])
    show  $(\lambda w. f w / (w - z)) \text{ holomorphic\_on } U$ 
      by (rule holomorphic_intros assms | use that in force)+
    qed (use ⟨open U⟩ pasz ⟨valid_path γ⟩ in auto)
    ultimately show ?thesis
      using z by (simp add: h_def)
    qed
    have  $z \text{ not: } z \notin \text{path\_image } \gamma$ 
      using pasz by blast
    obtain  $d0$  where  $d0 > 0$  and  $d0: \bigwedge x y. x \in \text{path\_image } \gamma \implies y \in - U \implies d0 \leq \text{dist } x y$ 
      using separate_compact_closed [of path_image γ - U] pasz ⟨open U⟩ ⟨path γ⟩ compact_path_image
      by blast
    obtain  $dd$  where  $0 < dd$  and  $dd: \{y + k \mid y k. y \in \text{path\_image } \gamma \wedge k \in \text{ball } 0 dd\} \subseteq U$ 
    proof
      show  $0 < d0 / 2$  using ⟨ $0 < d0$ ⟩ by auto
    qed (use ⟨ $0 < d0$ ⟩  $d0$  in ⟨force simp: dist_norm⟩)
    define  $T$  where  $T \equiv \{y + k \mid y k. y \in \text{path\_image } \gamma \wedge k \in \text{cball } 0 (dd / 2)\}$ 
    have  $\bigwedge x x'. \llbracket x \in \text{path\_image } \gamma; \text{dist } x x' * 2 < dd \rrbracket \implies \exists y k. x' = y + k \wedge y \in \text{path\_image } \gamma \wedge \text{dist } 0 k * 2 \leq dd$ 
      by (metis add.commute diff_add_cancel dist_0_norm dist_commute dist_norm
      less_eq_real_def)
    then have  $\text{subt: path\_image } \gamma \subseteq \text{interior } T$ 
      using ⟨ $0 < dd$ ⟩
      apply (clarsimp simp add: mem_interior T_def)
      apply (rule_tac  $x=dd/2$  in exI, auto)
    done
    have compact T

```

```

    unfolding T_def
    using ⟨valid_path γ⟩ compact_cball compact_sums compact_valid_path_image
  by blast
  have T:  $T \subseteq U$ 
    unfolding T_def using ⟨0 < dd⟩ dd by fastforce
  obtain L where L>0
    and L:  $\bigwedge f B. \llbracket f \text{ holomorphic\_on interior } T; \bigwedge z. z \in \text{interior } T \implies \text{cmod } (f z) \leq B \rrbracket \implies$ 
       $\text{cmod } (\text{contour\_integral } \gamma f) \leq L * B$ 
    using contour_integral_bound_exists [OF open_interior ⟨valid_path γ⟩ subt]
    by blast
  have bounded(f ‘ T)
    by (meson ⟨compact T⟩ compact_continuous_image compact_imp_bounded
  conf continuous_on_subset T)
  then obtain D where D>0 and D:  $\bigwedge x. x \in T \implies \text{norm } (f x) \leq D$ 
    by (auto simp: bounded_pos)
  obtain C where C>0 and C:  $\bigwedge x. x \in T \implies \text{norm } x \leq C$ 
    using ⟨compact T⟩ bounded_pos compact_imp_bounded by force
  have dist (h y) 0 ≤ e if 0 < e and le:  $D * L / e + C \leq \text{cmod } y$  for e y
  proof -
    have  $D * L / e > 0$  using ⟨D>0⟩ ⟨L>0⟩ ⟨e>0⟩ by simp
    with le have ybig:  $\text{norm } y > C$  by force
    with C have y ∉ T by force
    then have ynot:  $y \notin \text{path\_image } \gamma$ 
      using subt_interior_subset by blast
    have [simp]:  $\text{winding\_number } \gamma y = 0$ 
    proof (rule winding_number_zero_outside)
      show  $\text{path\_image } \gamma \subseteq \text{cball } 0 C$ 
      by (meson C interior_subset mem_cball_0 subset_eq subt)
    qed (use ybig loop ⟨path γ⟩ in auto)
    have [simp]:  $h y = \text{contour\_integral } \gamma (\lambda w. f w / (w - y))$ 
      by (rule contour_integral_unique [symmetric]) (simp add: v_def ynot V)
    have holint:  $(\lambda w. f w / (w - y)) \text{ holomorphic\_on interior } T$ 
    proof (intro holomorphic_intros)
      show  $f \text{ holomorphic\_on interior } T$ 
      using holf holomorphic_on_subset interior_subset T by blast
    qed (use ⟨y ∉ T⟩ interior_subset in auto)
    have leD:  $\text{cmod } (f z / (z - y)) \leq D * (e / L / D)$  if  $z: z \in \text{interior } T$  for z
    proof -
      have  $D * L / e + \text{cmod } z \leq \text{cmod } y$ 
      using le C [of z] z using interior_subset by force
      then have DL2:  $D * L / e \leq \text{cmod } (z - y)$ 
      using norm_triangle_ineq2 [of y z] by (simp add: norm_minus_commute)
      have  $\text{cmod } (f z / (z - y)) = \text{cmod } (f z) * \text{inverse } (\text{cmod } (z - y))$ 
      by (simp add: norm_mult norm_inverse Fields.field_class.field_divide_inverse)
      also have  $\dots \leq D * (e / L / D)$ 
      proof (rule mult_mono)
        show  $\text{cmod } (f z) \leq D$ 
        using D interior_subset z by blast
      qed
    qed
  end

```

```

    show inverse (cmod (z-y)) ≤ e / L / D D ≥ 0
    using ‹L>0› ‹e>0› ‹D>0› DL2 by (auto simp: norm_divide field_split_simps)
  qed auto
  finally show ?thesis .
qed
have dist (h y) 0 = cmod (contour_integral γ (λw. f w / (w-y)))
  by (simp add: dist_norm)
also have ... ≤ L * (D * (e / L / D))
  by (rule L [OF holint leD])
also have ... = e
  using ‹L>0› ‹0 < D› by auto
finally show ?thesis .
qed
then have (h ⟶ 0) at_infinity
  by (meson Lim_at_infinityI)
moreover have h holomorphic_on UNIV
proof -
  have con_ff: continuous (at (x,z)) (λ(x,y). (f y - f x) / (y-x))
    if x ∈ U z ∈ U x ≠ z for x z
  using that conf
  apply (simp add: split_def continuous_on_eq_continuous_at ‹open U›)
  apply (simp | rule continuous_intros continuous_within_compose2 [where
g=f])+
  done
  have con_fstsd: continuous_on UNIV (λx. (fst x - snd x) :: complex)
    by (rule continuous_intros)+
  have open_uu_Id: open (U × U - Id)
  proof (rule open_Diff)
    show open (U × U)
      by (simp add: open_Times ‹open U›)
    show closed (Id :: complex rel)
    using continuous_closed_preimage_constant [OF con_fstsd closed_UNIV,
of 0]
      by (auto simp: Id_fstsd_eq algebra_simps)
  qed
  have con_derf: continuous (at z) (deriv f) if z ∈ U for z
    by (meson analytic_at analytic_at_imp_isCont assms(1) holf holomor-
phic_deriv that)
  have tendsto_f': ((λ(x,y). if y = x then deriv f (x)
                        else (f (y) - f (x)) / (y-x)) ⟶ deriv f x)
    (at (x, x) within U × U) if x ∈ U for x
  proof (rule Lim_withinI)
    fix e::real assume 0 < e
    obtain k1 where k1>0 and k1: ∧x'. norm (x' - x) ≤ k1 ⟹ norm (deriv
f x' - deriv f x) < e
    using ‹0 < e› continuous_within_E [OF con_derf [OF ‹x ∈ U›]]
    by (metis UNIV_I dist_norm)
    obtain k2 where k2>0 and k2: ball x k2 ⊆ U
    by (blast intro: openE [OF ‹open U›] ‹x ∈ U›)

```



```

have neq: norm ((f z' - f x') / (z' - x') - deriv f x) ≤ e
      if z' ≠ x' and less_k1: norm (x'-x, z'-x) < k1 and less_k2:
norm (x'-x, z'-x) < k2
      for x' z'
proof -
  have cs_less: w ∈ closed_segment x' z' ⇒ cmod (w-x) ≤ norm (x'-x,
z'-x) for w
    using segment_furthest_le [of w x' z' x]
    by (metis (no_types) dist_commute dist_norm norm_fst_le norm_snd_le
order_trans)
  have derf_le: w ∈ closed_segment x' z' ⇒ z' ≠ x' ⇒ cmod (deriv f w -
deriv f x) ≤ e for w
    by (blast intro: cs_less less_k1 k1 [unfolded divide_const_simps dist_norm]
less_imp_le le_less_trans)
  have f_has_der: ∧x. x ∈ U ⇒ (f has_field_derivative deriv f x) (at x
within U)
    by (metis DERIV_deriv_iff_field_differentiable at_within_open holf
holomorphic_on_def ‹open U›)
  have closed_segment x' z' ⊆ U
    by (rule order_trans [OF _ k2]) (simp add: cs_less le_less_trans [OF _
less_k2] dist_complex_def norm_minus_commute subset_iff)
  then have cint_derf: (deriv f has_contour_integral f z' - f x') (linepath x'
z')
    using contour_integral_primitive [OF f_has_der valid_path_linepath]
pasz by simp
  then have *: ((λx. deriv f x / (z' - x')) has_contour_integral (f z' - f x')
/ (z' - x')) (linepath x' z')
    by (rule has_contour_integral_div)
  have norm ((f z' - f x') / (z' - x') - deriv f x) ≤ e/norm(z' - x') *
norm(z' - x')
    apply (rule has_contour_integral_bound_linepath [OF has_contour_integral_diff
[OF *]])
    using has_contour_integral_div [where c = z' - x', OF has_contour_integral_const_linepath
[of deriv f x z' x']]
    ‹e > 0› ‹z' ≠ x'›
    apply (auto simp: norm_divide divide_simps derf_le)
    done
  also have ... ≤ e using ‹0 < e› by simp
  finally show ?thesis .
qed
show ∃ d>0. ∀ xa∈U × U.
  0 < dist xa (x, x) ∧ dist xa (x, x) < d ⇒
  dist (case xa of (x, y) ⇒ if y = x then deriv f x else (f y - f x) /
(y-x)) (deriv f x) ≤ e
    apply (rule_tac x=min k1 k2 in exI)
    using ‹k1>0› ‹k2>0› ‹e>0›
    by (force simp: dist_norm neq intro: dual_order.strict_trans2 k1 less_imp_le
norm_fst_le)
qed

```

```

have con_pa_f: continuous_on (path_image  $\gamma$ ) f
by (meson holf holomorphic_on_imp_continuous_on holomorphic_on_subset
interior_subset subt T)
have le_B:  $\bigwedge T. T \in \{0..1\} \implies \text{cmod } (\text{vector\_derivative } \gamma \text{ (at } T)) \leq B$ 
using  $\gamma'$  B by (simp add: path_image_def vector_derivative_at rev_image_eqI)
have f_has_cint:  $\bigwedge w. w \in v - \text{path\_image } \gamma \implies ((\lambda u. f u / (u-w)) ^ 1)$ 
has_contour_integral h w)  $\gamma$ 
by (simp add: V)
have  $\bigwedge x y. \llbracket x \in U; y \in U; y \neq x \rrbracket \implies (\lambda(x, y). d x y) - (x, y) \rightarrow (f y - f x) / (y - x)$ 
unfolding d_def
apply (rule Lim_transform_within_open [OF _ open_uu_Id, where f =
 $(\lambda(x, y). (f y - f x) / (y-x))$ ])
using con_ff by (auto simp: continuous_within)
then have cond_uu: continuous_on  $(U \times U)$   $(\lambda(x, y). d x y)$ 
unfolding continuous_on_eq_continuous_within continuous_within_d_def
by (fastforce simp add: tendsto_f' intro: Lim_at_imp_Lim_at_within)
have hol_dw:  $(\lambda z. d z w)$  holomorphic_on U if  $w \in U$  for w
proof -
have continuous_on U  $((\lambda(x, y). d x y) \circ (\lambda z. (w, z)))$ 
by (rule continuous_on_compose continuous_intros continuous_on_subset
[OF cond_uu] | force intro: that)+
then have *: continuous_on U  $(\lambda z. \text{if } w = z \text{ then deriv } f z \text{ else } (f w - f z) / (w-z))$ 
by (rule rev_iffD1 [OF _ continuous_on_cong [OF refl]]) (simp add: d_def
field_simps)
have **:  $(\lambda z. \text{if } w = z \text{ then deriv } f z \text{ else } (f w - f z) / (w-z))$  field_differentiable
at x
if  $x \in U$   $x \neq w$  for x
proof (rule_tac f =  $\lambda x. (f w - f x) / (w-x)$  and d = dist x w in field_differentiable_transform_within)
show  $(\lambda x. (f w - f x) / (w-x))$  field_differentiable at x
using that <open U>
by (intro derivative_intros holomorphic_on_imp_differentiable_at [OF
holf]; force)
qed (use that <open U> in <auto simp: dist_commute>)
show ?thesis
unfolding d_def
proof (rule no_isolated_singularity [OF * _ <open U>])
show  $(\lambda z. \text{if } w = z \text{ then deriv } f z \text{ else } (f w - f z) / (w-z))$  holomorphic_on
 $U - \{w\}$ 
by (auto simp: field_differentiable_def [symmetric] holomorphic_on_open
open_Diff <open U> **)
qed auto
qed
{ fix a b
assume abu: closed_segment a b  $\subseteq U$ 
have cont_cint_d: continuous_on U  $(\lambda w. \text{contour\_integral } (\text{linepath } a b)$ 
 $(\lambda z. d z w))$ 
proof (rule contour_integral_continuous_on_linepath_2D [OF <open U> _
```

```

  _ abu])
  show  $\bigwedge w. w \in U \implies (\lambda z. d\ z\ w)$  contour_integrable_on (linepath a b)
  by (metis abu hol_dw continuous_on_subset contour_integrable_continuous_linepath
holomorphic_on_imp_continuous_on)
  show continuous_on ( $U \times U$ )  $(\lambda(x, y). d\ y\ x)$ 
  by (auto intro: continuous_on_swap_args cond_uu)
  qed
  have cont_cint_d $\gamma$ : continuous_on {0..1}  $((\lambda w. \text{contour\_integral } (\text{linepath } a\ b) (\lambda z. d\ z\ w)) \circ \gamma)$ 
  by (metis Diff_subset  $\langle \text{path } \gamma \rangle$  cont_cint_d continuous_on_compose continuous_on_subset
pasz path_def path_image_def)
  have continuous_on {0..1}  $(\lambda x. \text{vector\_derivative } \gamma\ (\text{at } x))$ 
  using pf $\gamma'$  by (simp add: continuous_on_polynomial_function vector\_derivative_at
[OF  $\gamma'$ ])
  then have cint_cint:  $(\lambda w. \text{contour\_integral } (\text{linepath } a\ b) (\lambda z. d\ z\ w))$ 
contour_integrable_on  $\gamma$ 
  apply (simp add: contour_integrable_on)
  apply (rule integrable_continuous_real)
  by (rule continuous_on_mult [OF cont_cint_d  $\gamma$  [unfolded o_def]])
  have contour_integral (linepath a b) h = contour_integral (linepath a b)  $(\lambda z. \text{contour\_integral } \gamma\ (d\ z))$ 
  using abu by (force simp: h_def intro: contour_integral_eq)
  also have ... = contour_integral  $\gamma\ (\lambda w. \text{contour\_integral } (\text{linepath } a\ b) (\lambda z. d\ z\ w))$ 
  proof (rule contour_integral_swap)
    show continuous_on (path_image (linepath a b)  $\times$  path_image  $\gamma$ )  $(\lambda(y1, y2). d\ y1\ y2)$ 
    using abu pasz by (auto intro: continuous_on_subset [OF cond_uu])
    show continuous_on {0..1}  $(\lambda t. \text{vector\_derivative } (\text{linepath } a\ b)\ (\text{at } t))$ 
    by (auto intro!: continuous_intros)
    show continuous_on {0..1}  $(\lambda t. \text{vector\_derivative } \gamma\ (\text{at } t))$ 
    by (metis  $\gamma'$  continuous_on_eq path_def path_polynomial_function pf $\gamma'$ 
vector\_derivative_at)
  qed (use  $\langle \text{valid\_path } \gamma \rangle$  in auto)
  finally have cint_h_eq:
    contour_integral (linepath a b) h =
      contour_integral  $\gamma\ (\lambda w. \text{contour\_integral } (\text{linepath } a\ b) (\lambda z. d\ z\ w))$  .
  note cint_cint_cint_h_eq
} note cint_h = this
have conth_u: continuous_on  $U\ h$ 
proof (simp add: continuous_on_sequentially, clarify)
  fix a x
  assume x:  $x \in U$  and au:  $\forall n. a\ n \in U$  and ax:  $a \longrightarrow x$ 
  then have A1:  $\forall_F n$  in sequentially.  $d\ (a\ n)$  contour_integrable_on  $\gamma$ 
  by (meson  $U$  contour_integrable_on_def eventuallyI)
  obtain dd where dd>0 and dd: cball x dd  $\subseteq U$  using open_contains_cball
 $\langle \text{open } U \rangle\ x$  by force
  have A2: uniform_limit (path_image  $\gamma$ )  $(\lambda n. d\ (a\ n))\ (d\ x)$  sequentially

```

```

    unfolding uniform_limit_iff dist_norm
  proof clarify
    fix ee::real
    assume 0 < ee
    show  $\forall_F n$  in sequentially.  $\forall \xi \in \text{path\_image } \gamma. \text{cmod } (d (a \ n) \ \xi - d \ x \ \xi) < ee$ 

    proof -
      let ?ddpa =  $\{(w,z) \mid w \ z. w \in \text{cball } x \ dd \wedge z \in \text{path\_image } \gamma\}$ 
      have uniformly_continuous_on ?ddpa  $(\lambda(x,y). d \ x \ y)$ 
      proof (rule compact_uniformly_continuous [OF continuous_on_subset [OF
cond_uu]])
        show compact  $\{(w, z) \mid w \ z. w \in \text{cball } x \ dd \wedge z \in \text{path\_image } \gamma\}$ 
        using  $\langle \text{valid\_path } \gamma \rangle$ 
        by (auto simp: compact_Times compact_valid_path_image simp del:
mem_cball)
      qed (use dd pasz in auto)
      then obtain kk where  $kk > 0$ 
      and  $kk: \bigwedge x \ x'. \llbracket x \in ?ddpa; x' \in ?ddpa; \text{dist } x' \ x < kk \rrbracket \implies$ 
 $\text{dist } ((\lambda(x,y). d \ x \ y) \ x') ((\lambda(x,y). d \ x \ y) \ x) < ee$ 
      by (rule uniformly_continuous_onE [where  $\varepsilon = ee$ ]) (use  $\langle 0 < ee \rangle$  in
auto)
      have  $kk: \llbracket \text{norm } (w-x) \leq dd; z \in \text{path\_image } \gamma; \text{norm } ((w, z) - (x, z)) < kk \rrbracket \implies \text{norm } (d \ w \ z - d \ x \ z) < ee$ 
      for  $w \ z$ 
      using  $\langle dd > 0 \rangle$   $kk$  [of  $(x,z) (w,z)$ ] by (force simp: norm_minus_commute
dist_norm)
      obtain no where  $\forall n \geq no. \text{dist } (a \ n) \ x < \min dd \ kk$ 
      using ax unfolding lim_sequentially
      by (meson  $\langle 0 < dd \rangle \langle 0 < kk \rangle$  min_less_iff_conj)
      then show ?thesis
      using  $\langle dd > 0 \rangle \langle kk > 0 \rangle$  by (fastforce simp: eventually_sequentially kk
dist_norm)
      qed
    qed
    have  $(\lambda n. \text{contour\_integral } \gamma \ (d \ (a \ n))) \longrightarrow \text{contour\_integral } \gamma \ (d \ x)$ 
    by (rule contour_integral_uniform_limit [OF A1 A2 le_B]) (auto simp:
 $\langle \text{valid\_path } \gamma \rangle$ )
    then have tendsto_hx:  $(\lambda n. \text{contour\_integral } \gamma \ (d \ (a \ n))) \longrightarrow h \ x$ 
    by (simp add: h_def x)
    then show  $(h \circ a) \longrightarrow h \ x$ 
    by (simp add: h_def x au_o_def)
  qed
  show ?thesis
  proof (simp add: holomorphic_on_open field_differentiable_def [symmetric],
clarify)
    fix z0
    consider  $z0 \in v \mid z0 \in U$  using uv_Un by blast
    then show  $h$  field_differentiable at z0
    proof cases

```

```

    assume  $z0 \in v$  then show ?thesis
      using Cauchy_next_derivative [OF con_pa_f le_B f_has_cint __ ov] V
    f_has_cint ⟨valid_path  $\gamma$ ⟩
      by (auto simp: field_differentiable_def v_def)
    next
      assume  $z0 \in U$  then
        obtain  $e$  where  $e > 0$  and  $e$ :  $\text{ball } z0 \ e \subseteq U$  by (blast intro: openE [OF
        ⟨open U⟩])
        have *:  $\text{contour\_integral (linepath } a \ b) \ h + \text{contour\_integral (linepath } b \ c) \ h + \text{contour\_integral (linepath } c \ a) \ h} = 0$ 
          if  $abc\_subset$ :  $\text{convex hull } \{a, b, c\} \subseteq \text{ball } z0 \ e$  for  $a \ b \ c$ 
          proof -
            have *:  $\bigwedge x1 \ x2 \ z. z \in U \implies \text{closed\_segment } x1 \ x2 \subseteq U \implies (\lambda w. d \ w \ z)$ 
            contour_integrable_on linepath  $x1 \ x2$ 
              using hol_dw holomorphic_on_imp_continuous_on ⟨open U⟩
              by (auto intro!: contour_integrable_holomorphic_simple)
            have  $abc$ :  $\text{closed\_segment } a \ b \subseteq U \ \text{closed\_segment } b \ c \subseteq U \ \text{closed\_segment } c \ a \subseteq U$ 
              using that  $e$  segments_subset_convex_hull by fastforce+
            have  $eq0$ :  $\bigwedge w. w \in U \implies \text{contour\_integral (linepath } a \ b \ +++ \text{ linepath } b \ c \ +++ \text{ linepath } c \ a) (\lambda z. d \ z \ w)} = 0$ 
              proof (rule contour_integral_unique [OF Cauchy_theorem_triangle])
                show  $\bigwedge w. w \in U \implies (\lambda z. d \ z \ w)$  holomorphic_on  $\text{convex hull } \{a, b, c\}$ 
                  using  $e \ abc\_subset$  by (auto intro: holomorphic_on_subset [OF
                  hol_dw])
              qed
            qed
            have  $\bigwedge z. z \in \text{path\_image } \gamma \implies$ 
               $\text{contour\_integral (linepath } a \ b) (\lambda x. d \ x \ z) +$ 
               $(\text{contour\_integral (linepath } b \ c) (\lambda x. d \ x \ z) +$ 
               $\text{contour\_integral (linepath } c \ a) (\lambda x. d \ x \ z)) = 0$ 
              using  $abc \ \text{pasz } U * eq0$  by auto
            then show ?thesis
              by (simp add: contour_integral_eq_0 cint_h  $abc$  contour_integrable_add
              contour_integral_add [symmetric] add_ac)
            qed
            show ?thesis
              using  $e \ \langle e > 0 \rangle$ 
              by (auto intro!: holomorphic_on_imp_differentiable_at [OF __ open_ball]
              analytic_imp_holomorphic
              Morera_triangle_continuous_on_subset [OF conthu] *)
            qed
          qed
        ultimately have [simp]:  $h \ z = 0$  for  $z$ 
          by (meson Liouville_weak)
        have  $((\lambda w. 1 / (w - z)) \text{ has\_contour\_integral complex\_of\_real } (2 * \pi) * i * \text{winding\_number } \gamma \ z) \ \gamma$ 
          by (rule has_contour_integral_winding_number [OF ⟨valid_path  $\gamma$ ⟩ znot])
        then have  $((\lambda w. f \ z * (1 / (w - z))) \text{ has\_contour\_integral complex\_of\_real } (2$ 

```

```

* pi) * i * winding_number  $\gamma$  z * f z)  $\gamma$ 
  by (metis mult.commute has_contour_integral_lmul)
  then have 1: (( $\lambda w$ . f z / (w-z)) has_contour_integral complex_of_real (2 *
pi) * i * winding_number  $\gamma$  z * f z)  $\gamma$ 
  by (simp add: field_split_simps)
  moreover have 2: (( $\lambda w$ . (f w - f z) / (w-z)) has_contour_integral 0)  $\gamma$ 
  using U [OF z] pasz d_def by (force elim: has_contour_integral_eq [where
g =  $\lambda w$ . (f w - f z)/(w-z)])
  show ?thesis
  using has_contour_integral_add [OF 1 2] by (simp add: diff_divide_distrib)
qed

theorem Cauchy_integral_formula_global:
  assumes S: open S and holf: f holomorphic_on S
  and z: z  $\in$  S and vpg: valid_path  $\gamma$ 
  and pasz: path_image  $\gamma \subseteq S - \{z\}$  and loop: pathfinish  $\gamma$  = pathstart  $\gamma$ 
  and zero:  $\bigwedge w$ . w  $\notin$  S  $\implies$  winding_number  $\gamma$  w = 0
  shows (( $\lambda w$ . f w / (w-z)) has_contour_integral (2*pi * i * winding_number
 $\gamma$  z * f z))  $\gamma$ 
proof -
  have path  $\gamma$  using vpg by (blast intro: valid_path_imp_path)
  have hols: ( $\lambda w$ . f w / (w-z)) holomorphic_on S - {z} ( $\lambda w$ . 1 / (w-z)) holo-
morphic_on S - {z}
  by (rule holomorphic_intros holomorphic_on_subset [OF holf] | force)+
  then have cint_fw: ( $\lambda w$ . f w / (w-z)) contour_integrable_on  $\gamma$ 
  by (meson contour_integrable_holomorphic_simple holomorphic_on_imp_continuous_on
open_delete S vpg pasz)
  obtain d where d>0
  and d:  $\bigwedge g$  h.  $\llbracket$ valid_path g; valid_path h;  $\forall t \in \{0..1\}$ . cmod (g t -  $\gamma$  t) < d
 $\wedge$  cmod (h t -  $\gamma$  t) < d;
  pathstart h = pathstart g  $\wedge$  pathfinish h = pathfinish g
 $\implies$  path_image h  $\subseteq S - \{z\} \wedge (\forall f$ . f holomorphic_on S - {z}
 $\longrightarrow$  contour_integral h f = contour_integral g f)
  using contour_integral_nearby_ends [OF _  $\langle$ path  $\gamma \rangle$  pasz] S by (simp add:
open_Diff) metis
  obtain p where polyp: polynomial_function p
  and ps: pathstart p = pathstart  $\gamma$  and pf: pathfinish p = pathfinish  $\gamma$ 
and led:  $\forall t \in \{0..1\}$ . cmod (p t -  $\gamma$  t) < d
  using path_approx_polynomial_function [OF  $\langle$ path  $\gamma \rangle \langle d > 0 \rangle$ ] by metis
  then have ploop: pathfinish p = pathstart p using loop by auto
  have vpp: valid_path p using polyp valid_path_polynomial_function by blast
  have [simp]: z  $\notin$  path_image  $\gamma$  using pasz by blast
  have paps: path_image p  $\subseteq S - \{z\}$  and cint_eq: ( $\bigwedge f$ . f holomorphic_on S -
{z}  $\implies$  contour_integral p f = contour_integral  $\gamma$  f)
  using pf ps led d [OF vpg vpp]  $\langle d > 0 \rangle$  by auto
  have wn_eq: winding_number p z = winding_number  $\gamma$  z
  using vpp paps
  by (simp add: subset_Diff_insert vpg valid_path_polynomial_function wind-
ing_number_valid_path cint_eq hols)

```

```

have winding_number p w = winding_number  $\gamma$  w if  $w \notin S$  for w
proof -
  have hol:  $(\lambda v. 1 / (v-w))$  holomorphic_on  $S - \{z\}$ 
  using that by (force intro: holomorphic_intros holomorphic_on_subset [OF
hol])
  have  $w \notin \text{path\_image } p$   $w \notin \text{path\_image } \gamma$  using paps pasz that by auto
  then show ?thesis
  using vpp vpg by (simp add: subset_Diff_insert valid_path_polynomial_function
winding_number_valid_path cint_eq [OF hol])
qed
then have wn0:  $\bigwedge w. w \notin S \implies \text{winding\_number } p \ w = 0$ 
  by (simp add: zero)
show ?thesis
  using Cauchy_integral_formula_global_weak [OF S hol f z polyp paps ploop
wn0] hols
  by (metis wn_eq cint_eq has_contour_integral_eqpath cint_fw cint_eq)
qed

```

theorem Cauchy_theorem_global:

```

assumes S: open S and hol: f holomorphic_on S
  and vpg: valid_path  $\gamma$  and loop: pathfinish  $\gamma$  = pathstart  $\gamma$ 
  and pas: path_image  $\gamma \subseteq S$ 
  and zero:  $\bigwedge w. w \notin S \implies \text{winding\_number } \gamma \ w = 0$ 
shows (f has_contour_integral 0)  $\gamma$ 
proof -
  have path_image  $\gamma \neq S$ 
  by (metis compact_valid_path_image vpg compact_open path_image_nonempty
S)
  then obtain z where  $z \in S$  and znot:  $z \notin \text{path\_image } \gamma$  and pasz: path_image
 $\gamma \subseteq S - \{z\}$ 
  using pas by blast
  have hol:  $(\lambda w. (w-z) * f \ w)$  holomorphic_on S
  by (rule holomorphic_intros hol)+
  show ?thesis
  using Cauchy_integral_formula_global [OF S hol  $\langle z \in S \rangle$  vpg pasz loop zero]
  by (auto simp: znot elim!: has_contour_integral_eq)
qed

```

corollary Cauchy_theorem_global_outside:

```

assumes open S f holomorphic_on S valid_path  $\gamma$  pathfinish  $\gamma$  = pathstart  $\gamma$ 
path_image  $\gamma \subseteq S$ 
   $\bigwedge w. w \notin S \implies w \in \text{outside}(\text{path\_image } \gamma)$ 
shows (f has_contour_integral 0)  $\gamma$ 
by (metis Cauchy_theorem_global assms winding_number_zero_in_outside valid_path_imp_path)

```

lemma simply_connected_imp_winding_number_zero:

```

assumes simply_connected S path g
  path_image g  $\subseteq S$  pathfinish g = pathstart g  $z \notin S$ 
shows winding_number g z = 0

```

proof –

have *hom*: *homotopic_loops* *S* *g* (*linepath* (*pathstart* *g*) (*pathstart* *g*))
 by (*meson* *assms* *homotopic_paths_imp_homotopic_loops* *pathfinish_linepath*
simply_connected_eq_contractible_path)
 then have *homotopic_paths* ($-\{z\}$) *g* (*linepath* (*pathstart* *g*) (*pathstart* *g*))
 by (*meson* $\langle z \notin S \rangle$ *homotopic_loops_imp_homotopic_paths_null* *homotopic_paths_subset*
subset_Compl_singleton)
 then have *winding_number* *g* *z* = *winding_number* (*linepath* (*pathstart* *g*) (*pathstart*
g)) *z*
 by (*rule* *winding_number_homotopic_paths*)
 also have $\dots = 0$
 using *assms* **by** (*force* *intro*: *winding_number_trivial*)
 finally show *?thesis* .
qed

lemma *Cauchy_theorem_simply_connected*:

assumes *open* *S* *simply_connected* *S* *f* *holomorphic_on* *S* *valid_path* *g*
 path_image *g* \subseteq *S* *pathfinish* *g* = *pathstart* *g*
 shows (*f* *has_contour_integral* 0) *g*
 by (*meson* *assms* *Cauchy_theorem_global* *simply_connected_imp_winding_number_zero*
valid_path_imp_path)

proposition *holomorphic_logarithm_exists*:

assumes *A*: *convex* *A* *open* *A*
 and *f*: *f* *holomorphic_on* *A* $\bigwedge x. x \in A \implies f\ x \neq 0$
 and *z0*: *z0* \in *A*
 obtains *g* **where** *g* *holomorphic_on* *A* **and** $\bigwedge x. x \in A \implies \exp (g\ x) = f\ x$
proof –
 note *f'* = *holomorphic_derivI* [*OF* *f*(1) *A*(2)]
 obtain *g* **where** *g*: $\bigwedge x. x \in A \implies (g\ \text{has_field_derivative}\ \text{deriv}\ f\ x / f\ x)\ (at\ x)$
 proof (*rule* *holomorphic_convex_primitive'* [*OF* *A*])
 show $(\lambda x. \text{deriv}\ f\ x / f\ x)$ *holomorphic_on* *A*
 by (*intro* *holomorphic_intros* *f* *A*)
qed (*auto* *simp*: *A* *at_within_open*[*of* *A*])
 define *h* **where** *h* = $(\lambda x. -g\ z0 + \ln (f\ z0) + g\ x)$
 from *g* **and** *A* **have** *g_holo*: *g* *holomorphic_on* *A*
 by (*auto* *simp*: *holomorphic_on_def* *at_within_open*[*of* *A*] *field_differentiable_def*)
 hence *h_holo*: *h* *holomorphic_on* *A*
 by (*auto* *simp*: *h_def* *intro*!: *holomorphic_intros*)
 note [*simp*] = *at_within_open*[*OF* $\langle \text{open}\ A \rangle$]
 have $\exists c. \forall x \in A. f\ x / \exp (h\ x) - 1 = c$
 using $\langle \text{convex}\ A \rangle$ *z0* *f*
 by (*force* *simp*: *h_def* *exp_diff_field_simps* *intro*!: *has_field_derivative_zero_constant*
derivative_eq_intros *g* *f'*)
 then obtain *c* **where** *c*: $\bigwedge x. x \in A \implies f\ x / \exp (h\ x) - 1 = c$
 by *blast*
 from *c*[*OF* *z0*] **and** *z0* **and** *f* **have** *c* = 0
 by (*simp* *add*: *h_def*)
 with *c* **have** $\bigwedge x. x \in A \implies \exp (h\ x) = f\ x$ **by** *simp*

from that[OF h_holo this] show ?thesis .
qed

4.14 Cauchy's inequality and more versions of Liouville

lemma *Cauchy_higher_deriv_bound:*

```

  assumes holf: f holomorphic_on (ball z r)
    and conf: continuous_on (cball z r) f
    and fin :  $\bigwedge w. w \in \text{ball } z \ r \implies f \ w \in \text{ball } y \ B0$ 
    and 0 < r and 0 < n
  shows cmod ((deriv  $\hat{\sim}$  n) f z)  $\leq$  (fact n) * B0 / rn
proof -
  have 0 < B0
  using <0 < r> fin [of z] by (metis ball_eq_empty ex_in_conv fin not_less)
  have le_B0: cmod (f w - y)  $\leq$  B0 if cmod (w-z)  $\leq$  r for w
  proof (rule continuous_on_closure_norm_le [of ball z r  $\lambda w. f \ w - y$ ], use <0
  < r> in simp_all)
    show continuous_on (cball z r) ( $\lambda w. f \ w - y$ )
    by (intro continuous_intros conf)
    show dist z w  $\leq$  r
    by (simp add: dist_commute dist_norm that)
  qed (use fin in <auto simp: dist_norm less_eq_real_def norm_minus_commute>)
  have (deriv  $\hat{\sim}$  n) f z = (deriv  $\hat{\sim}$  n) ( $\lambda w. f \ w$ ) z - (deriv  $\hat{\sim}$  n) ( $\lambda w. y$ ) z
  using <0 < n> by simp
  also have ... = (deriv  $\hat{\sim}$  n) ( $\lambda w. f \ w - y$ ) z
  using <0 < r> higher_deriv_diff holf by auto
  finally have (deriv  $\hat{\sim}$  n) f z = (deriv  $\hat{\sim}$  n) ( $\lambda w. f \ w - y$ ) z .
  have conf': continuous_on (cball z r) ( $\lambda u. f \ u - y$ )
  by (rule conf continuous_intros)+
  have holf': ( $\lambda u. (f \ u - y)$ ) holomorphic_on (ball z r)
  by (simp add: holf holomorphic_on_diff)
  define a where a = (2 * pi)/(fact n)
  have 0 < a by (simp add: a_def)
  have B0/r(Suc n)*2 * pi * r = a*((fact n)*B0/rn)
  using <0 < r> by (simp add: a_def field_split_simps)
  have der_dif: (deriv  $\hat{\sim}$  n) ( $\lambda w. f \ w - y$ ) z = (deriv  $\hat{\sim}$  n) f z
  using <0 < r> <0 < n>
  by (auto simp: higher_deriv_diff [OF holf holomorphic_on_const])
  have norm ((2 * of_real pi * i)/(fact n) * (deriv  $\hat{\sim}$  n) ( $\lambda w. f \ w - y$ ) z)
     $\leq$  (B0/r(Suc n)) * (2 * pi * r)
  apply (rule has_contour_integral_bound_circlepath [of ( $\lambda u. (f \ u - y)/(u-z)$ )(Suc n)) _ z])
  using Cauchy_has_contour_integral_higher_derivative_circlepath [OF conf' holf]
  using <0 < B0> <0 < r>
  apply (auto simp: norm_divide norm_mult norm_power divide_simps le_B0)
  done
then show ?thesis
  using <0 < r>
```

by (auto simp: norm_divide norm_mult norm_power field_simps der_dif
le_B0)
qed

lemma *Cauchy_inequality:*

assumes holf: f holomorphic_on (ball ξ r)
and conf: continuous_on (cball ξ r) f
and $0 < r$
and nof: $\bigwedge x. \text{norm}(\xi - x) = r \implies \text{norm}(f x) \leq B$
shows $\text{norm} ((\text{deriv } ^n) f \xi) \leq (\text{fact } n) * B / r^n$
proof -
obtain x where $\text{norm}(\xi - x) = r$
by (metis $0 < r$ dist_norm order_less_imp_le vector_choose_dist)
then have $0 \leq B$
by (metis nof norm_not_less_zero not_le order_trans)
have $\xi \in \text{ball } \xi r$
using $0 < r$ by simp
then have $((\lambda u. f u / (u - \xi)^{\wedge \text{Suc } n}) \text{ has_contour_integral } (2 * \pi) * i / \text{fact } n * (\text{deriv } ^n) f \xi)$
 $(\text{circlepath } \xi r)$
by (rule Cauchy_has_contour_integral_higher_derivative_circlepath [OF conf holf])
have $\text{norm} ((2 * \pi * i) / (\text{fact } n) * (\text{deriv } ^n) f \xi) \leq (B / r^{\wedge \text{Suc } n}) * (2 * \pi * r)$
proof (rule has_contour_integral_bound_circlepath)
have $\xi \in \text{ball } \xi r$
using $0 < r$ by simp
then show $((\lambda u. f u / (u - \xi)^{\wedge \text{Suc } n}) \text{ has_contour_integral } (2 * \pi) * i / \text{fact } n * (\text{deriv } ^n) f \xi)$
 $(\text{circlepath } \xi r)$
by (rule Cauchy_has_contour_integral_higher_derivative_circlepath [OF conf holf])
show $\bigwedge x. \text{cmod}(x - \xi) = r \implies \text{cmod}(f x / (x - \xi)^{\wedge \text{Suc } n}) \leq B / r^{\wedge \text{Suc } n}$
using $0 \leq B$ $0 < r$
by (simp add: norm_divide norm_power nof frac_le norm_minus_commute del: power_Suc)
qed (use $0 \leq B$ $0 < r$ in auto)
then show ?thesis using $0 < r$
by (simp add: norm_divide norm_mult field_simps)
qed

lemma *Liouville_polynomial:*

assumes holf: f holomorphic_on UNIV
and nof: $\bigwedge z. A \leq \text{norm } z \implies \text{norm}(f z) \leq B * \text{norm } z^{\wedge n}$
shows $f \xi = (\sum_{k \leq n}. (\text{deriv } ^k) f 0 / \text{fact } k * \xi^{\wedge k})$
proof (cases rule: le_less_linear [THEN disjE])
assume $B \leq 0$
then have $\bigwedge z. A \leq \text{norm } z \implies \text{norm}(f z) = 0$
by (metis nof less_le_trans zero_less_mult_iff neqE norm_not_less_zero)

```

norm_power not_le)
  then have f0: (f ⟶ 0) at_infinity
    using Lim_at_infinity by force
  then have [simp]: f = (λw. 0)
    using Liouville_weak [OF holf, of 0]
    by (simp add: eventually_at_infinity f0) meson
  show ?thesis by simp
next
  assume 0 < B
  have ((λk. (deriv ~^k) f 0 / (fact k) * (ξ - 0) ^ k) sums f ξ)
  proof (rule holomorphic_power_series [where r = norm ξ + 1])
    show f holomorphic_on ball 0 (cmod ξ + 1) ξ ∈ ball 0 (cmod ξ + 1)
      using holf holomorphic_on_subset by auto
    qed
  then have sumsf: ((λk. (deriv ~^k) f 0 / (fact k) * ξ ^ k) sums f ξ) by simp
  have (deriv ~^k) f 0 / fact k * ξ ^ k = 0 if k > n for k
  proof (cases (deriv ~^k) f 0 = 0)
    case True then show ?thesis by simp
  next
    case False
    define w where w = complex_of_real (fact k * B / cmod ((deriv ~^k) f 0)
+ (|A| + 1))
    have 1 ≤ abs (fact k * B / cmod ((deriv ~^k) f 0) + (|A| + 1))
      using ‹0 < B› by simp
    then have wge1: 1 ≤ norm w
      by (metis norm_of_real w_def)
    then have w ≠ 0 by auto
    have kB: 0 < fact k * B
      using ‹0 < B› by simp
    then have 0 ≤ fact k * B / cmod ((deriv ~^k) f 0)
      by simp
    then have wgeA: A ≤ cmod w
      by (simp only: w_def norm_of_real)
    have fact k * B / cmod ((deriv ~^k) f 0) < abs (fact k * B / cmod ((deriv ~^k) f 0) + (|A| + 1))
      using ‹0 < B› by simp
    then have wge: fact k * B / cmod ((deriv ~^k) f 0) < norm w
      by (metis norm_of_real w_def)
    then have fact k * B / norm w < cmod ((deriv ~^k) f 0)
      using False by (simp add: field_split_simps mult.commute split: if_split_asm)
    also have ... ≤ fact k * (B * norm w ^ n) / norm w ^ k
    proof (rule Cauchy_inequality)
      show f holomorphic_on ball 0 (cmod w)
        using holf holomorphic_on_subset by force
      show continuous_on (cball 0 (cmod w)) f
        using holf holomorphic_on_imp_continuous_on holomorphic_on_subset
    by blast
    show ∧x. cmod (0 - x) = cmod w ⟹ cmod (f x) ≤ B * cmod w ^ n
      by (metis nof wgeA dist_0_norm dist_norm)

```

```

qed (use <w ≠ 0> in auto)
also have ... = fact k * B / cmod w ^ (k-n)
  using <k>n> by (simp add: divide_simps flip: power_add)
finally have fact k * B / cmod w < fact k * B / cmod w ^ (k-n) .
then have 1 / cmod w < 1 / cmod w ^ (k-n)
  by (metis kB divide_inverse inverse_eq_divide mult_less_cancel_left_pos)
then have cmod w ^ (k-n) < cmod w
  by (smt (verit, best) <w ≠ 0> frac_le zero_less_norm_iff)
with self_le_power [OF wge1] show ?thesis
  by (meson diff_is_0_eq not_gr0 not_le that)
qed
then have (deriv ~^ (k + Suc n)) f 0 / fact (k + Suc n) * ξ ^ (k + Suc n) =
0 for k
  using not_less_eq by blast
then have (λi. (deriv ~^ (i + Suc n)) f 0 / fact (i + Suc n) * ξ ^ (i + Suc n))
sums 0
  by (rule sums_0)
with sums_split_initial_segment [OF sumsf, where n = Suc n]
show ?thesis
  using atLeast0AtMost lessThan_Suc_atMost sums_unique2 by fastforce
qed

```

Every bounded entire function is a constant function.

```

theorem Liouville_theorem:
  assumes holf: f holomorphic_on UNIV
    and bf: bounded (range f)
  shows f constant_on UNIV
  using Liouville_polynomial [OF holf, of 0 0, simplified]
  by (metis bf bounded_iff constant_on_def rangeI)

```

A holomorphic function f has only isolated zeros unless f is 0.

```

lemma powser_0_nonzero:
  fixes a :: nat ⇒ 'a:: {real_normed_field, banach}
  assumes r: 0 < r
    and sm: (λx. norm (x-ξ) < r ⇒ (λn. a n * (x-ξ) ^ n) sums (f x))
    and [simp]: f ξ = 0
    and m0: a m ≠ 0 and m>0
  obtains s where 0 < s and (λz. z ∈ cball ξ s - {ξ} ⇒ f z ≠ 0)
proof -
  have r ≤ conv_radius a
    using sm sums_summable by (auto simp: le_conv_radius_iff [where ξ=ξ])
  obtain m where am: a m ≠ 0 and az [simp]: (λn. n < m ⇒ a n = 0)
  proof
    show a (LEAST n. a n ≠ 0) ≠ 0
      by (metis (mono_tags, lifting) m0 LeastI)
  qed (fastforce dest!: not_less_Least)
  define b where b i = a (i+m) / a m for i
  define g where g x = suminf (λi. b i * (x-ξ) ^ i) for x
  have [simp]: b 0 = 1

```

```

  by (simp add: am b_def)
{ fix x::'a
  assume norm (x-ξ) < r
  then have (λn. (a m * (x-ξ) ^ m) * (b n * (x-ξ) ^ n)) sums (f x)
    using am az sm sums_zero_iff_shift [of m (λn. a n * (x-ξ) ^ n) f x]
    by (simp add: b_def monoid_mult_class.power_add algebra_simps)
  then have x ≠ ξ ⟹ (λn. b n * (x-ξ) ^ n) sums (f x / (a m * (x-ξ) ^ m))
    using am by (simp add: sums_mult_D)
} note bsums = this
then have norm (x-ξ) < r ⟹ summable (λn. b n * (x-ξ) ^ n) for x
  using sums_summable by (cases x=ξ) auto
then have r ≤ conv_radius b
  by (simp add: le_conv_radius_iff [where ξ=ξ])
then have r/2 < conv_radius b
  using not_le order_trans r by fastforce
then have continuous_on (cball ξ (r/2)) g
  using power_continuous suminf [of r/2 b ξ] by (simp add: g_def)
then obtain s where s>0 ∧ x. ‖norm (x-ξ) ≤ s; norm (x-ξ) ≤ r/2‖ ⟹ dist
(g x) (g ξ) < 1/2
proof (rule continuous_onE)
  show ξ ∈ cball ξ (r / 2) 1/2 > (0::real)
    using r by auto
qed (auto simp: dist_commute dist_norm)
moreover have g ξ = 1
  by (simp add: g_def)
ultimately have gnz: ∧x. ‖norm (x-ξ) ≤ s; norm (x-ξ) ≤ r/2‖ ⟹ (g x) ≠ 0
  by fastforce
show ?thesis
proof
  have *: f x ≠ 0 if x ≠ ξ norm (x-ξ) ≤ s norm (x-ξ) ≤ r/2 for x
    using bsums [of x] that gnz [of x] r sums_iff unfolding g_def by fastforce
  show ∧z. z ∈ cball ξ (min s (r / 2)) - {ξ} ⟹ f z ≠ 0
    by (simp add: * dist_norm norm_minus_commute)
qed (use ‹0 < r› ‹0 < s› in auto)
qed

```

4.15 Complex functions and power series

The following defines the power series expansion of a complex function at a given point (assuming that it is analytic at that point).

definition *fps_expansion* :: (complex ⇒ complex) ⇒ complex ⇒ complex *fps*
where

$$\text{fps_expansion } f \ z0 = \text{Abs_fps } (\lambda n. (\text{deriv } \frown n) f \ z0 / \text{fact } n)$$

lemma *fps_expansion_cong*:

assumes $\forall_F w \text{ in } nhds \ x. f \ w = g \ w$

shows $\text{fps_expansion } f \ x = \text{fps_expansion } g \ x$

unfolding *fps_expansion_def* **using** *assms higher_deriv_cong_ev* **by** *fastforce*

```

lemma
  fixes r :: ereal
  assumes f holomorphic_on eball z0 r
  shows conv_radius_fps_expansion: fps_conv_radius (fps_expansion f z0) ≥ r
    and eval_fps_expansion:  $\bigwedge z. z \in \text{eball } z0 \ r \implies \text{eval\_fps } (\text{fps\_expansion } f \ z0) \ (z - z0) = f \ z$ 
    and eval_fps_expansion':  $\bigwedge z. \text{norm } z < r \implies \text{eval\_fps } (\text{fps\_expansion } f \ z0) \ z = f \ (z0 + z)$ 
  proof -
    have  $(\lambda n. \text{fps\_nth } (\text{fps\_expansion } f \ z0) \ n * (z - z0) ^ n) \text{ sums } f \ z$ 
      if  $z \in \text{ball } z0 \ r' \text{ ereal } r' < r$  for  $z \ r'$ 
    proof -
      have f holomorphic_on ball z0 r'
        using holomorphic_on_subset[OF _ ball_eball_mono] assms that by force
      then show ?thesis
        using fps_expansion_def holomorphic_power_series that by auto
    qed
    hence *:  $(\lambda n. \text{fps\_nth } (\text{fps\_expansion } f \ z0) \ n * (z - z0) ^ n) \text{ sums } f \ z$ 
      if  $z \in \text{eball } z0 \ r$  for  $z$ 
      using that by (subst (asm) eball_conv_UNION_balls) blast
    show fps_conv_radius (fps_expansion f z0) ≥ r unfolding fps_conv_radius_def
    proof (rule conv_radius_geI_ex)
      fix r' :: real assume r':  $r' > 0 \text{ ereal } r' < r$ 
      thus  $\exists z. \text{norm } z = r' \wedge \text{summable } (\lambda n. \text{fps\_nth } (\text{fps\_expansion } f \ z0) \ n * z ^ n)$ 
        using *[of z0 + of_real r']
        by (intro exI[of _ of_real r']) (auto simp: summable_def dist_norm)
    qed
    show eval_fps (fps_expansion f z0) (z - z0) = f z if  $z \in \text{eball } z0 \ r$  for  $z$ 
      using *[OF that] by (simp add: eval_fps_def sums_iff)
    show eval_fps (fps_expansion f z0) z = f (z0 + z) if  $\text{ereal } (\text{norm } z) < r$  for  $z$ 
      using *[of z0 + z] and that by (simp add: eval_fps_def sums_iff dist_norm)
  qed

```

We can now show several more facts about power series expansions (at least in the complex case) with relative ease that would have been trickier without complex analysis.

```

lemma
  fixes f :: complex fps and r :: ereal
  assumes  $\bigwedge z. \text{ereal } (\text{norm } z) < r \implies \text{eval\_fps } f \ z \neq 0$ 
  shows fps_conv_radius_inverse:  $\text{fps\_conv\_radius } (\text{inverse } f) \geq \min r \ (\text{fps\_conv\_radius } f)$ 
    and eval_fps_inverse:  $\bigwedge z. \text{ereal } (\text{norm } z) < \text{fps\_conv\_radius } f \implies \text{ereal } (\text{norm } z) < r \implies$ 
       $\text{eval\_fps } (\text{inverse } f) \ z = \text{inverse } (\text{eval\_fps } f \ z)$ 
  proof -
    define R where  $R = \min (\text{fps\_conv\_radius } f) \ r$ 
    have *:  $\text{fps\_conv\_radius } (\text{inverse } f) \geq \min r \ (\text{fps\_conv\_radius } f) \wedge$ 
       $(\forall z \in \text{eball } 0 \ (\min (\text{fps\_conv\_radius } f) \ r). \text{eval\_fps } (\text{inverse } f) \ z = \text{inverse } (\text{eval\_fps } f \ z))$ 

```

```

(eval_fps f z))
proof (cases min r (fps_conv_radius f) > 0)
  case True
    define f' where f' = fps_expansion (λz. inverse (eval_fps f z)) 0
    have holo: (λz. inverse (eval_fps f z)) holomorphic_on eball 0 (min r (fps_conv_radius
f))
      using assms by (intro holomorphic_intros) auto
    from holo have radius: fps_conv_radius f' ≥ min r (fps_conv_radius f)
      unfolding f'_def by (rule conv_radius_fps_expansion)
    have eval_f': eval_fps f' z = inverse (eval_fps f z)
      if norm z < fps_conv_radius f norm z < r for z
      using that unfolding f'_def by (subst eval_fps_expansion'[OF holo]) auto

    have f * f' = 1
    proof (rule eval_fps_eqD)
      from radius and True have 0 < min (fps_conv_radius f) (fps_conv_radius
f')
        by (auto simp: min_def split: if_splits)
      also have ... ≤ fps_conv_radius (f * f') by (rule fps_conv_radius_mult)
      finally show ... > 0 .
    next
      from True have R > 0 by (auto simp: R_def)
      hence eventually (λz. z ∈ eball 0 R) (nhds 0)
        by (intro eventually_nhds_in_open) (auto simp: zero_ereal_def)
      thus eventually (λz. eval_fps (f * f') z = eval_fps 1 z) (nhds 0)
        proof eventually_elim
          case (elim z)
          hence eval_fps (f * f') z = eval_fps f z * eval_fps f' z
            using radius by (intro eval_fps_mult)
            (auto simp: R_def min_def split: if_splits intro: less_trans)
          also have eval_fps f' z = inverse (eval_fps f z)
            using elim by (intro eval_f') (auto simp: R_def)
          also from elim have eval_fps f z ≠ 0
            by (intro assms) (auto simp: R_def)
          hence eval_fps f z * inverse (eval_fps f z) = eval_fps 1 z
            by simp
          finally show eval_fps (f * f') z = eval_fps 1 z .
        qed
      qed simp_all
      hence f' = inverse f
        by (intro fps_inverse_unique [symmetric]) (simp_all add: mult_ac)
      with eval_f' and radius show ?thesis by simp
    next
      case False
      hence *: eball 0 R = {}
        by (intro eball_empty) (auto simp: R_def min_def split: if_splits)
      show ?thesis
    proof
      from False have min r (fps_conv_radius f) ≤ 0

```

```

    by (simp add: min_def)
  also have  $0 \leq \text{fps\_conv\_radius } (\text{inverse } f)$ 
    by (simp add: fps_conv_radius_def conv_radius_nonneg)
  finally show  $\min r (\text{fps\_conv\_radius } f) \leq \dots$  .
qed (unfold * [unfolded R_def], auto)
qed

show  $\text{fps\_conv\_radius } (\text{inverse } f) \geq \min r (\text{fps\_conv\_radius } f)$ 
  using * by blast
show  $\text{eval\_fps } (\text{inverse } f) z = \text{inverse } (\text{eval\_fps } f z)$ 
  if  $\text{ereal } (\text{norm } z) < \text{fps\_conv\_radius } f$   $\text{ereal } (\text{norm } z) < r$  for  $z$ 
  using that * by auto
qed

lemma
  fixes  $f g :: \text{complex fps}$  and  $r :: \text{ereal}$ 
  defines  $R \equiv \text{Min } \{r, \text{fps\_conv\_radius } f, \text{fps\_conv\_radius } g\}$ 
  assumes  $\text{fps\_conv\_radius } f > 0$   $\text{fps\_conv\_radius } g > 0$   $r > 0$ 
  assumes  $\text{nz}: \bigwedge z. z \in \text{eball } 0 r \implies \text{eval\_fps } g z \neq 0$ 
  shows  $\text{fps\_conv\_radius\_divide}': \text{fps\_conv\_radius } (f / g) \geq R$ 
    and  $\text{eval\_fps\_divide}':$ 
       $\text{ereal } (\text{norm } z) < R \implies \text{eval\_fps } (f / g) z = \text{eval\_fps } f z / \text{eval\_fps } g z$ 
proof -
  from  $\text{nz}[of 0]$  and  $\langle r > 0 \rangle$  have  $\text{nz}': \text{fps\_nth } g 0 \neq 0$ 
    by (auto simp: eval_fps_at_0 zero_ereal_def)
  have  $R \leq \min r (\text{fps\_conv\_radius } g)$ 
    by (auto simp: R_def intro: min.coboundedI2)
  also have  $\min r (\text{fps\_conv\_radius } g) \leq \text{fps\_conv\_radius } (\text{inverse } g)$ 
    by (intro fps_conv_radius_inverse assms) (auto simp: zero_ereal_def)
  finally have  $\text{radius}: \text{fps\_conv\_radius } (\text{inverse } g) \geq R$  .
  have  $R \leq \min (\text{fps\_conv\_radius } f) (\text{fps\_conv\_radius } (\text{inverse } g))$ 
    by (intro radius min.boundedI) (auto simp: R_def intro: min.coboundedI1 min.coboundedI2)
  also have  $\dots \leq \text{fps\_conv\_radius } (f * \text{inverse } g)$ 
    by (rule fps_conv_radius_mult)
  also have  $f * \text{inverse } g = f / g$ 
    by (intro fps_divide_unit [symmetric] nz')
  finally show  $\text{fps\_conv\_radius } (f / g) \geq R$  .

  assume  $z: \text{ereal } (\text{norm } z) < R$ 
  have  $\text{eval\_fps } (f * \text{inverse } g) z = \text{eval\_fps } f z * \text{eval\_fps } (\text{inverse } g) z$ 
    using radius by (intro eval_fps_mult less_le_trans[OF z])
    (auto simp: R_def intro: min.coboundedI1 min.coboundedI2)
  also have  $\text{eval\_fps } (\text{inverse } g) z = \text{inverse } (\text{eval\_fps } g z)$  using  $\langle r > 0 \rangle$ 
    by (intro eval_fps_inverse[where  $r = r$ ] less_le_trans[OF z] nz)
    (auto simp: R_def intro: min.coboundedI1 min.coboundedI2)
  also have  $f * \text{inverse } g = f / g$  by fact
  finally show  $\text{eval\_fps } (f / g) z = \text{eval\_fps } f z / \text{eval\_fps } g z$ 
    by (simp add: field_split_simps)

```


qed

lemma

```

  fixes f g :: complex fps and r :: ereal
  defines R ≡ Min {r, fps_conv_radius f, fps_conv_radius g}
  assumes subdegree g ≤ subdegree f
  assumes fps_conv_radius f > 0 fps_conv_radius g > 0 r > 0
  assumes  $\bigwedge z. z \in \text{eball } 0 \ r \implies z \neq 0 \implies \text{eval\_fps } g \ z \neq 0$ 
  shows fps_conv_radius_divide: fps_conv_radius (f / g) ≥ R
    and eval_fps_divide:
      ereal (norm z) < R  $\implies c = \text{fps\_nth } f \ (\text{subdegree } g) / \text{fps\_nth } g \ (\text{subdegree } g) \implies$ 
        eval_fps (f / g) z = (if z = 0 then c else eval_fps f z / eval_fps g z)
proof -
  define f' g' where f' = fps_shift (subdegree g) f and g' = fps_shift (subdegree g) g
  have f_eq: f = f' * fps_X ^ subdegree g and g_eq: g = g' * fps_X ^ subdegree g
  unfolding f'_def g'_def by (rule subdegree_decompose' le_refl | fact)+
  have subdegree: subdegree f' = subdegree f - subdegree g subdegree g' = 0
  using assms(2) by (simp_all add: f'_def g'_def)
  have [simp]: fps_conv_radius f' = fps_conv_radius f fps_conv_radius g' =
    fps_conv_radius g
  by (simp_all add: f'_def g'_def)
  have [simp]: fps_nth f' 0 = fps_nth f (subdegree g)
    fps_nth g' 0 = fps_nth g (subdegree g) by (simp_all add: f'_def g'_def)
  have g_nz: g ≠ 0
  proof -
    define z :: complex where z = (if r = ∞ then 1 else of_real (real_of_ereal r / 2))
    have z ∈ eball 0 r
    using ⟨r > 0⟩ ereal_less_real_iff z_def by fastforce
    moreover have z ≠ 0 using ⟨r > 0⟩
    by (cases r) (auto simp: z_def)
    ultimately have eval_fps g z ≠ 0 by (rule assms(6))
    thus g ≠ 0 by auto
  qed
  have fg: f / g = f' * inverse g'
  by (subst f_eq, subst (2) g_eq) (insert g_nz, simp add: fps_divide_unit)

  have g'_nz: eval_fps g' z ≠ 0 if z: norm z < min r (fps_conv_radius g) for z
  proof (cases z = 0)
    case False
    with assms and z have eval_fps g z ≠ 0 by auto
    also from z have eval_fps g z = eval_fps g' z * z ^ subdegree g
    by (subst g_eq) (auto simp: eval_fps_mult)
    finally show ?thesis by auto
  qed (use ⟨g ≠ 0⟩ in ⟨auto simp: g'_def eval_fps_at_0⟩)

```

```

have  $R \leq \min (\min r (fps\_conv\_radius\ g)) (fps\_conv\_radius\ g')$ 
  by (auto simp:  $R\_def$  min.coboundedI1 min.coboundedI2)
also have  $\dots \leq fps\_conv\_radius (inverse\ g')$ 
  using  $g'\_nz$  by (rule fps_conv_radius_inverse)
finally have  $conv\_radius\_inv: R \leq fps\_conv\_radius (inverse\ g')$  .
hence  $R \leq fps\_conv\_radius (f' * inverse\ g')$ 
  by (intro order.trans[OF fps_conv_radius_mult])
  (auto simp:  $R\_def$  intro: min.coboundedI1 min.coboundedI2)
thus  $fps\_conv\_radius (f / g) \geq R$  by (simp add: fg)

fix  $z\ c :: complex$  assume  $z: ereal (norm\ z) < R$ 
assume  $c: c = fps\_nth\ f (subdegree\ g) / fps\_nth\ g (subdegree\ g)$ 
show  $eval\_fps (f / g)\ z = (if\ z = 0\ then\ c\ else\ eval\_fps\ f\ z / eval\_fps\ g\ z)$ 
proof (cases  $z = 0$ )
  case False
    from  $z$  and  $conv\_radius\_inv$  have  $ereal (norm\ z) < fps\_conv\_radius (inverse\ g')$ 
    by simp
    with  $z$  have  $eval\_fps (f / g)\ z = eval\_fps\ f'\ z * eval\_fps (inverse\ g')\ z$ 
    unfolding fg by (subst eval_fps_mult) (auto simp:  $R\_def$ )
    also have  $eval\_fps (inverse\ g')\ z = inverse (eval\_fps\ g'\ z)$ 
    using  $z$  by (intro eval_fps_inverse[of min r (fps_conv_radius g')]  $g'\_nz$ )
    (auto simp:  $R\_def$ )
    also have  $eval\_fps\ f'\ z * \dots = eval\_fps\ f\ z / eval\_fps\ g\ z$ 
    using  $z$  False assms(2) by (simp add:  $f'\_def\ g'\_def\ eval\_fps\_shift\ R\_def$ )
    finally show ?thesis using False by simp
  qed (simp_all add:  $eval\_fps\_at\_0\ fg\ field\_simps\ c$ )
qed

lemma has_fps_expansion_fps_expansion [intro]:
  assumes open  $A$   $0 \in A$   $f$  holomorphic_on  $A$ 
  shows  $f$  has_fps_expansion fps_expansion  $f\ 0$ 
proof -
  from assms obtain  $r$  where  $r > 0$  and  $r: ball\ 0\ r \subseteq A$ 
  by (auto simp: open_contains_ball)
  with assms have holo:  $f$  holomorphic_on eball  $0 (ereal\ r)$ 
  by auto
  have  $r \leq fps\_conv\_radius (fps\_expansion\ f\ 0)$ 
  using holo by (intro conv_radius_fps_expansion) auto
  then have  $\dots > 0$ 
  by (simp add:  $ereal\_le\_less\ \langle r > 0 \rangle\ zero\_ereal\_def$ )
  moreover have eventually  $(\lambda z. z \in ball\ 0\ r) (nhds\ 0)$ 
  using  $\langle r > 0 \rangle$  by (intro eventually_nhds_in_open) auto
  hence eventually  $(\lambda z. eval\_fps (fps\_expansion\ f\ 0)\ z = f\ z) (nhds\ 0)$ 
  by eventually_elim (subst eval_fps_expansion'[OF holo], auto)
  ultimately show ?thesis
  using  $\langle r > 0 \rangle$  by (auto simp: has_fps_expansion_def)
qed

```

```

lemma fps_conv_radius_tan:
  fixes c :: complex
  assumes c ≠ 0
  shows fps_conv_radius (fps_tan c) ≥ pi / (2 * norm c)
proof -
  have fps_conv_radius (fps_tan c) ≥
    Min {pi / (2 * norm c), fps_conv_radius (fps_sin c), fps_conv_radius
(fps_cos c)}
    unfolding fps_tan_def
  proof (rule fps_conv_radius_divide)
    fix z :: complex assume z ∈ eball 0 (pi / (2 * norm c))
    with cos_eq_zero_imp_norm_ge[of c*z] assms
    show eval_fps (fps_cos c) z ≠ 0 by (auto simp: norm_mult field_simps)
  qed (insert assms, auto)
  thus ?thesis by (simp add: min_def)
qed

lemma eval_fps_tan:
  fixes c :: complex
  assumes norm z < pi / (2 * norm c)
  shows eval_fps (fps_tan c) z = tan (c * z)
proof (cases c = 0)
  case False
  show ?thesis unfolding fps_tan_def
  proof (subst eval_fps_divide'[where r = pi / (2 * norm c)])
    fix z :: complex assume z ∈ eball 0 (pi / (2 * norm c))
    with cos_eq_zero_imp_norm_ge[of c*z] assms
    show eval_fps (fps_cos c) z ≠ 0 using False by (auto simp: norm_mult
field_simps)
  qed (use False assms in ‹auto simp: field_simps tan_def›)
qed simp_all

end

```

5 Conformal Mappings and Consequences of Cauchy's Integral Theorem

By John Harrison et al. Ported from HOL Light by L C Paulson (2016)

Also Cauchy's residue theorem by Wenda Li (2016)

```

theory Conformal_Mappings
imports Cauchy_Integral_Formula

```

```

begin

```

5.1 Analytic continuation

proposition *isolated_zeros*:

assumes *holf*: f holomorphic_on S
 and open S connected S $\xi \in S$ $f \xi = 0$ $\beta \in S$ $f \beta \neq 0$
 obtains r where $0 < r$ and ball ξ $r \subseteq S$ and
 $\bigwedge z. z \in \text{ball } \xi \ r - \{\xi\} \implies f z \neq 0$
proof –
 obtain r where $0 < r$ and r : ball ξ $r \subseteq S$
 using $\langle \text{open } S \rangle \langle \xi \in S \rangle$ open_contains_ball_eq by blast
 have powf: $((\lambda n. (\text{deriv } \sim^n) f \xi / (\text{fact } n) * (z - \xi)^{\sim n}) \text{ sums } f z)$ if $z \in \text{ball } \xi \ r$ for z
 by (intro holomorphic_power_series [OF _ that] holomorphic_on_subset [OF holf r])
 obtain m where m : $(\text{deriv } \sim^m) f \xi / (\text{fact } m) \neq 0$
 using holomorphic_fun_eq_0_on_connected [OF holf $\langle \text{open } S \rangle \langle \text{connected } S \rangle$
 $_ \langle \xi \in S \rangle \langle \beta \in S \rangle \langle f \beta \neq 0 \rangle$
 by auto
 then have $m \neq 0$ using assms(5) funpow_0 by fastforce
 obtain s where $0 < s$ and s : $\bigwedge z. z \in \text{cball } \xi \ s - \{\xi\} \implies f z \neq 0$
 using powser_0_nonzero [OF $\langle 0 < r \rangle$ powf $\langle f \xi = 0 \rangle m$]
 by (metis $\langle m \neq 0 \rangle$ dist_norm mem_ball norm_minus_commute not_gr_zero)
 have $0 < \min r \ s$ by (simp add: $\langle 0 < r \rangle \langle 0 < s \rangle$)
 then show thesis
 proof qed (use $r \ s$ in auto)
 qed

proposition *analytic_continuation*:

assumes *holf*: f holomorphic_on S
 and open S and connected S
 and $U \subseteq S$ and $\xi \in S$
 and ξ islimpt U
 and $fU0$ [simp]: $\bigwedge z. z \in U \implies f z = 0$
 and $w \in S$
 shows $f w = 0$
proof –
 obtain e where $0 < e$ and e : cball ξ $e \subseteq S$
 using $\langle \text{open } S \rangle \langle \xi \in S \rangle$ open_contains_cball_eq by blast
 define T where $T = \text{cball } \xi \ e \cap U$
 have conf: continuous_on (closure T) f
 by (metis T_def closed_cball closure_minimal e holf holomorphic_on_imp_continuous_on
 holomorphic_on_subset inf.cobounded1)
 have fT0 [simp]: $\bigwedge x. x \in T \implies f x = 0$
 by (simp add: T_def)
 have $\bigwedge r. [\forall e > 0. \exists x' \in U. x' \neq \xi \wedge \text{dist } x' \xi < e; 0 < r] \implies \exists x' \in \text{cball } \xi \ e \cap U. x' \neq \xi \wedge \text{dist } x' \xi < r$
 by (metis $\langle 0 < e \rangle$ IntI dist_commute less_eq_real_def mem_cball min_less_iff_conj)
 then have ξ islimpt T using $\langle \xi$ islimpt $U \rangle$
 by (auto simp: T_def islimpt_approachable)
 then have $\xi \in \text{closure } T$

```

    by (simp add: closure_def)
  then have  $f \xi = 0$ 
    by (auto simp: continuous_constant_on_closure [OF contf])
  moreover have  $\bigwedge r. \llbracket 0 < r; \bigwedge z. z \in \text{ball } \xi \ r - \{\xi\} \implies f z \neq 0 \rrbracket \implies \text{False}$ 
    by (metis open_ball  $\langle \xi \text{ islimpt } T \rangle$  centre_in_ball fT0 insertE insert_Diff
    islimptE)
  ultimately show ?thesis
    by (metis  $\langle \text{open } S \rangle$   $\langle \text{connected } S \rangle$   $\langle \xi \in S \rangle$   $\langle w \in S \rangle$  holf isolated_zeros)
qed

```

```

corollary analytic_continuation_open:
  assumes open s and open s' and  $s \neq \{\}$  and connected s'
    and  $s \subseteq s'$ 
  assumes f holomorphic_on s' and g holomorphic_on s'
    and  $\bigwedge z. z \in s \implies f z = g z$ 
  assumes  $z \in s'$ 
  shows  $f z = g z$ 
proof -
  from  $\langle s \neq \{\} \rangle$  obtain  $\xi$  where  $\xi \in s$  by auto
  with  $\langle \text{open } s \rangle$  have  $\xi: \xi \text{ islimpt } s$ 
    by (intro interior_limit_point) (auto simp: interior_open)
  have  $f z - g z = 0$ 
    by (rule analytic_continuation[of  $\lambda z. f z - g z \ s' \ s \ \xi$ ])
      (insert assms  $\langle \xi \in s \rangle \ \xi$ , auto intro: holomorphic_intros)
  thus ?thesis by simp
qed

```

```

corollary analytic_continuation':
  assumes f holomorphic_on S open S connected S
    and  $U \subseteq S \ \xi \in S \ \xi \text{ islimpt } U$ 
    and f constant_on U
  shows f constant_on S
proof -
  obtain c where  $c: \bigwedge x. x \in U \implies f x - c = 0$ 
    by (metis  $\langle f \text{ constant_on } U \rangle$  constant_on_def diff_self)
  have  $(\lambda z. f z - c)$  holomorphic_on S
    using assms by (intro holomorphic_intros)
  with c analytic_continuation assms have  $\bigwedge x. x \in S \implies f x - c = 0$ 
    by blast
  then show ?thesis
    unfolding constant_on_def by force
qed

```

```

lemma holomorphic_compact_finite_zeros:
  assumes S: f holomorphic_on S open S connected S
    and compact K  $K \subseteq S$ 
    and  $\neg f \text{ constant_on } S$ 
  shows finite  $\{z \in K. f z = 0\}$ 
proof (rule ccontr)

```

```

assume infinite  $\{z \in K. f z = 0\}$ 
then obtain  $z$  where  $z \in K$  and  $z: z \text{ islimpt } \{z \in K. f z = 0\}$ 
  using  $\langle \text{compact } K \rangle$  by (auto simp: compact_eq_Bolzano_Weierstrass)
moreover have  $\{z \in K. f z = 0\} \subseteq S$ 
  using  $\langle K \subseteq S \rangle$  by blast
ultimately show False
  using assms analytic_continuation [OF S] unfolding constant_on_def
  by blast
qed

```

lemma *holomorphic_countable_zeros*:

```

assumes  $S: f \text{ holomorphic\_on } S$   $open\ S$   $connected\ S$  and  $f \text{ sigma } S$ 
  and  $\neg f \text{ constant\_on } S$ 
shows countable  $\{z \in S. f z = 0\}$ 
proof –
  obtain  $F::nat \Rightarrow \text{complex set}$ 
    where  $F: \text{range } F \subseteq \text{Collect compact}$  and  $Seq: S = (\bigcup i. F i)$ 
    using  $\langle f \text{ sigma } S \rangle$  by (meson fsigma_Union_compact)
  have  $fin: \text{finite } \{z \in F i. f z = 0\}$  for  $i$ 
    using holomorphic_compact_finite_zeros assms F Seq Union_iff by blast
  have  $\{z \in S. f z = 0\} = (\bigcup i. \{z \in F i. f z = 0\})$ 
    using Seq by auto
  with fin show ?thesis
    by (simp add: countable_finite)
qed

```

lemma *holomorphic_countable_equal*:

```

assumes  $f \text{ holomorphic\_on } S$   $g \text{ holomorphic\_on } S$   $open\ S$   $connected\ S$  and  $f \text{ sigma } S$ 
  and  $eq: \text{uncountable } \{z \in S. f z = g z\}$ 
shows  $S \subseteq \{z \in S. f z = g z\}$ 
proof –
  obtain  $z$  where  $z: z \in S \wedge f z = g z$ 
    using eq not_finite_existsD uncountable_infinite by blast
  have  $(\lambda x. f x - g x) \text{ holomorphic\_on } S$ 
    by (simp add: assms holomorphic_on_diff)
  then have  $(\lambda x. f x - g x) \text{ constant\_on } S$ 
    using holomorphic_countable_zeros assms by force
  with  $z$  have  $\bigwedge x. x \in S \implies f x - g x = 0$ 
    unfolding constant_on_def by force
  then show ?thesis
    by auto
qed

```

lemma *holomorphic_countable_equal_UNIV*:

```

assumes  $fg: f \text{ holomorphic\_on } UNIV$   $g \text{ holomorphic\_on } UNIV$ 
  and  $eq: \text{uncountable } \{z. f z = g z\}$ 
shows  $f=g$ 
using holomorphic_countable_equal [OF fg] eq by fastforce

```

5.2 Open mapping theorem

lemma *holomorphic_contract_to_zero*:

assumes *contf*: *continuous_on* (*cball* ξ r) *f*
and *holf*: *f* *holomorphic_on* *ball* ξ r
and $0 < r$
and *norm_less*: $\bigwedge z. \text{norm}(\xi - z) = r \implies \text{norm}(f \xi) < \text{norm}(f z)$
obtains *z* **where** $z \in \text{ball } \xi \ r \wedge f z = 0$

proof –

{ **assume** *fnz*: $\bigwedge w. w \in \text{ball } \xi \ r \implies f w \neq 0$
then have $0 < \text{norm } (f \xi)$
by (*simp add*: $\langle 0 < r \rangle$)
have *fnz'*: $\bigwedge w. w \in \text{cball } \xi \ r \implies f w \neq 0$
using *dist_complex_def* *fnz* *norm_less* *order_le_less* **by** *fastforce*
have *frontier*(*cball* ξ r) $\neq \{\}$
using $\langle 0 < r \rangle$ **by** *simp*
define *g* **where** [*abs_def*]: $g z = \text{inverse } (f z)$ **for** *z*
have *contg*: *continuous_on* (*cball* ξ r) *g*
unfolding *g_def* **using** *contf* *continuous_on_inverse* *fnz'* **by** *blast*
have *holg*: *g* *holomorphic_on* *ball* ξ r
unfolding *g_def* **using** *fnz* *holf* *holomorphic_on_inverse* **by** *blast*
have *frontier* (*cball* ξ r) \subseteq *cball* ξ r
by (*simp add*: *subset_iff*)
then have *contf'*: *continuous_on* (*frontier* (*cball* ξ r)) *f*
and *contg'*: *continuous_on* (*frontier* (*cball* ξ r)) *g*
by (*blast intro*: *contf* *contg* *continuous_on_subset*)
have *froc*: *frontier*(*cball* ξ r) $\neq \{\}$
using $\langle 0 < r \rangle$ **by** *simp*
moreover have *continuous_on* (*frontier* (*cball* ξ r)) (*norm* \circ *f*)
using *contf'* *continuous_on_compose* *continuous_on_norm_id* **by** *blast*
ultimately obtain *w* **where** $w \in \text{frontier}(\text{cball } \xi \ r)$
and *now*: $\bigwedge x. x \in \text{frontier}(\text{cball } \xi \ r) \implies \text{norm } (f w) \leq \text{norm}$
(*f x*)
using *continuous_attains_inf* [*OF compact_frontier* [*OF compact_cball*]]
by (*metis comp_apply*)
then have *fw*: $0 < \text{norm } (f w)$
by (*simp add*: *fnz'*)
have *continuous_on* (*frontier* (*cball* ξ r)) (*norm* \circ *g*)
using *contg'* *continuous_on_compose* *continuous_on_norm_id* **by** *blast*
then obtain *v* **where** $v \in \text{frontier}(\text{cball } \xi \ r)$
and *nov*: $\bigwedge x. x \in \text{frontier}(\text{cball } \xi \ r) \implies \text{norm } (g v) \geq \text{norm } (g x)$
using *continuous_attains_sup* [*OF compact_frontier* [*OF compact_cball*]]
froc] **by** *force*
then have *fv*: $0 < \text{norm } (f v)$
by (*simp add*: *fnz'*)
have $\text{norm } ((\text{deriv } \sim 0) \ g \ \xi) \leq \text{fact } 0 * \text{norm } (g v) / r \wedge 0$
by (*rule Cauchy_inequality* [*OF holg* *contg* $\langle 0 < r \rangle$]) (*simp add*: *dist_norm*
nov)
then have *cmod* (*g* ξ) \leq *cmod* (*g* v)
by *simp*

```

moreover have cmod ( $\xi - w$ ) =  $r$ 
  by (metis (no_types) dist_norm frontier_cball mem_sphere w)
ultimately obtain wr: norm ( $\xi - w$ ) =  $r$  and nfw: norm ( $f w$ )  $\leq$  norm ( $f \xi$ )
  unfolding g_def
  by (smt (verit, del_insts)  $\langle 0 < \text{cmod } (f \xi) \rangle$  inverse_le_imp_le norm_inverse
now v)
  with fw have False
    using norm_less by force
}
with that show ?thesis by blast
qed

```

theorem open_mapping_thm:

```

  assumes holf:  $f$  holomorphic_on  $S$ 
    and  $S$ : open  $S$  and connected  $S$ 
    and open  $U$  and  $U \subseteq S$ 
    and fne:  $\neg f$  constant_on  $S$ 
  shows open ( $f^{-1} U$ )
proof -
  have *: open ( $f^{-1} U$ )
    if  $U \neq \{\}$  and  $U$ : open  $U$  connected  $U$  and  $f$  holomorphic_on  $U$  and
fneU:  $\bigwedge x. \exists y \in U. f y \neq x$ 
    for  $U$ 
  proof (clarsimp simp: open_contains_ball)
  fix  $\xi$  assume  $\xi \in U$ 
  show  $\exists e > 0. \text{ball } (f \xi) e \subseteq f^{-1} U$ 
  proof -
  have hol:  $(\lambda z. f z - f \xi)$  holomorphic_on  $U$ 
    by (rule holomorphic_intros that)+
  obtain  $s$  where  $0 < s$  and sbU:  $\text{ball } \xi s \subseteq U$ 
    and sne:  $\bigwedge z. z \in \text{ball } \xi s - \{\xi\} \implies (\lambda z. f z - f \xi) z \neq 0$ 
    using isolated_zeros [OF hol U  $\xi$ ] by (metis fneU right_minus_eq)
  obtain  $r$  where  $0 < r$  and r:  $\text{cball } \xi r \subseteq \text{ball } \xi s$ 
    using  $\langle 0 < s \rangle$  by (rule_tac  $r=s/2$  in that) auto
  have cball  $\xi r \subseteq U$ 
    using sbU  $r$  by blast
  then have frsbU:  $\text{frontier } (\text{cball } \xi r) \subseteq U$ 
    using Diff_subset frontier_def order_trans by fastforce
  then have cof: compact ( $\text{frontier } (\text{cball } \xi r)$ )
    by blast
  have frne:  $\text{frontier } (\text{cball } \xi r) \neq \{\}$ 
    using  $\langle 0 < r \rangle$  by auto
  have conf: continuous_on ( $\text{frontier } (\text{cball } \xi r)$ )  $(\lambda z. \text{norm } (f z - f \xi))$ 
    by (metis continuous_on_norm continuous_on_subset frsbU hol holomorphic_on_imp_continuous_on)
  obtain  $w$  where norm ( $\xi - w$ ) =  $r$ 
    and  $w$ :  $(\bigwedge z. \text{norm } (\xi - z) = r \implies \text{norm } (f w - f \xi) \leq \text{norm } (f z - f \xi))$ 
    using continuous_attains_inf [OF cof frne conf] by (auto simp: dist_norm)

```



```

moreover define  $\varepsilon$  where  $\varepsilon \equiv \text{norm } (f\ w - f\ \xi) / 3$ 
ultimately have  $0 < \varepsilon$ 
  using  $\langle 0 < r \rangle$  dist_complex_def  $r$  sne by auto
have  $\text{ball } (f\ \xi)\ \varepsilon \subseteq f\ ' U$ 
proof
  fix  $\gamma$ 
  assume  $\gamma: \gamma \in \text{ball } (f\ \xi)\ \varepsilon$ 
  have  $*$ :  $\text{cmod } (\gamma - f\ \xi) < \text{cmod } (\gamma - f\ z)$  if  $\text{cmod } (\xi - z) = r$  for  $z$ 
  proof  $-$ 
    have  $\text{lt}$ :  $\text{cmod } (f\ w - f\ \xi) / 3 < \text{cmod } (\gamma - f\ z)$ 
    using  $w$  [OF that]  $\gamma$ 
    using dist_triangle2 [of  $f\ \xi\ \gamma\ f\ z$ ] dist_triangle2 [of  $f\ \xi\ f\ z\ \gamma$ ]
    by (simp add: _def dist_norm norm_minus_commute)
    show ?thesis
    by (metis _def dist_commute dist_norm less_trans lt mem_ball  $\gamma$ )
  qed
have continuous_on (cball  $\xi\ r$ ) ( $\lambda z. \gamma - f\ z$ )
  using  $\langle \text{cball } \xi\ r \subseteq U \rangle$  f holomorphic_on U
by (force intro: continuous_intros continuous_on_subset holomorphic_on_imp_continuous_on)
moreover have ( $\lambda z. \gamma - f\ z$ ) holomorphic_on  $\text{ball } \xi\ r$ 
  using  $\langle \text{cball } \xi\ r \subseteq U \rangle$  ball_subset_cball holomorphic_on_subset that(4)
  by (intro holomorphic_intros) blast
ultimately obtain  $z$  where  $z \in \text{ball } \xi\ r$   $\gamma - f\ z = 0$ 
  using  $*$   $\langle 0 < r \rangle$  holomorphic_contract_to_zero by blast
then show  $\gamma \in f\ ' U$ 
  using  $\langle \text{cball } \xi\ r \subseteq U \rangle$  by fastforce
qed
then show ?thesis using  $\langle 0 < \varepsilon \rangle$  by blast
qed
have open ( $f\ ' X$ ) if  $X \in \text{components } U$  for  $X$ 
proof  $-$ 
  have holfU:  $f$  holomorphic_on  $U$ 
  using  $\langle U \subseteq S \rangle$  holf holomorphic_on_subset by blast
have  $X \neq \{\}$ 
  using that by (simp add: in_components_nonempty)
moreover have open  $X$ 
  using that  $\langle \text{open } U \rangle$  open_components by auto
moreover have connected  $X$ 
  using that in_components_maximal by blast
moreover have  $f$  holomorphic_on  $X$ 
  by (meson that holfU holomorphic_on_subset in_components_maximal)
moreover have  $\exists y \in X. f\ y \neq x$  for  $x$ 
proof (rule ccontr)
  assume not:  $\neg (\exists y \in X. f\ y \neq x)$ 
  have  $X \subseteq S$ 
  using  $\langle U \subseteq S \rangle$  in_components_subset that by blast
obtain  $w$  where  $w \in X$  using  $\langle X \neq \{\} \rangle$  by blast
have wis:  $w$  islimpt  $X$ 

```

```

    using  $w \langle \text{open } X \rangle \text{interior\_eq}$  by auto
  have hol:  $(\lambda z. f z - x)$  holomorphic_on  $S$ 
    by (simp add: holf holomorphic_on_diff)
  with fne [unfolded constant_on_def]
    analytic_continuation[OF hol  $S \langle \text{connected } S \rangle \langle X \subseteq S \rangle \_ \text{wis}$ ] not  $\langle X \subseteq S \rangle w$ 
  show False by auto
qed
ultimately show ?thesis
  by (rule *)
qed
then have open  $(f ' \bigcup (\text{components } U))$ 
  by (metis (no_types, lifting) imageE image_Union open_Union)
then show ?thesis
  by force
qed

```

No need for S to be connected. But the nonconstant condition is stronger.

corollary *open_mapping_thm2:*

```

  assumes holf:  $f$  holomorphic_on  $S$ 
    and  $S$ : open  $S$ 
    and open  $U$   $U \subseteq S$ 
    and fnc:  $\bigwedge X. [\text{open } X; X \subseteq S; X \neq \{\}] \implies \neg f \text{constant\_on } X$ 
  shows open  $(f ' U)$ 
proof -
  have  $S = \bigcup (\text{components } S)$  by simp
  with  $\langle U \subseteq S \rangle$  have  $U = (\bigcup C \in \text{components } S. C \cap U)$  by auto
  then have  $f ' U = (\bigcup C \in \text{components } S. f ' (C \cap U))$ 
    using image_UN by fastforce
  moreover
  { fix  $C$  assume  $C \in \text{components } S$ 
    with  $S \langle C \in \text{components } S \rangle \text{open\_components in\_components\_connected}$ 
    have  $C$ : open  $C$  connected  $C$  by auto
    have  $C \subseteq S$ 
      by (metis  $\langle C \in \text{components } S \rangle \text{in\_components\_maximal}$ )
    have nf:  $\neg f \text{constant\_on } C$ 
      using  $\langle \text{open } C \rangle \langle C \in \text{components } S \rangle \langle C \subseteq S \rangle$  fnc in_components_nonempty
  }
by blast
  have  $f$  holomorphic_on  $C$ 
    by (metis holf holomorphic_on_subset  $\langle C \subseteq S \rangle$ )
  then have open  $(f ' (C \cap U))$ 
    by (meson  $C \langle \text{open } U \rangle \text{inf\_le1 nf open\_Int open\_mapping\_thm}$ )
} ultimately show ?thesis
  by force
qed

```

corollary *open_mapping_thm3:*

```

  assumes  $f$  holomorphic_on  $S$ 
    and open  $S$  and inj_on  $f$   $S$ 

```

```

  shows open (f ` S)
  by (meson assms inj_on_subset injective_not_constant open_mapping_thm2
order.refl)

```

5.3 Maximum modulus principle

If f is holomorphic, then its norm (modulus) cannot exhibit a true local maximum that is properly within the domain of f .

proposition *maximum_modulus_principle*:

```

  assumes holf: f holomorphic_on S
    and S: open S and connected S
    and open U and U ⊆ S and ξ ∈ U
    and no: ⋀z. z ∈ U ⇒ norm(f z) ≤ norm(f ξ)
  shows f constant_on S
proof (rule ccontr)
  assume ¬ f constant_on S
  then have open (f ` U)
    using open_mapping_thm assms by blast
  moreover have ¬ open (f ` U)
  proof -
    have ∃ t. cmod (f ξ - t) < e ∧ t ∉ f ` U if 0 < e for e
      using that
      apply (rule_tac x=if 0 < Re(f ξ) then f ξ + (e/2) else f ξ - (e/2) in exI)
      apply (simp add: dist_norm)
      apply (fastforce simp: cmod_Re_le_iff dest!: no dest: sym)
      done
    then show ?thesis
      unfolding open_contains_ball by (metis ⟨ξ ∈ U⟩ contra_subsetD dist_norm
imageI mem_ball)
  qed
  ultimately show False
    by blast
qed

```

proposition *maximum_modulus_frontier*:

```

  assumes holf: f holomorphic_on (interior S)
    and conf: continuous_on (closure S) f
    and bos: bounded S
    and leB: ⋀z. z ∈ frontier S ⇒ norm(f z) ≤ B
    and ξ ∈ S
  shows norm(f ξ) ≤ B
proof -
  have compact (closure S) using bos
    by (simp add: bounded_closure compact_eq_bounded_closed)
  moreover have continuous_on (closure S) (cmod ∘ f)
    using conf continuous_on_compose continuous_on_norm_id by blast
  ultimately obtain z where z ∈ closure S and z: ⋀y. y ∈ closure S ⇒ (cmod
  ∘ f) y ≤ (cmod ∘ f) z
    using continuous_attains_sup [of closure S norm o f] ⟨ξ ∈ S⟩ by auto

```

```

then consider  $z \in \text{frontier } S \mid z \in \text{interior } S$  using frontier_def by auto
then have  $\text{norm}(f z) \leq B$ 
proof cases
  case 1 then show ?thesis using leB by blast
next
  case 2
  have  $f \text{ constant\_on } (\text{connected\_component\_set } (\text{interior } S) z)$ 
  proof (rule maximum_modulus_principle)
    show  $f \text{ holomorphic\_on } \text{connected\_component\_set } (\text{interior } S) z$ 
    by (metis connected_component_subset holf holomorphic_on_subset)
    show  $z \in \text{connected\_component\_set } (\text{interior } S) z$ 
    by (simp add: 2)
    show  $\bigwedge W. W \in \text{connected\_component\_set } (\text{interior } S) z \implies \text{cmod } (f W)$ 
     $\leq \text{cmod } (f z)$ 
    using closure_def connected_component_subset z by fastforce
  qed (auto simp: open_connected_component)
  then obtain  $c$  where  $c: \bigwedge w. w \in \text{connected\_component\_set } (\text{interior } S) z$ 
   $\implies f w = c$ 
  by (auto simp: constant_on_def)
  have  $f \text{ `closure}(\text{connected\_component\_set } (\text{interior } S) z) \subseteq \{c\}$ 
  proof (rule image_closure_subset)
    show  $\text{continuous\_on } (\text{closure } (\text{connected\_component\_set } (\text{interior } S) z)) f$ 
    by (meson closure_mono connected_component_subset contf continuous_on_subset interior_subset)
  qed (use c in auto)
  then have  $cc: \bigwedge w. w \in \text{closure}(\text{connected\_component\_set } (\text{interior } S) z) \implies$ 
   $f w = c$  by blast
  have  $\text{connected\_component } (\text{interior } S) z z$ 
  by (simp add: 2)
  moreover have  $\text{connected\_component\_set } (\text{interior } S) z \neq \text{UNIV}$ 
  by (metis bos bounded_interior connected_component_eq_UNIV not_bounded_UNIV)
  ultimately have  $\text{frontier}(\text{connected\_component\_set } (\text{interior } S) z) \neq \{\}$ 
  by (meson 2 connected_component_eq_empty frontier_not_empty)
  then obtain  $w$  where  $w: w \in \text{frontier}(\text{connected\_component\_set } (\text{interior } S)$ 
   $z)$ 
  by auto
  then have  $\text{norm } (f z) = \text{norm } (f w)$  by (simp add: 2 c cc frontier_def)
  also have  $\dots \leq B$ 
  using  $w \text{ frontier\_interior\_subset frontier\_of\_connected\_component\_subset}$ 
  by (blast intro: leB)
  finally show ?thesis .
qed
then show ?thesis
  using  $z \in \langle S \rangle \text{ closure\_subset}$  by fastforce
qed

corollary maximum_real_frontier:
  assumes  $\text{holf: } f \text{ holomorphic\_on } (\text{interior } S)$ 
  and  $\text{contf: continuous\_on } (\text{closure } S) f$ 

```

```

and bos: bounded S
and leB:  $\bigwedge z. z \in \text{frontier } S \implies \text{Re}(f z) \leq B$ 
and  $\xi \in S$ 
shows  $\text{Re}(f \xi) \leq B$ 
using maximum_modulus_frontier [of exp o f S exp B]
  Transcendental.continuous_on_exp holomorphic_on_compose holomorphic_on_exp
  assms
by auto

```

5.4 Factoring out a zero according to its order

```

lemma holomorphic_factor_order_of_zero:
  assumes holf: f holomorphic_on S
    and os: open S
    and  $\xi \in S$   $0 < n$ 
    and dnz:  $(\text{deriv } \sim^n) f \xi \neq 0$ 
    and dfz:  $\bigwedge i. [0 < i; i < n] \implies (\text{deriv } \sim^i) f \xi = 0$ 
  obtains g r where  $0 < r$ 
    g holomorphic_on ball  $\xi$  r
     $\bigwedge w. w \in \text{ball } \xi r \implies f w - f \xi = (w - \xi)^n * g w$ 
     $\bigwedge w. w \in \text{ball } \xi r \implies g w \neq 0$ 
proof -
  obtain r where  $r > 0$  and r: ball  $\xi$  r  $\subseteq S$  using assms by (blast elim!: openE)
  then have holfb: f holomorphic_on ball  $\xi$  r
    using holf holomorphic_on_subset by blast
  define g where  $g w = \text{suminf } (\lambda i. (\text{deriv } \sim^{(i+n)}) f \xi / (\text{fact}(i+n)) * (w - \xi)^i)$ 
  for w
  have sumsg:  $(\lambda i. (\text{deriv } \sim^{(i+n)}) f \xi / (\text{fact}(i+n)) * (w - \xi)^i) \text{ sums } g w$ 
  and feq:  $f w - f \xi = (w - \xi)^n * g w$ 
    if w: w  $\in \text{ball } \xi r$  for w
  proof -
    define powf where  $\text{powf} = (\lambda i. (\text{deriv } \sim^i) f \xi / (\text{fact } i) * (w - \xi)^i)$ 
    have [simp]: powf 0 = f  $\xi$ 
      by (simp add: powf_def)
    have sing:  $\{..<n\} - \{i. \text{powf } i = 0\} = (\text{if } f \xi = 0 \text{ then } \{\} \text{ else } \{0\})$ 
    unfolding powf_def using  $\langle 0 < n \rangle$  dfz by (auto simp: dfz; metis funpow_0
  not_gr0)
    have powf sums f w
      unfolding powf_def by (rule holomorphic_power_series [OF holfb w])
    moreover have  $(\sum i < n. \text{powf } i) = f \xi$ 
      by (subst sum.setdiff_irrelevant [symmetric]; simp add: dfz sing)
    ultimately have fsums:  $(\lambda i. \text{powf } (i+n)) \text{ sums } (f w - f \xi)$ 
      using w sums_iff_shift' by metis
    then have *: summable  $(\lambda i. (w - \xi)^n * ((\text{deriv } \sim^{(i+n)}) f \xi * (w - \xi)^{i-n} / \text{fact } (i+n)))$ 
      unfolding powf_def using sums_summable
      by (auto simp: power_add mult_ac)
    have summable  $(\lambda i. (\text{deriv } \sim^{(i+n)}) f \xi * (w - \xi)^i / \text{fact } (i+n))$ 
      proof (cases w= $\xi$ )

```

```

    case False then show ?thesis
      using summable_mult [OF *, of 1 / (w - ξ) ^ n] by simp
    next
      case True then show ?thesis
        by (auto simp: Power.semiring_1_class.power_0_left intro!: summable_finite
            [of {0}])
            split: if_split_asm)
      qed
      then show sumsg: (λi. (deriv ^^ (i + n)) f ξ / (fact(i + n)) * (w - ξ) ^ i)
sums g w
        by (simp add: summable_sums_iff g_def)
      show f w - f ξ = (w - ξ) ^ n * g w
        using sums_mult [OF sumsg, of (w - ξ) ^ n]
        by (intro sums_unique2 [OF fsums]) (auto simp: power_add mult_ac powf_def)
      qed
      then have holg: g holomorphic_on ball ξ r
        by (meson sumsg power_series_holomorphic)
      then have contg: continuous_on (ball ξ r) g
        by (blast intro: holomorphic_on_imp_continuous_on)
      have g ξ ≠ 0
        using dnz unfolding g_def
        by (subst suminf_finite [of {0}]) auto
      obtain d where 0 < d and d: ∧w. w ∈ ball ξ d ⟹ g w ≠ 0
        using ⟨0 < r⟩ continuous_on_avoid [OF contg _ ⟨g ξ ≠ 0⟩]
        by (metis centre_in_ball le_cases mem_ball mem_ball_leI)
      show ?thesis
      proof
        show g holomorphic_on ball ξ (min r d)
          using holg by (auto simp: feq holomorphic_on_subset subset_ball d)
        qed (use ⟨0 < r⟩ ⟨0 < d⟩ in ⟨auto simp: feq d⟩)
      qed

```

```

lemma holomorphic_factor_order_of_zero_strong:
  assumes holf: f holomorphic_on S open S ξ ∈ S 0 < n
    and (deriv ^^ n) f ξ ≠ 0
    and ∧i. [0 < i; i < n] ⟹ (deriv ^^ i) f ξ = 0
  obtains g r where 0 < r
    g holomorphic_on ball ξ r
    ∧w. w ∈ ball ξ r ⟹ f w - f ξ = ((w - ξ) * g w) ^ n
    ∧w. w ∈ ball ξ r ⟹ g w ≠ 0
proof -
  obtain g r where 0 < r
    and holg: g holomorphic_on ball ξ r
    and feq: ∧w. w ∈ ball ξ r ⟹ f w - f ξ = (w - ξ) ^ n * g w
    and gne: ∧w. w ∈ ball ξ r ⟹ g w ≠ 0
  by (auto intro: holomorphic_factor_order_of_zero [OF assms])
  have con: continuous_on (ball ξ r) (λz. deriv g z / g z)
    by (rule continuous_intros) (auto simp: gne holg holomorphic_deriv holomor-

```

```

phic_on_imp_continuous_on)
  have cd: ( $\lambda z. \text{deriv } g \, z / g \, z$ ) field_differentiable at  $x$  if  $\text{dist } \xi \, x < r$  for  $x$ 
  proof (intro derivative_intros)
    show  $\text{deriv } g$  field_differentiable at  $x$ 
    using that holg mem_ball by (blast intro: holomorphic_deriv holomorphic_on_imp_differentiable_at)
    show  $g$  field_differentiable at  $x$ 
    by (metis that open_ball_at_within_open holg holomorphic_on_def mem_ball)
  qed (simp add: gne that)
  obtain  $h$  where  $h: \bigwedge x. x \in \text{ball } \xi \, r \implies (h \text{ has\_field\_derivative } \text{deriv } g \, x / g \, x) \text{ (at } x)$ 
  using holomorphic_convex_primitive [of ball  $\xi \, r$   $\{\}$   $\lambda z. \text{deriv } g \, z / g \, z$ ]
  by (metis (no_types, lifting) Diff_empty_at_within_interior cd con_convex_ball_infinite_imp_nonempty_interior_ball mem_ball)
  then have continuous_on (ball  $\xi \, r$ )  $h$ 
  by (metis open_ball_holomorphic_on_imp_continuous_on holomorphic_on_open)
  then have con: continuous_on (ball  $\xi \, r$ ) ( $\lambda x. \text{exp } (h \, x) / g \, x$ )
  by (auto intro!: continuous_intros simp add: holg holomorphic_on_imp_continuous_on gne)
  have gfd:  $\text{dist } \xi \, x < r \implies g \text{ field\_differentiable at } x$  if  $\text{dist } \xi \, x < r$  for  $x$ 
  using holg holomorphic_on_imp_differentiable_at by auto
  have 0:  $\text{dist } \xi \, x < r \implies ((\lambda x. \text{exp } (h \, x) / g \, x) \text{ has\_field\_derivative } 0) \text{ (at } x)$ 
  for  $x$ 
  by (rule gfd h derivative_eq_intros DERIV_deriv_iff_field_differentiable [THEN iffD2] | simp add: gne)+
  obtain  $c$  where  $c: \bigwedge x. x \in \text{ball } \xi \, r \implies \text{exp } (h \, x) / g \, x = c$ 
  by (rule DERIV_zero_connected_constant [of ball  $\xi \, r$   $\{\}$   $\lambda x. \text{exp } (h \, x) / g \, x$ ])
  (auto simp: con 0)
  have hol: ( $\lambda z. \text{exp } ((Ln (\text{inverse } c) + h \, z) / \text{of\_nat } n)$ ) holomorphic_on ball  $\xi \, r$ 
  proof (intro holomorphic_intros holomorphic_on_compose [unfolded o_def, where  $g = \text{exp}$ ])
    show  $h$  holomorphic_on ball  $\xi \, r$ 
    using h holomorphic_on_open by blast
  qed (use  $\langle 0 < n \rangle$  in auto)
  show ?thesis
  proof
    show  $\bigwedge w. w \in \text{ball } \xi \, r \implies f \, w - f \, \xi = ((w - \xi) * \text{exp } ((Ln (\text{inverse } c) + h \, w) / \text{of\_nat } n)) ^ n$ 
    using  $\langle 0 < n \rangle$ 
    by (auto simp: feq power_mult_distrib exp_divide_power_eq exp_add gne simp flip: c)
  qed (use hol  $\langle 0 < r \rangle$  in auto)
qed

```

lemma

```

fixes  $k :: 'a::wellorder$ 
assumes  $a\_def: a == \text{LEAST } x. P \, x$  and  $P: P \, k$ 
shows  $\text{def\_LeastI}: P \, a$  and  $\text{def\_Least\_le}: a \leq k$ 

```

```

unfolding a_def
by (rule LeastI Least_le; rule P)+

lemma holomorphic_factor_zero_nonconstant:
  assumes holf:  $f$  holomorphic_on  $S$  and  $S$ : open  $S$  connected  $S$ 
    and  $\xi \in S$   $f \xi = 0$ 
    and nonconst:  $\neg f$  constant_on  $S$ 
  obtains  $g$   $r$   $n$ 
    where  $0 < n$   $0 < r$   $\text{ball } \xi \ r \subseteq S$ 
       $g$  holomorphic_on  $\text{ball } \xi \ r$ 
       $\bigwedge w. w \in \text{ball } \xi \ r \implies f w = (w - \xi)^n * g w$ 
       $\bigwedge w. w \in \text{ball } \xi \ r \implies g w \neq 0$ 
proof (cases  $\forall n > 0. (\text{deriv } ^n) f \xi = 0$ )
  case True then show ?thesis
    using holomorphic_fun_eq_const_on_connected [OF holf  $S$   $\langle \xi \in S \rangle$ ] non-
    const by (simp add: constant_on_def)
  next
    case False
    then obtain  $n0$  where  $n0 > 0$  and  $n0$ :  $(\text{deriv } ^{n0}) f \xi \neq 0$  by blast
    obtain  $r0$  where  $r0 > 0$   $\text{ball } \xi \ r0 \subseteq S$  using  $S$  openE  $\langle \xi \in S \rangle$  by auto
    define  $n$  where  $n \equiv \text{LEAST } n. (\text{deriv } ^n) f \xi \neq 0$ 
    have  $n\_ne$ :  $(\text{deriv } ^n) f \xi \neq 0$ 
      by (rule def_LeastI [OF  $n\_def$ ]) (rule  $n0$ )
    then have  $0 < n$  using  $\langle f \xi = 0 \rangle$ 
      using funpow_0 by fastforce
    have  $n\_min$ :  $\bigwedge k. k < n \implies (\text{deriv } ^k) f \xi = 0$ 
      using def_Least_le [OF  $n\_def$ ] not_le by blast
    then obtain  $g$   $r1$ 
      where  $g$ :  $0 < r1$   $g$  holomorphic_on  $\text{ball } \xi \ r1$ 
        and  $geq$ :  $\bigwedge w. w \in \text{ball } \xi \ r1 \implies f w = (w - \xi)^n * g w$ 
        and  $g0$ :  $\bigwedge w. w \in \text{ball } \xi \ r1 \implies g w \neq 0$ 
      by (auto intro: holomorphic_factor_order_of_zero [OF holf  $\langle \text{open } S \rangle \langle \xi \in S \rangle$ 
         $\langle n > 0 \rangle$   $n\_ne$ ] simp:  $\langle f \xi = 0 \rangle$ )
    show ?thesis
  proof
    show  $g$  holomorphic_on  $\text{ball } \xi \ (\min r0 \ r1)$ 
      using  $g$  by auto
    show  $\bigwedge w. w \in \text{ball } \xi \ (\min r0 \ r1) \implies f w = (w - \xi)^n * g w$ 
      by (simp add:  $geq$ )
    qed (use  $\langle 0 < n \rangle \langle 0 < r0 \rangle \langle 0 < r1 \rangle \langle \text{ball } \xi \ r0 \subseteq S \rangle g0$  in auto)
  qed

```

```

lemma holomorphic_lower_bound_difference:
  assumes holf:  $f$  holomorphic_on  $S$  and  $S$ : open  $S$  connected  $S$ 
    and  $\xi \in S$  and  $\varphi \in S$ 
    and  $fne$ :  $f \varphi \neq f \xi$ 
  obtains  $k$   $n$   $r$ 
    where  $0 < k$   $0 < r$ 

```



```

      ball ξ r ⊆ S
      ∧ w. w ∈ ball ξ r ⟹ k * norm(w - ξ) ^ n ≤ norm(f w - f ξ)
proof -
  define n where n = (LEAST n. 0 < n ∧ (deriv ~ n) f ξ ≠ 0)
  obtain n0 where 0 < n0 and n0: (deriv ~ n0) f ξ ≠ 0
  using fne holomorphic_fun_eq_const_on_connected [OF holf S] ⟨ξ ∈ S⟩ ⟨φ ∈
S⟩ by blast
  then have 0 < n and n_ne: (deriv ~ n) f ξ ≠ 0
  unfolding n_def by (metis (mono_tags, lifting) LeastI)+
  have n_min: ∧k. [0 < k; k < n] ⟹ (deriv ~ k) f ξ = 0
  unfolding n_def by (blast dest: not_less_Least)
  then obtain g r
  where 0 < r and holg: g holomorphic_on ball ξ r
  and fne: ∧w. w ∈ ball ξ r ⟹ f w - f ξ = (w - ξ) ^ n * g w
  and gnz: ∧w. w ∈ ball ξ r ⟹ g w ≠ 0
  by (auto intro: holomorphic_factor_order_of_zero [OF holf ⟨open S⟩ ⟨ξ ∈
S⟩ ⟨n > 0⟩ n_ne])
  obtain e where e>0 and e: ball ξ e ⊆ S using assms by (blast elim!: openE)
  then have holfb: f holomorphic_on ball ξ e
  using holf holomorphic_on_subset by blast
  define d where d = (min e r) / 2
  have 0 < d using ⟨0 < r⟩ ⟨0 < e⟩ by (simp add: d_def)
  have d < r
  using ⟨0 < r⟩ by (auto simp: d_def)
  then have cbb: cball ξ d ⊆ ball ξ r
  by (auto simp: cball_subset_ball_iff)
  then have g holomorphic_on cball ξ d
  by (rule holomorphic_on_subset [OF holg])
  then have closed (g ' cball ξ d)
  by (simp add: compact_imp_closed compact_continuous_image holomorphic_on_imp_continuous_on)
  moreover have g ' cball ξ d ≠ {}
  using ⟨0 < d⟩ by auto
  ultimately obtain x where x: x ∈ g ' cball ξ d and ∧y. y ∈ g ' cball ξ d ⟹
dist 0 x ≤ dist 0 y
  by (rule distance_attains_inf) blast
  then have leg: ∧w. w ∈ cball ξ d ⟹ norm x ≤ norm (g w)
  by auto
  have ball ξ d ⊆ cball ξ d by auto
  also have ... ⊆ ball ξ e using ⟨0 < d⟩ d_def by auto
  also have ... ⊆ S by (rule e)
  finally have dS: ball ξ d ⊆ S .
  have x ≠ 0 using gnz x ⟨d < r⟩ by auto
  show thesis
proof
  show ∧w. w ∈ ball ξ d ⟹ cmod x * cmod (w - ξ) ^ n ≤ cmod (f w - f ξ)
  using ⟨d < r⟩ leg by (auto simp: fne norm_mult norm_power algebra_simps
mult_right_mono)
qed (use dS ⟨x ≠ 0⟩ ⟨d > 0⟩ in auto)
qed

```

lemma

assumes *hol* f : f *holomorphic_on* $(S - \{\xi\})$ **and** ξ : $\xi \in \text{interior } S$
shows *holomorphic_on_extend_lim*:
 $(\exists g. g \text{ holomorphic_on } S \wedge (\forall z \in S - \{\xi\}. g\ z = f\ z)) \longleftrightarrow$
 $((\lambda z. (z - \xi) * f\ z) \longrightarrow 0) \text{ (at } \xi))$
(is $?P = ?Q$ **)**
and *holomorphic_on_extend_bounded*:
 $(\exists g. g \text{ holomorphic_on } S \wedge (\forall z \in S - \{\xi\}. g\ z = f\ z)) \longleftrightarrow$
 $(\exists B. \text{eventually } (\lambda z. \text{norm}(f\ z) \leq B) \text{ (at } \xi))$
(is $?P = ?R$ **)**
proof –
obtain δ **where** $0 < \delta$ **and** δ : $\text{ball } \xi\ \delta \subseteq S$
using $\xi \text{ mem_interior}$ **by** *blast*
have $?R$ **if** *hol* g : g *holomorphic_on* S **and** *gf*: $\bigwedge z. z \in S - \{\xi\} \implies g\ z = f\ z$
for g
proof –
have \S : $\text{cmod } (f\ x) \leq \text{cmod } (g\ \xi) + 1$ **if** $x \neq \xi$ $\text{dist } x\ \xi < \delta$ $\text{dist } (g\ x)\ (g\ \xi) <$
 1 **for** x
proof –
have $x \in S$
by (*metis* $\delta\ \text{dist_commute mem_ball subsetD that}(2)$)
with *that gf [of x]* **show** $?thesis$
using *norm_triangle_ineq2 [of f x g ξ]* *dist_complex_def* **by** *auto*
qed
then have $*$: $\forall_F z \text{ in at } \xi. \text{dist } (g\ z)\ (g\ \xi) < 1 \longrightarrow \text{cmod } (f\ z) \leq \text{cmod } (g\ \xi)$
 $+ 1$
using $\langle 0 < \delta \rangle$ *eventually_at* **by** *blast*
have *continuous_on* $(\text{interior } S)$ g
by (*meson continuous_on_subset holg holomorphic_on_imp_continuous_on interior_subset*)
then have $\bigwedge x. x \in \text{interior } S \implies (g \longrightarrow g\ x) \text{ (at } x)$
using *continuous_on_interior continuous_within holg holomorphic_on_imp_continuous_on*
by *blast*
then have $(g \longrightarrow g\ \xi) \text{ (at } \xi)$
by (*simp add: ξ*)
then have $\forall_F z \text{ in at } \xi. \text{cmod } (f\ z) \leq \text{cmod } (g\ \xi) + 1$
by (*rule eventually_mp [OF * tendstoD [where e=1]], auto*)
then show $?thesis$
by *blast*
qed
moreover have $?Q$ **if** $\forall_F z \text{ in at } \xi. \text{cmod } (f\ z) \leq B$ **for** B
by (*rule lim_null_mult_right_bounded [OF _ that]*) (*simp add: LIM_zero*)
moreover have $?P$ **if** $(\lambda z. (z - \xi) * f\ z) - \xi \rightarrow 0$
proof –
define h **where** [*abs_def*]: $h\ z = (z - \xi)^2 * f\ z$ **for** z
have $(\lambda y. (y - \xi)^2 * f\ y / (y - \xi)) - \xi \rightarrow 0$
by (*simp add: LIM_cong power2_eq_square that*)
then have $h0$: $(h \text{ has_field_derivative } 0) \text{ (at } \xi)$

```

    by (simp add: h_def has_field_derivative_iff)
  have holh: h holomorphic_on S
proof (simp add: holomorphic_on_def, clarify)
  fix z assume z ∈ S
  show h field_differentiable at z within S
proof (cases z = ξ)
  case True then show ?thesis
    using field_differentiable_at_within field_differentiable_def h0 by blast
  next
  case False
  then have f field_differentiable at z within S
    using holomorphic_onD [OF holh, of z] ‹z ∈ S›
    unfolding field_differentiable_def has_field_derivative_iff
    by (force intro: exI [where x=dist ξ z] elim: Lim_transform_within_set
[unfolded eventually_at])
  then show ?thesis
    by (simp add: h_def power2_eq_square derivative_intros)
qed
qed
define g where [abs_def]: g z = (if z = ξ then deriv h ξ else (h z - h ξ) / (z
- ξ)) for z
have §: ∀ z ∈ S - {ξ}. (g z - g ξ) / (z - ξ) = f z
using h0 by (auto simp: g_def power2_eq_square divide_simps DERIV_imp_deriv
h_def)
have g holomorphic_on S
  unfolding g_def by (rule pole_lemma [OF holh ξ])
then have (λz. if z = ξ then deriv g ξ else (g z - g ξ) / (z - ξ)) holomorphic_on
S
  using ξ pole_lemma by blast
then show ?thesis
  using § by (smt (verit, best) DiffD2 singletonI)
qed
ultimately show ?P = ?Q and ?P = ?R
  by meson+
qed

lemma pole_at_infinity:
  assumes holh: f holomorphic_on UNIV and lim: ((inverse o f) ⟶ l) at_infinity
  obtains a n where ∧z. f z = (∑ i ≤ n. a i * zi)
proof (cases l = 0)
  case False
  show thesis
proof
  show f z = (∑ i ≤ 0. inverse l * zi) for z
    using False tendsto_inverse [OF lim] by (simp add: Liouville_weak [OF
holh])
  qed
next
  case True

```

```

then have [simp]:  $l = 0$  .
show ?thesis
proof (cases  $\exists r. 0 < r \wedge (\forall z \in \text{ball } 0 \ r - \{0\}. f(\text{inverse } z) \neq 0)$ )
  case True
    then obtain  $r$  where  $0 < r$  and  $r$ :  $\bigwedge z. z \in \text{ball } 0 \ r - \{0\} \implies f(\text{inverse } z) \neq 0$ 
      by auto
      have 1:  $\text{inverse} \circ f \circ \text{inverse}$  holomorphic_on ball 0 r - {0}
      by (rule holomorphic_on_compose holomorphic_intros holomorphic_on_subset [OF holf] | force simp: r)+
      have 2:  $0 \in \text{interior } (\text{ball } 0 \ r)$ 
      using  $\langle 0 < r \rangle$  by simp
      obtain  $g$  where holg:  $g$  holomorphic_on ball 0 r
      and geg:  $\bigwedge z. z \in \text{ball } 0 \ r - \{0\} \implies g \ z = (\text{inverse} \circ f \circ \text{inverse}) \ z$ 
      using tendstoD [OF lim [unfolded lim_at_infinity_0] zero_less_one] holomorphic_on_extend_bounded [OF 1 2]
      by (smt (verit, del_insts)  $\langle l = 0 \rangle$  eventually_mono norm_conv_dist)
      have ifi0:  $(\text{inverse} \circ f \circ \text{inverse}) - 0 \rightarrow 0$ 
      using  $\langle l = 0 \rangle$  lim lim_at_infinity_0 by blast
      have g2g0:  $g - 0 \rightarrow g \ 0$ 
      using  $\langle 0 < r \rangle$  centre_in_ball continuous_at continuous_on_eq continuous_at holg
      by (blast intro: holomorphic_on_imp_continuous_on)
      have g2g1:  $g - 0 \rightarrow 0$ 
      proof (rule Lim_transform_within_open [OF ifi0 open_ball])
        show  $\bigwedge x. [x \in \text{ball } 0 \ r; x \neq 0] \implies (\text{inverse} \circ f \circ \text{inverse}) \ x = g \ x$ 
        by (auto simp: geg)
      qed (auto simp:  $\langle 0 < r \rangle$ )
      have [simp]:  $g \ 0 = 0$ 
      by (rule tendsto_unique [OF _ g2g0 g2g1]) simp
      have ball 0 r - {0::complex}  $\neq \{\}$ 
      using  $\langle 0 < r \rangle$  by (metis 2 Diff_iff insert_Diff interior_ball interior_singleton)
      then obtain  $w::\text{complex}$  where  $w \neq 0$  and  $w$ :  $\text{norm } w < r$  by force
      then have  $g \ w \neq 0$  by (simp add: geg r)
      obtain  $B \ n \ e$  where  $0 < B \ 0 < e \ e \leq r$ 
      and leg:  $\bigwedge w. \text{norm } w < e \implies B * \text{cmod } w ^ n \leq \text{cmod } (g \ w)$ 
      proof (rule holomorphic_lower_bound_difference [OF holg open_ball connected_ball])
        show  $g \ w \neq g \ 0$ 
        by (simp add:  $\langle g \ w \neq 0 \rangle$ )
        show  $w \in \text{ball } 0 \ r$ 
        using mem_ball_0 w by blast
      qed (use  $\langle 0 < r \rangle$  in  $\langle \text{auto simp: ball_subset_ball_iff} \rangle$ )
      have  $\S$ :  $\text{cmod } (f \ z) \leq \text{cmod } z ^ n / B$  if  $2/e \leq \text{cmod } z$  for  $z$ 
      proof -
        have ize:  $\text{inverse } z \in \text{ball } 0 \ e - \{0\}$  using that  $\langle 0 < e \rangle$ 
        by (auto simp: norm_divide field_split_simps algebra_simps)
        then have [simp]:  $z \neq 0$  and izr:  $\text{inverse } z \in \text{ball } 0 \ r - \{0\}$  using  $\langle e \leq r \rangle$ 

```

```

    by auto
  then have [simp]:  $f z \neq 0$ 
    using  $r$  [of inverse  $z$ ] by simp
  have [simp]:  $f z = \text{inverse } (g (\text{inverse } z))$ 
    using  $\text{izr\_geq}$  [of inverse  $z$ ] by simp
  show ?thesis using  $\text{ize\_leg}$  [of inverse  $z$ ]  $\langle 0 < B \rangle$   $\langle 0 < e \rangle$ 
    by (simp add: field_split_simps norm_divide algebra_simps)
qed
show thesis
proof
  show  $f z = (\sum_{i \leq n}. (\text{deriv } \wedge^i) f 0 / \text{fact } i * z^i)$  for  $z$ 
    using § by (rule_tac  $A = 2/e$  and  $B = 1/B$  in Liouville_polynomial
[OF holf], simp)
  qed
next
case False
  then have  $\text{fi0}: \bigwedge r. r > 0 \implies \exists z \in \text{ball } 0 r - \{0\}. f (\text{inverse } z) = 0$ 
    by simp
  have  $\text{fz0}: f z = 0$  if  $0 < r$  and  $\text{lt1}: \bigwedge x. x \neq 0 \implies \text{cmod } x < r \implies \text{inverse}$ 
    ( $\text{cmod } (f (\text{inverse } x)) < 1$ )
    for  $z r$ 
  proof -
    have  $\text{f0}: (f \longrightarrow 0)$  at_infinity
    proof -
      have  $\text{DIM\_complex}[\text{intro}]: 2 \leq \text{DIM}(\text{complex})$  — should not be necessary!
      by simp
      have  $f (\text{inverse } x) \neq 0 \implies x \neq 0 \implies \text{cmod } x < r \implies 1 < \text{cmod } (f (\text{inverse}$ 
 $x))$  for  $x$ 
        using  $\text{lt1}$  [of  $x$ ] by (auto simp: field_simps)
      then have **:  $\text{cmod } (f (\text{inverse } x)) \leq 1 \implies x \neq 0 \implies \text{cmod } x < r \implies f$ 
        ( $\text{inverse } x$ ) = 0 for  $x$ 
        by force
      then have *:  $(f \circ \text{inverse}) ' (\text{ball } 0 r - \{0\}) \subseteq \{0\} \cup - \text{ball } 0 1$ 
        by force
      have continuous_on (inverse ' ( $\text{ball } 0 r - \{0\}$ ))  $f$ 
        using continuous_on_subset holf holomorphic_on_imp_continuous_on
by blast
      then have connected (( $f \circ \text{inverse}$ ) ' ( $\text{ball } 0 r - \{0\}$ ))
        using connected_punctured_ball
      by (intro connected_continuous_image continuous_intros; force)
      then have  $\{0\} \cap (f \circ \text{inverse}) ' (\text{ball } 0 r - \{0\}) = \{0\} \vee - \text{ball } 0 1 \cap (f \circ$ 
 $\text{inverse}) ' (\text{ball } 0 r - \{0\}) = \{0\}$ 
        by (rule connected_closedD) (use * in auto)
      then have  $f (\text{inverse } w) = 0$  if  $w \neq 0$   $\text{cmod } w < r$  for  $w$ 
        using ** [of  $w$ ]  $\text{fi0}$   $\langle 0 < r \rangle$  that by force
      then show ?thesis
        unfolding lim_at_infinity_0
        using eventually_at  $\langle r > 0 \rangle$  by (force simp: intro: tendsto_eventually)
    qed

```

```

    obtain w where w ∈ ball 0 r - {0} and f (inverse w) = 0
    using False ⟨0 < r⟩ by blast
  then show ?thesis
    by (auto simp: f0 Liouville_weak [OF holf, of 0])
qed
show thesis
proof
  show  $\bigwedge z. f z = (\sum_{i \leq 0}. 0 * z^i)$ 
    using lim
    apply (simp add: lim_at_infinity_0 Lim_at dist_norm norm_inverse)
    using f0 zero_less_one by blast
qed
qed
qed

```

5.5 Entire proper functions are precisely the non-trivial polynomials

```

lemma proper_map_polyfun:
  fixes c :: nat ⇒ 'a::{real_normed_div_algebra,heine_borel}
  assumes closed S and compact K and c: c i ≠ 0 1 ≤ i i ≤ n
  shows compact (S ∩ {z. (∑_{i ≤ n}. c i * z^i) ∈ K})
proof -
  obtain B where B > 0 and B:  $\bigwedge x. x \in K \implies \text{norm } x \leq B$ 
  by (metis compact_imp_bounded ⟨compact K⟩ bounded_pos)
  have *: norm x ≤ b
    if  $\bigwedge x. b \leq \text{norm } x \implies B + 1 \leq \text{norm } (\sum_{i \leq n}. c i * x^i)$ 
     $(\sum_{i \leq n}. c i * x^i) \in K$  for b x
  proof -
    have norm (∑_{i ≤ n}. c i * x^i) ≤ B
      using B that by blast
    moreover have  $\neg B + 1 \leq B$ 
      by simp
    ultimately show norm x ≤ b
      using that by (metis (no_types) less_eq_real_def not_less order_trans)
  qed
  have bounded {z. (∑_{i ≤ n}. c i * z^i) ∈ K}
    using Limits.polyfun_extremal [where c=c and B=B+1, OF c]
    by (auto simp: bounded_pos eventually_at_infinity_pos *)
  moreover have closed ((λz. (∑_{i ≤ n}. c i * z^i)) - `K)
    using ⟨compact K⟩ compact_eq_bounded_closed
    by (intro allI continuous_closed_vimage continuous_intros; force)
  ultimately show ?thesis
    using closed_Int_compact [OF ⟨closed S⟩] compact_eq_bounded_closed
    by (auto simp add: vimage_def)
qed

```

```

lemma proper_map_polyfun_univ:
  fixes c :: nat ⇒ 'a::{real_normed_div_algebra,heine_borel}

```

```

assumes compact  $K$   $c\ i \neq 0$   $1 \leq i \leq n$ 
shows compact  $\{z. (\sum_{i \leq n}. c\ i * z^i) \in K\}$ 
using proper_map_polyfun [of UNIV  $K\ c\ i\ n$ ] assms by simp

lemma proper_map_polyfun_eq:
assumes  $f$  holomorphic_on UNIV
shows  $(\forall k. \text{compact } k \longrightarrow \text{compact } \{z. f\ z \in k\}) \longleftrightarrow$ 
 $(\exists c\ n. 0 < n \wedge (c\ n \neq 0) \wedge f = (\lambda z. \sum_{i \leq n}. c\ i * z^i))$ 
(is ?lhs = ?rhs)

proof
assume compf [rule_format]: ?lhs
have 2:  $\exists k. 0 < k \wedge a\ k \neq 0 \wedge f = (\lambda z. \sum_{i \leq k}. a\ i * z^i)$ 
if  $\bigwedge z. f\ z = (\sum_{i \leq n}. a\ i * z^i)$  for  $a\ n$ 
proof (cases  $\forall i \leq n. 0 < i \longrightarrow a\ i = 0$ )
case True
then have [simp]:  $\bigwedge z. f\ z = a\ 0$ 
by (simp add: that sum.atMost_shift)
have False using compf [of  $\{a\ 0\}$ ] by simp
then show ?thesis ..
next
case False
then obtain  $k$  where  $k: 0 < k \leq n \wedge a\ k \neq 0$  by force
define  $m$  where  $m = (\text{GREATEST } k. k \leq n \wedge a\ k \neq 0)$ 
have  $m: m \leq n \wedge a\ m \neq 0$ 
unfolding m_def
using GreatestI_nat [where  $b = n$ ]  $k$  by (metis (mono_tags, lifting))
have [simp]:  $a\ i = 0$  if  $m < i \leq n$  for  $i$ 
using Greatest_le_nat [where  $b = n$  and  $P = \lambda k. k \leq n \wedge a\ k \neq 0$ ]
using m_def not_le that by auto
have  $k \leq m$ 
unfolding m_def
using Greatest_le_nat [where  $b = n$ ]  $k$  by (metis (mono_tags, lifting))
with  $k\ m$  show ?thesis
by (rule_tac  $x=m$  in exI) (auto simp: that comm_monoid_add_class.sum.mono_neutral_right)
qed
have *:  $((\text{inverse} \circ f) \longrightarrow 0)$  at_infinity
proof (rule Lim_at_infinityI)
fix  $e::\text{real}$  assume  $0 < e$ 
with compf [of cball 0 (inverse e)]
show  $\exists B. \forall x. B \leq \text{cmod } x \longrightarrow \text{dist } ((\text{inverse} \circ f)\ x)\ 0 \leq e$ 
apply (clarsimp simp: compact_eq_bounded_closed norm_divide divide_simps
mult.commute elim!: bounded_normE_less)
by (meson linorder_not_le nle_le)
qed
then obtain  $a\ n$  where  $\bigwedge z. f\ z = (\sum_{i \leq n}. a\ i * z^i)$ 
using assms pole_at_infinity by blast
with * 2 show ?rhs by blast
next
assume ?rhs

```

```

then obtain c n where  $0 < n$   $c n \neq 0$   $f = (\lambda z. \sum_{i \leq n}. c i * z ^ i)$  by blast
then have compact  $\{z. f z \in k\}$  if compact k for k
  by (auto intro: proper_map_polyfun_univ [OF that])
then show ?lhs by blast
qed

```

5.6 Relating invertibility and nonvanishing of derivative

```

lemma has_complex_derivative_locally_injective:
  assumes holf: f holomorphic_on S
    and S:  $\xi \in S$  open S
    and dnz: deriv f  $\xi \neq 0$ 
  obtains r where  $r > 0$  ball  $\xi$  r  $\subseteq S$  inj_on f (ball  $\xi$  r)
proof -
  have *:  $\exists d > 0. \forall x. \text{dist } \xi \ x < d \longrightarrow \text{onorm } (\lambda v. \text{deriv } f \ x * v - \text{deriv } f \ \xi * v) < e$ 
    if  $e > 0$  for e
  proof -
    have contdf: continuous_on S (deriv f)
    by (simp add: holf holomorphic_deriv holomorphic_on_imp_continuous_on ⟨open S⟩)
    obtain  $\delta$  where  $\delta > 0$  and  $\delta: \bigwedge x. [x \in S; \text{dist } x \ \xi \leq \delta] \Longrightarrow \text{cmod } (\text{deriv } f \ x - \text{deriv } f \ \xi) \leq e/2$ 
    using continuous_onE [OF contdf ⟨ $\xi \in S$ ⟩, of  $e/2$ ] ⟨ $0 < e$ ⟩
    by (metis dist_complex_def half_gt_zero less_imp_le)
    have  $\S: \bigwedge \zeta z. [\zeta \in S; \text{dist } \xi \ \zeta < \delta] \Longrightarrow \text{cmod } (\text{deriv } f \ \zeta - \text{deriv } f \ \xi) * \text{cmod } z \leq e/2 * \text{cmod } z$ 
    by (intro mult_right_mono [OF  $\delta$ ]) (auto simp: dist_commute)
    obtain  $\varepsilon$  where  $\varepsilon > 0$  ball  $\xi$   $\varepsilon \subseteq S$ 
    by (metis openE [OF ⟨open S⟩ ⟨ $\xi \in S$ ⟩])
    with ⟨ $\delta > 0$ ⟩ have  $\exists \delta > 0. \forall x. \text{dist } \xi \ x < \delta \longrightarrow \text{onorm } (\lambda v. \text{deriv } f \ x * v - \text{deriv } f \ \xi * v) \leq e/2$ 
    using  $\S$ 
    apply (rule_tac  $x = \min \delta \ \varepsilon$  in exI)
    apply (intro conjI allI impI Operator_Norm.onorm_le)
    apply (force simp: mult_right_mono norm_mult [symmetric] left_diff_distrib  $\delta$ )
    done
    with ⟨ $e > 0$ ⟩ show ?thesis by force
  qed
  have inj ((* ) (deriv f  $\xi$ ))
  using dnz by simp
  then obtain g' where g': linear g' g'  $\circ$  ((* ) (deriv f  $\xi$ )) = id
  using linear_injective_left_inverse [of ((* ) (deriv f  $\xi$ ))] by auto

  have fder:  $\bigwedge x. x \in S \Longrightarrow (f \text{ has\_derivative } (* ) (\text{deriv } f \ x)) \text{ (at } x)$ 
  using ⟨open S⟩ has_field_derivative_imp_has_derivative holf holomorphic_derivI
  by blast
  show ?thesis
  apply (rule has_derivative_locally_injective [OF S, where f=f and f' =  $\lambda z$ 

```



```

h. deriv f z * h and g' = g']
  using g' * by (simp_all add: fder linear_conv_bounded_linear that)
qed

```

```

lemma has_complex_derivative_locally_invertible:
  assumes holf: f holomorphic_on S
    and S:  $\xi \in S$  open S
    and dnz: deriv f  $\xi \neq 0$ 
  obtains r where  $r > 0$  ball  $\xi$  r  $\subseteq S$  open (f ' (ball  $\xi$  r)) inj_on f (ball  $\xi$  r)
proof -
  obtain r where  $r > 0$  ball  $\xi$  r  $\subseteq S$  inj_on f (ball  $\xi$  r)
    by (blast intro: that has_complex_derivative_locally_injective [OF assms])
  then have  $\xi: \xi \in \text{ball } \xi \text{ } r$  by simp
  then have nc:  $\neg f \text{ constant\_on ball } \xi \text{ } r$ 
    using <inj_on f (ball  $\xi$  r)> injective_not_constant by fastforce
  have holf': f holomorphic_on ball  $\xi$  r
    using <ball  $\xi$  r  $\subseteq S$ > holf holomorphic_on_subset by blast
  have open (f ' ball  $\xi$  r)
    by (simp add: <inj_on f (ball  $\xi$  r)> holf' open_mapping_thm3)
  then show ?thesis
    using <0 < r> <ball  $\xi$  r  $\subseteq S$ > <inj_on f (ball  $\xi$  r)> that by blast
qed

```

```

lemma holomorphic_injective_imp_regular:
  assumes holf: f holomorphic_on S
    and open S and injf: inj_on f S
    and  $\xi \in S$ 
  shows deriv f  $\xi \neq 0$ 
proof -
  obtain r where  $r > 0$  and r: ball  $\xi$  r  $\subseteq S$  using assms by (blast elim!: openE)
  have holf': f holomorphic_on ball  $\xi$  r
    using <ball  $\xi$  r  $\subseteq S$ > holf holomorphic_on_subset by blast
  show ?thesis
proof (cases  $\forall n > 0. (\text{deriv } \sim n) f \xi = 0$ )
  case True
    have fcon: f w = f  $\xi$  if  $w \in \text{ball } \xi \text{ } r$  for w
      by (meson open_ball True <0 < r> centre_in_ball connected_ball holf'
        holomorphic_fun_eq_const_on_connected that)
    have False
      using fcon [of  $\xi + r/2$ ] <0 < r> r injf unfolding inj_on_def
    by (metis < $\xi \in S$ > contra_subsetD dist_commute fcon mem_ball perfect_choose_dist)
    then show ?thesis ..
  next
  case False
    then obtain n0 where n0:  $n0 > 0 \wedge (\text{deriv } \sim n0) f \xi \neq 0$  by blast
    define n where [abs_def]:  $n = (\text{LEAST } n. n > 0 \wedge (\text{deriv } \sim n) f \xi \neq 0)$ 
    have n_ne:  $n > 0$  (deriv  $\sim n$ ) f  $\xi \neq 0$ 
      using def_LeastI [OF n_def n0] by auto
    have n_min:  $\bigwedge k. 0 < k \implies k < n \implies (\text{deriv } \sim k) f \xi = 0$ 

```

```

    using def_Least_le [OF n_def] not_le by auto
  obtain g δ where 0 < δ
    and holg: g holomorphic_on ball ξ δ
    and fd:  $\bigwedge w. w \in \text{ball } \xi \ \delta \implies f \ w - f \ \xi = ((w - \xi) * g \ w) ^ \wedge n$ 
    and gnz:  $\bigwedge w. w \in \text{ball } \xi \ \delta \implies g \ w \neq 0$ 
    by (blast intro: n_min holomorphic_factor_order_of_zero_strong [OF holg
    ‹open S› ‹ξ ∈ S› n_ne])
  show ?thesis
  proof (cases n=1)
    case True
      with n_ne show ?thesis by auto
    next
      case False
        have g holomorphic_on ball ξ (min r δ)
          using holg by (simp add: holomorphic_on_subset subset_ball)
        then have holgw:  $(\lambda w. (w - \xi) * g \ w)$  holomorphic_on ball ξ (min r δ)
          by (intro holomorphic_intros)
        have gd:  $\bigwedge w. \text{dist } \xi \ w < \delta \implies (g \text{ has\_field\_derivative } \text{deriv } g \ w) \ (at \ w)$ 
          using holg
          by (simp add: DERIV_deriv_iff_field_differentiable holomorphic_on_def
          at_within_open_NO_MATCH)
        have *:  $\bigwedge w. w \in \text{ball } \xi \ (\min r \ \delta) \implies ((\lambda w. (w - \xi) * g \ w) \text{ has\_field\_derivative } ((w - \xi) * \text{deriv } g \ w + g \ w))$ 
          (at w)
          by (rule gd derivative_eq_intros | simp)+
        have [simp]:  $\text{deriv } (\lambda w. (w - \xi) * g \ w) \ \xi \neq 0$ 
          using * [of ξ] ‹0 < δ› ‹0 < r› by (simp add: DERIV_imp_deriv gnz)
        obtain T where ξ ∈ T open T and Tsb:  $T \subseteq \text{ball } \xi \ (\min r \ \delta)$  and oimT:
          open  $((\lambda w. (w - \xi) * g \ w) ^ \wedge T)$ 
          using ‹0 < r› ‹0 < δ› has_complex_derivative_locally_invertible [OF
          holgw, of ξ]
          by force
        define U where  $U = (\lambda w. (w - \xi) * g \ w) ^ \wedge T$ 
        have open U by (metis oimT U_def)
        moreover have 0 ∈ U
          using ‹ξ ∈ T› by (auto simp: U_def intro: image_eqI [where x = ξ])
        ultimately obtain ε where ε>0 and ε:  $cball \ 0 \ \varepsilon \subseteq U$ 
          using ‹open U› open_contains_cball by blast
        then have ε *  $\exp(2 * \text{of\_real } \pi * i * (0/n)) \in cball \ 0 \ \varepsilon$ 
          ε *  $\exp(2 * \text{of\_real } \pi * i * (1/n)) \in cball \ 0 \ \varepsilon$ 
          by (auto simp: norm_mult)
        with ε have ε *  $\exp(2 * \text{of\_real } \pi * i * (0/n)) \in U$ 
          ε *  $\exp(2 * \text{of\_real } \pi * i * (1/n)) \in U$  by blast+
        then obtain y0 y1 where y0 ∈ T and y0:  $(y0 - \xi) * g \ y0 = \varepsilon * \exp(2 * \text{of\_real } \pi * i * (0/n))$ 
          and y1 ∈ T and y1:  $(y1 - \xi) * g \ y1 = \varepsilon * \exp(2 * \text{of\_real } \pi * i * (1/n))$ 
          by (auto simp: U_def)

```

```

    then have  $y0 \in \text{ball } \xi \ \delta \ y1 \in \text{ball } \xi \ \delta$  using Tsb by auto
    then have  $f \ y0 - f \ \xi = ((y0 - \xi) * g \ y0) \wedge^n f \ y1 - f \ \xi = ((y1 - \xi) * g \ y1) \wedge^n$ 
    using fd by blast+
    moreover have  $y0 \neq y1$ 
    using  $y0 \ y1 \ \langle \varepsilon > 0 \rangle \ \text{complex\_root\_unity\_eq\_1} \ [\text{of } n \ 1] \ \langle n > 0 \rangle \ \text{False}$  by
auto
    moreover have  $T \subseteq S$ 
    by (meson Tsb min.cobounded1 order_trans r subset_ball)
    ultimately have False
    using inj_onD [OF injf, of y0 y1]  $\langle y0 \in T \rangle \ \langle y1 \in T \rangle$ 
    using complex_root_unity [of n 1]
    by (auto simp: y0 y1 power_mult_distrib diff_eq_eq n_ne)
    then show ?thesis ..
  qed
qed
qed

```

5.6.1 Hence a nice clean inverse function theorem

lemma *has_field_derivative_inverse_strong*:

```

  fixes  $f :: 'a :: \{\text{euclidean\_space}, \text{real\_normed\_field}\} \Rightarrow 'a$ 
  shows  $\llbracket \text{DERIV } f \ x :> f'; f' \neq 0; \text{open } S; x \in S; \text{continuous\_on } S \ f; \wedge z. z \in S \implies g \ (f \ z) = z \rrbracket$ 
     $\implies \text{DERIV } g \ (f \ x) :> \text{inverse } (f')$ 
  unfolding has_field_derivative_def
  by (rule has_derivative_inverse_strong [of S x f g]) auto

```

lemma *has_field_derivative_inverse_strong_x*:

```

  fixes  $f :: 'a :: \{\text{euclidean\_space}, \text{real\_normed\_field}\} \Rightarrow 'a$ 
  shows  $\llbracket \text{DERIV } f \ (g \ y) :> f'; f' \neq 0; \text{open } S; \text{continuous\_on } S \ f; g \ y \in S; f(g \ y) = y; \wedge z. z \in S \implies g \ (f \ z) = z \rrbracket$ 
     $\implies \text{DERIV } g \ y :> \text{inverse } (f')$ 
  unfolding has_field_derivative_def
  by (rule has_derivative_inverse_strong_x [of S g y f]) auto

```

proposition *holomorphic_has_inverse*:

```

  assumes holf:  $f \text{ holomorphic\_on } S$ 
  and open S and injf:  $\text{inj\_on } f \ S$ 
  obtains  $g$  where  $g \text{ holomorphic\_on } (f \ ' S)$ 
     $\wedge z. z \in S \implies \text{deriv } f \ z * \text{deriv } g \ (f \ z) = 1$ 
     $\wedge z. z \in S \implies g(f \ z) = z$ 

```

proof –

```

  have ofs:  $\text{open } (f \ ' S)$ 
  by (rule open_mapping_thm3 [OF assms])
  have contf:  $\text{continuous\_on } S \ f$ 
  by (simp add: holf holomorphic_on_imp_continuous_on)
  have *:  $(\text{the\_inv\_into } S \ f \text{ has\_field\_derivative\_inverse } (\text{deriv } f \ z)) \ (\text{at } (f \ z))$  if

```

```

 $z \in S$  for  $z$ 
proof -
  have 1:  $(f \text{ has\_field\_derivative } \text{deriv } f \ z) \ (at \ z)$ 
    using DERIV_deriv_iff_field_differentiable  $\langle z \in S \rangle \langle \text{open } S \rangle$  holf holomorphic_on_imp_differentiable_at
    by blast
  have 2:  $\text{deriv } f \ z \neq 0$ 
    using  $\langle z \in S \rangle \langle \text{open } S \rangle$  holf holomorphic_injective_imp_regular injf by blast
  show ?thesis
proof (rule has_field_derivative_inverse_strong [OF 1 2  $\langle \text{open } S \rangle \langle z \in S \rangle$ ])
  show continuous_on  $S \ f$ 
    by (simp add: holf holomorphic_on_imp_continuous_on)
  show  $\bigwedge z. z \in S \implies \text{the\_inv\_into } S \ f \ (f \ z) = z$ 
    by (simp add: injf the_inv_into_f_f)
qed
qed
show ?thesis
proof
  show  $\text{the\_inv\_into } S \ f \text{ holomorphic\_on } f^{-1} S$ 
    by (simp add: holomorphic_on_open ofs) (blast intro: *)
next
  fix  $z$  assume  $z \in S$ 
  have  $\text{deriv } f \ z \neq 0$ 
    using  $\langle z \in S \rangle \langle \text{open } S \rangle$  holf holomorphic_injective_imp_regular injf by blast
  then show  $\text{deriv } f \ z * \text{deriv } (\text{the\_inv\_into } S \ f) \ (f \ z) = 1$ 
    using * [OF  $\langle z \in S \rangle$ ] by (simp add: DERIV_imp_deriv)
next
  fix  $z$  assume  $z \in S$ 
  show  $\text{the\_inv\_into } S \ f \ (f \ z) = z$ 
    by (simp add:  $\langle z \in S \rangle$  injf the_inv_into_f_f)
qed
qed

```

5.6.2 Holomorphism of covering maps and lifts.

lemma covering_space_lift_is_holomorphic:

```

assumes cov: covering_space  $C \ p \ S$ 
  and  $C$ :  $\text{open } C \ p \text{ holomorphic\_on } C$ 
  and holf:  $f \text{ holomorphic\_on } U$  and fim:  $f \in U \rightarrow S$  and gim:  $g \in U \rightarrow C$ 
  and contg: continuous_on  $U \ g$  and pgf:  $\bigwedge x. x \in U \implies p(g \ x) = f \ x$ 
shows  $g \text{ holomorphic\_on } U$ 
unfolding holomorphic_on_def
proof (intro strip)
  fix  $z$ 
  assume  $z \in U$ 
  with fim have  $f \ z \in S$  by blast
  then obtain  $T \ \mathcal{V}$  where  $f \ z \in T$  and opeT:  $\text{openin } (\text{top\_of\_set } S) \ T$ 
    and UV:  $\bigcup \mathcal{V} = C \cap p^{-1} T$ 

```

```

    and  $\bigwedge W. W \in \mathcal{V} \implies \text{openin } (\text{top\_of\_set } C) \ W$ 
    and  $\text{disV}$ : pairwise disjoint  $\mathcal{V}$  and  $\text{homeV}$ :  $\bigwedge W. W \in \mathcal{V} \implies \exists q. \text{homeomor-}$ 
    phism  $W \ T \ p \ q$ 
    using cov fim unfolding covering_space_def by meson
    then have  $\bigwedge W. W \in \mathcal{V} \implies \text{open } W \wedge W \subseteq C$ 
    by (metis  $\langle \text{open } C \rangle \text{ inf\_le1 open\_Int openin\_open}$ )
    then obtain  $V$  where  $V \in \mathcal{V} \ g \ z \in V \ \text{open } V \ V \subseteq C$ 
    by (metis IntI UnionE image_subset_iff vimageI UV  $\langle f \ z \in T \rangle \langle z \in U \rangle \text{ gim}$ 
    pg_f image_subset_iff_funcset)
    have holp:  $p$  holomorphic_on  $V$ 
    using  $\langle V \subseteq C \rangle \langle p \text{ holomorphic\_on } C \rangle \text{ holomorphic\_on\_subset}$  by blast
    moreover have injp:  $\text{inj\_on } p \ V$ 
    by (metis  $\langle V \in \mathcal{V} \rangle \text{ homeV homeomorphism\_def inj\_on\_inverseI}$ )
    ultimately
    obtain  $p'$  where holp':  $p'$  holomorphic_on  $(p^{-1} V)$  and pp':  $\bigwedge z. z \in V \implies p'(p$ 
     $z) = z$ 
    using  $\langle \text{open } V \rangle \text{ holomorphic\_has\_inverse}$  by metis
    have  $z \in U \cap g^{-1} V$ 
    using  $\langle g \ z \in V \rangle \langle z \in U \rangle$  by blast
    moreover have  $\text{openin } (\text{top\_of\_set } U) \ (U \cap g^{-1} V)$ 
    using continuous_openin_preimage [OF contg gim]
    by (meson  $\langle \text{open } V \rangle \text{ contg continuous\_openin\_preimage\_eq}$ )
    ultimately obtain  $\varepsilon$  where  $\varepsilon > 0$  and  $e$ :  $\text{ball } z \ \varepsilon \cap U \subseteq g^{-1} V$ 
    by (force simp: openin_contains_ball)
    show  $g$  field_differentiable at  $z$  within  $U$ 
    proof (rule field_differentiable_transform_within)
      show  $(0::\text{real}) < \varepsilon$ 
      by (simp add:  $\langle 0 < \varepsilon \rangle$ )
      show  $z \in U$ 
      by (simp add:  $\langle z \in U \rangle$ )
      show  $(p' \circ f) \ x' = g \ x'$  if  $x' \in U \ \text{dist } x' \ z < \varepsilon$  for  $x'$ 
      using that
      by (metis Int_iff comp_apply dist_commute e mem_ball pg_f pp' subsetD
      vimage_eq)
      have  $\text{open } (p^{-1} V)$ 
      using  $\langle \text{open } V \rangle \text{ holp injp open\_mapping\_thm3}$  by force
      moreover have  $f \ z \in p^{-1} V$ 
      by (metis  $\langle z \in U \rangle \text{ image\_iff pg\_f } \langle g \ z \in V \rangle$ )
      ultimately have  $p'$  field_differentiable at  $(f \ z)$ 
      using holomorphic_on_imp_differentiable_at holp' by blast
      moreover have  $f$  field_differentiable at  $z$  within  $U$ 
      by (metis (no_types)  $\langle z \in U \rangle \text{ holf holomorphic\_on\_def}$ )
      ultimately show  $(p' \circ f)$  field_differentiable at  $z$  within  $U$ 
      by (metis (no_types) field_differentiable_at_within field_differentiable_compose_within)
    qed
  qed

```

lemma *covering_space_lift_holomorphic*:
 assumes *cov*: *covering_space* $C \ p \ S$

```

    and C: open C p holomorphic_on C
    and f: f holomorphic_on U f ∈ U → S
    and U: simply_connected U locally_path_connected U
  obtains g where g holomorphic_on U g ∈ U → C ∧ y. y ∈ U ⇒ p(g y) = f y
proof -
  obtain g where continuous_on U g g ∈ U → C ∧ y. y ∈ U ⇒ p(g y) = f y
  using covering_space_lift [OF cov U] f holomorphic_on_imp_continuous_on
  by blast
  then show ?thesis
  by (metis C cov covering_space_lift_is_holomorphic f image_subset_iff_funcset
  that)
qed

```

5.7 The Schwarz Lemma

lemma Schwarz1:

```

  assumes holf: f holomorphic_on S
    and conf: continuous_on (closure S) f
    and S: open S connected S
    and boS: bounded S
    and S ≠ {}
  obtains w where w ∈ frontier S
    ∧ z. z ∈ closure S ⇒ norm (f z) ≤ norm (f w)
proof -
  have connf: continuous_on (closure S) (norm o f)
  using conf continuous_on_compose continuous_on_norm_id by blast
  have coc: compact (closure S)
  by (simp add: ⟨bounded S⟩ bounded_closure compact_eq_bounded_closed)
  then obtain x where x: x ∈ closure S and xmax: ∧ z. z ∈ closure S ⇒ norm(f
  z) ≤ norm(f x)
  using continuous_attains_sup [OF _ _ connf] ⟨S ≠ {}⟩ by auto
  then show ?thesis
  proof (cases x ∈ frontier S)
    case True
    then show ?thesis using that xmax by blast
  next
    case False
    then have x ∈ S
    using ⟨open S⟩ frontier_def interior_eq x by auto
    then have f_constant_on S
    proof (rule maximum_modulus_principle [OF holf S ⟨open S⟩ order_refl])
      show ∧ z. z ∈ S ⇒ cmod (f z) ≤ cmod (f x)
      using closure_subset by (blast intro: xmax)
    qed
    then have f_constant_on (closure S)
    by (rule constant_on_closureI [OF _ conf])
    then obtain c where c: ∧ x. x ∈ closure S ⇒ f x = c
    by (meson constant_on_def)
    obtain w where w ∈ frontier S

```

```

    by (metis coc all_not_in_conv assms(6) closure_UNIV frontier_eq_empty
not_compact_UNIV)
    then show ?thesis
    by (simp add: c frontier_def that)
qed
qed

```

lemma Schwarz2:

```

[[f holomorphic_on ball 0 r;
  0 < s; ball w s ⊆ ball 0 r;
  ∧z. norm (w-z) < s ⇒ norm(f z) ≤ norm(f w)]
⇒ f constant_on ball 0 r]
by (rule maximum_modulus_principle [where U = ball w s and ξ = w]) (simp_all
add: dist_norm)

```

lemma Schwarz3:

```

assumes holf: f holomorphic_on (ball 0 r) and [simp]: f 0 = 0
obtains h where h holomorphic_on (ball 0 r) and ∧z. norm z < r ⇒ f z =
z * (h z) and deriv f 0 = h 0
proof -
  define h where h z = (if z = 0 then deriv f 0 else f z / z) for z
  have d0: deriv f 0 = h 0
  by (simp add: h_def)
  moreover have h holomorphic_on (ball 0 r)
  by (rule pole_theorem_open_0 [OF holf, of 0]) (auto simp: h_def)
  moreover have norm z < r ⇒ f z = z * h z for z
  by (simp add: h_def)
  ultimately show ?thesis
  using that by blast
qed

```

proposition Schwarz_Lemma:

```

assumes holf: f holomorphic_on (ball 0 1) and [simp]: f 0 = 0
and no: ∧z. norm z < 1 ⇒ norm (f z) < 1
and ξ: norm ξ < 1
shows norm (f ξ) ≤ norm ξ and norm(deriv f 0) ≤ 1
and ((∃z. norm z < 1 ∧ z ≠ 0 ∧ norm(f z) = norm z)
  ∨ norm(deriv f 0) = 1)
⇒ ∃α. (∀z. norm z < 1 → f z = α * z) ∧ norm α = 1
(is ?P ⇒ ?Q)
proof -
  obtain h where holh: h holomorphic_on (ball 0 1)
  and fz_eq: ∧z. norm z < 1 ⇒ f z = z * (h z) and df0: deriv f 0 = h
  0
  by (rule Schwarz3 [OF holf]) auto
  have noh_le: norm (h z) ≤ 1 if z: norm z < 1 for z
  proof -
    have norm (h z) < a if a: 1 < a for a
    proof -

```

```

have max (inverse a) (norm z) < 1
  using z a by (simp_all add: inverse_less_1_iff)
then obtain r where r: max (inverse a) (norm z) < r and r < 1
  using Rats_dense_in_real by blast
then have nzs: norm z < r and ira: inverse r < a
  using z a less_imp_inverse_less by force+
then have 0 < r
  by (meson norm_not_less_zero not_le order.strict_trans2)
have holh': h holomorphic_on ball 0 r
  by (meson holh ⟨r < 1⟩ holomorphic_on_subset less_eq_real_def subset_ball)
have conth': continuous_on (cball 0 r) h
  by (meson ⟨r < 1⟩ dual_order.trans holh holomorphic_on_imp_continuous_on holomorphic_on_subset mem_ball_0 mem_cball_0 not_less subsetI)
obtain w where w: norm w = r and lenw:  $\bigwedge z. \text{norm } z < r \implies \text{norm}(h z) \leq \text{norm}(h w)$ 
  using conth' ⟨0 < r⟩ by (auto simp add: intro: Schwarz1 [OF holh'])
have h w = f w / w using fz_eq ⟨r < 1⟩ nzs w by auto
then have cmod (h z) < inverse r
  by (metis ⟨0 < r⟩ ⟨r < 1⟩ divide_strict_right_mono inverse_eq_divide le_less_trans lenw no_norm_divide nzs w)
then show ?thesis using ira by linarith
qed
then show norm (h z) ≤ 1
  using not_le by blast
qed
show cmod (f ξ) ≤ cmod ξ
proof (cases ξ = 0)
  case True then show ?thesis by auto
next
  case False
  then show ?thesis
    by (simp add: noh_le fz_eq ξ mult_left_le norm_mult)
qed
show no_df0: norm (deriv f 0) ≤ 1
  by (simp add: ⟨ $\bigwedge z. \text{cmod } z < 1 \implies \text{cmod}(h z) \leq 1$ ⟩ df0)
show ?Q if ?P
  using that
proof
  assume  $\exists z. \text{cmod } z < 1 \wedge z \neq 0 \wedge \text{cmod}(f z) = \text{cmod } z$ 
  then obtain γ where γ: cmod γ < 1 γ ≠ 0 cmod (f γ) = cmod γ by blast
  then have [simp]: norm (h γ) = 1
    by (simp add: fz_eq norm_mult)
  have §: ball γ (1 - cmod γ) ⊆ ball 0 1
    by (simp add: ball_subset_ball_iff)
  moreover have  $\bigwedge z. \text{cmod}(\gamma - z) < 1 - \text{cmod } \gamma \implies \text{cmod}(h z) \leq \text{cmod}(h \gamma)$ 
    by (metis ⟨cmod (h γ) = 1⟩ § dist_0_norm dist_complex_def in_mono mem_ball noh_le)

```



```

ultimately obtain c where c:  $\bigwedge z. \text{norm } z < 1 \implies h z = c$ 
  using Schwarz2 [OF holh, of 1 - norm  $\gamma$ , unfolded constant_on_def]  $\gamma$ 
by auto
  then have norm c = 1
    using  $\gamma$  by force
  with c show ?thesis
    using fz_eq by auto
next
  assume [simp]: cmod (deriv f 0) = 1
  then obtain c where c:  $\bigwedge z. \text{norm } z < 1 \implies h z = c$ 
    using Schwarz2 [OF holh zero_less_one, of 0, unfolded constant_on_def]
df0 noh_le
  by auto
  moreover have norm c = 1 using df0 c by auto
  ultimately show ?thesis
    using fz_eq by auto
qed
qed

```

corollary Schwarz_Lemma':

```

assumes holh: f holomorphic_on (ball 0 1) and [simp]: f 0 = 0
  and no:  $\bigwedge z. \text{norm } z < 1 \implies \text{norm } (f z) < 1$ 
shows (( $\forall \xi. \text{norm } \xi < 1 \longrightarrow \text{norm } (f \xi) \leq \text{norm } \xi$ )
   $\wedge \text{norm } (\text{deriv } f 0) \leq 1$ )
   $\wedge ((\exists z. \text{norm } z < 1 \wedge z \neq 0 \wedge \text{norm } (f z) = \text{norm } z)$ 
     $\vee \text{norm } (\text{deriv } f 0) = 1)$ 
     $\longrightarrow (\exists \alpha. (\forall z. \text{norm } z < 1 \longrightarrow f z = \alpha * z) \wedge \text{norm } \alpha = 1))$ 
using Schwarz_Lemma [OF assms]
by (metis (no_types) norm_eq_zero zero_less_one)

```

5.8 The Schwarz reflection principle

lemma hol_pal_lem0:

assumes $d \cdot a \leq k$ $k \leq d \cdot b$

obtains c where

```

  c  $\in$  closed_segment a b  $d \cdot c = k$ 
   $\bigwedge z. z \in \text{closed\_segment } a c \implies d \cdot z \leq k$ 
   $\bigwedge z. z \in \text{closed\_segment } c b \implies k \leq d \cdot z$ 

```

proof –

obtain c where cin: $c \in \text{closed_segment } a b$ and keq: $k = d \cdot c$

using connected_ivt_hyperplane [of closed_segment a b a b d k]

by (auto simp: assms)

have closed_segment a c $\subseteq \{z. d \cdot z \leq k\}$ closed_segment c b $\subseteq \{z. k \leq d \cdot z\}$

unfolding segment_convex_hull using assms keq

by (auto simp: convex_halfspace_le convex_halfspace_ge hull_minimal)

then show ?thesis using cin that by fastforce

qed

lemma hol_pal_lem1:

```

assumes convex  $S$  open  $S$ 
and abc:  $a \in S$   $b \in S$   $c \in S$ 
and  $d \neq 0$  and lek:  $d \cdot a \leq k$   $d \cdot b \leq k$   $d \cdot c \leq k$ 
and holf1:  $f$  holomorphic_on  $\{z. z \in S \wedge d \cdot z < k\}$ 
and contf: continuous_on  $S$   $f$ 
shows contour_integral (linepath  $a$   $b$ )  $f$  +
  contour_integral (linepath  $b$   $c$ )  $f$  +
  contour_integral (linepath  $c$   $a$ )  $f$  = 0
proof -
have interior (convex hull  $\{a, b, c\}$ )  $\subseteq$  interior( $S \cap \{x. d \cdot x \leq k\}$ )
proof (intro interior_mono hull_minimal)
show  $\{a, b, c\} \subseteq S \cap \{x. d \cdot x \leq k\}$ 
by (simp add: abc lek)
show convex ( $S \cap \{x. d \cdot x \leq k\}$ )
by (rule convex_Int [OF convex  $S$  convex_halfspace_le])
qed
also have  $\dots \subseteq \{z \in S. d \cdot z < k\}$ 
by (force simp: interior_open [OF open  $S$ ]  $\langle d \neq 0 \rangle$ )
finally have *: interior (convex hull  $\{a, b, c\}$ )  $\subseteq \{z \in S. d \cdot z < k\}$  .
have continuous_on (convex hull  $\{a, b, c\}$ )  $f$ 
using convex  $S$  contf abc continuous_on_subset subset_hull
by fastforce
moreover have  $f$  holomorphic_on interior (convex hull  $\{a, b, c\}$ )
by (rule holomorphic_on_subset [OF holf1 *])
ultimately show ?thesis
using Cauchy_theorem_triangle_interior has_chain_integral_chain_integral3
by blast
qed

lemma hol_pal_lem2:
assumes  $S$ : convex  $S$  open  $S$ 
and abc:  $a \in S$   $b \in S$   $c \in S$ 
and  $d \neq 0$  and lek:  $d \cdot a \leq k$   $d \cdot b \leq k$ 
and holf1:  $f$  holomorphic_on  $\{z. z \in S \wedge d \cdot z < k\}$ 
and holf2:  $f$  holomorphic_on  $\{z. z \in S \wedge k < d \cdot z\}$ 
and contf: continuous_on  $S$   $f$ 
shows contour_integral (linepath  $a$   $b$ )  $f$  +
  contour_integral (linepath  $b$   $c$ )  $f$  +
  contour_integral (linepath  $c$   $a$ )  $f$  = 0
proof (cases  $d \cdot c \leq k$ )
case True show ?thesis
by (rule hol_pal_lem1 [OF  $S$  abc  $\langle d \neq 0 \rangle$  lek True holf1 contf])
next
case False
then have  $d \cdot c > k$  by force
obtain  $a'$  where  $a'$ :  $a' \in \text{closed\_segment } b \ c$  and  $d \cdot a' = k$ 
and  $ba'$ :  $\bigwedge z. z \in \text{closed\_segment } b \ a' \implies d \cdot z \leq k$ 
and  $a'c$ :  $\bigwedge z. z \in \text{closed\_segment } a' \ c \implies k \leq d \cdot z$ 
using False hol_pal_lem0 [of  $d \ b \ k \ c$ , OF  $\langle d \cdot b \leq k \rangle$ ] by auto

```

```

obtain  $b'$  where  $b': b' \in \text{closed\_segment } a \ c$  and  $d \cdot b' = k$ 
  and  $ab': \bigwedge z. z \in \text{closed\_segment } a \ b' \implies d \cdot z \leq k$ 
  and  $b'c: \bigwedge z. z \in \text{closed\_segment } b' \ c \implies k \leq d \cdot z$ 
  using False hol_pal_lem0 [of d a k c, OF <d · a ≤ k>] by auto
have  $a'b': a' \in S \wedge b' \in S$ 
  using  $a' \ abc \ b' \ \text{convex\_contains\_segment } \langle \text{convex } S \rangle$  by auto
have continuous_on (closed_segment c a) f
  by (meson abc contf continuous_on_subset convex_contains_segment <convex S>)
then have 1: contour_integral (linepath c a) f =
  contour_integral (linepath c b') f + contour_integral (linepath b' a) f
  using  $b' \ \text{closed\_segment\_commute } \text{contour\_integral\_split\_linepath}$  by blast
have continuous_on (closed_segment b c) f
  by (meson abc contf continuous_on_subset convex_contains_segment <convex S>)
then have 2: contour_integral (linepath b c) f =
  contour_integral (linepath b a') f + contour_integral (linepath a' c) f
  by (rule contour_integral_split_linepath [OF _ a'])
have 3: contour_integral (reversepath (linepath b' a')) f =
  - contour_integral (linepath b' a') f
  by (rule contour_integral_reversepath [OF valid_path_linepath])
have fcd_le: f field_differentiable at x
  if  $x \in \text{interior } S \wedge x \in \text{interior } \{x. d \cdot x \leq k\}$  for x
proof -
  have f holomorphic_on  $S \cap \{x. d \cdot x \leq k\}$ 
  by (metis (no_types) Collect_conj_eq Collect_mem_eq holf1)
  then have  $\exists C \ D. x \in \text{interior } C \cap \text{interior } D \wedge f \ \text{holomorphic\_on } \text{interior } C$ 
   $\cap \text{interior } D$ 
  using that
  by (metis Collect_mem_eq Int_Collect <d ≠ 0> interior_halfspace_le interior_open <open S>)
  then show f field_differentiable at x
  by (metis at_within_interior holomorphic_on_def interior_Int interior_interior)
qed
have ab_le:  $\bigwedge x. x \in \text{closed\_segment } a \ b \implies d \cdot x \leq k$ 
proof -
  fix x :: complex
  assume  $x \in \text{closed\_segment } a \ b$ 
  then have  $\bigwedge C. x \in C \vee b \notin C \vee a \notin C \vee \neg \text{convex } C$ 
  by (meson contra_subsetD convex_contains_segment)
  then show  $d \cdot x \leq k$ 
  by (metis lek convex_halfspace_le mem_Collect_eq)
qed
have cs:  $\text{closed\_segment } a' \ b' \subseteq \{x. d \cdot x \leq k\} \wedge \text{closed\_segment } b' \ a \subseteq \{x. d \cdot x \leq k\}$ 
  by (simp add:  $\langle d \cdot a' = k \rangle \langle d \cdot b' = k \rangle \text{closed\_segment\_subset convex\_halfspace\_le lek(1)})
have continuous_on ( $S \cap \{x. d \cdot x \leq k\}$ ) f using contf
  by (simp add: continuous_on_subset)$ 
```

```

then have (f has_contour_integral 0)
  (linepath a b +++ linepath b a' +++ linepath a' b' +++ linepath b' a)
  apply (rule Cauchy_theorem_convex [where K = {}])
  by (simp_all add: path_image_join convex_Int convex_halfspace_le <convex
S> fcd_le ab_le
      closed_segment_subset abc a'b' ba' cs)
then have 4: contour_integral (linepath a b) f +
  contour_integral (linepath b a') f +
  contour_integral (linepath a' b') f +
  contour_integral (linepath b' a) f = 0
  by (rule has_chain_integral_chain_integral4)
have fcd_ge: f field_differentiable at x
  if x ∈ interior S ∧ x ∈ interior {x. k ≤ d · x} for x
proof -
  have f2: f holomorphic_on S ∩ {c. k < d · c}
    by (metis (full_types) Collect_conj_eq Collect_mem_eq holf2)
  have f3: interior S = S
    by (simp add: interior_open <open S>)
  then have x ∈ S ∩ interior {c. k ≤ d · c}
    using that by simp
  then show f field_differentiable at x
    using f3 f2 unfolding holomorphic_on_def
    by (metis (no_types) <d ≠ 0> at_within_interior interior_Int interior_halfspace_ge
interior_interior)
  qed
  have cs: closed_segment c b' ⊆ {x. k ≤ d · x} ∧ closed_segment b' a' ⊆ {x. k
≤ d · x}
    by (simp add: <d · a' = k> b'c closed_segment_subset convex_halfspace_ge)
  have continuous_on (S ∩ {x. k ≤ d · x}) f using contf
    by (simp add: continuous_on_subset)
  then have (f has_contour_integral 0) (linepath a' c +++ linepath c b' +++
linepath b' a')
    apply (rule Cauchy_theorem_convex [where K = {}])
    by (simp_all add: path_image_join convex_Int convex_halfspace_ge <convex
S>
        fcd_ge closed_segment_subset abc a'b' a'c cs)
  then have 5: contour_integral (linepath a' c) f + contour_integral (linepath c
b') f + contour_integral (linepath b' a') f = 0
    by (rule has_chain_integral_chain_integral3)
  show ?thesis
    using 1 2 3 4 5 by (metis add.assoc eq_neg_iff_add_eq_0 reversepath_linepath)
qed

```

lemma hol_pal_lem3:

```

assumes S: convex S open S
  and abc: a ∈ S b ∈ S c ∈ S
  and d ≠ 0 and lek: d · a ≤ k
  and holf1: f holomorphic_on {z. z ∈ S ∧ d · z < k}
  and holf2: f holomorphic_on {z. z ∈ S ∧ k < d · z}

```

```

    and conf: continuous_on S f
  shows contour_integral (linepath a b) f +
        contour_integral (linepath b c) f +
        contour_integral (linepath c a) f = 0
proof (cases d • b ≤ k)
  case True show ?thesis
    by (rule hol_pal_lem2 [OF S abc ⟨d ≠ 0⟩ lek True holf1 holf2 conf])
next
  case False
  show ?thesis
  proof (cases d • c ≤ k)
    case True
    have contour_integral (linepath c a) f +
          contour_integral (linepath a b) f +
          contour_integral (linepath b c) f = 0
      by (rule hol_pal_lem2 [OF S ⟨c ∈ S⟩ ⟨a ∈ S⟩ ⟨b ∈ S⟩ ⟨d ≠ 0⟩ ⟨d • c ≤ k⟩
lek holf1 holf2 conf])
    then show ?thesis
      by (simp add: algebra_simps)
  next
    case False
    have contour_integral (linepath b c) f +
          contour_integral (linepath c a) f +
          contour_integral (linepath a b) f = 0
      using hol_pal_lem2 [OF S ⟨b ∈ S⟩ ⟨c ∈ S⟩ ⟨a ∈ S⟩, of -d -k]
      using ⟨d ≠ 0⟩ ⟨¬ d • b ≤ k⟩ False by (simp_all add: holf1 holf2 conf)
    then show ?thesis
      by (simp add: algebra_simps)
  qed
qed

lemma hol_pal_lem4:
  assumes S: convex S open S
    and abc: a ∈ S b ∈ S c ∈ S and d ≠ 0
    and holf1: f holomorphic_on {z. z ∈ S ∧ d • z < k}
    and holf2: f holomorphic_on {z. z ∈ S ∧ k < d • z}
    and conf: continuous_on S f
  shows contour_integral (linepath a b) f +
        contour_integral (linepath b c) f +
        contour_integral (linepath c a) f = 0
proof (cases d • a ≤ k)
  case True show ?thesis
    by (rule hol_pal_lem3 [OF S abc ⟨d ≠ 0⟩ True holf1 holf2 conf])
next
  case False
  show ?thesis
    using ⟨d ≠ 0⟩ hol_pal_lem3 [OF S abc, of -d -k] False
    by (simp_all add: holf1 holf2 conf)
qed

```

lemma *holomorphic_on_paste_across_line*:

assumes S : *open* S **and** $d \neq 0$
and $holf1$: f *holomorphic_on* $(S \cap \{z. d \cdot z < k\})$
and $holf2$: f *holomorphic_on* $(S \cap \{z. k < d \cdot z\})$
and $contf$: *continuous_on* S f
shows f *holomorphic_on* S
proof –
have *: $\exists t. \text{open } t \wedge p \in t \wedge \text{continuous_on } t \ f \wedge$
 $(\forall a \ b \ c. \text{convex_hull } \{a, b, c\} \subseteq t \longrightarrow$
 $\text{contour_integral } (\text{linepath } a \ b) \ f +$
 $\text{contour_integral } (\text{linepath } b \ c) \ f +$
 $\text{contour_integral } (\text{linepath } c \ a) \ f = 0)$
if $p \in S$ **for** p
proof –
obtain e **where** $e > 0$ **and** e : $\text{ball } p \ e \subseteq S$
using $\langle p \in S \rangle$ *openE* S **by** *blast*
then have *continuous_on* $(\text{ball } p \ e)$ f
using $contf$ *continuous_on_subset* **by** *blast*
moreover
have $\{z. \text{dist } p \ z < e \wedge d \cdot z < k\} \subseteq S \cap \{z. d \cdot z < k\}$
 $\{z. \text{dist } p \ z < e \wedge k < d \cdot z\} \subseteq S \cap \{z. k < d \cdot z\}$
using e **by** *auto*
then have f *holomorphic_on* $\{z. \text{dist } p \ z < e \wedge d \cdot z < k\}$
 f *holomorphic_on* $\{z. \text{dist } p \ z < e \wedge k < d \cdot z\}$
using *holomorphic_on_subset* $holf1$ $holf2$ **by** *presburger+*
ultimately show *?thesis*
apply $(\text{rule_tac } x = \text{ball } p \ e \text{ in } exI)$
using $\langle e > 0 \rangle$ $e \langle d \neq 0 \rangle$ *hol_pal_lem4* [*of ball p e _ _ d _ k*]
by $(\text{force simp: subset_hull})$
qed
show *?thesis*
by $(\text{blast intro: } * \text{ Morera_local_triangle_analytic_imp_holomorphic})$
qed

proposition *Schwarz_reflection*:

assumes *open* S **and** $cnjs$: $cnj \ ' S \subseteq S$
and $holf$: f *holomorphic_on* $(S \cap \{z. 0 < \text{Im } z\})$
and $contf$: *continuous_on* $(S \cap \{z. 0 \leq \text{Im } z\})$ f
and f : $\bigwedge z. \llbracket z \in S; z \in \mathbb{R} \rrbracket \implies (f \ z) \in \mathbb{R}$
shows $(\lambda z. \text{if } 0 \leq \text{Im } z \text{ then } f \ z \text{ else } cnj(f(cnjs \ z)))$ *holomorphic_on* S
proof –
have 1: $(\lambda z. \text{if } 0 \leq \text{Im } z \text{ then } f \ z \text{ else } cnj(f(cnjs \ z)))$ *holomorphic_on* $(S \cap \{z. 0 < \text{Im } z\})$
by $(\text{force intro: iffD1 } [OF \ \text{holomorphic_cong } [OF \ refl] \ holf])$
have $cont_cfc$: *continuous_on* $(S \cap \{z. \text{Im } z \leq 0\})$ $(cnj \ o \ f \ o \ cnj)$
using $cnjs$
by $(\text{intro continuous_intros continuous_on_compose continuous_on_subset } [OF \ contf])$ *auto*

```

  have cnj ∘ f ∘ cnj field_differentiable at x within S ∩ {z. Im z < 0}
    if x ∈ S Im x < 0 f field_differentiable at (cnj x) within S ∩ {z. 0 < Im z}
for x
  using that
  apply (clarsimp simp add: field_differentiable_def has_field_derivative_iff
Lim_within dist_norm)
  apply (rule_tac x=cnj f' in exI)
  apply (elim_all_forward ex_forward conj_forward imp_forward asm_rl, clarify)
  apply (drule_tac x=cnj xa in bspec)
  using cnjs apply force
  apply (metis complex_cnj_cnj complex_cnj_diff complex_cnj_divide complex_mod_cnj)
done
then have hol_cfc: (cnj ∘ f ∘ cnj) holomorphic_on (S ∩ {z. Im z < 0})
  using holf cnjs
  by (force simp: holomorphic_on_def)
have 2: (λz. if 0 ≤ Im z then f z else cnj (f (cnj z))) holomorphic_on (S ∩ {z.
Im z < 0})
  by (smt (verit) Int_Collect comp_def hol_cfc holomorphic_cong)
have [simp]: (S ∩ {z. 0 ≤ Im z}) ∪ (S ∩ {z. Im z ≤ 0}) = S
  by force
have eq: ⋀z. [z ∈ S; Im z ≤ 0; 0 ≤ Im z] ⟹ f z = cnj (f (cnj z))
  using f Reals_cnj_iff complex_is_Real_iff by auto
have continuous_on ((S ∩ {z. 0 ≤ Im z}) ∪ (S ∩ {z. Im z ≤ 0}))
  (λz. if 0 ≤ Im z then f z else cnj (f (cnj z)))
  apply (rule continuous_on_cases_local)
  using cont_cfc contf
  by (simp_all add: closedin_closed_Int closed_halfspace_Im_le closed_halfspace_Im_ge
eq)
then have 3: continuous_on S (λz. if 0 ≤ Im z then f z else cnj (f (cnj z)))
  by force
show ?thesis
  using holomorphic_on_paste_across_line [OF ‹open S›, of - i 0]
  using 1 2 3 by auto
qed

```

5.9 Bloch's theorem

lemma Bloch_lemma_0:

```

  assumes holf: f holomorphic_on cball 0 r and 0 < r
    and [simp]: f 0 = 0
    and le: ⋀z. norm z < r ⟹ norm (deriv f z) ≤ 2 * norm (deriv f 0)
  shows ball 0 ((3 - 2 * sqrt 2) * r * norm (deriv f 0)) ⊆ f ` ball 0 r

```

proof -

```

  have sqrt 2 < 3/2
    by (rule real_less_sqrt) (auto simp: power2_eq_square)
  then have sq3: 0 < 3 - 2 * sqrt 2 by simp
  show ?thesis

```

```

proof (cases deriv f 0 = 0)
  case True then show ?thesis by simp
next
  case False
  define C where C = 2 * norm(deriv f 0)
  have 0 < C using False by (simp add: C_def)
  have hol_f': f holomorphic_on ball 0 r using hol_f
    using ball_subset_cball holomorphic_on_subset by blast
  then have holdf': deriv f holomorphic_on ball 0 r
    by (rule holomorphic_deriv [OF _ open_ball])
  have Le1: norm(deriv f z - deriv f 0) ≤ norm z / (r - norm z) * C
    if norm z < r for z
  proof -
    have T1: norm(deriv f z - deriv f 0) ≤ norm z / (R - norm z) * C
      if R: norm z < R R < r for R
    proof -
      have 0 < R using R
      by (metis less_trans norm_zero zero_less_norm_iff)
      have df_le:  $\bigwedge x. \text{norm } x < r \implies \text{norm } (\text{deriv } f \ x) \leq C$ 
        using le by (simp add: C_def)
      have hol_df: deriv f holomorphic_on cball 0 R
        using R holdf' holomorphic_on_subset by auto
      have *: (( $\lambda w. \text{deriv } f \ w / (w - z)$ ) has_contour_integral 2 * pi * i * deriv
f z) (circlepath 0 R)
        if norm z < R for z
        using <0 < R> that Cauchy_integral_formula_convex_simple [OF convex_cball hol_df, of _ circlepath 0 R]
        by (force simp: winding_number_circlepath)
      have **: (( $\lambda x. \text{deriv } f \ x / (x - z) - \text{deriv } f \ x / x$ ) has_contour_integral
of_real (2 * pi) * i * (deriv f z - deriv f 0))
(circlepath 0 R)
        using has_contour_integral_diff [OF * [of z] * [of 0]] <0 < R> that
by (simp add: algebra_simps)
      have [simp]:  $\bigwedge x. \text{norm } x = R \implies x \neq z$  using that(1) by blast
      have norm (deriv f x / (x - z) - deriv f x / x)
        ≤ C * norm z / (R * (R - norm z))
        if norm x = R for x
    proof -
      have [simp]: norm (deriv f x * x - deriv f x * (x - z)) =
        norm (deriv f x) * norm z
      by (simp add: norm_mult right_diff_distrib')
      show ?thesis
      using <0 < R> <0 < C> R that
      by (auto simp add: norm_mult norm_divide divide_simps df_le
mult_mono norm_triangle_ineq2)
    qed
  then show ?thesis
    using has_contour_integral_bound_circlepath
      [OF **, of C * norm z / (R * (R - norm z))]

```



```

      ‹0 < R› ‹0 < C› R
    apply (simp add: norm_mult norm_divide)
    apply (simp add: divide_simps mult.commute)
  done
qed
obtain r' where r': norm z < r' r' < r
  using Rats_dense_in_real [of norm z r] ‹norm z < r› by blast
then have [simp]: closure {r' <..<r} = {r'..r} by simp
show ?thesis
  apply (rule continuous_ge_on_closure
    [where f = λr. norm z / (r - norm z) * C and S = {r' <..<r},
      OF _ _ T1])
  using that r'
  by (auto simp: not_le intro!: continuous_intros)
qed
have *: (norm z - norm z^2 / (r - norm z)) * norm (deriv f 0) ≤ norm (f z)
  if r: norm z < r for z
proof -
  have 1: ∧x. x ∈ ball 0 r ⟹
    ((λz. f z - deriv f 0 * z) has_field_derivative deriv f x - deriv f 0)
    (at x within ball 0 r)
  by (rule derivative_eq_intros holomorphic_derivI holf' | simp)+
  have 2: closed_segment 0 z ⊆ ball 0 r
  by (metis ‹0 < r› convex_ball convex_contains_segment dist_self mem_ball
    mem_ball_0 that)
  have 4: norm (deriv f (x *R z) - deriv f 0) * norm z ≤ norm z * norm z *
    x * C / (r - norm z)
  if x: 0 ≤ x x ≤ 1 for x
proof -
  have [simp]: x * norm z < r
  using r x by (meson le_less_trans mult_le_cancel_right2 norm_not_less_zero)
  then have cmod (x *R z) < r
  by (simp add: x)
  then have norm (deriv f (x *R z) - deriv f 0) ≤ norm (x *R z) / (r -
    norm (x *R z)) * C
  by (metis Le1)
  also have ... ≤ norm (x *R z) / (r - norm z) * C
  using r x ‹0 < r› ‹0 < C› by (simp add: frac_le mult_left_le_one_le)
  finally have norm (deriv f (x *R z) - deriv f 0) * norm z ≤ norm (x *R
    z) / (r - norm z) * C * norm z
  by (rule mult_right_mono) simp
  with x show ?thesis by (simp add: algebra_simps)
qed
have le_norm: abc ≤ norm d - e ⟹ norm (f - d) ≤ e ⟹ abc ≤ norm f
for abc d e and f::complex
  by (metis add_diff_cancel_left' add_diff_eq diff_left_mono norm_diff_ineq
    order_trans)
have norm (integral {0..1} (λx. (deriv f (x *R z) - deriv f 0) * z))
  ≤ integral {0..1} (λt. (norm z)^2 * t / (r - norm z) * C)

```

```

proof (rule integral_norm_bound_integral)
  show ( $\lambda x. (\text{deriv } f (x *_{\mathbb{R}} z) - \text{deriv } f 0) * z$ ) integrable_on {0..1}
    using contour_integral_primitive [OF 1, of linepath 0 z] 2
  by (simp add: has_contour_integral_linepath has_integral_integrable_integral)
  have (*) ((cmod z)2) integrable_on {0..1}
  by (metis ident_integrable_on integrable_0 integrable_eq integrable_on_cmult_iff
    lambda_zero)
  then show ( $\lambda t. (\text{norm } z)^2 * t / (r - \text{norm } z) * C$ ) integrable_on {0..1}
    using integrable_on_cmult_right[where 'b=real, simplified] integrable_on_divide
  [where 'b=real, simplified]
    by blast
  qed (simp add: norm_mult power2_eq_square 4)
  then have int_le:  $\text{norm } (f z - \text{deriv } f 0 * z) \leq (\text{norm } z)^2 * \text{norm } (\text{deriv } f 0)$ 
  / ((r - norm z))
    using contour_integral_primitive [OF 1, of linepath 0 z] 2
  by (simp add: has_contour_integral_linepath has_integral_integrable_integral
    C_def)
  have  $\text{norm } z * (\text{norm } (\text{deriv } f 0) * (r - \text{norm } z - \text{norm } z)) \leq \text{norm } z * (\text{norm } (\text{deriv } f 0) * (r - \text{norm } z) - \text{norm } (\text{deriv } f 0) * \text{norm } z)$ 
    by (simp add: algebra_simps)
  then have §:  $(\text{norm } z * (r - \text{norm } z) - \text{norm } z * \text{norm } z) * \text{norm } (\text{deriv } f 0) \leq \text{norm } (\text{deriv } f 0) * \text{norm } z * (r - \text{norm } z) - \text{norm } z * \text{norm } z * \text{norm } (\text{deriv } f 0)$ 
    by (simp add: algebra_simps)
  show ?thesis
    using ‹norm z < r›
    by (force simp add: power2_eq_square divide_simps C_def norm_mult §
    intro!: le_norm [OF _ int_le])
  qed
  have sq201 [simp]:  $0 < (1 - \sqrt{2} / 2) (1 - \sqrt{2} / 2) < 1$ 
    by (auto simp: sqrt2_less_2)
  have 1: continuous_on (closure (ball 0 ((1 - sqrt 2 / 2) * r))) f
  proof (rule continuous_on_subset [OF holomorphic_on_imp_continuous_on
    [OF holf]])
    show closure (ball 0 ((1 - sqrt 2 / 2) * r))  $\subseteq$  cball 0 r
    proof -
      have  $(1 - \sqrt{2} / 2) * r \leq r$ 
        by (simp add: ‹0 < r›)
      then show ?thesis
        by (meson ball_subset_ccball closed_ccball closure_minimal dual_order.trans
        subset_ball)
    qed
  qed
  have 2: open (f ` interior (ball 0 ((1 - sqrt 2 / 2) * r)))
  proof (rule open_mapping_thm [OF holf' open_ball connected_ball])
    show interior (ball 0 ((1 - sqrt 2 / 2) * r))  $\subseteq$  ball (0::complex) r
      using ‹0 < r› mult_pos_pos sq201 by (simp add: ball_subset_ball_iff)
    show  $\neg f \text{ constant\_on ball } 0 r$ 
      using False ‹0 < r› centre_in_ball holf' holomorphic_nonconstant by blast

```

```

qed auto
have ball 0 ((3 - 2 * sqrt 2) * r * norm (deriv f 0)) =
  ball (f 0) ((3 - 2 * sqrt 2) * r * norm (deriv f 0))
by simp
also have ...  $\subseteq f^{-1}$  ball 0 ((1 - sqrt 2 / 2) * r)
proof -
  have 3: (3 - 2 * sqrt 2) * r * norm (deriv f 0)  $\leq$  norm (f z)
  if norm z = (1 - sqrt 2 / 2) * r for z
  proof (rule order_trans [OF _ *])
    show (3 - 2 * sqrt 2) * r * cmod (deriv f 0)
       $\leq$  (cmod z - (cmod z)2 / (r - cmod z)) * cmod (deriv f 0)
    by (simp add: le_less algebra_simps divide_simps power2_eq_square that)
  qed (use <0 < r> that in auto)
  show ?thesis
  using <0 < r> sq201 3 C_def <0 < C> sq3
  by (intro ball_subset_open_map_image [OF 1 2 _ bounded_ball]) auto
qed
also have ...  $\subseteq f^{-1}$  ball 0 r
proof -
  have  $\bigwedge x. (1 - \text{sqrt } 2 / 2) * r \leq r$ 
  using <0 < r> by (auto simp: field_simps)
  then show ?thesis
  by auto
qed
finally show ?thesis .
qed
qed

```

lemma Bloch_lemma:

```

assumes holf: f holomorphic_on cball a r and 0 < r
and le:  $\bigwedge z. z \in \text{ball } a \ r \implies \text{norm}(\text{deriv } f \ z) \leq 2 * \text{norm}(\text{deriv } f \ a)$ 
shows ball (f a) ((3 - 2 * sqrt 2) * r * norm (deriv f a))  $\subseteq f^{-1}$  ball a r (is ?lhs
 $\subseteq$  ?rhs)
proof -
  have fz: ( $\lambda z. f \ (a + z)$ ) = f o ( $\lambda z. (a + z)$ )
  by (simp add: o_def)
  have hol0: ( $\lambda z. f \ (a + z)$ ) holomorphic_on cball 0 r
  unfolding fz by (intro holomorphic_intros holf holomorphic_on_compose |
simp)+
  then have [simp]:  $\bigwedge x. \text{norm } x < r \implies (\lambda z. f \ (a + z)) \text{ field\_differentiable at } x$ 
  by (metis open_ball at_within_open ball_subset_cball diff_0 dist_norm holomorphic_on_def holomorphic_on_subset mem_ball norm_minus_cancel)
  have [simp]:  $\bigwedge z. \text{norm } z < r \implies f \text{ field\_differentiable at } (a + z)$ 
  by (metis holf open_ball add_diff_cancel_left' dist_complex_def holomorphic_on_imp_differentiable_at holomorphic_on_subset interior_cball interior_subset mem_ball norm_minus_commute)
  then have [simp]: f field_differentiable at a
  by (metis add.comm_neutral <0 < r> norm_eq_zero)
  have hol1: ( $\lambda z. f \ (a + z) - f \ a$ ) holomorphic_on cball 0 r

```

```

    by (intro holomorphic_intros hol0)
  moreover have  $\bigwedge z. \text{cmod } z < r \implies$ 
     $\text{cmod } (\text{deriv } (\lambda z. f (a + z)) z) \leq 2 * \text{cmod } (\text{deriv } (\lambda z. f (a + z)) 0)$ 
    by (simp add: fz deriv_chain dist_norm le)
  ultimately have  $\S: \text{ball } 0 ((3 - 2 * \text{sqrt } 2) * r * \text{norm } (\text{deriv } (\lambda z. f (a + z)) - f a) 0))$ 
     $\subseteq (\lambda z. f (a + z) - f a) ' \text{ball } 0 r$ 
    using  $\langle 0 < r \rangle$  by (intro Bloch_lemma_0) auto
  show ?thesis
proof clarify
  fix x
  assume  $x \in ?lhs$ 
  with subsetD [OF  $\S$ , of  $x - f a$ ] show  $x \in ?rhs$ 
    by (force simp: fz  $\langle 0 < r \rangle$  dist_norm deriv_chain field_differentiable_compose)
qed
qed

```

proposition *Bloch_unit*:

assumes *holf*: *f* holomorphic_on ball *a* 1 and [*simp*]: *deriv f a* = 1
 obtains *b r* where $1/12 < r$ and $\text{ball } b r \subseteq f ' (\text{ball } a 1)$

proof –

```

define r :: real where r = 249/256
have  $0 < r$  and  $r < 1$  by (auto simp: r_def)
define g where  $g z = \text{deriv } f z * \text{of\_real}(r - \text{norm}(z - a))$  for z
have deriv f holomorphic_on ball a 1
  by (rule holomorphic_deriv [OF holf open_ball])
then have continuous_on (ball a 1) (deriv f)
  using holomorphic_on_imp_continuous_on by blast
then have continuous_on (cball a r) (deriv f)
  by (rule continuous_on_subset) (simp add: cball_subset_ball_iff  $\langle r < 1 \rangle$ )
then have continuous_on (cball a r) g
  by (simp add: g_def continuous_intros)
then have 1: compact ( $g ' \text{cball } a r$ )
  by (rule compact_continuous_image [OF __ compact_cball])
have 2:  $g ' \text{cball } a r \neq \{\}$ 
  using  $\langle r > 0 \rangle$  by auto
obtain p where pr:  $p \in \text{cball } a r$ 
  and pge:  $\bigwedge y. y \in \text{cball } a r \implies \text{norm } (g y) \leq \text{norm } (g p)$ 
  using distance_attains_sup [OF 1 2, of 0] by force
define t where  $t = (r - \text{norm}(p - a)) / 2$ 
have  $\text{norm } (p - a) \neq r$ 
  using pge [of a]  $\langle r > 0 \rangle$  by (auto simp: g_def norm_mult)
then have  $\text{norm } (p - a) < r$  using pr
  by (simp add: norm_minus_commute dist_norm)
then have  $0 < t$ 
  by (simp add: t_def)
have cpt:  $\text{cball } p t \subseteq \text{ball } a r$ 
  using  $\langle 0 < t \rangle$  by (simp add: cball_subset_ball_iff dist_norm t_def field_simps)
have gen_le_dfp:  $\text{norm } (\text{deriv } f y) * (r - \text{norm } (y - a)) / (r - \text{norm } (p - a))$ 

```

```

≤ norm (deriv f p)
  if y ∈ cball a r for y
proof -
  have [simp]: norm (y - a) ≤ r
    using that by (simp add: dist_norm norm_minus_commute)
  have norm (g y) ≤ norm (g p)
    using pge [OF that] by simp
  then have norm (deriv f y) * abs (r - norm (y - a)) ≤ norm (deriv f p) *
    abs (r - norm (p - a))
    by (simp only: dist_norm g_def norm_mult norm_of_real)
  with that ⟨norm (p - a) < r⟩ show ?thesis
    by (simp add: dist_norm field_split_simps)
qed
have le_norm_dfp: r / (r - norm (p - a)) ≤ norm (deriv f p)
  using gen_le_dfp [of a] ⟨r > 0⟩ by auto
have 1: f holomorphic_on cball p t
  using cpt ⟨r < 1⟩ order_subst1 subset_ball
  by (force simp: intro!: holomorphic_on_subset [OF holf])
have 2: norm (deriv f z) ≤ 2 * norm (deriv f p) if z ∈ ball p t for z
proof -
  have z: z ∈ cball a r
    by (meson ball_subset_cball subsetD cpt that)
  then have norm(z - a) < r
    by (metis ball_subset_cball contra_subsetD cpt dist_norm mem_ball norm_minus_commute
    that)
  have norm (deriv f z) * (r - norm (z - a)) / (r - norm (p - a)) ≤ norm
    (deriv f p)
    using gen_le_dfp [OF z] by simp
  with ⟨norm (z - a) < r⟩ ⟨norm (p - a) < r⟩
  have norm (deriv f z) ≤ (r - norm (p - a)) / (r - norm (z - a)) * norm
    (deriv f p)
    by (simp add: field_simps)
  also have ... ≤ 2 * norm (deriv f p)
  proof (rule mult_right_mono)
    show (r - cmod (p - a)) / (r - cmod (z - a)) ≤ 2
      using that ⟨norm (p - a) < r⟩ ⟨norm (z - a) < r⟩ dist_triangle3 [of z a p]
      by (simp add: field_simps t_def dist_norm [symmetric])
    qed auto
  finally show ?thesis .
qed
have sqrt2: sqrt 2 < 2113/1494
  by (rule real_less_sqrt) (auto simp: power2_eq_square)
then have sq3: 0 < 3 - 2 * sqrt 2 by simp
have 1 / 12 / ((3 - 2 * sqrt 2) / 2) < r
  using sq3 sqrt2 by (auto simp: field_simps r_def)
also have ... ≤ cmod (deriv f p) * (r - cmod (p - a))
  using ⟨norm (p - a) < r⟩ le_norm_dfp by (simp add: pos_divide_le_eq)
finally have 1 / 12 < cmod (deriv f p) * (r - cmod (p - a)) * ((3 - 2 * sqrt
2) / 2)

```

```

    using pos_divide_less_eq half_gt_zero_iff sq3 by blast
  then have **:  $1 / 12 < (3 - 2 * \text{sqrt } 2) * t * \text{norm } (\text{deriv } f p)$ 
    using sq3 by (simp add: mult.commute t_def)
  have ball (f p)  $((3 - 2 * \text{sqrt } 2) * t * \text{norm } (\text{deriv } f p)) \subseteq f^{-1} \text{ ball } p t$ 
    by (rule Bloch_lemma [OF 1 <0 < t> 2])
  also have ...  $\subseteq f^{-1} \text{ ball } a 1$ 
    by (meson <r < 1> ball_subset_cball cpt dual_order.trans image_mono less_le_not_le
subset_ball)
  finally have ball (f p)  $((3 - 2 * \text{sqrt } 2) * t * \text{norm } (\text{deriv } f p)) \subseteq f^{-1} \text{ ball } a 1$  .
  with ** show ?thesis
    by (rule that)
qed

```

theorem Bloch:

```

  assumes holf:  $f \text{ holomorphic\_on ball } a r$  and  $0 < r$ 
    and r':  $r' \leq r * \text{norm } (\text{deriv } f a) / 12$ 
  obtains b where  $\text{ball } b r' \subseteq f^{-1} (\text{ball } a r)$ 
proof (cases  $\text{deriv } f a = 0$ )
  case True with r' show ?thesis
    using ball_eq_empty that by fastforce
next
  case False
  define C where  $C = \text{deriv } f a$ 
  have  $0 < \text{norm } C$  using False by (simp add: C_def)
  have dfa:  $f \text{ field\_differentiable at } a$ 
    using <0 < r> holomorphic_on_imp_differentiable_at [OF holf] by auto
  have fo:  $(\lambda z. f (a + \text{of\_real } r * z)) = f \circ (\lambda z. (a + \text{of\_real } r * z))$ 
    by (simp add: o_def)
  have holf':  $f \text{ holomorphic\_on } (\lambda z. a + \text{complex\_of\_real } r * z)^{-1} \text{ ball } 0 1$ 
    using <0 < r> holomorphic_on_subset [OF holf] by (force simp: dist_norm
norm_mult)
  have 1:  $(\lambda z. f (a + r * z) / (C * r)) \text{ holomorphic\_on ball } 0 1$ 
    using <0 < r> <0 < norm C>
    by (intro holomorphic_intros holomorphic_on_compose holf'; simp add: fo)+
  have  $((\lambda z. f (a + \text{of\_real } r * z) / (C * \text{of\_real } r)) \text{ has\_field\_derivative } (\text{deriv } f (a + \text{of\_real } r * z) / C)) \text{ at } z$ 
    if  $\text{norm } z < 1$  for z
  proof -
    have fd:  $f \text{ field\_differentiable at } (a + \text{complex\_of\_real } r * z)$ 
      using <0 < r> by (simp_all add: dist_norm norm_mult holomorphic_on_imp_differentiable_at
[OF holf] that)
    have *:  $((\lambda x. f (a + \text{of\_real } r * x)) \text{ has\_field\_derivative } (\text{deriv } f (a + \text{of\_real } r * z) * \text{of\_real } r)) \text{ at } z$ 
      by (rule fd DERIV_chain [OF field_differentiable_derivI] derivative_eq_intros
| simp add: fo)+
    show ?thesis
      apply (rule derivative_eq_intros * | simp)+
      using <0 < r> by (auto simp: C_def False)
  qed
qed

```

```

obtain  $f'$  where ( $f$  has_field_derivative  $f'$ ) (at  $a$ )
using dfa field_differentiable_def by blast
then have  $\exists c. ((\lambda c. f (a + \text{complex\_of\_real } r * c)) \text{ has\_field\_derivative } c)$  (at
0)
by (metis (no_types) DERIV_chain2 add_cancel_left_right field_differentiable_add_const

    field_differentiable_def field_differentiable_linear mult_eq_0_iff)
then have ( $\lambda w. f (a + \text{complex\_of\_real } r * w)$ ) field_differentiable at 0
by (simp add: field_differentiable_def)
then have deriv ( $\lambda z. f (a + \text{of\_real } r * z) / (C * \text{of\_real } r)$ ) 0
    = deriv ( $\lambda z. f (a + \text{of\_real } r * z)$ ) 0 / ( $C * \text{of\_real } r$ )
by (rule deriv_cdivide_right)
also have ... = 1
using  $\langle 0 < r \rangle$  by (simp add: C_def False fo_derivative_intros dfa deriv_chain)
finally have 2: deriv ( $\lambda z. f (a + \text{of\_real } r * z) / (C * \text{of\_real } r)$ ) 0 = 1 .
have sb1:  $(*) (C * r) \text{ ' } (\lambda z. f (a + \text{of\_real } r * z) / (C * r)) \text{ ' ball } 0 \ 1$ 
     $\subseteq f \text{ ' ball } a \ r$ 
using  $\langle 0 < r \rangle$  by (auto simp: dist_norm norm_mult C_def False)
have sb2: ball  $(C * r * b) \ r' \subseteq (*) (C * r) \text{ ' ball } b \ t$ 
    if  $1 / 12 < t$  for  $b \ t$ 
proof –
have *:  $r * \text{cmod} (\text{deriv } f \ a) / 12 \leq r * (t * \text{cmod} (\text{deriv } f \ a))$ 
using that  $\langle 0 < r \rangle$  less_eq_real_def mult.commute mult.right_neutral mult_left_mono
norm_ge_zero times_divide_eq_right
by auto
show ?thesis
apply clarify
apply (rule_tac  $x=x / (C * r)$  in image_eqI)
using  $\langle 0 < r \rangle$  apply (simp_all add: dist_norm norm_mult norm_divide
C_def False field_simps)
using *  $r'$  by linarith
qed
show ?thesis
apply (rule Bloch_unit [OF 1 2])
using image_mono sb1 sb2 that by fastforce
qed

corollary Bloch_general:
assumes holf:  $f$  holomorphic_on  $S$  and  $a \in S$ 
and tle:  $\bigwedge z. z \in \text{frontier } S \implies t \leq \text{dist } a \ z$ 
and rle:  $r \leq t * \text{norm}(\text{deriv } f \ a) / 12$ 
obtains  $b$  where ball  $b \ r \subseteq f \text{ ' } S$ 
proof –
consider  $r \leq 0 \mid 0 < t * \text{norm}(\text{deriv } f \ a) / 12$  using rle by force
then show ?thesis
proof cases
case 1 then show ?thesis
by (simp add: ball_empty that)
next

```

```

case 2
show ?thesis
proof (cases deriv f a = 0)
  case True then show ?thesis
    using rle by (simp add: ball_empty that)
  next
  case False
  then have t > 0
    using 2 by (force simp: zero_less_mult_iff)
  have  $\neg \text{ball } a \ t \subseteq S \implies \text{ball } a \ t \cap \text{frontier } S \neq \{\}$ 
    by (metis Diff_eq_empty_iff  $\langle 0 < t \rangle \langle a \in S \rangle \text{closure\_Int\_ball\_not\_empty}$ 
closure_subset connected_Int_frontier connected_ball inf.commute)
  with tle have *:  $\text{ball } a \ t \subseteq S$  by fastforce
  then have 1:  $f \text{ holomorphic\_on } \text{ball } a \ t$ 
    using holf using holomorphic_on_subset by blast
  show ?thesis
    using Bloch [OF 1  $\langle t > 0 \rangle$  rle] * by (metis image_mono order_trans that)
qed
qed
qed
end

```

6 The Great Picard Theorem and its Applications

Ported from HOL Light (cauchy.ml) by L C Paulson, 2017

```

theory Great_Picard
  imports Conformal_Mappings
begin

```

6.1 Schottky's theorem

```

lemma Schottky_lemma0:
  assumes holf:  $f \text{ holomorphic\_on } S$  and cons:  $\text{contractible } S$  and  $a \in S$ 
  and f:  $\bigwedge z. z \in S \implies f z \neq 1 \wedge f z \neq -1$ 
  obtains g where  $g \text{ holomorphic\_on } S$ 
     $\text{norm}(g a) \leq 1 + \text{norm}(f a) / 3$ 
     $\bigwedge z. z \in S \implies f z = \cos(\text{of\_real } \pi i * g z)$ 
proof -
  obtain g where holg:  $g \text{ holomorphic\_on } S$  and g:  $\text{norm}(g a) \leq \pi + \text{norm}(f a)$ 
    and f_eq_cos:  $\bigwedge z. z \in S \implies f z = \cos(g z)$ 
  using contractible_imp_holomorphic_arccos_bounded [OF assms]
  by blast
  show ?thesis
  proof
    show  $(\lambda z. g z / \pi i) \text{ holomorphic\_on } S$ 
      by (auto intro: holomorphic_intros holg)
    have  $3 \leq \pi$ 

```



```

    using pi_approx by force
  have  $3 * \text{norm}(g \ a) \leq 3 * (\text{pi} + \text{norm}(f \ a))$ 
    using g by auto
  also have  $\dots \leq \text{pi} * 3 + \text{pi} * \text{cmod}(f \ a)$ 
    using  $\langle 3 \leq \text{pi} \rangle$  by (simp add: mult_right_mono algebra_simps)
  finally show  $\text{cmod}(g \ a / \text{complex\_of\_real} \ \text{pi}) \leq 1 + \text{cmod}(f \ a) / 3$ 
    by (simp add: field_simps norm_divide)
  show  $\bigwedge z. z \in S \implies f \ z = \cos(\text{complex\_of\_real} \ \text{pi} * (g \ z / \text{complex\_of\_real} \ \text{pi}))$ 
    by (simp add: f_eq_cos)
qed
qed

```

```

lemma Schottky_lemma1:
  fixes n::nat
  assumes  $0 < n$ 
  shows  $0 < n + \text{sqrt}(\text{real } n^2 - 1)$ 
proof -
  have  $0 < n * n$ 
    by (simp add: assms)
  then show ?thesis
    by (metis add.commute add.right_neutral add_pos_nonneg assms diff_ge_0_iff_ge
      nat_less_real_le of_nat_0 of_nat_0_less_iff of_nat_power power2_eq_square
      real_sqrt_ge_0_iff)
qed

```

```

lemma Schottky_lemma2:
  fixes x::real
  assumes  $0 \leq x$ 
  obtains n where  $0 < n \wedge |x - \ln(\text{real } n + \text{sqrt}((\text{real } n)^2 - 1)) / \text{pi}| < 1/2$ 
proof -
  obtain n0::nat where  $0 < n0 \wedge \ln(n0 + \text{sqrt}(\text{real } n0^2 - 1)) / \text{pi} \leq x$ 
  proof
    show  $\ln(\text{real } 1 + \text{sqrt}(\text{real } 1^2 - 1)) / \text{pi} \leq x$ 
      by (auto simp: assms)
    qed auto
  moreover
  obtain M::nat where  $\bigwedge n. [0 < n; \ln(n + \text{sqrt}(\text{real } n^2 - 1)) / \text{pi} \leq x] \implies n \leq M$ 
  proof
    fix n::nat
    assume  $0 < n \wedge \ln(n + \text{sqrt}((\text{real } n)^2 - 1)) / \text{pi} \leq x$ 
    then have  $\ln(n + \text{sqrt}((\text{real } n)^2 - 1)) \leq x * \text{pi}$ 
      by (simp add: field_split_simps)
    then have  $*: \exp(\ln(n + \text{sqrt}((\text{real } n)^2 - 1))) \leq \exp(x * \text{pi})$ 
      by blast
    have  $0: 0 \leq \text{sqrt}((\text{real } n)^2 - 1)$ 

```

```

    using ‹0 < n› by auto
    have  $n + \sqrt{(\text{real } n)^2 - 1} = \exp(\ln(n + \sqrt{(\text{real } n)^2 - 1}))$ 
    by (simp add: Suc_leI ‹0 < n› add_pos_nonneg real_of_nat_ge_one_iff)
    also have  $\dots \leq \exp(x * \pi)$ 
    using * by blast
    finally have  $\text{real } n \leq \exp(x * \pi)$ 
    using 0 by linarith
    then show  $n \leq \text{nat}(\text{ceiling}(\exp(x * \pi)))$ 
    by linarith
qed
ultimately obtain n where
  0 < n and le_x:  $\ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi \leq x$ 
  and le_n:  $\bigwedge k. \llbracket 0 < k; \ln(k + \sqrt{(\text{real } k)^2 - 1}) / \pi \leq x \rrbracket \implies k \leq$ 
n
  using bounded_Max_nat [of  $\lambda n. 0 < n \wedge \ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi \leq$ 
x] by metis
define a where  $a \equiv \ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi$ 
define b where  $b \equiv \ln(1 + \text{real } n + \sqrt{(1 + \text{real } n)^2 - 1}) / \pi$ 
have le_xa:  $a \leq x$ 
and le_na:  $\bigwedge k. \llbracket 0 < k; \ln(k + \sqrt{(\text{real } k)^2 - 1}) / \pi \leq x \rrbracket \implies k \leq n$ 
using le_x le_n by (auto simp: a_def)
moreover have  $x < b$ 
using le_n [of Suc n] by (force simp: b_def)
moreover have  $b - a < 1$ 
proof -
  have  $\ln(1 + \text{real } n + \sqrt{(1 + \text{real } n)^2 - 1}) - \ln(\text{real } n + \sqrt{(\text{real } n)^2 - 1}) =$ 
 $\ln((1 + \text{real } n + \sqrt{(1 + \text{real } n)^2 - 1}) / (\text{real } n + \sqrt{(\text{real } n)^2 - 1}))$ 
by (simp add: ‹0 < n› Schottky_lemma1 add_pos_nonneg ln_divide_pos
[symmetric])
  also have  $\dots \leq 3$ 
proof (cases n = 1)
  case True
  have  $\sqrt{3} \leq 2$ 
  by (simp add: real_le_sqrt)
  then have  $(2 + \sqrt{3}) \leq 4$ 
  by simp
  also have  $\dots \leq \exp 3$ 
  using exp_ge_add_one_self [of 3::real] by simp
  finally have  $\ln(2 + \sqrt{3}) \leq 3$ 
  by (metis add_nonneg_nonneg add_pos_nonneg dbl_def dbl_inc_def
dbl_inc_simps(3) exp_gt_zero ln_exp ln_le_cancel_iff real_sqrt_ge_0_iff
zero_le_one zero_less_one)
  then show ?thesis
  by (simp add: True)
next
  case False with ‹0 < n› have  $1 < n \wedge 2 \leq n$ 

```

```

    by linarith+
  then have 1:  $1 \leq \text{real } n * \text{real } n$ 
  by (metis less_imp_le_nat one_le_power power2_eq_square real_of_nat_ge_one_iff)
  have *:  $4 + (m+2) * 2 \leq (m+2) * ((m+2) * 3)$  for  $m::\text{nat}$ 
    by simp
  have  $4 + n * 2 \leq n * (n * 3)$ 
    using * [of n-2]  $\langle 2 \leq n \rangle$ 
    by (metis le_add_diff_inverse2)
  then have **:  $4 + \text{real } n * 2 \leq \text{real } n * (\text{real } n * 3)$ 
  by (metis (mono_tags, opaque_lifting) of_nat_le_iff of_nat_add of_nat_mult
of_nat_numeral)
  have  $\text{sqrt } ((1 + \text{real } n)^2 - 1) \leq 2 * \text{sqrt } ((\text{real } n)^2 - 1)$ 
    by (auto simp: real_le_sqrt power2_eq_square algebra_simps 1 **)
  then
    have  $((1 + \text{real } n + \text{sqrt } ((1 + \text{real } n)^2 - 1)) / (\text{real } n + \text{sqrt } ((\text{real } n)^2 - 1))) \leq 2$ 
      using Schottky_lemma1  $\langle 0 < n \rangle$  by (simp add: field_split_simps)
    then have  $\ln ((1 + \text{real } n + \text{sqrt } ((1 + \text{real } n)^2 - 1)) / (\text{real } n + \text{sqrt } ((\text{real } n)^2 - 1))) \leq \ln 2$ 
      using Schottky_lemma1 [of n]  $\langle 0 < n \rangle$ 
      by (simp add: field_split_simps add_pos_nonneg)
    also have  $\dots \leq 3$ 
      using ln_add_one_self_le_self [of 1] by auto
    finally show ?thesis .
  qed
  also have  $\dots < \pi$ 
    using pi_approx by simp
  finally show ?thesis
    by (simp add: a_def b_def field_split_simps)
  qed
  ultimately have  $|x - a| < 1/2 \vee |x - b| < 1/2$ 
    by (auto simp: abs_if)
  then show thesis
  proof
    assume  $|x - a| < 1/2$ 
    then show ?thesis
      by (rule_tac  $n=n$  in that) (auto simp: a_def  $\langle 0 < n \rangle$ )
  next
    assume  $|x - b| < 1/2$ 
    then show ?thesis
      by (rule_tac  $n=\text{Suc } n$  in that) (auto simp: b_def  $\langle 0 < n \rangle$ )
  qed
qed

```

lemma Schottky_lemma3:

```

  fixes  $z::\text{complex}$ 
  assumes  $z \in (\bigcup m \in \text{Ints. } \bigcup n \in \{0 < ..\}. \{\text{Complex } m (\ln(n + \text{sqrt}(\text{real } n^2 - 1)) / \pi i)\})$ 

```

```

    ∪ (∪ m ∈ Ints. ∪ n ∈ {0<..}. {Complex m (-ln(n + sqrt(real n ^ 2 -
1))) / pi}))
  shows cos(pi * cos(pi * z)) = 1 ∨ cos(pi * cos(pi * z)) = -1
proof -
  have sqrt2 [simp]: complex_of_real (sqrt x) * complex_of_real (sqrt x) = x if x
≥ 0 for x::real
  by (metis abs_of_nonneg of_real_mult real_sqrt_mult_self that)
  define plusi where plusi (e::complex) ≡ e + inverse e for e
  have 1: ∃ k. plusi (exp (i * (of_int m * complex_of_real pi) - ln (real n + sqrt
((real n)^2 - 1)))) = of_int k * 2
    (is ∃ k. ?Φ k)
    if n > 0 for m n
proof -
  have eeq: e ≠ 0 ⇒ plusi e = n ⇔ (inverse e) ^ 2 = n/e - 1 for n e::complex
    by (auto simp: plusi_def field_simps power2_eq_square)
  have [simp]: 1 ≤ real n * real n
    using nat_0_less_mult_iff nat_less_real_le that by force
  consider odd m | even m
  by blast
  then have ∃ k. ?Φ k
proof cases
  case 1
  then have ?Φ (- n)
    using Schottky_lemma1 [OF that]
    by (simp add: eeq) (simp add: power2_eq_square exp_diff exp_Euler
exp_of_real algebra_simps sin_int_times_real cos_int_times_real)
  then show ?thesis ..
next
  case 2
  then have ?Φ n
    using Schottky_lemma1 [OF that]
    by (simp add: eeq) (simp add: power2_eq_square exp_diff exp_Euler
exp_of_real algebra_simps)
  then show ?thesis ..
qed
then show ?thesis by blast
qed
have 2: ∃ k. plusi (exp (i * (of_int m * complex_of_real pi) +
(ln (real n + sqrt ((real n)^2 - 1)))))) = of_int k * 2
  (is ∃ k. ?Φ k)
  if n > 0 for m n
proof -
  have eeq: e ≠ 0 ⇒ plusi e = n ⇔ e^2 - n*e + 1 = 0 for n e::complex
    by (auto simp: plusi_def field_simps power2_eq_square)
  have [simp]: 1 ≤ real n * real n
    by (metis One_nat_def add.commute nat_less_real_le of_nat_1 of_nat_Suc
one_le_power power2_eq_square that)
  consider odd m | even m
  by blast

```

```

then have  $\exists k. ?\Phi k$ 
proof cases
  case 1
  then have  $?\Phi (-n)$ 
    using Schottky_lemma1 [OF that]
    by (simp add: eeq) (simp add: power2_eq_square exp_add exp_Euler
exp_of_real algebra_simps sin_int_times_real cos_int_times_real)
  then show ?thesis ..
next
  case 2
  then have  $?\Phi n$ 
    using Schottky_lemma1 [OF that]
    by (simp add: eeq) (simp add: power2_eq_square exp_add exp_Euler
exp_of_real algebra_simps)
  then show ?thesis ..
qed
then show ?thesis by blast
qed
have  $\exists x. \cos(\text{complex\_of\_real } \pi * z) = \text{of\_int } x$ 
  using assms
  apply (auto simp: Ints_def cos_exp_eq exp_minus Complex_eq simp flip:
plusi_def)
  apply (auto simp: algebra_simps dest: 1 2)
done
then have  $\sin(\pi * \cos(\pi * z)) ^ 2 = 0$ 
  by (simp add: Complex_Transcendental.sin_eq_0)
then have  $1 - \cos(\pi * \cos(\pi * z)) ^ 2 = 0$ 
  by (simp add: sin_squared_eq)
then show ?thesis
  using power2_eq_1_iff by auto
qed

```

theorem Schottky:

```

assumes holf:  $f \text{ holomorphic\_on } \text{cball } 0 \ 1$ 
  and nof0:  $\text{norm}(f \ 0) \leq r$ 
  and not01:  $\bigwedge z. z \in \text{cball } 0 \ 1 \implies \neg(f \ z = 0 \vee f \ z = 1)$ 
  and 0 < t t < 1  $\text{norm } z \leq t$ 
  shows  $\text{norm}(f \ z) \leq \exp(\pi * \exp(\pi * (2 + 2 * r + 12 * t / (1 - t))))$ 
proof -
  obtain h where holf:  $h \text{ holomorphic\_on } \text{cball } 0 \ 1$ 
    and nh0:  $\text{norm}(h \ 0) \leq 1 + \text{norm}(2 * f \ 0 - 1) / 3$ 
    and h:  $\bigwedge z. z \in \text{cball } 0 \ 1 \implies 2 * f \ z - 1 = \cos(\text{of\_real } \pi * h \ z)$ 
  proof (rule Schottky_lemma0 [of  $\lambda z. 2 * f \ z - 1 \ \text{cball } 0 \ 1 \ 0$ ])
    show  $(\lambda z. 2 * f \ z - 1) \text{ holomorphic\_on } \text{cball } 0 \ 1$ 
      by (intro holomorphic_intros holf)
    show contractible (cball (0::complex) 1)
      by (auto simp: convex_imp_contractible)
    show  $\bigwedge z. z \in \text{cball } 0 \ 1 \implies 2 * f \ z - 1 \neq 1 \wedge 2 * f \ z - 1 \neq -1$ 

```

```

    using not01 by force
  qed auto
  obtain g where holg: g holomorphic_on cball 0 1
    and ng0: norm(g 0) ≤ 1 + norm(h 0) / 3
    and g:  $\bigwedge z. z \in \text{cball } 0 \ 1 \implies h \ z = \cos(\text{of\_real } \pi * g \ z)$ 
  proof (rule Schottky_lemma0 [OF holf convex_imp_contractible, of 0])
    show  $\bigwedge z. z \in \text{cball } 0 \ 1 \implies h \ z \neq 1 \wedge h \ z \neq -1$ 
      using h not01 by fastforce+
  qed auto
  have g0_2_f0: norm(g 0) ≤ 2 + norm(f 0)
  proof -
    have cmod (2 * f 0 - 1) ≤ cmod (2 * f 0) + 1
      by (metis norm_one norm_triangle_ineq4)
    also have ... ≤ 6 + 9 * cmod (f 0)
      by auto
    finally have 1 + norm(2 * f 0 - 1) / 3 ≤ (2 + norm(f 0) - 1) * 3
      by (simp add: divide_simps)
    with nh0 have norm(h 0) ≤ (2 + norm(f 0) - 1) * 3
      by linarith
    then have 1 + norm(h 0) / 3 ≤ 2 + norm(f 0)
      by simp
    with ng0 show ?thesis
      by auto
  qed
  have z ∈ ball 0 1
    using assms by auto
  have norm_g_12: norm(g z - g 0) ≤ (12 * t) / (1 - t)
  proof -
    obtain g' where g':  $\bigwedge x. x \in \text{cball } 0 \ 1 \implies (g \text{ has\_field\_derivative } g' \ x)$  (at x
    within cball 0 1)
    using holg [unfolded holomorphic_on_def field_differentiable_def] by metis
    have int_g': (g' has_contour_integral g z - g 0) (linepath 0 z)
    using contour_integral_primitive [OF g' valid_path_linepath, of 0 z]
    using ⟨z ∈ ball 0 1⟩ segment_bound1 by fastforce
    have cmod (g' w) ≤ 12 / (1 - t) if w ∈ closed_segment 0 z for w
  proof -
    have w: w ∈ ball 0 1
      using segment_bound [OF that] ⟨z ∈ ball 0 1⟩ by simp
    have *:  $\llbracket \bigwedge b. (\exists w \in T \cup U. w \in \text{ball } b \ 1); \bigwedge x. x \in D \implies g \ x \notin T \cup U \rrbracket$ 
 $\implies \nexists b. \text{ball } b \ 1 \subseteq g^{-1} D$  for T U D
      by force
    have ttt: 1 - t ≤ dist w u if cmod u = 1 for u
      using ⟨norm z ≤ t⟩ segment_bound1 [OF ⟨w ∈ closed_segment 0 z⟩]
    norm_triangle_ineq2 [of u w] that
      by (simp add: dist_norm norm_minus_commute)
    have  $\nexists b. \text{ball } b \ 1 \subseteq g^{-1} \text{cball } 0 \ 1$ 
    proof (rule *)
      show  $(\exists w \in (\bigcup m \in \text{Ints}. \bigcup n \in \{0 < ..\}. \{\text{Complex } m \ (\ln(n + \text{sqrt}(\text{real } n$ 
 $\wedge 2 - 1)) / \pi i\})) \cup$ 

```

```

      ( $\bigcup m \in \text{Ints}. \bigcup n \in \{0 < ..\}. \{ \text{Complex } m \ (-\ln(n + \sqrt{\text{real } n^2 - 1})) / \pi i \}$ ).  $w \in \text{ball } b \ 1$ ) for  $b$ 
    proof -
      obtain  $m$  where  $m: m \in \mathbb{Z} \mid \text{Re } b - m \leq 1/2$ 
      by (metis Ints_of_int abs_minus_commute of_int_round_abs_le)
      show ?thesis
      proof (cases  $0::\text{real Im } b$  rule: le_cases)
        case le
          then obtain  $n$  where  $0 < n$  and  $n: |\text{Im } b - \ln(n + \sqrt{(\text{real } n)^2 - 1})| / \pi i < 1/2$ 
          using Schottky_lemma2 [of  $\text{Im } b$ ] by blast
          have  $\text{dist } b (\text{Complex } m (\text{Im } b)) \leq 1/2$ 
          by (metis cancel_comm_monoid_add_class.diff_cancel cmod_eq_Re complex.sel(1) complex.sel(2) dist_norm m(2) minus_complex.code)
          moreover
            have  $\text{dist} (\text{Complex } m (\text{Im } b)) (\text{Complex } m (\ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi i)) < 1/2$ 
            using  $n$  by (simp add: complex_norm cmod_eq_Re complex_diff dist_norm del: Complex_eq)
            ultimately have  $\text{dist } b (\text{Complex } m (\ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi i)) < 1$ 
            by (simp add: dist_triangle_lt [of  $b$   $\text{Complex } m (\text{Im } b)$ ] dist_commute)
            with  $\text{le } m \langle 0 < n \rangle$  show ?thesis
            apply (rule_tac  $x = \text{Complex } m (\ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi i)$  in bexI)
            by (force simp del: Complex_eq greaterThan_0)+
          next
            case ge
              then obtain  $n$  where  $0 < n$  and  $n: |-\text{Im } b - \ln(n + \sqrt{(\text{real } n)^2 - 1})| / \pi i < 1/2$ 
              using Schottky_lemma2 [of  $-\text{Im } b$ ] by auto
              have  $\text{dist } b (\text{Complex } m (\text{Im } b)) \leq 1/2$ 
              by (metis cancel_comm_monoid_add_class.diff_cancel cmod_eq_Re complex.sel(1) complex.sel(2) dist_norm m(2) minus_complex.code)
              moreover
                have  $\text{dist} (\text{Complex } m (-\ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi i)) (\text{Complex } m (\text{Im } b))$ 
                 $= |-\text{Im } b - \ln(n + \sqrt{(\text{real } n)^2 - 1})| / \pi i$ 
                by (simp add: complex_norm dist_norm cmod_eq_Re complex_diff)
                ultimately have  $\text{dist } b (\text{Complex } m (-\ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi i)) < 1$ 
                using  $n$  by (simp add: dist_triangle_lt [of  $b$   $\text{Complex } m (\text{Im } b)$ ] dist_commute)
                with  $\text{ge } m \langle 0 < n \rangle$  show ?thesis
                by (rule_tac  $x = \text{Complex } m (-\ln(n + \sqrt{(\text{real } n)^2 - 1}) / \pi i)$  in bexI) auto
              qed
            qed
          show  $g \ v \notin (\bigcup m \in \text{Ints}. \bigcup n \in \{0 < ..\}. \{ \text{Complex } m (\ln(n + \sqrt{\text{real } n^2 - 1})) / \pi i \})$ 

```

```

2 - 1)) / pi)) ∪
  (∪ m ∈ Ints. ∪ n ∈ {0 < ..}. {Complex m (-ln(n + sqrt(real n ^ 2
- 1)) / pi)})
  if v ∈ cball 0 1 for v
  using not01 [OF that]
  by (force simp: g [OF that, symmetric] h [OF that, symmetric] dest!:
Schottky_lemma3 [of g v])
qed
then have 12: (1 - t) * cmod (deriv g w) / 12 < 1
  using Bloch_general [OF holg __ ttt, of 1] w by force
have g field_differentiable at w within cball 0 1
  using holg w by (simp add: holomorphic_on_def)
then have g field_differentiable at w within ball 0 1
  using ball_subset_cball field_differentiable_within_subset by blast
with w have der_gw: (g has_field_derivative deriv g w) (at w)
by (simp add: field_differentiable_within_open [of __ ball 0 1] field_differentiable_derivI)
with DERIV_unique [OF der_gw] g' w have deriv g w = g' w
by (metis open_ball at_within_open ball_subset_cball has_field_derivative_subset
subsetCE)
then show cmod (g' w) ≤ 12 / (1 - t)
  using g' 12 < t < 1 by (simp add: field_simps)
qed
then have cmod (g z - g 0) ≤ 12 / (1 - t) * cmod z
  using has_contour_integral_bound_linepath [OF int_g', of 12/(1 - t)] assms
  by simp
with <cmod z ≤ t> <t < 1> show ?thesis
  by (simp add: field_split_simps)
qed
have fz: f z = (1 + cos(of_real pi * h z)) / 2
  using h <z ∈ ball 0 1> by (auto simp: field_simps)
have cmod (f z) ≤ exp (cmod (complex_of_real pi * h z))
  by (simp add: fz mult.commute norm_cos_plus1_le)
also have ... ≤ exp (pi * exp (pi * (2 + 2 * r + 12 * t / (1 - t))))
proof (simp add: norm_mult)
  have cmod (g z - g 0) ≤ 12 * t / (1 - t)
    using norm_g 12 <t < 1> by (simp add: norm_mult)
  then have cmod (g z) - cmod (g 0) ≤ 12 * t / (1 - t)
    using norm_triangle_ineq2 order_trans by blast
  then have *: cmod (g z) ≤ 2 + 2 * r + 12 * t / (1 - t)
    using g0_2_f0 norm_ge_zero [of f 0] nof0
    by linarith
  have cmod (h z) ≤ exp (cmod (complex_of_real pi * g z))
    using <z ∈ ball 0 1> by (simp add: g norm_cos_le)
  also have ... ≤ exp (pi * (2 + 2 * r + 12 * t / (1 - t)))
    using <t < 1> nof0 * by (simp add: norm_mult)
  finally show cmod (h z) ≤ exp (pi * (2 + 2 * r + 12 * t / (1 - t))) .
qed
finally show ?thesis .
qed

```


6.2 The Little Picard Theorem

theorem *Landau_Picard*:

obtains R

where $\bigwedge z. 0 < R\ z$

$\bigwedge f. \llbracket f \text{ holomorphic_on } \text{cball } 0\ (R(f\ 0));$

$\bigwedge z. \text{norm } z \leq R(f\ 0) \implies f\ z \neq 0 \wedge f\ z \neq 1 \rrbracket \implies \text{norm}(deriv\ f\ 0)$

< 1

proof –

define R **where** $R \equiv \lambda z. 3 * \exp(\pi * \exp(\pi * (2 + 2 * \text{cmod } z + 12)))$

show *?thesis*

proof

show $Rpos: \bigwedge z. 0 < R\ z$

by (*auto simp: R_def*)

show $\text{norm}(deriv\ f\ 0) < 1$

if $holf: f \text{ holomorphic_on } \text{cball } 0\ (R(f\ 0))$

and $Rf: \bigwedge z. \text{norm } z \leq R(f\ 0) \implies f\ z \neq 0 \wedge f\ z \neq 1$ **for** f

proof –

let $?r = R(f\ 0)$

define g **where** $g \equiv f \circ (\lambda z. \text{of_real } ?r * z)$

have $0 < ?r$

using $Rpos$ **by** *blast*

have $holg: g \text{ holomorphic_on } \text{cball } 0\ 1$

unfolding g_def

proof (*intro holomorphic_intros holomorphic_on_compose holomorphic_on_subset [OF holf]*)

show $(*) (\text{complex_of_real } (R(f\ 0)))\ ' \text{cball } 0\ 1 \subseteq \text{cball } 0\ (R(f\ 0))$

using $Rpos$ **by** (*auto simp: less_imp_le norm_mult*)

qed

have $*$: $\text{norm}(g\ z) \leq \exp(\pi * \exp(\pi * (2 + 2 * \text{norm } (f\ 0) + 12 * t / (1 - t))))$

if $0 < t < 1$ $\text{norm } z \leq t$ **for** $t\ z$

proof (*rule Schottky [OF holg]*)

show $\text{cmod } (g\ 0) \leq \text{cmod } (f\ 0)$

by (*simp add: g_def*)

show $\bigwedge z. z \in \text{cball } 0\ 1 \implies \neg (g\ z = 0 \vee g\ z = 1)$

using $Rpos$ **by** (*simp add: g_def less_imp_le norm_mult Rf*)

qed (*auto simp: that*)

have $C1: g \text{ holomorphic_on } \text{ball } 0\ (1/2)$

by (*rule holomorphic_on_subset [OF holg]*) *auto*

have $C2: \text{continuous_on } (\text{cball } 0\ (1/2))\ g$

by (*meson cball_divide_subset_numeral holg holomorphic_on_imp_continuous_on holomorphic_on_subset*)

have $C3: \text{cmod } (g\ z) \leq R(f\ 0) / 3$ **if** $\text{cmod } (0 - z) = 1/2$ **for** z

proof –

have $\text{norm}(g\ z) \leq \exp(\pi * \exp(\pi * (2 + 2 * \text{norm } (f\ 0) + 12)))$

using $*$ [*of 1/2*] **that** **by** *simp*

also have $\dots = ?r / 3$

by (*simp add: R_def*)

finally show *?thesis* .

```

qed
then have cmod_g'_le: cmod (deriv g 0) * 3 ≤ R (f 0) * 2
  using Cauchy_inequality [OF C1 C2_ C3, of 1] by simp
have holf': f holomorphic_on ball 0 (R(f 0))
  by (rule holomorphic_on_subset [OF holf]) auto
then have fd0: f field_differentiable at 0
  by (rule holomorphic_on_imp_differentiable_at [OF _ open_ball])
    (auto simp: Rpos [of f 0])
have g_eq: deriv g 0 = of_real ?r * deriv f 0
  unfolding g_def
  by (metis DERIV_imp_deriv DERIV_chain DERIV_cmult_Id fd0 field_differentiable_derivI
mult.commute mult_zero_right)
show ?thesis
  using cmod_g'_le Rpos [of f 0] by (simp add: g_eq norm_mult)
qed
qed
qed

lemma little_Picard_01:
  assumes holf: f holomorphic_on UNIV and f01:  $\bigwedge z. f z \neq 0 \wedge f z \neq 1$ 
  obtains c where f = ( $\lambda x. c$ )
proof -
  obtain R
    where Rpos:  $\bigwedge z. 0 < R z$ 
    and R:  $\bigwedge h. \llbracket h \text{ holomorphic\_on cball } 0 (R(h\ 0));$ 
       $\bigwedge z. \text{norm } z \leq R(h\ 0) \implies h\ z \neq 0 \wedge h\ z \neq 1 \rrbracket \implies \text{norm}(\text{deriv}$ 
 $h\ 0) < 1$ 
    using Landau_Picard by metis
  have conf: continuous_on UNIV f
    by (simp add: holf holomorphic_on_imp_continuous_on)
  show ?thesis
  proof (cases  $\forall x. \text{deriv } f\ x = 0$ )
    case True
      have (f has_field_derivative 0) (at x) for x
        by (metis True UNIV_I holf holomorphic_derivI open_UNIV)
      then obtain c where  $\bigwedge x. f(x) = c$ 
        by (meson UNIV_I DERIV_zero_connected_constant [OF connected_UNIV
open_UNIV finite.emptyI conf])
      then show ?thesis
        using that by auto
    next
      case False
      then obtain w where w: deriv f w  $\neq 0$  by auto
      define fw where fw  $\equiv (f \circ (\lambda z. w + z / \text{deriv } f\ w))$ 
      have norm_let1: norm(deriv fw 0) < 1
      proof (rule R)
        show fw holomorphic_on cball 0 (R (fw 0))
          unfolding fw_def
          by (intro holomorphic_intros holomorphic_on_compose w holomorphic_on_subset

```

```

[OF holf] subset_UNIV)
  show  $fw\ z \neq 0 \wedge fw\ z \neq 1$  if  $cmod\ z \leq R\ (fw\ 0)$  for  $z$ 
    using f01 by (simp add: fw_def)
  qed
  have (fw has_field_derivative deriv f w * inverse (deriv f w)) (at 0)
    unfolding fw_def
    apply (intro DERIV_chain derivative_eq_intros w)+
    using holf holomorphic_derivI by (force simp: field_simps)+
  then show ?thesis
    using norm_let1 w by (simp add: DERIV_imp_deriv)
  qed
qed

```

```

theorem little_Picard:
  assumes holf:  $f$  holomorphic_on UNIV
    and  $a \neq b$  range  $f \cap \{a, b\} = \{\}$ 
    obtains  $c$  where  $f = (\lambda x. c)$ 
proof -
  let ?g =  $\lambda x. 1/(b-a)*(f\ x - b) + 1$ 
  obtain  $c$  where ?g =  $(\lambda x. c)$ 
  proof (rule little_Picard_01)
    show ?g holomorphic_on UNIV
      by (intro holomorphic_intros holf)
    show  $\bigwedge z. ?g\ z \neq 0 \wedge ?g\ z \neq 1$ 
      using assms by (auto simp: field_simps)
  qed auto
  then have ?g  $x = c$  for  $x$ 
    by meson
  then have  $f\ x = c * (b-a) + a$  for  $x$ 
    using assms by (auto simp: field_simps)
  then show ?thesis
    using that by blast
qed

```

A couple of little applications of Little Picard

```

lemma holomorphic_periodic_fixpoint:
  assumes holf:  $f$  holomorphic_on UNIV
    and  $p \neq 0$  and per:  $\bigwedge z. f(z + p) = f\ z$ 
    obtains  $x$  where  $f\ x = x$ 
proof -
  have False if non:  $\bigwedge x. f\ x \neq x$ 
  proof -
    obtain  $c$  where  $(\lambda z. f\ z - z) = (\lambda z. c)$ 
    proof (rule little_Picard)
      show  $(\lambda z. f\ z - z)$  holomorphic_on UNIV
        by (simp add: holf holomorphic_on_diff)
      show range  $(\lambda z. f\ z - z) \cap \{p, 0\} = \{\}$ 
        using assms non by auto (metis add.commute diff_eq_eq)
    qed
  qed

```

```

    qed (auto simp: assms)
  with per show False
    by (metis add.commute add_cancel_left_left ‹p ≠ 0› diff_add_cancel)
  qed
  then show ?thesis
    using that by blast
  qed

lemma holomorphic_involution_point:
  assumes holfU: f holomorphic_on UNIV and non:  $\bigwedge a. f \neq (\lambda x. a + x)$ 
  obtains x where f(f x) = x
proof -
  { assume non_ff [simp]:  $\bigwedge x. f(f x) \neq x$ 
    then have non_fp [simp]: f z ≠ z for z
      by metis
    have holf: f holomorphic_on X for X
      using assms holomorphic_on_subset by blast
    obtain c where c:  $(\lambda x. (f(f x) - x)/(f x - x)) = (\lambda x. c)$ 
    proof (rule little_Picard_01)
      show  $(\lambda x. (f(f x) - x)/(f x - x))$  holomorphic_on UNIV
        using non_fp
      by (intro holomorphic_intros holf holomorphic_on_compose [unfolded o_def,
        OF holf]) auto
    qed auto
    then obtain c ≠ 0 c ≠ 1
      by (metis (no_types, opaque_lifting) non_ff diff_zero divide_eq_0_iff right_inverse_eq)
    have eq: f(f x) - c * f x = x * (1 - c) for x
      using fun_cong [OF c, of x] by (simp add: field_simps)
    have df_times_dff: deriv f z * (deriv f (f z) - c) = 1 * (1 - c) for z
      proof (rule DERIV_unique)
        show (( $\lambda x. f(f x) - c * f x$ ) has_field_derivative
          deriv f z * (deriv f (f z) - c)) (at z)
          by (rule derivative_eq_intros holomorphic_derivI [OF holfU]
            DERIV_chain [unfolded o_def, where f=f and g=f] | simp add:
            algebra_simps)+
        show (( $\lambda x. f(f x) - c * f x$ ) has_field_derivative 1 * (1 - c)) (at z)
          by (simp add: eq mult_commute_abs)
      qed
    { fix z::complex
      obtain k where k: deriv f ∘ f = ( $\lambda x. k$ )
      proof (rule little_Picard)
        show (deriv f ∘ f) holomorphic_on UNIV
          by (meson holfU holomorphic_deriv holomorphic_on_compose holomor-
            phic_on_subset open_UNIV subset_UNIV)
        obtain deriv f (f x) ≠ 0 deriv f (f x) ≠ c for x
          using df_times_dff ‹c ≠ 1› eq_iff_diff_eq_0
          by (metis lambda_one mult_zero_left mult_zero_right)
        then show range (deriv f ∘ f) ∩ {0,c} = {}
      }
    }
  }

```

```

    by force
  qed (use ⟨c ≠ 0⟩ in auto)
  have ¬ f constant_on UNIV
    by (meson UNIV_I non_ff constant_on_def)
  with holf open_mapping_thm have open(range f)
    by blast
  obtain l where l:  $\bigwedge x. f\ x - k * x = l$ 
  proof (rule DERIV_zero_connected_constant [of UNIV { }  $\lambda x. f\ x - k * x$ ],
    simp_all)
    have deriv f w - k = 0 for w
      proof (rule analytic_continuation [OF open_UNIV connected_UNIV
subset_UNIV, of  $\lambda z. deriv\ f\ z - k\ f\ z\ range\ f\ w$ ])
        show  $(\lambda z. deriv\ f\ z - k)$  holomorphic_on UNIV
          by (intro holomorphic_intros holf open_UNIV)
        show f z islimpt range f
          by (metis (no_types, lifting) IntI UNIV_I ⟨open (range f)⟩ im-
age_eqI inf.absorb_iff2 inf_aci(1) islimpt_UNIV islimpt_eq_acc_point open_Int
top_greatest)
        show  $\bigwedge z. z \in range\ f \implies deriv\ f\ z - k = 0$ 
          by (metis comp_def diff_self image_iff k)
      qed auto
    moreover
    have  $((\lambda x. f\ x - k * x)$  has_field_derivative deriv f x - k) (at x) for x
      by (metis DERIV_cmult_Id Deriv.field_differentiable_diff UNIV_I
field_differentiable_derivI holf holomorphic_on_def)
    ultimately
    show  $\forall x. ((\lambda x. f\ x - k * x)$  has_field_derivative 0) (at x)
      by auto
    show continuous_on UNIV  $(\lambda x. f\ x - k * x)$ 
      by (simp add: continuous_on_diff holf holomorphic_on_imp_continuous_on)
  qed (auto simp: connected_UNIV)
  have False
  proof (cases k=1)
    case True
      then have  $\exists x. k * x + l \neq a + x$  for a
        using l non [of a] ext [of f (+) a]
        by (metis add.commute diff_eq_eq)
      with True show ?thesis by auto
    next
    case False
      have  $\bigwedge x. (1 - k) * x \neq f\ 0$ 
        using l [of 0]
        by (simp add: algebra_simps) (metis diff_add_cancel l mult.commute
non_fp)
      then show False
        by (metis False eq_iff diff_eq_0 mult.commute nonzero_mult_div_cancel_right
times_divide_eq_right)
  qed
}

```

```

}
then show thesis
  using that by blast
qed

```

6.3 The Arzelà–Ascoli theorem

lemma *subsequence_diagonalization_lemma*:

```

fixes  $P :: \text{nat} \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow \text{bool}$ 
assumes  $\text{sub}: \bigwedge i r. \exists k. \text{strict\_mono } (k :: \text{nat} \Rightarrow \text{nat}) \wedge P\ i\ (r \circ k)$ 
and  $P\_P: \bigwedge i r::\text{nat} \Rightarrow 'a. \bigwedge k1\ k2\ N. \llbracket P\ i\ (r \circ k1); \bigwedge j. N \leq j \implies \exists j'. j \leq j' \wedge k2\ j = k1\ j' \rrbracket \implies P\ i\ (r \circ k2)$ 
obtains  $k$  where  $\text{strict\_mono } (k :: \text{nat} \Rightarrow \text{nat}) \wedge \bigwedge i. P\ i\ (r \circ k)$ 
proof –
obtain  $kk$  where  $\bigwedge i r. \text{strict\_mono } (kk\ i\ r :: \text{nat} \Rightarrow \text{nat}) \wedge P\ i\ (r \circ (kk\ i\ r))$ 
using  $\text{sub}$  by metis
then have  $\text{sub\_kk}: \bigwedge i r. \text{strict\_mono } (kk\ i\ r)$  and  $P\_kk: \bigwedge i r. P\ i\ (r \circ (kk\ i\ r))$ 
by auto
define  $rr$  where  $rr \equiv \text{rec\_nat } (kk\ 0\ r) (\lambda n x. x \circ kk\ (\text{Suc } n) (r \circ x))$ 
then have  $[\text{simp}]: rr\ 0 = kk\ 0\ r \wedge \bigwedge n. rr(\text{Suc } n) = rr\ n \circ kk\ (\text{Suc } n) (r \circ rr\ n)$ 
by auto
show thesis
proof
have  $\text{sub\_rr}: \text{strict\_mono } (rr\ i)$  for  $i$ 
using  $\text{sub\_kk}$  by (induction  $i$ ) (auto simp: strict_mono_def o_def)
have  $P\_rr: P\ i\ (r \circ rr\ i)$  for  $i$ 
using  $P\_kk$  by (induction  $i$ ) (auto simp: o_def)
have  $i \leq i+d \implies rr\ i\ n \leq rr\ (i+d)\ n$  for  $d\ i\ n$ 
proof (induction  $d$ )
case 0 then show ?case
by simp
next
case ( $\text{Suc } d$ ) then show ?case
using  $\text{seq\_suble } [OF\ \text{sub\_kk}] \text{strict\_mono\_less\_eq } [OF\ \text{sub\_rr}]$ 
by (simp add: order_subst1)
qed
then have  $\bigwedge i\ j\ n. i \leq j \implies rr\ i\ n \leq rr\ j\ n$ 
by (metis le_iff_add)
show  $\text{strict\_mono } (\lambda n. rr\ n\ n)$ 
unfolding  $\text{strict\_mono\_Suc\_iff}$ 
by (simp add: Suc_le_lessD strict_monoD strict_mono_imp_increasing sub_kk sub_rr)
have  $\exists j. i \leq j \wedge rr\ (n+d)\ i = rr\ n\ j$  for  $d\ n\ i$ 
proof (induction  $d$  arbitrary: i)
case ( $\text{Suc } d$ )
then show ?case
using  $\text{seq\_suble } [OF\ \text{sub\_kk}]$  by simp (meson order_trans)

```

```

qed auto
then have  $\bigwedge m\ n\ i. n \leq m \implies \exists j. i \leq j \wedge rr\ m\ i = rr\ n\ j$ 
  by (metis le_iff_add)
then show  $P\ i\ (r \circ (\lambda n. rr\ n\ n))$  for  $i$ 
  by (meson P_rr P_P)
qed
qed

lemma function_convergent_subsequence:
  fixes  $f :: [nat, 'a] \Rightarrow 'b :: \{real\_normed\_vector, heine\_borel\}$ 
  assumes countable  $S$  and  $M: \bigwedge n::nat. \bigwedge x. x \in S \implies norm(f\ n\ x) \leq M$ 
  obtains  $k$  where  $strict\_mono\ (k::nat \Rightarrow nat) \bigwedge x. x \in S \implies \exists l. (\lambda n. f\ (k\ n)\ x)$ 
 $\longrightarrow l$ 
proof (cases  $S = \{\}$ )
  case True
  then show ?thesis
    using strict_mono_id that by fastforce
next
  case False
  with  $\langle countable\ S \rangle$  obtain  $\sigma :: nat \Rightarrow 'a$  where  $\sigma: S = range\ \sigma$ 
    using uncountable_def by blast
  obtain  $k$  where  $strict\_mono\ k$  and  $k: \bigwedge i. \exists l. (\lambda n. (f \circ k)\ n\ (\sigma\ i)) \longrightarrow l$ 
  proof (rule subsequence_diagonalization_lemma
    [of  $\lambda i\ r. \exists l. ((\lambda n. (f \circ r)\ n\ (\sigma\ i)) \longrightarrow l)$  sequentially id])
    show  $\exists k::nat \Rightarrow nat. strict\_mono\ k \wedge (\exists l. (\lambda n. (f \circ (r \circ k))\ n\ (\sigma\ i)) \longrightarrow l)$ 
  for  $i\ r$ 
  proof -
    have  $f\ (r\ n)\ (\sigma\ i) \in cball\ 0\ M$  for  $n$ 
      by (simp add:  $\sigma\ M$ )
    then show ?thesis
      using compact_def [of  $cball\ (0::'b)\ M$ ] by (force simp: o_def)
  qed
  show  $\exists l. (\lambda n. (f \circ (r \circ k2))\ n\ (\sigma\ i)) \longrightarrow l$ 
  if  $\exists l. (\lambda n. (f \circ (r \circ k1))\ n\ (\sigma\ i)) \longrightarrow l \bigwedge j. N \leq j \implies \exists j' \geq j. k2\ j = k1\ j'$ 
  for  $i\ N$  and  $r\ k1\ k2 :: nat \Rightarrow nat$ 
  using that
  by (simp add: lim_sequentially) (metis (no_types, opaque_lifting) le_cases
order_trans)
qed auto
with  $\sigma$  that show ?thesis
  by force
qed

```

theorem *Arzela_Ascoli:*

```

fixes  $\mathcal{F} :: [nat, 'a::euclidean\_space] \Rightarrow 'b::\{real\_normed\_vector, heine\_borel\}$ 
assumes compact  $S$ 
  and  $M: \bigwedge n\ x. x \in S \implies norm(\mathcal{F}\ n\ x) \leq M$ 
  and equicont:

```

```


$$\bigwedge x e. \llbracket x \in S; 0 < e \rrbracket$$


$$\implies \exists d. 0 < d \wedge (\forall n y. y \in S \wedge \text{norm}(x - y) < d \longrightarrow \text{norm}(\mathcal{F} n x - \mathcal{F} n y) < e)$$

obtains  $g k$  where continuous_on  $S$   $g$  strict_mono  $(k :: \text{nat} \Rightarrow \text{nat})$ 

$$\bigwedge e. 0 < e \implies \exists N. \forall n x. n \geq N \wedge x \in S \longrightarrow \text{norm}(\mathcal{F}(k n) x - g x) < e$$

proof –
  have UEQ:  $\bigwedge e. 0 < e \implies \exists d. 0 < d \wedge (\forall n. \forall x \in S. \forall x' \in S. \text{dist } x' x < d \longrightarrow \text{dist } (\mathcal{F} n x') (\mathcal{F} n x) < e)$ 
  apply (rule compact_uniformly_equicontinuous [OF  $\langle \text{compact } S \rangle$ , of range  $\mathcal{F}$ ])
  using equicont by (force simp: dist_commute dist_norm)+
  have continuous_on  $S$   $g$ 
    if  $\bigwedge e. 0 < e \implies \exists N. \forall n x. n \geq N \wedge x \in S \longrightarrow \text{norm}(\mathcal{F}(r n) x - g x) < e$ 
    for  $g :: 'a \Rightarrow 'b$  and  $r :: \text{nat} \Rightarrow \text{nat}$ 
  proof (rule uniform_limit_theorem [of  $\_ \mathcal{F} \circ r$ ])
    have continuous_on  $S$   $(\mathcal{F} (r n))$  for  $n$ 
    using UEQ by (force simp: continuous_on_iff)
    then show  $\forall_F n$  in sequentially. continuous_on  $S$   $((\mathcal{F} \circ r) n)$ 
    by (simp add: eventually_sequentially)
    show uniform_limit  $S$   $(\mathcal{F} \circ r) g$  sequentially
    using that by (metis (mono_tags, opaque_lifting) comp_apply dist_norm uniform_limit_sequentially_iff)
  qed auto
  moreover
  obtain  $R$  where countable  $R$   $R \subseteq S$  and  $SR: S \subseteq \text{closure } R$ 
  by (metis separable that)
  obtain  $k$  where strict_mono  $k$  and  $k: \bigwedge x. x \in R \implies \exists l. (\lambda n. \mathcal{F} (k n) x) \longrightarrow l$ 
  using  $\langle R \subseteq S \rangle$  by (force intro: function_convergent_subsequence [OF  $\langle \text{countable } R \rangle M$ ])
  then have Cauchy: Cauchy  $((\lambda n. \mathcal{F} (k n) x))$  if  $x \in R$  for  $x$ 
  using convergent_eq_Cauchy that by blast
  have  $\exists N. \forall m n x. N \leq m \wedge N \leq n \wedge x \in S \longrightarrow \text{dist } ((\mathcal{F} \circ k) m x) ((\mathcal{F} \circ k) n x) < e$ 
  if  $0 < e$  for  $e$ 
  proof –
    obtain  $d$  where  $0 < d$ 
    and  $d: \bigwedge n. \forall x \in S. \forall x' \in S. \text{dist } x' x < d \longrightarrow \text{dist } (\mathcal{F} n x') (\mathcal{F} n x) < e/3$ 
    by (metis UEQ  $\langle 0 < e \rangle$  divide_pos_pos zero_less_numeral)
    obtain  $T$  where  $T \subseteq R$  and finite  $T$  and  $T: S \subseteq (\bigcup_{c \in T. \text{ball } c d)$ 
    proof (rule compactE_image [OF  $\langle \text{compact } S \rangle$ , of  $R (\lambda x. \text{ball } x d)$ ])
    have closure  $R \subseteq (\bigcup_{c \in R. \text{ball } c d)$ 
    using  $\langle 0 < d \rangle$  by (auto simp: closure_approachable)
    with  $SR$  show  $S \subseteq (\bigcup_{c \in R. \text{ball } c d)$ 
    by auto
  qed auto
  have  $\exists M. \forall m \geq M. \forall n \geq M. \text{dist } (\mathcal{F} (k m) x) (\mathcal{F} (k n) x) < e/3$  if  $x \in R$  for  $x$ 
  using Cauchy  $\langle 0 < e \rangle$  that unfolding Cauchy_def
  by (metis less_divide_eq_numeral1  $(1)$  mult_zero_left)

```



```

then obtain  $MF$  where  $MF: \bigwedge x m n. \llbracket x \in R; m \geq MF\ x; n \geq MF\ x \rrbracket \implies$ 
 $norm\ (\mathcal{F}\ (k\ m)\ x - \mathcal{F}\ (k\ n)\ x) < e/3$ 
using  $dist\_norm$  by  $metis$ 
have  $dist\ ((\mathcal{F} \circ k)\ m\ x)\ ((\mathcal{F} \circ k)\ n\ x) < e$ 
if  $m: Max\ (MF\ 'T) \leq m$  and  $n: Max\ (MF\ 'T) \leq n$   $x \in S$  for  $m\ n\ x$ 
proof -
obtain  $t$  where  $t \in T$  and  $t: x \in ball\ t\ d$ 
using  $\langle x \in S \rangle\ T$  by  $auto$ 
have  $norm(\mathcal{F}\ (k\ m)\ t - \mathcal{F}\ (k\ m)\ x) < e / 3$ 
by ( $metis\ \langle R \subseteq S \rangle\ \langle T \subseteq R \rangle\ \langle t \in T \rangle\ d\ dist\_norm\ mem\_ball\ subset\_iff\ t\ \langle x \in S \rangle$ )
moreover
have  $norm(\mathcal{F}\ (k\ n)\ t - \mathcal{F}\ (k\ n)\ x) < e / 3$ 
by ( $metis\ \langle R \subseteq S \rangle\ \langle T \subseteq R \rangle\ \langle t \in T \rangle\ subsetD\ d\ dist\_norm\ mem\_ball\ t\ \langle x \in S \rangle$ )
moreover
have  $norm(\mathcal{F}\ (k\ m)\ t - \mathcal{F}\ (k\ n)\ t) < e / 3$ 
proof ( $rule\ MF$ )
show  $t \in R$ 
using  $\langle T \subseteq R \rangle\ \langle t \in T \rangle$  by  $blast$ 
show  $MF\ t \leq m\ MF\ t \leq n$ 
by ( $meson\ Max\_ge\ \langle finite\ T \rangle\ \langle t \in T \rangle\ finite\_imageI\ imageI\ le\_trans\ m$ 
 $n)+$ 
qed
ultimately
show  $?thesis$ 
unfolding  $dist\_norm\ [symmetric]\ o\_def$ 
by ( $metis\ dist\_triangle\_third\ dist\_commute$ )
qed
then show  $?thesis$ 
by  $force$ 
qed
then obtain  $g$  where  $\forall e > 0. \exists N. \forall n\ x. N \leq n \wedge x \in S \longrightarrow norm\ ((\mathcal{F} \circ k)\ n\ x - g\ x) < e$ 
using  $uniformly\_convergent\_eq\_cauchy$  [ $of\ \lambda x. x \in S\ \mathcal{F} \circ k$ ] by ( $auto\ simp\ add: dist\_norm$ )
ultimately show  $thesis$ 
by ( $metis\ \langle strict\_mono\ k \rangle\ comp\_apply\ that$ )
qed

```

6.3.1 Montel's theorem

a sequence of holomorphic functions uniformly bounded on compact subsets of an open set S has a subsequence that converges to a holomorphic function, and converges *uniformly* on compact subsets of S .

theorem *Montel*:

fixes $\mathcal{F} :: [nat, complex] \Rightarrow complex$

assumes $open\ S$

and $\mathcal{H}: \bigwedge h. h \in \mathcal{H} \implies h\ holomorphic_on\ S$

```

    and bounded:  $\bigwedge K. \llbracket \text{compact } K; K \subseteq S \rrbracket \implies \exists B. \forall h \in \mathcal{H}. \forall z \in K. \text{norm}(h z) \leq B$ 
    and rng_f:  $\text{range } \mathcal{F} \subseteq \mathcal{H}$ 
    obtains  $g \ r$ 
    where  $g$  holomorphic_on  $S$  strict_mono  $(r :: \text{nat} \Rightarrow \text{nat})$ 
       $\bigwedge x. x \in S \implies ((\lambda n. \mathcal{F} (r \ n) \ x) \longrightarrow g \ x)$  sequentially
       $\bigwedge K. \llbracket \text{compact } K; K \subseteq S \rrbracket \implies \text{uniform\_limit } K \ (\mathcal{F} \circ r) \ g$  sequentially

  proof -
    obtain  $K$  where comK:  $\bigwedge n. \text{compact}(K \ n)$  and KS:  $\bigwedge n :: \text{nat}. K \ n \subseteq S$ 
      and subK:  $\bigwedge X. \llbracket \text{compact } X; X \subseteq S \rrbracket \implies \exists N. \forall n \geq N. X \subseteq K \ n$ 
    using open_Union_compact_subsets [OF  $\langle \text{open } S \rangle$ ] by metis
    then have  $\bigwedge i. \exists B. \forall h \in \mathcal{H}. \forall z \in K \ i. \text{norm}(h \ z) \leq B$ 
      by (simp add: bounded)
    then obtain  $B$  where  $B$ :  $\bigwedge i \ h \ z. \llbracket h \in \mathcal{H}; z \in K \ i \rrbracket \implies \text{norm}(h \ z) \leq B \ i$ 
      by metis
    have *:  $\exists r \ g. \text{strict\_mono } (r :: \text{nat} \Rightarrow \text{nat}) \wedge (\forall e > 0. \exists N. \forall n \geq N. \forall x \in K \ i. \text{norm}((\mathcal{F} \circ r) \ n \ x - g \ x) < e)$ 
      if  $\bigwedge n. \mathcal{F} \ n \in \mathcal{H}$  for  $\mathcal{F} \ i$ 
    proof -
      obtain  $g \ k$  where continuous_on  $(K \ i)$   $g$  strict_mono  $(k :: \text{nat} \Rightarrow \text{nat})$ 
         $\bigwedge e. 0 < e \implies \exists N. \forall n \geq N. \forall x \in K \ i. \text{norm}(\mathcal{F}(k \ n) \ x - g \ x) < e$ 
      proof (rule Arzela_Ascoli [of  $K \ i \ \mathcal{F} \ B \ i$ ])
        show  $\exists d > 0. \forall n \ y. y \in K \ i \wedge \text{cmod}(z - y) < d \longrightarrow \text{cmod}(\mathcal{F} \ n \ z - \mathcal{F} \ n \ y) < e$ 
          if  $z: z \in K \ i$  and  $0 < e$  for  $z \ e$ 
      proof -
        obtain  $r$  where  $0 < r$  and  $r$ :  $\text{cball } z \ r \subseteq S$ 
          using  $z \ KS$  [of  $i$ ]  $\langle \text{open } S \rangle$  by (force simp: open_contains_cball)
        have  $\text{cball } z \ (2/3 * r) \subseteq \text{cball } z \ r$ 
          using  $\langle 0 < r \rangle$  by (simp add: cball_subset_cball_iff)
        then have  $z23S$ :  $\text{cball } z \ (2/3 * r) \subseteq S$ 
          using  $r$  by blast
        obtain  $M$  where  $0 < M$  and  $M$ :  $\bigwedge n \ w. \text{dist } z \ w \leq 2/3 * r \implies \text{norm}(\mathcal{F} \ n \ w) \leq M$ 
      proof -
        obtain  $N$  where  $N$ :  $\forall n \geq N. \text{cball } z \ (2/3 * r) \subseteq K \ n$ 
          using subK compact_cball [of  $z \ (2/3 * r)$ ]  $z23S$  by force
        have  $\text{cmod}(\mathcal{F} \ n \ w) \leq |B \ N| + 1$  if  $\text{dist } z \ w \leq 2/3 * r$  for  $n \ w$ 
      proof -
        have  $w \in K \ N$ 
          using  $N$  mem_cball that by blast
        then have  $\text{cmod}(\mathcal{F} \ n \ w) \leq B \ N$ 
          using  $B \ \langle \bigwedge n. \mathcal{F} \ n \in \mathcal{H} \rangle$  by blast
        also have  $\dots \leq |B \ N| + 1$ 
          by simp
        finally show ?thesis .
      qed
    then show ?thesis
  end

```

```

    by (rule_tac M=|B N| + 1 in that) auto
  qed
  have cmod (F n z - F n y) < e
    if y ∈ K i and y_near_z: cmod (z - y) < r/3 cmod (z - y) < e * r
  / (6 * M)
    for n y
  proof -
    have ((λw. F n w / (w - ξ)) has_contour_integral
      (2 * pi) * i * winding_number (circlepath z (2/3 * r)) ξ * F n ξ)
      (circlepath z (2/3 * r))
    if dist ξ z < (2/3 * r) for ξ
  proof (rule Cauchy_integral_formula_convex_simple)
    have F n holomorphic_on S
      by (simp add: H ⟨Λ n. F n ∈ H⟩)
    with z23S show F n holomorphic_on cball z (2/3 * r)
      using holomorphic_on_subset by blast
  qed (use that ⟨0 < r⟩ in ⟨auto simp: dist_commute⟩)
  then have *: ((λw. F n w / (w - ξ)) has_contour_integral (2 * pi) * i
    * F n ξ)
    (circlepath z (2/3 * r))
    if dist ξ z < (2/3 * r) for ξ
    using that by (simp add: winding_number_circlepath dist_norm)
  have y: ((λw. F n w / (w - y)) has_contour_integral (2 * pi) * i * F
n y)
    (circlepath z (2/3 * r))
  proof (rule *)
    show dist y z < 2/3 * r
      using that ⟨0 < r⟩ by (simp only: dist_norm norm_minus_commute)
  qed
  have z: ((λw. F n w / (w - z)) has_contour_integral (2 * pi) * i * F n
z)
    (circlepath z (2/3 * r))
    using ⟨0 < r⟩ by (force intro!: *)
  have le_er: cmod (F n x / (x - y) - F n x / (x - z)) ≤ e / r
    if cmod (x - z) = r/3 + r/3 for x
  proof -
    have ¬ (cmod (x - y) < r/3)
      using y_near_z(1) that ⟨M > 0⟩ ⟨r > 0⟩
    by (metis (full_types) norm_diff_triangle_less norm_minus_commute
order_less_irrefl)
    then have r4_le_xy: r/4 ≤ cmod (x - y)
      using ⟨r > 0⟩ by simp
    then have neg: x ≠ y x ≠ z
      using that ⟨r > 0⟩ by (auto simp: field_split_simps norm_minus_commute)
    have leM: cmod (F n x) ≤ M
      by (simp add: M dist_commute dist_norm that)
    have cmod (F n x / (x - y) - F n x / (x - z)) = cmod (F n x) *
cmod (1 / (x - y) - 1 / (x - z))
      by (metis (no_types, lifting) divide_inverse mult.left_neutral norm_mult

```

```

right_diff_distrib')
  also have ... = cmod (F n x) * cmod ((y - z) / ((x - y) * (x - z)))
    using neq by (simp add: field_split_simps)
  also have ... = cmod (F n x) * (cmod (y - z) / (cmod(x - y) * (2/3
* r)))
    by (simp add: norm_mult norm_divide that)
  also have ... ≤ M * (cmod (y - z) / (cmod(x - y) * (2/3 * r)))
    using ⟨r > 0⟩ ⟨M > 0⟩ by (intro mult_mono [OF leM]) auto
  also have ... < M * ((e * r / (6 * M)) / (cmod(x - y) * (2/3 * r)))
    unfolding mult_less_cancel_left
    using y_near_z(2) ⟨M > 0⟩ ⟨r > 0⟩ neq
    by (simp add: field_simps mult_less_0_iff norm_minus_commute)
  also have ... ≤ e/r
    using ⟨e > 0⟩ ⟨r > 0⟩ r4_le_xy by (simp add: field_split_simps)
  finally show ?thesis by simp
qed
have (2 * pi) * cmod (F n y - F n z) = cmod ((2 * pi) * i * F n y -
(2 * pi) * i * F n z)
  by (simp add: right_diff_distrib [symmetric] norm_mult)
  also have cmod ((2 * pi) * i * F n y - (2 * pi) * i * F n z) ≤ e / r *
(2 * pi * (2/3 * r))

proof (rule has_contour_integral_bound_circlepath [OF has_contour_integral_diff
[OF y z]])
  show ∧x. cmod (x - z) = 2/3 * r ⟹ cmod (F n x / (x - y) - F n
x / (x - z)) ≤ e / r
    using le_er by auto
  qed (use ⟨e > 0⟩ ⟨r > 0⟩ in auto)
  also have ... = (2 * pi) * e * ((2/3))
    using ⟨r > 0⟩ by (simp add: field_split_simps)
  finally have cmod (F n y - F n z) ≤ e * (2/3)
    by simp
  also have ... < e
    using ⟨e > 0⟩ by simp
  finally show ?thesis by (simp add: norm_minus_commute)
qed
then show ?thesis
  apply (rule_tac x=min (r/3) ((e * r)/(6 * M)) in exI)
  using ⟨0 < e⟩ ⟨0 < r⟩ ⟨0 < M⟩ by simp
qed
show ∧n x. x ∈ K i ⟹ cmod (F n x) ≤ B i
  using B ⟨∧n. F n ∈ H⟩ by blast
next
  fix g :: complex ⇒ complex and k :: nat ⇒ nat
  assume *: ∧(g::complex⇒complex) (k::nat⇒nat). continuous_on (K i) g ⟹
    strict_mono k ⟹
    (∧e. 0 < e ⟹ ∃ N. ∀ n ≥ N. ∀ x ∈ K i. cmod (F (k n) x - g x) <
e) ⟹ thesis
    continuous_on (K i) g

```

```

      strict_mono k
       $\bigwedge e. 0 < e \implies \exists N. \forall n x. N \leq n \wedge x \in K i \longrightarrow \text{cmod } (\mathcal{F} (k n) x - g$ 
 $x) < e$ 
      show ?thesis
      by (rule *(1)[OF *(2,3)], drule *(4)) auto
    qed (use comK in simp_all)
  then show ?thesis
  by auto
qed
define  $\Phi$  where  $\Phi \equiv \lambda g i r. \lambda k::\text{nat} \Rightarrow \text{nat}. \forall e > 0. \exists N. \forall n \geq N. \forall x \in K i. \text{cmod}$ 
 $((\mathcal{F} \circ (r \circ k)) n x - g x) < e$ 
obtain  $k :: \text{nat} \Rightarrow \text{nat}$  where strict_mono k and  $k: \bigwedge i. \exists g. \Phi g i \text{ id } k$ 
proof (rule subsequence_diagonalization_lemma [where r=id])
  show  $\exists g. \Phi g i \text{ id } (r \circ k2)$ 
  if  $ex: \exists g. \Phi g i \text{ id } (r \circ k1)$  and  $\bigwedge j. N \leq j \implies \exists j' \geq j. k2 j = k1 j'$ 
  for  $i k1 k2 N$  and  $r::\text{nat} \Rightarrow \text{nat}$ 
proof -
  obtain  $g$  where  $\Phi g i \text{ id } (r \circ k1)$ 
  using ex by blast
  then have  $\Phi g i \text{ id } (r \circ k2)$ 
  using that
  by (simp add:  $\Phi\_def$ ) (metis (no_types, opaque_lifting) le_trans linear)
  then show ?thesis
  by metis
qed
have  $\exists k g. \text{strict\_mono } (k::\text{nat} \Rightarrow \text{nat}) \wedge \Phi g i \text{ id } (r \circ k)$  for  $i r$ 
  unfolding  $\Phi\_def$  o_assoc using rng_f by (force intro!: *)
  then show  $\bigwedge i r. \exists k. \text{strict\_mono } (k::\text{nat} \Rightarrow \text{nat}) \wedge (\exists g. \Phi g i \text{ id } (r \circ k))$ 
  by force
qed fastforce
have  $\exists l. \forall e > 0. \exists N. \forall n \geq N. \text{norm}(\mathcal{F} (k n) z - l) < e$  if  $z \in S$  for  $z$ 
proof -
  obtain  $G$  where  $G: \bigwedge i e. e > 0 \implies \exists M. \forall n \geq M. \forall x \in K i. \text{cmod } ((\mathcal{F} \circ k) n$ 
 $x - G i x) < e$ 
  using k unfolding  $\Phi\_def$  by (metis id_comp)
  obtain  $N$  where  $\bigwedge n. n \geq N \implies z \in K n$ 
  using subK [of {z}] that  $\langle z \in S \rangle$  by auto
  moreover have  $\bigwedge e. e > 0 \implies \exists M. \forall n \geq M. \forall x \in K N. \text{cmod } ((\mathcal{F} \circ k) n x -$ 
 $G N x) < e$ 
  using G by auto
  ultimately show ?thesis
  by (metis comp_apply order_refl)
qed
then obtain  $g$  where  $g: \bigwedge z e. \llbracket z \in S; e > 0 \rrbracket \implies \exists N. \forall n \geq N. \text{norm}(\mathcal{F} (k n)$ 
 $z - g z) < e$ 
  by metis
show ?thesis
proof
  show  $g\_lim: \bigwedge x. x \in S \implies (\lambda n. \mathcal{F} (k n) x) \longrightarrow g x$ 

```

```

    by (simp add: lim_sequentially g dist_norm)
  have dg_le_e:  $\exists N. \forall n \geq N. \forall x \in T. \text{cmod } (\mathcal{F} (k n) x - g x) < e$ 
    if  $T: \text{compact } T \subseteq S$  and  $0 < e$  for  $T e$ 
  proof -
    obtain N where  $N: \bigwedge n. n \geq N \implies T \subseteq K n$ 
      using subK [OF T] by blast
    obtain h where  $h: \bigwedge e. e > 0 \implies \exists M. \forall n \geq M. \forall x \in K N. \text{cmod } ((\mathcal{F} \circ k) n x - h x) < e$ 
      using k unfolding  $\Phi\_def$  by (metis id_comp)
    have geq:  $g w = h w$  if  $w \in T$  for  $w$ 
    proof (rule LIMSEQ_unique)
      show  $(\lambda n. \mathcal{F} (k n) w) \longrightarrow g w$ 
        using  $\langle T \subseteq S \rangle g\_lim$  that by blast
      show  $(\lambda n. \mathcal{F} (k n) w) \longrightarrow h w$ 
        using h N that by (force simp: lim_sequentially dist_norm)
    qed
    show ?thesis
      using T h N  $\langle 0 < e \rangle$  by (fastforce simp add: geq)
  qed
  then show  $\bigwedge K. \llbracket \text{compact } K; K \subseteq S \rrbracket \implies \text{uniform\_limit } K (\mathcal{F} \circ k) g \text{ sequentially}$ 
    by (simp add: uniform_limit_iff dist_norm eventually_sequentially)
  show  $g \text{ holomorphic\_on } S$ 
  proof (rule holomorphic_uniform_sequence [OF  $\langle \text{open } S \rangle \mathcal{H}$ ])
    show  $\bigwedge n. (\mathcal{F} \circ k) n \in \mathcal{H}$ 
      by (simp add: range_subsetD rng_f)
    show  $\exists d > 0. \text{cball } z d \subseteq S \wedge \text{uniform\_limit } (\text{cball } z d) (\lambda n. (\mathcal{F} \circ k) n) g \text{ sequentially}$ 
      if  $z \in S$  for  $z$ 
    proof -
      obtain d where  $d: d > 0 \text{cball } z d \subseteq S$ 
        using  $\langle \text{open } S \rangle \langle z \in S \rangle \text{open\_contains\_cball}$  by blast
      then have  $\text{uniform\_limit } (\text{cball } z d) (\mathcal{F} \circ k) g \text{ sequentially}$ 
        using dg_le_e compact_cball by (auto simp: uniform_limit_iff eventually_sequentially dist_norm)
      with d show ?thesis by blast
    qed
  qed
  qed (auto simp:  $\langle \text{strict\_mono } k \rangle$ )
qed

```

6.4 Some simple but useful cases of Hurwitz's theorem

proposition *Hurwitz_no_zeros:*

```

  assumes  $S: \text{open } S \text{ connected } S$ 
    and  $holf: \bigwedge n::\text{nat}. \mathcal{F} n \text{ holomorphic\_on } S$ 
    and  $holg: g \text{ holomorphic\_on } S$ 
    and  $ul\_g: \bigwedge K. \llbracket \text{compact } K; K \subseteq S \rrbracket \implies \text{uniform\_limit } K \mathcal{F} g \text{ sequentially}$ 
    and  $\text{nonconst}: \neg g \text{ constant\_on } S$ 

```

```

    and nz:  $\bigwedge n z. z \in S \implies \mathcal{F} n z \neq 0$ 
    and z0  $\in S$ 
    shows  $g z0 \neq 0$ 
  proof
    assume g0:  $g z0 = 0$ 
    obtain h r m
      where 0 < m 0 < r and subS:  $\text{ball } z0 \ r \subseteq S$ 
      and holh:  $h \text{ holomorphic\_on } \text{ball } z0 \ r$ 
      and geq:  $\bigwedge w. w \in \text{ball } z0 \ r \implies g w = (w - z0)^m * h w$ 
      and hnz:  $\bigwedge w. w \in \text{ball } z0 \ r \implies h w \neq 0$ 
      by (blast intro: holomorphic_factor_zero_nonconstant [OF holg S  $\langle z0 \in S \rangle$  g0
nonconst])
    then have holf0:  $\mathcal{F} n \text{ holomorphic\_on } \text{ball } z0 \ r$  for n
      by (meson holf holomorphic_on_subset)
    have *:  $((\lambda z. \text{deriv } (\mathcal{F} n) z / \mathcal{F} n z) \text{ has\_contour\_integral } 0) (\text{circlepath } z0$ 
 $(r/2))$  for n
    proof (rule Cauchy_theorem_disc_simple)
      show  $(\lambda z. \text{deriv } (\mathcal{F} n) z / \mathcal{F} n z) \text{ holomorphic\_on } \text{ball } z0 \ r$ 
      by (metis (no_types)  $\langle \text{open } S \rangle$  holf holomorphic_deriv holomorphic_on_divide
holomorphic_on_subset nz subS)
    qed (use  $\langle 0 < r \rangle$  in auto)
    have hol_dg:  $\text{deriv } g \text{ holomorphic\_on } S$ 
      by (simp add:  $\langle \text{open } S \rangle$  holg holomorphic_deriv)
    have continuous_on (sphere z0 (r/2)) (deriv g)
      using  $\langle 0 < r \rangle$  subS
      by (intro holomorphic_on_imp_continuous_on holomorphic_on_subset [OF
hol_dg]) auto
    then have compact (deriv g ' (sphere z0 (r/2)))
      by (rule compact_continuous_image [OF compact_sphere])
    then have bo_dg: bounded (deriv g ' (sphere z0 (r/2)))
      using compact_imp_bounded by blast
    have continuous_on (sphere z0 (r/2)) (cmod  $\circ$  g)
      using  $\langle 0 < r \rangle$  subS
      by (intro continuous_intros holomorphic_on_imp_continuous_on holomor-
phic_on_subset [OF holg]) auto
    then have compact ((cmod  $\circ$  g) ' sphere z0 (r/2))
      by (rule compact_continuous_image [OF compact_sphere])
    moreover have (cmod  $\circ$  g) ' sphere z0 (r/2)  $\neq \{\}$ 
      using  $\langle 0 < r \rangle$  by auto
    ultimately obtain b where b:  $b \in (\text{cmod } \circ g) ' \text{sphere } z0 \ (r/2)$ 
       $\bigwedge t. t \in (\text{cmod } \circ g) ' \text{sphere } z0 \ (r/2) \implies b \leq t$ 
      using compact_attains_inf [of (norm  $\circ$  g) ' (sphere z0 (r/2))] by blast
    have  $(\lambda n. \text{contour\_integral } (\text{circlepath } z0 \ (r/2)) (\lambda z. \text{deriv } (\mathcal{F} n) z / \mathcal{F} n z))$ 
 $\longrightarrow$ 
       $\text{contour\_integral } (\text{circlepath } z0 \ (r/2)) (\lambda z. \text{deriv } g z / g z)$ 
    proof (rule contour_integral_uniform_limit_circlepath)
      show  $\forall_F n$  in sequentially.  $(\lambda z. \text{deriv } (\mathcal{F} n) z / \mathcal{F} n z) \text{ contour\_integrable\_on}$ 
 $\text{circlepath } z0 \ (r/2)$ 
      using * contour_integrable_on_def eventually_sequentiallyI by meson

```

```

show uniform_limit (sphere z0 (r/2)) (λn z. deriv (F n) z / F n z) (λz. deriv
g z / g z) sequentially
proof (rule uniform_lim_divide [OF _ _ bo_dg])
show uniform_limit (sphere z0 (r/2)) (λa. deriv (F a)) (deriv g) sequentially
proof (rule uniform_limitI)
  fix e::real
  assume 0 < e

  show ∀F n in sequentially. ∀x ∈ sphere z0 (r/2). dist (deriv (F n) x)
(deriv g x) < e
  proof -
    have dist (deriv (F n) w) (deriv g w) < e
      if e8: ∧x. dist z0 x ≤ 3 * r / 4 ⇒ dist (F n x) (g x) * 8 < r * e
      and w: w ∈ sphere z0 (r/2) for n w
    proof -
      have ball w (r/4) ⊆ ball z0 r cball w (r/4) ⊆ ball z0 r
      using ⟨0 < r⟩ w by (simp_all add: ball_subset_ball_iff cball_subset_ball_iff
dist_commute)
      with subS have wr4_sub: ball w (r/4) ⊆ S cball w (r/4) ⊆ S by force+
      moreover
        have (λz. F n z - g z) holomorphic_on S
          by (intro holomorphic_intros holf holg)
        ultimately have hol: (λz. F n z - g z) holomorphic_on ball w (r/4)
          and cont: continuous_on (cball w (r / 4)) (λz. F n z - g z)
      using holomorphic_on_subset by (blast intro: holomorphic_on_imp_continuous_on)+
      have w ∈ S
        using ⟨0 < r⟩ wr4_sub by auto
      have dist z0 y ≤ 3 * r / 4 if dist w y < r/4 for y
      proof (rule dist_triangle_le [where z=w])
        show dist z0 w + dist y w ≤ 3 * r / 4
        using w that by (simp add: dist_commute)
      qed
      with e8 have in_ball: ∧y. y ∈ ball w (r/4) ⇒ F n y - g y ∈ ball 0
(r/4 * e/2)
        by (simp add: dist_norm [symmetric])
      have F n field_differentiable at w
        by (metis holomorphic_on_imp_differentiable_at ⟨w ∈ S⟩ holf ⟨open
S⟩)
      moreover
        have g field_differentiable at w
          using ⟨w ∈ S⟩ ⟨open S⟩ holg holomorphic_on_imp_differentiable_at
by auto
      moreover
        have cmod (deriv (λw. F n w - g w) w) * 2 ≤ e
          using Cauchy_higher_deriv_bound [OF hol cont in_ball, of 1] ⟨r >
0⟩ by auto
      ultimately have dist (deriv (F n) w) (deriv g w) ≤ e/2
        by (simp add: dist_norm)
      then show ?thesis

```



```

      using ‹e > 0› by auto
    qed
  moreover
  have cball z0 (3 * r / 4) ⊆ ball z0 r
    by (simp add: cball_subset_ball_iff ‹0 < r›)
  with subS have uniform_limit (cball z0 (3 * r / 4))  $\mathcal{F}$  g sequentially
    by (force intro: ul_g)
  then have  $\forall_F n$  in sequentially.  $\forall x \in \text{cball } z0 \ (3 * r / 4). \text{dist } (\mathcal{F} \ n \ x) \ (g$ 
 $x) < r / 4 * e / 2$ 
    using ‹0 < e› ‹0 < r› by (force simp: intro!: uniform_limitD)
  ultimately show ?thesis
    by (force simp add: eventually_sequentially)
  qed
qed
show uniform_limit (sphere z0 (r/2))  $\mathcal{F}$  g sequentially
proof (rule uniform_limitI)
  fix e::real
  assume 0 < e
  have sphere z0 (r/2) ⊆ ball z0 r
    using ‹0 < r› by auto
  with subS have uniform_limit (sphere z0 (r/2))  $\mathcal{F}$  g sequentially
    by (force intro: ul_g)
  then show  $\forall_F n$  in sequentially.  $\forall x \in \text{sphere } z0 \ (r/2). \text{dist } (\mathcal{F} \ n \ x) \ (g \ x)$ 
< e
    using ‹0 < e› uniform_limit_iff by blast
  qed
  show b > 0  $\wedge x. x \in \text{sphere } z0 \ (r/2) \implies b \leq \text{cmod } (g \ x)$ 
    using b ‹0 < r› by (fastforce simp: geq hnz)+
  qed
qed (use ‹0 < r› in auto)
then have  $(\lambda n. 0) \longrightarrow \text{contour\_integral } (\text{circlepath } z0 \ (r/2)) \ (\lambda z. \text{deriv } g \ z$ 
/ g z)
  by (simp add: contour_integral_unique [OF *])
then have contour_integral (circlepath z0 (r/2))  $(\lambda z. \text{deriv } g \ z / g \ z) = 0$ 
  by (simp add: LIMSEQ_const_iff)
moreover
have contour_integral (circlepath z0 (r/2))  $(\lambda z. \text{deriv } g \ z / g \ z) =$ 
contour_integral (circlepath z0 (r/2))  $(\lambda z. m / (z - z0) + \text{deriv } h \ z / h \ z)$ 
proof (rule contour_integral_eq, use ‹0 < r› in simp)
  fix w
  assume w: dist z0 w * 2 = r
  then have w_inb: w ∈ ball z0 r
    using ‹0 < r› by auto
  have h_der: (h has_field_derivative deriv h w) (at w)
    using holh holomorphic_derivI w_inb by blast
  have deriv g w = ((of_nat m * h w + deriv h w * (w - z0)) * (w - z0) ^ m)
/ (w - z0)
    if r = dist z0 w * 2 w ≠ z0
  proof -

```

```

    have (( $\lambda w. (w - z0) ^ m * h w$ ) has_field_derivative
      ( $m * h w + \text{deriv } h w * (w - z0)$ ) * ( $w - z0$ ) ^  $m / (w - z0)$ ) (at  $w$ )
    apply (rule derivative_eq_intros  $h\_der$  refl) +
    using that  $\langle m > 0 \rangle \langle 0 < r \rangle$  apply (simp add: divide_simps distrib_right)
    by (metis Suc_pred mult.commute power_Suc)
  then show ?thesis
proof (rule DERIV_imp_deriv [OF has_field_derivative_transform_within_open])
  show  $\bigwedge x. x \in \text{ball } z0 \ r \implies (x - z0) ^ m * h x = g x$ 
    by (simp add: hnz geq)
  qed (use that  $\langle m > 0 \rangle \langle 0 < r \rangle$  in auto)
qed
with  $\langle 0 < r \rangle \langle 0 < m \rangle$   $w \ w\_inb$  show  $\text{deriv } g \ w / g \ w = \text{of\_nat } m / (w - z0)$ 
+  $\text{deriv } h \ w / h \ w$ 
  by (auto simp: geq field_split_simps hnz)
qed
moreover
have contour_integral (circlepath z0 (r/2)) ( $\lambda z. m / (z - z0) + \text{deriv } h \ z / h$ 
 $z$ ) =
   $2 * \text{of\_real } \pi * i * m + 0$ 
proof (rule contour_integral_unique [OF has_contour_integral_add])
  show (( $\lambda x. m / (x - z0)$ ) has_contour_integral  $2 * \text{of\_real } \pi * i * m$ )
(circlepath  $z0 \ (r/2)$ )
    by (force simp:  $\langle 0 < r \rangle$  intro: Cauchy_integral_circlepath_simple)
  show (( $\lambda x. \text{deriv } h \ x / h \ x$ ) has_contour_integral 0) (circlepath  $z0 \ (r/2)$ )
    using hnz holh holomorphic_deriv holomorphic_on_divide  $\langle 0 < r \rangle$ 
    by (fastforce intro!: Cauchy_theorem_disc_simple [of _ z0 r])
  qed
ultimately show False using  $\langle 0 < m \rangle$  by auto
qed

corollary Hurwitz_injective:
  assumes  $S$ : open  $S$  connected  $S$ 
    and holf:  $\bigwedge n::\text{nat}. \mathcal{F} \ n$  holomorphic_on  $S$ 
    and holg:  $g$  holomorphic_on  $S$ 
    and ul_g:  $\bigwedge K. \llbracket \text{compact } K; K \subseteq S \rrbracket \implies \text{uniform\_limit } K \ \mathcal{F} \ g \text{ sequentially}$ 
    and nonconst:  $\neg g$  constant_on  $S$ 
    and inj:  $\bigwedge n. \text{inj\_on } (\mathcal{F} \ n) \ S$ 
  shows inj_on  $g \ S$ 
proof -
  have False if  $z12: z1 \in S \ z2 \in S \ z1 \neq z2 \ g \ z2 = g \ z1$  for  $z1 \ z2$ 
  proof -
    obtain  $z0$  where  $z0 \in S$  and  $z0: g \ z0 \neq g \ z2$ 
      using constant_on_def nonconst by blast
    have ( $\lambda z. g \ z - g \ z1$ ) holomorphic_on  $S$ 
      by (intro holomorphic_intros holg)
    then obtain  $r$  where  $0 < r \ \text{ball } z2 \ r \subseteq S \ \bigwedge z. \text{dist } z2 \ z < r \wedge z \neq z2 \implies g$ 
 $z \neq g \ z1$ 
      using isolated_zeros [of  $\lambda z. g \ z - g \ z1 \ S \ z2 \ z0$ ]  $S \ \langle z0 \in S \rangle \ z0 \ z12$  by auto
    have  $g \ z2 - g \ z1 \neq 0$ 

```

```

proof (rule Hurwitz_no_zeros [of  $S - \{z1\}$   $\lambda n z. \mathcal{F} n z - \mathcal{F} n z1$   $\lambda z. g z - g z1$ ])
  show open  $(S - \{z1\})$ 
    by (simp add: S open_delete)
  show connected  $(S - \{z1\})$ 
    by (simp add: connected_open_delete [OF S])
  show  $\bigwedge n. (\lambda z. \mathcal{F} n z - \mathcal{F} n z1)$  holomorphic_on  $S - \{z1\}$ 
    by (intro holomorphic_intros holomorphic_on_subset [OF holf]) blast
  show  $(\lambda z. g z - g z1)$  holomorphic_on  $S - \{z1\}$ 
    by (intro holomorphic_intros holomorphic_on_subset [OF holg]) blast
  show uniform_limit  $K (\lambda n z. \mathcal{F} n z - \mathcal{F} n z1) (\lambda z. g z - g z1)$  sequentially
    if compact  $K$   $K \subseteq S - \{z1\}$  for  $K$ 
proof (rule uniform_limitI)
  fix  $e::\text{real}$ 
  assume  $e > 0$ 
  have uniform_limit  $K \mathcal{F} g$  sequentially
    using that ul_g by fastforce
  then have  $K: \forall_F n$  in sequentially.  $\forall x \in K. \text{dist } (\mathcal{F} n x) (g x) < e/2$ 
    using  $\langle 0 < e \rangle$  by (force simp: intro!: uniform_limitD)
  have uniform_limit  $\{z1\} \mathcal{F} g$  sequentially
    by (intro ul_g) (auto simp: z12)
  then have  $\forall_F n$  in sequentially.  $\forall x \in \{z1\}. \text{dist } (\mathcal{F} n x) (g x) < e/2$ 
    using  $\langle 0 < e \rangle$  by (force simp: intro!: uniform_limitD)
  then have  $z1: \forall_F n$  in sequentially.  $\text{dist } (\mathcal{F} n z1) (g z1) < e/2$ 
    by simp
  show  $\forall_F n$  in sequentially.  $\forall x \in K. \text{dist } (\mathcal{F} n x - \mathcal{F} n z1) (g x - g z1) < e$ 
    apply (intro eventually_mono [OF eventually_conj [OF K z1]])
  by (metis (no_types, opaque_lifting) diff_add_eq diff_diff_eq2 dist_commute
dist_norm dist_triangle_add_half)
qed
show  $\neg (\lambda z. g z - g z1)$  constant_on  $S - \{z1\}$ 
  unfolding constant_on_def
  by (metis Diff_iff  $\langle z0 \in S \rangle$  empty_iff insert_iff right_minus_eq  $z0 z12$ )
show  $\bigwedge n z. z \in S - \{z1\} \implies \mathcal{F} n z - \mathcal{F} n z1 \neq 0$ 
  by (metis DiffD1 DiffD2 eq_iff_diff_eq_0 inj inj_onD insertI1  $\langle z1 \in S \rangle$ )
show  $z2 \in S - \{z1\}$ 
  using  $\langle z2 \in S \rangle \langle z1 \neq z2 \rangle$  by auto
qed
with z12 show False by auto
qed
then show ?thesis by (auto simp: inj_on_def)
qed

```

6.5 The Great Picard theorem

lemma GPicard1:

```

assumes  $S$ : open  $S$  connected  $S$  and  $w \in S$   $0 < r$   $Y \subseteq X$ 
and holX:  $\bigwedge h. h \in X \implies h$  holomorphic_on  $S$ 
and X01:  $\bigwedge h z. \llbracket h \in X; z \in S \rrbracket \implies h z \neq 0 \wedge h z \neq 1$ 

```

```

    and r:  $\bigwedge h. h \in Y \implies \text{norm}(h\ w) \leq r$ 
  obtains B Z where  $0 < B$  open Z  $w \in Z$   $Z \subseteq S$   $\bigwedge h\ z. \llbracket h \in Y; z \in Z \rrbracket \implies$ 
 $\text{norm}(h\ z) \leq B$ 
proof -
  obtain e where  $e > 0$  and  $e: \text{cball } w\ e \subseteq S$ 
  using assms open_contains_cball_eq by blast
  show ?thesis
proof
  show  $0 < \exp(\pi * \exp(\pi * (2 + 2 * r + 12)))$ 
  by simp
  show  $\text{ball } w\ (e / 2) \subseteq S$ 
  using e ball_divide_subset_numeral ball_subset_cball by blast
  show  $\text{cmod } (h\ z) \leq \exp(\pi * \exp(\pi * (2 + 2 * r + 12)))$ 
  if  $h \in Y\ z \in \text{ball } w\ (e / 2)$  for h z
proof -
  have  $h \in X$ 
  using  $\langle Y \subseteq X \rangle \langle h \in Y \rangle$  by blast
  have  $\text{hol\_h\_o}: (h \circ (\lambda z. (w + \text{of\_real } e * z))) \text{ holomorphic\_on } \text{cball } 0\ 1$ 
proof (intro holomorphic_intros holomorphic_on_compose)
  have  $h \text{ holomorphic\_on } S$ 
  using holX  $\langle h \in X \rangle$  by auto
  then have  $h \text{ holomorphic\_on } \text{cball } w\ e$ 
  by (metis e holomorphic_on_subset)
  moreover have  $(\lambda z. w + \text{complex\_of\_real } e * z) \text{ ` } \text{cball } 0\ 1 \subseteq \text{cball } w\ e$ 
  using that  $\langle e > 0 \rangle$  by (auto simp: dist_norm norm_mult)
  ultimately show  $h \text{ holomorphic\_on } (\lambda z. w + \text{complex\_of\_real } e * z) \text{ ` }$ 
 $\text{cball } 0\ 1$ 
  by (rule holomorphic_on_subset)
qed
  have  $\text{norm\_le\_r}: \text{cmod } ((h \circ (\lambda z. w + \text{complex\_of\_real } e * z))\ 0) \leq r$ 
  by (auto simp: r  $\langle h \in Y \rangle$ )
  have le12:  $\text{norm } (\text{of\_real } (\text{inverse } e) * (z - w)) \leq 1/2$ 
  using that  $\langle e > 0 \rangle$  by (simp add: inverse_eq_divide dist_norm norm_minus_commute
 $\text{norm\_divide}$ )
  have non01:  $h\ (w + e * z) \neq 0 \wedge h\ (w + e * z) \neq 1$  if  $z \in \text{cball } 0\ 1$  for
 $z::\text{complex}$ 
proof (rule X01 [OF  $\langle h \in X \rangle$ ])
  have  $w + \text{complex\_of\_real } e * z \in \text{cball } w\ e$ 
  using  $\langle 0 < e \rangle$  that by (auto simp: dist_norm norm_mult)
  then show  $w + \text{complex\_of\_real } e * z \in S$ 
  by (rule subsetD [OF e])
qed
  have  $\text{cmod } (h\ z) \leq \text{cmod } (h\ (w + \text{of\_real } e * (\text{inverse } e * (z - w))))$ 
  using  $\langle 0 < e \rangle$  by (simp add: field_split_simps)
  also have  $\dots \leq \exp(\pi * \exp(\pi * (14 + 2 * r)))$ 
  using r [OF  $\langle h \in Y \rangle$ ] Schottky [OF hol_h_o norm_le_r le12] non01
by auto
finally
  show ?thesis by simp

```

qed
 qed (use <e> 0> in auto)
 qed

lemma GPicard2:

assumes $S \subseteq T$ connected $T \setminus S \neq \{\}$ open $S \wedge x. \llbracket x \text{ islimpt } S; x \in T \rrbracket \implies x \in S$
 shows $S = T$
 by (metis assms open_subset connected_clopen closedin_limpt)

lemma GPicard3:

assumes S : open S connected $S \neq \{\}$ and $Y \subseteq X$
 and holX: $\bigwedge h. h \in X \implies h \text{ holomorphic_on } S$
 and X01: $\bigwedge h z. \llbracket h \in X; z \in S \rrbracket \implies h z \neq 0 \wedge h z \neq 1$
 and no_hw_le1: $\bigwedge h. h \in Y \implies \text{norm}(h w) \leq 1$
 and compact $K \subseteq S$
 obtains B where $\bigwedge h z. \llbracket h \in Y; z \in K \rrbracket \implies \text{norm}(h z) \leq B$
 proof –
 define U where $U \equiv \{z \in S. \exists B Z. 0 < B \wedge \text{open } Z \wedge z \in Z \wedge Z \subseteq S \wedge$
 $(\forall h z'. h \in Y \wedge z' \in Z \longrightarrow \text{norm}(h z') \leq B)\}$
 then have $U \subseteq S$ by blast
 have $U = S$
 proof (rule GPicard2 [OF <U ⊆ S> <connected S>])
 show $U \neq \{\}$
 proof –
 obtain $B Z$ where $0 < B$ open $Z \neq \{\}$ $w \in Z \wedge Z \subseteq S$
 and $\bigwedge h z. \llbracket h \in Y; z \in Z \rrbracket \implies \text{norm}(h z) \leq B$
 using GPicard1 [OF $S \neq \{\}$ zero_less_one <Y ⊆ X> holX] X01 no_hw_le1 by
 blast
 then show ?thesis
 unfolding U_def using <w ∈ S> by blast
 qed
 show open U
 unfolding open_subopen [of U] by (auto simp: U_def)
 fix v
 assume v : $v \text{ islimpt } U \wedge v \in S$
 have $\neg (\forall r > 0. \exists h \in Y. r < \text{cmod}(h v))$
 proof
 assume $\forall r > 0. \exists h \in Y. r < \text{cmod}(h v)$
 then have $\forall n. \exists h \in Y. \text{Suc } n < \text{cmod}(h v)$
 by simp
 then obtain \mathcal{F} where $FY: \bigwedge n. \mathcal{F} n \in Y$ and $ltF: \bigwedge n. \text{Suc } n < \text{cmod}(\mathcal{F} n v)$
 by metis
 define \mathcal{G} where $\mathcal{G} \equiv \lambda n z. \text{inverse}(\mathcal{F} n z)$
 have holG: $\mathcal{G} n \text{ holomorphic_on } S$ for n
 proof (simp add: G_def)
 show $(\lambda z. \text{inverse}(\mathcal{F} n z)) \text{ holomorphic_on } S$
 using FY X01 <Y ⊆ X> holX by (blast intro: holomorphic_on_inverse)

```

qed
have  $\mathcal{G} \text{ not } 0$ :  $\mathcal{G} \ n \ z \neq 0$  and  $\mathcal{G} \text{ not } 1$ :  $\mathcal{G} \ n \ z \neq 1$  if  $z \in S$  for  $n \ z$ 
  using FY X01  $\langle Y \subseteq X \rangle$  that by (force simp:  $\mathcal{G}_{\text{def}}$ ) +
have  $\mathcal{G}_{\text{le}} 1$ :  $\text{cmod } (\mathcal{G} \ n \ v) \leq 1$  for  $n$ 
  using less_le_trans linear ltF
  by (fastforce simp add:  $\mathcal{G}_{\text{def}}$  norm_inverse inverse_le_1_iff)
define  $W$  where  $W \equiv \{h. h \text{ holomorphic\_on } S \wedge (\forall z \in S. h \ z \neq 0 \wedge h \ z \neq 1)\}$ 
obtain  $B \ Z$  where  $0 < B$  open  $Z \ v \in Z \ Z \subseteq S$ 
  and  $B$ :  $\bigwedge h \ z. \llbracket h \in \text{range } \mathcal{G}; z \in Z \rrbracket \implies \text{norm}(h \ z) \leq B$ 
  apply (rule GPicard1 [OF  $\langle \text{open } S \rangle \langle \text{connected } S \rangle \langle v \in S \rangle \text{zero\_less\_one,}$ 
of range  $\mathcal{G} \ W$ ])
  using hol $\mathcal{G}$   $\mathcal{G} \text{ not } 0$   $\mathcal{G} \text{ not } 1$   $\mathcal{G}_{\text{le}} 1$  by (force simp:  $W_{\text{def}}$ ) +
then obtain  $e$  where  $e > 0$  and  $e$ :  $\text{ball } v \ e \subseteq Z$ 
  by (meson open_contains_ball)
obtain  $h \ j$  where hol $h$ :  $h \text{ holomorphic\_on ball } v \ e$  and strict_mono  $j$ 
  and lim:  $\bigwedge x. x \in \text{ball } v \ e \implies (\lambda n. \mathcal{G} \ (j \ n) \ x) \longrightarrow h \ x$ 
  and ulim:  $\bigwedge K. \llbracket \text{compact } K; K \subseteq \text{ball } v \ e \rrbracket \implies \text{uniform\_limit } K \ (\mathcal{G} \circ j) \ h \text{ sequentially}$ 
proof (rule Montel)
  show  $\bigwedge h. h \in \text{range } \mathcal{G} \implies h \text{ holomorphic\_on ball } v \ e$ 
    by (metis  $\langle Z \subseteq S \rangle e \text{ hol } \mathcal{G} \text{ holomorphic\_on\_subset image } E$ )
  show  $\bigwedge K. \llbracket \text{compact } K; K \subseteq \text{ball } v \ e \rrbracket \implies \exists B. \forall h \in \text{range } \mathcal{G}. \forall z \in K. \text{cmod}$ 
     $(h \ z) \leq B$ 
    using  $B \ e$  by blast
qed auto
have  $h \ v = 0$ 
proof (rule LIMSEQ_unique)
  show  $(\lambda n. \mathcal{G} \ (j \ n) \ v) \longrightarrow h \ v$ 
    using  $\langle e > 0 \rangle$  lim by simp
  have lt_Fj:  $\text{real } x \leq \text{cmod } (\mathcal{F} \ (j \ x) \ v)$  for  $x$ 
    by (metis of_nat_Suc ltF  $\langle \text{strict\_mono } j \rangle$  add commute less_eq_real_def
less_le_trans nat_le_real_less seq_suble)
  show  $(\lambda n. \mathcal{G} \ (j \ n) \ v) \longrightarrow 0$ 
proof (rule Lim_null_comparison [OF eventually_sequentiallyI lim_inverse_n])
  show  $\text{cmod } (\mathcal{G} \ (j \ x) \ v) \leq \text{inverse } (\text{real } x)$  if  $1 \leq x$  for  $x$ 
    using that by (simp add:  $\mathcal{G}_{\text{def}}$  norm_inverse_le_norm [OF lt_Fj])
qed
qed
have  $h \ v \neq 0$ 
proof (rule Hurwitz_no_zeros [of ball  $v \ e \ \mathcal{G} \circ j \ h$ ])
  show  $\bigwedge n. (\mathcal{G} \circ j) \ n \text{ holomorphic\_on ball } v \ e$ 
    using  $\langle Z \subseteq S \rangle e \text{ hol } \mathcal{G}$  by force
  show  $\bigwedge n \ z. z \in \text{ball } v \ e \implies (\mathcal{G} \circ j) \ n \ z \neq 0$ 
    using  $\mathcal{G} \text{ not } 0 \ \langle Z \subseteq S \rangle e$  by fastforce
  show  $\neg h \text{ constant\_on ball } v \ e$ 
proof (clarsimp simp: constant_on_def)
  fix  $c$ 
  have False if  $\bigwedge z. \text{dist } v \ z < e \implies h \ z = c$ 

```

```

proof -
  have  $h\ v = c$ 
    by (simp add:  $\langle 0 < e \rangle$  that)
  obtain  $y$  where  $y \in U$   $y \neq v$  and  $y$ :  $\text{dist } y\ v < e$ 
    using  $v \langle e > 0 \rangle$  by (auto simp: islimpt_approachable)
  then obtain  $C\ T$  where  $y \in S$   $C > 0$  open  $T$   $y \in T$   $T \subseteq S$ 
    and  $\bigwedge h\ z'. \llbracket h \in Y; z' \in T \rrbracket \implies \text{cmod } (h\ z') \leq C$ 
    using  $\langle y \in U \rangle$  by (auto simp:  $U\_def$ )
  then have  $le\_C: \bigwedge n. \text{cmod } (\mathcal{F}\ n\ y) \leq C$ 
    using  $FY$  by blast

  have  $\forall_F\ n$  in sequentially.  $\text{dist } (\mathcal{G}\ (j\ n)\ y)\ (h\ y) < \text{inverse } C$ 
    using uniform_limitD [OF ulim [of  $\{y\}$ ], of inverse C]  $\langle C > 0 \rangle\ y$ 
    by (simp add: dist_commute)
  then obtain  $n$  where  $\text{dist } (\mathcal{G}\ (j\ n)\ y)\ (h\ y) < \text{inverse } C$ 
    by (meson eventually_at_top_linorder order_refl)
  moreover
  have  $h\ y = h\ v$ 
    by (metis  $\langle h\ v = c \rangle$  dist_commute that  $y$ )
  ultimately have  $\text{cmod } (\text{inverse } (\mathcal{F}\ (j\ n)\ y)) < \text{inverse } C$ 
    by (simp add:  $\langle h\ v = 0 \rangle\ \mathcal{G\_def}$ )
  then have  $C < \text{norm } (\mathcal{F}\ (j\ n)\ y)$ 
    by (metis  $\mathcal{G\_def}\ \mathcal{Gnot0}\ \langle y \in S \rangle$  inverse_less_imp_less inverse_zero
    norm_inverse_zero_less_norm_iff)
  show False
    using  $\langle C < \text{cmod } (\mathcal{F}\ (j\ n)\ y) \rangle\ le\_C\ \text{not\_less}$  by blast
  qed
  then show  $\exists x \in \text{ball } v\ e. h\ x \neq c$  by force
  qed
  show  $h$  holomorphic_on ball  $v\ e$ 
    by (simp add: holh)
  show  $\bigwedge K. \llbracket \text{compact } K; K \subseteq \text{ball } v\ e \rrbracket \implies \text{uniform\_limit } K\ (\mathcal{G} \circ j)\ h$ 
    sequentially
    by (simp add: ulim)
  qed (use  $\langle e > 0 \rangle$  in auto)
  with  $\langle h\ v = 0 \rangle$  show False by blast
  qed
  then obtain  $r$  where  $0 < r$  and  $r$ :  $\bigwedge h. h \in Y \implies \text{cmod } (h\ v) \leq r$ 
    by (metis not_le)
  moreover
  obtain  $B\ Z$  where  $0 < B$  open  $Z$   $v \in Z$   $Z \subseteq S$   $\bigwedge h\ z. \llbracket h \in Y; z \in Z \rrbracket \implies$ 
    norm( $h\ z$ )  $\leq B$ 
    using  $X01$ 
    by (auto simp:  $r$  intro: GPicard1 [OF  $\langle \text{open } S \rangle\ \langle \text{connected } S \rangle\ \langle v \in S \rangle\ \langle r > 0 \rangle$ 
     $\langle Y \subseteq X \rangle\ \text{holX} \rangle\ X01)$ 
  ultimately show  $v \in U$ 
    using  $v$  by (simp add:  $U\_def$ ) meson
  qed
  have  $\bigwedge x. x \in K \longrightarrow x \in U$ 

```

```

    using ‹ $U = S$ › ‹ $K \subseteq S$ › by blast
  then have  $\bigwedge x. x \in K \longrightarrow (\exists B Z. 0 < B \wedge \text{open } Z \wedge x \in Z \wedge$ 
     $(\forall h z'. h \in Y \wedge z' \in Z \longrightarrow \text{norm}(h z') \leq B))$ 
    unfolding  $U\_def$  by blast
  then obtain  $F Z$  where  $F: \bigwedge x. x \in K \implies \text{open } (Z x) \wedge x \in Z x \wedge$ 
     $(\forall h z'. h \in Y \wedge z' \in Z x \longrightarrow \text{norm}(h z') \leq F x)$ 
    by metis
  then obtain  $L$  where  $L \subseteq K$  finite  $L$  and  $L: K \subseteq (\bigcup c \in L. Z c)$ 
    by (auto intro: compactE_image [OF ‹compact  $K$ ›, of  $K Z$ ])
  then have *:  $\bigwedge x h z'. \llbracket x \in L; h \in Y \wedge z' \in Z x \rrbracket \implies \text{cmod } (h z') \leq F x$ 
    using  $F$  by blast
  have  $\exists B. \forall h z. h \in Y \wedge z \in K \longrightarrow \text{norm}(h z) \leq B$ 
  proof (cases  $L = \{\}$ )
    case True with  $L$  show ?thesis by simp
  next
    case False
    then have  $\forall h z. h \in Y \wedge z \in K \longrightarrow (\exists x \in L. \text{cmod } (h z) \leq F x)$ 
      by (metis *  $L \text{ UN\_E subset\_iff}$ )
    with  $False$  ‹finite  $L$ › show ?thesis
      by (rule_tac  $x = \text{Max } (F \restriction L)$  in exI) (simp add: linorder_class.Max_ge_iff)
  qed
  with that show ?thesis by metis
qed

```

lemma GPicard4:

```

  assumes  $0 < k$  and holf:  $f \text{ holomorphic\_on } (\text{ball } 0 k - \{0\})$ 
    and AE:  $\bigwedge e. \llbracket 0 < e; e < k \rrbracket \implies \exists d. 0 < d \wedge d < e \wedge (\forall z \in \text{sphere } 0 d. \text{norm}(f z) \leq B)$ 
  obtains  $\varepsilon$  where  $0 < \varepsilon \varepsilon < k \wedge \bigwedge z. z \in \text{ball } 0 \varepsilon - \{0\} \implies \text{norm}(f z) \leq B$ 
  proof -
    obtain  $\varepsilon$  where  $0 < \varepsilon \varepsilon < k/2$  and  $\varepsilon: \bigwedge z. \text{norm } z = \varepsilon \implies \text{norm}(f z) \leq B$ 
      using AE [of  $k/2$ ] ‹ $0 < k$ › by auto
    show ?thesis
    proof
      show  $\varepsilon < k$ 
        using ‹ $0 < k$ › ‹ $\varepsilon < k/2$ › by auto
      show  $\text{cmod } (f \xi) \leq B$  if  $\xi: \xi \in \text{ball } 0 \varepsilon - \{0\}$  for  $\xi$ 
        proof -
          obtain  $d$  where  $0 < d d < \text{norm } \xi$  and  $d: \bigwedge z. \text{norm } z = d \implies \text{norm}(f z) \leq B$ 
            using AE [of  $\text{norm } \xi$ ] ‹ $\varepsilon < k$ ›  $\xi$  by auto
          have [simp]:  $\text{closure } (\text{cball } 0 \varepsilon - \text{ball } 0 d) = \text{cball } 0 \varepsilon - \text{ball } 0 d$ 
            by (blast intro!: closure_closed)
          have [simp]:  $\text{interior } (\text{cball } 0 \varepsilon - \text{ball } 0 d) = \text{ball } 0 \varepsilon - \text{cball } (0::\text{complex}) d$ 
            using ‹ $0 < \varepsilon$ › ‹ $0 < d$ › by (simp add: interior_diff)
          have *:  $\text{norm}(f w) \leq B$  if  $w \in \text{cball } 0 \varepsilon - \text{ball } 0 d$  for  $w$ 
            proof (rule maximum_modulus_frontier [of  $f \text{ cball } 0 \varepsilon - \text{ball } 0 d$ ])
              show  $f \text{ holomorphic\_on interior } (\text{cball } 0 \varepsilon - \text{ball } 0 d)$ 

```



```

    using  $\langle \varepsilon < k \rangle \langle 0 < d \rangle$  that by (auto intro: holomorphic_on_subset [OF
holf])
    show continuous_on (closure (cball 0  $\varepsilon$  - ball 0  $d$ )) f
    proof (intro holomorphic_on_imp_continuous_on holomorphic_on_subset
[OF holf])
      show closure (cball 0  $\varepsilon$  - ball 0  $d$ )  $\subseteq$  ball 0  $k$  - {0}
      using  $\langle 0 < d \rangle \langle \varepsilon < k \rangle$  by auto
    qed
    show  $\bigwedge z. z \in \text{frontier (cball 0 } \varepsilon \text{ - ball 0 } d) \implies cmod (f z) \leq B$ 
    unfolding frontier_def
    using  $\varepsilon d \text{ less\_eq\_real\_def}$  by force
  qed (use that in auto)
  show ?thesis
  using *  $\langle d < cmod \xi \rangle$  that by auto
  qed
  qed (use  $\langle 0 < \varepsilon \rangle$  in auto)
  qed

```

lemma GPicard5:

```

  assumes holf: f holomorphic_on (ball 0 1 - {0})
  and f01:  $\bigwedge z. z \in \text{ball 0 1} - \{0\} \implies f z \neq 0 \wedge f z \neq 1$ 
  obtains e B where  $0 < e \wedge e < 1 \wedge 0 < B$ 
    ( $\forall z \in \text{ball 0 } e - \{0\}. \text{norm}(f z) \leq B$ )  $\vee$ 
    ( $\forall z \in \text{ball 0 } e - \{0\}. \text{norm}(f z) \geq B$ )
  proof -
    have [simp]:  $1 + \text{of\_nat } n \neq (0::\text{complex})$  for n
    using of_nat_eq_0_iff by fastforce
    have [simp]:  $cmod (1 + \text{of\_nat } n) = 1 + \text{of\_nat } n$  for n
    by (metis norm_of_nat of_nat_Suc)
    have *:  $(\lambda x::\text{complex}. x / \text{of\_nat } (\text{Suc } n)) \text{ ` } (\text{ball 0 1} - \{0\}) \subseteq \text{ball 0 1} - \{0\}$ 
  for n
    by (auto simp: norm_divide field_split_simps split: if_split_asm)
  define h where  $h \equiv \lambda n z::\text{complex}. f (z / (\text{Suc } n))$ 
  have holh: (h n) holomorphic_on ball 0 1 - {0} for n
    unfolding h_def
  proof (rule holomorphic_on_compose_gen [unfolded o_def, OF _ holf *])
    show  $(\lambda x. x / \text{of\_nat } (\text{Suc } n)) \text{ holomorphic\_on ball 0 1} - \{0\}$ 
    by (intro holomorphic_intros) auto
  qed
  have h01:  $\bigwedge n z. z \in \text{ball 0 1} - \{0\} \implies h n z \neq 0 \wedge h n z \neq 1$ 
    unfolding h_def
    using * by (force intro!: f01)
  obtain w where  $w \in \text{ball 0 1} - \{0::\text{complex}\}$ 
    by (rule_tac  $w = 1/2$  in that) auto
  consider infinite {n.  $\text{norm}(h n w) \leq 1$ } | infinite {n.  $1 \leq \text{norm}(h n w)$ }
  by (metis (mono_tags, lifting) infinite_nat_iff_unbounded_le le_cases mem_Collect_eq)
  then show ?thesis
  proof cases

```

```

case 1
with infinite_enumerate obtain r :: nat ⇒ nat
  where strict_mono r and r:  $\bigwedge n. r\ n \in \{n. \text{norm}(h\ n\ w) \leq 1\}$ 
  by blast
obtain B where B:  $\bigwedge j\ z. [\text{norm } z = 1/2; j \in \text{range } (h \circ r)] \implies \text{norm}(j\ z)$ 
≤ B
proof (rule GPicard3 [OF _ _ w, where K = sphere 0 (1/2)])
  show range (h ∘ r) ⊆
    {g. g holomorphic_on ball 0 1 - {0} ∧ (∀ z ∈ ball 0 1 - {0}. g z ≠ 0
  ∧ g z ≠ 1)}
  using h01 by (auto intro: holomorphic_intros holomorphic_on_compose
holh)
  show connected (ball 0 1 - {0::complex})
    by (simp add: connected_open_delete)
qed (use r in auto)
have normf_le_B: cmod(f z) ≤ B if norm z = 1 / (2 * (1 + of_nat (r n)))
for z n
proof -
  have *:  $\bigwedge w. \text{norm } w = 1/2 \implies \text{cmod}((f\ (w / (1 + \text{of\_nat } (r\ n)))))) \leq B$ 
  using B by (auto simp: h_def o_def)
  have half: norm (z * (1 + of_nat (r n))) = 1/2
    by (simp add: norm_mult divide_simps that)
  show ?thesis
    using * [OF half] by simp
qed
obtain ε where 0 < ε ε < 1  $\bigwedge z. z \in \text{ball } 0\ \varepsilon - \{0\} \implies \text{cmod}(f\ z) \leq B$ 
proof (rule GPicard4 [OF zero_less_one holf, of B])
  fix e::real
  assume 0 < e e < 1
  obtain n where (1/e - 2) / 2 < real n
    using reals_Archimedean2 by blast
  also have ... ≤ r n
    using ‹strict_mono r› by (simp add: seq_suble)
  finally have (1/e - 2) / 2 < real (r n) .
  with ‹0 < e› have e: e > 1 / (2 + 2 * real (r n))
    by (simp add: field_simps)
  show ∃ d>0. d < e ∧ (∀ z∈sphere 0 d. cmod (f z) ≤ B)
    apply (rule_tac x=1 / (2 * (1 + of_nat (r n))) in exI)
    using normf_le_B by (simp add: e)
qed blast
then have ε: cmod (f z) ≤ |B| + 1 if cmod z < ε z ≠ 0 for z
  using that by fastforce
have 0 < |B| + 1
  by simp
then show ?thesis
  using ε by (force intro!: that [OF ‹0 < ε› ‹ε < 1›])
next
case 2
with infinite_enumerate obtain r :: nat ⇒ nat

```

```

    where strict_mono r and r:  $\bigwedge n. r\ n \in \{n. \text{norm}(h\ n\ w) \geq 1\}$ 
  by blast
  obtain B where B:  $\bigwedge j\ z. [\text{norm}\ z = 1/2; j \in \text{range}(\lambda n. \text{inverse} \circ h\ (r\ n))]$ 
 $\implies \text{norm}(j\ z) \leq B$ 
  proof (rule GPicard3 [OF _ _ w, where K = sphere 0 (1/2)])
    show range  $(\lambda n. \text{inverse} \circ h\ (r\ n)) \subseteq$ 
       $\{g. g \text{ holomorphic\_on ball } 0\ 1 - \{0\} \wedge (\forall z \in \text{ball } 0\ 1 - \{0\}. g\ z \neq 0 \wedge$ 
 $g\ z \neq 1)\}$ 
    using h01 by (auto intro!: holomorphic_intros holomorphic_on_compose_gen
      [unfolded o_def, OF _ holh] holomorphic_on_compose)
    show connected (ball 0 1 - {0::complex})
      by (simp add: connected_open_delete)
    show  $\bigwedge j. j \in \text{range}(\lambda n. \text{inverse} \circ h\ (r\ n)) \implies \text{cmod}(j\ w) \leq 1$ 
      using r norm_inverse_le_norm by fastforce
  qed (use r in auto)
  have norm_if_le_B:  $\text{cmod}(\text{inverse}(f\ z)) \leq B$  if  $\text{norm}\ z = 1 / (2 * (1 +$ 
 $\text{of\_nat}(r\ n)))$  for z
  proof -
    have *:  $\text{inverse}(\text{cmod}((f\ (z / (1 + \text{of\_nat}(r\ n)))))) \leq B$  if  $\text{norm}\ z = 1/2$ 
  for z
    using B [OF that] by (force simp: norm_inverse h_def)
    have half:  $\text{norm}(z * (1 + \text{of\_nat}(r\ n))) = 1/2$ 
      by (simp add: norm_mult divide_simps that)
    show ?thesis
      using * [OF half] by (simp add: norm_inverse)
  qed
  have hol_if:  $(\text{inverse} \circ f)$  holomorphic_on (ball 0 1 - {0})
    by (metis (no_types, lifting) hol_comp_apply f01 holomorphic_on_inverse
      holomorphic_transform)
  obtain  $\varepsilon$  where  $0 < \varepsilon < 1$  and leB:  $\bigwedge z. z \in \text{ball } 0\ \varepsilon - \{0\} \implies \text{cmod}((\text{inverse}$ 
 $\circ f)\ z) \leq B$ 
  proof (rule GPicard4 [OF zero_less_one hol_if, of B])
    fix e::real
    assume  $0 < e < 1$ 
    obtain n where  $(1/e - 2) / 2 < \text{real}\ n$ 
      using reals_Archimedean2 by blast
    also have  $\dots \leq r\ n$ 
      using <strict_mono r> by (simp add: seq_suble)
    finally have  $(1/e - 2) / 2 < \text{real}(r\ n)$  .
    with  $\langle 0 < e \rangle$  have e:  $e > 1 / (2 + 2 * \text{real}(r\ n))$ 
      by (simp add: field_simps)
    show  $\exists d > 0. d < e \wedge (\forall z \in \text{sphere } 0\ d. \text{cmod}((\text{inverse} \circ f)\ z) \leq B)$ 
      apply (rule_tac x=1 / (2 * (1 + of_nat (r n))) in exI)
      using norm_if_le_B by (simp add: e)
  qed blast
  have  $\varepsilon$ :  $\text{cmod}(f\ z) \geq \text{inverse}\ B$  and  $B > 0$  if  $\text{cmod}\ z < \varepsilon\ z \neq 0$  for z
  proof -
    have inverse (cmod (f z))  $\leq B$ 
      using leB that by (simp add: norm_inverse)

```

```

moreover
have  $f\ z \neq 0$ 
  using  $\langle \varepsilon < 1 \rangle$  f01 that by auto
ultimately show  $\text{cmod } (f\ z) \geq \text{inverse } B$ 
  by (simp add: norm_inverse inverse_le_imp_le)
show  $B > 0$ 
  using  $\langle f\ z \neq 0 \rangle$   $\langle \text{inverse } (\text{cmod } (f\ z)) \leq B \rangle$  not_le order.trans by fastforce
qed
then have  $B > 0$ 
  by (metis  $\langle 0 < \varepsilon \rangle$  dense leI order.asym vector_choose_size)
then have  $\text{inverse } B > 0$ 
  by (simp add: field_split_simps)
then show ?thesis
  using  $\varepsilon$  that  $[OF\ \langle 0 < \varepsilon \rangle\ \langle \varepsilon < 1 \rangle]$ 
  by (metis Diff_iff dist_0_norm insert_iff mem_ball)
qed
qed

```

lemma *GPicard6*:

```

assumes open  $M\ z \in M\ a \neq 0$  and holf:  $f\ \text{holomorphic\_on } (M - \{z\})$ 
and f0a:  $\bigwedge w. w \in M - \{z\} \implies f\ w \neq 0 \wedge f\ w \neq a$ 
obtains  $r$  where  $0 < r$   $\text{ball } z\ r \subseteq M$ 
   $\text{bounded}(f\ '(\text{ball } z\ r - \{z\})) \vee$ 
   $\text{bounded}((\text{inverse} \circ f)\ '(\text{ball } z\ r - \{z\}))$ 
proof –
  obtain  $r$  where  $0 < r$  and  $r: \text{ball } z\ r \subseteq M$ 
    using assms openE by blast
  let  $?g = \lambda w. f\ (z + \text{of\_real } r * w) / a$ 
  obtain  $e\ B$  where  $0 < e\ e < 1\ 0 < B$ 
    and  $B: (\forall z \in \text{ball } 0\ e - \{0\}. \text{norm}(?g\ z) \leq B) \vee (\forall z \in \text{ball } 0\ e - \{0\}. \text{norm}(?g\ z) \geq B)$ 
  proof (rule GPicard5)
    show  $?g\ \text{holomorphic\_on } \text{ball } 0\ 1 - \{0\}$ 
    proof (intro holomorphic_intros holomorphic_on_compose_gen [unfolded o_def, OF_ holf])
      show  $(\lambda x. z + \text{complex\_of\_real } r * x)\ '(\text{ball } 0\ 1 - \{0\}) \subseteq M - \{z\}$ 
        using  $\langle 0 < r \rangle\ r$ 
        by (auto simp: dist_norm norm_mult subset_eq)
      qed (use  $\langle a \neq 0 \rangle$  in auto)
      show  $\bigwedge w. w \in \text{ball } 0\ 1 - \{0\} \implies f\ (z + \text{of\_real } r * w) / a \neq 0 \wedge f\ (z + \text{of\_real } r * w) / a \neq 1$ 
        using f0a  $\langle 0 < r \rangle\ \langle a \neq 0 \rangle\ r$ 
        by (auto simp: field_split_simps dist_norm norm_mult subset_eq)
      qed
    show ?thesis
  proof
    show  $0 < e * r$ 
      by (simp add:  $\langle 0 < e \rangle\ \langle 0 < r \rangle$ )
  qed

```

```

have ball z (e * r) ⊆ ball z r
  by (simp add: ⟨0 < r⟩ ⟨e < 1⟩ order.strict_implies_order subset_ball)
then show ball z (e * r) ⊆ M
  using r by blast
consider ∧z. z ∈ ball 0 e - {0} ⇒ norm(?g z) ≤ B | ∧z. z ∈ ball 0 e - {0}
⇒ norm(?g z) ≥ B
  using B by blast
then show bounded (f ‘ (ball z (e * r) - {z})) ∨
  bounded ((inverse ∘ f) ‘ (ball z (e * r) - {z}))
proof cases
case 1
have ‖dist z w < e * r; w ≠ z‖ ⇒ cmod (f w) ≤ B * norm a for w
  using ⟨a ≠ 0⟩ ⟨0 < r⟩ 1 [of (w - z) / r]
  by (simp add: norm_divide dist_norm field_split_simps)
then show ?thesis
  by (force simp: intro!: boundedI)
next
case 2
have ‖dist z w < e * r; w ≠ z‖ ⇒ cmod (f w) ≥ B * norm a for w
  using ⟨a ≠ 0⟩ ⟨0 < r⟩ 2 [of (w - z) / r]
  by (simp add: norm_divide dist_norm field_split_simps)
then have ‖dist z w < e * r; w ≠ z‖ ⇒ inverse (cmod (f w)) ≤ inverse (B
* norm a) for w
  by (metis ⟨0 < B⟩ ⟨a ≠ 0⟩ mult_pos_pos norm_inverse norm_inverse_le_norm
zero_less_norm_iff)
then show ?thesis
  by (force simp: norm_inverse intro!: boundedI)
qed
qed
qed

```

theorem great_Picard:

```

assumes open M z ∈ M a ≠ b and holf: f holomorphic_on (M - {z})
and fab: ∧w. w ∈ M - {z} ⇒ f w ≠ a ∧ f w ≠ b
obtains l where (f ⟶ l) (at z) ∨ ((inverse ∘ f) ⟶ l) (at z)
proof -
obtain r where 0 < r and zrM: ball z r ⊆ M
  and r: bounded((λz. f z - a) ‘ (ball z r - {z})) ∨
    bounded((inverse ∘ (λz. f z - a)) ‘ (ball z r - {z}))
proof (rule GPicard6 [OF ⟨open M⟩ ⟨z ∈ M⟩])
show b - a ≠ 0
  using assms by auto
show (λz. f z - a) holomorphic_on M - {z}
  by (intro holomorphic_intros holf)
qed (use fab in auto)
have holfb: f holomorphic_on ball z r - {z}
  using zrM by (auto intro: holomorphic_on_subset [OF holf])
have holfb_i: (λz. inverse(f z - a)) holomorphic_on ball z r - {z}

```

```

    using fab zrM by (fastforce intro!: holomorphic_intros holfb)
  show ?thesis
    using r
  proof
    assume bounded ((λz. f z - a) ' (ball z r - {z}))
    then obtain B where B: ∧w. w ∈ (λz. f z - a) ' (ball z r - {z}) ⇒ norm
w ≤ B
    by (force simp: bounded_iff)
    then have ∀x. x ≠ z ∧ dist x z < r ⟶ cmod (f x - a) ≤ B
    by (simp add: dist_commute)
    with ⟨0 < r⟩ have ∀F w in at z. cmod (f w - a) ≤ B
    by (force simp add: eventually_at)
    moreover have ∧x. cmod (f x - a) ≤ B ⇒ cmod (f x) ≤ B + cmod a
    by (metis add.commute add_le_cancel_right norm_triangle_sub order.trans)
    ultimately have ∃B. ∀F w in at z. cmod (f w) ≤ B
    by (metis (mono_tags, lifting) eventually_at)
    then obtain g where holg: g holomorphic_on ball z r and gf: ∧w. w ∈ ball
z r - {z} ⇒ g w = f w
    using ⟨0 < r⟩ holomorphic_on_extend_bounded [OF holfb] by auto
    then have g -z→ g z
    unfolding continuous_at [symmetric]
    using ⟨0 < r⟩ centre_in_ball field_differentiable_imp_continuous_at
    holomorphic_on_imp_differentiable_at by blast
    then have (f ⟶ g z) (at z)
    using Lim_transform_within_open [of g g z z]
    using ⟨0 < r⟩ centre_in_ball gf by blast
    then show ?thesis
    using that by blast
  next
    assume bounded((inverse ∘ (λz. f z - a)) ' (ball z r - {z}))
    then obtain B where B: ∧w. w ∈ (inverse ∘ (λz. f z - a)) ' (ball z r - {z})
⇒ norm w ≤ B
    by (force simp: bounded_iff)
    then have ∀x. x ≠ z ∧ dist x z < r ⟶ cmod (inverse (f x - a)) ≤ B
    by (simp add: dist_commute)
    with ⟨0 < r⟩ have ∀F w in at z. cmod (inverse (f w - a)) ≤ B
    by (auto simp add: eventually_at)
    then have ∃B. ∀F z in at z. cmod (inverse (f z - a)) ≤ B
    by blast
    then obtain g where holg: g holomorphic_on ball z r and gf: ∧w. w ∈ ball
z r - {z} ⇒ g w = inverse (f w - a)
    using ⟨0 < r⟩ holomorphic_on_extend_bounded [OF holfb_i] by auto
    then have gz: g -z→ g z
    unfolding continuous_at [symmetric]
    using ⟨0 < r⟩ centre_in_ball field_differentiable_imp_continuous_at
    holomorphic_on_imp_differentiable_at by blast
    have gnz: ∧w. w ∈ ball z r - {z} ⇒ g w ≠ 0
    using gf fab zrM by fastforce
    show ?thesis

```

```

proof (cases  $g\ z = 0$ )
  case True
    have *:  $\llbracket g \neq 0; \text{inverse } g = f - a \rrbracket \implies g / (1 + a * g) = \text{inverse } f$  for  $f$ 
    g::complex
    by (auto simp: field_simps)
    have (inverse  $\circ$   $f$ )  $-z \rightarrow 0$ 
    proof (rule Lim_transform_within_open [of  $\lambda w. g\ w / (1 + a * g\ w)$   $_{\_}$   $_{\_}$ 
    UNIV ball  $z\ r$ ])
      show ( $\lambda w. g\ w / (1 + a * g\ w)$ )  $-z \rightarrow 0$ 
      using True by (auto simp: intro!: tendsto_eq_intros gz)
      show  $\bigwedge x. \llbracket x \in \text{ball } z\ r; x \neq z \rrbracket \implies g\ x / (1 + a * g\ x) = (\text{inverse} \circ f)\ x$ 
      using * gf gnz by simp
    qed (use  $\langle 0 < r \rangle$  in auto)
    with that show ?thesis by blast
  next
    case False
    show ?thesis
    proof (cases  $1 + a * g\ z = 0$ )
      case True
        have ( $f \longrightarrow 0$ ) (at  $z$ )
        proof (rule Lim_transform_within_open [of  $\lambda w. (1 + a * g\ w) / g\ w$   $_{\_}$   $_{\_}$ 
         $_{\_}$  ball  $z\ r$ ])
          show ( $\lambda w. (1 + a * g\ w) / g\ w$ )  $-z \rightarrow 0$ 
          by (rule tendsto_eq_intros refl gz  $\langle g\ z \neq 0 \rangle$  | simp add: True) +
          show  $\bigwedge x. \llbracket x \in \text{ball } z\ r; x \neq z \rrbracket \implies (1 + a * g\ x) / g\ x = f\ x$ 
          using fab fab zrM by (fastforce simp add: gf field_split_simps)
        qed (use  $\langle 0 < r \rangle$  in auto)
        then show ?thesis
        using that by blast
      next
        case False
        have *:  $\llbracket g \neq 0; \text{inverse } g = f - a \rrbracket \implies g / (1 + a * g) = \text{inverse } f$  for  $f$ 
        g::complex
        by (auto simp: field_simps)
        have (inverse  $\circ$   $f$ )  $-z \rightarrow g\ z / (1 + a * g\ z)$ 
        proof (rule Lim_transform_within_open [of  $\lambda w. g\ w / (1 + a * g\ w)$   $_{\_}$   $_{\_}$ 
        UNIV ball  $z\ r$ ])
          show ( $\lambda w. g\ w / (1 + a * g\ w)$ )  $-z \rightarrow g\ z / (1 + a * g\ z)$ 
          using False by (auto simp: False intro!: tendsto_eq_intros gz)
          show  $\bigwedge x. \llbracket x \in \text{ball } z\ r; x \neq z \rrbracket \implies g\ x / (1 + a * g\ x) = (\text{inverse} \circ f)\ x$ 
          using * gf gnz by simp
        qed (use  $\langle 0 < r \rangle$  in auto)
        with that show ?thesis by blast
      qed
    qed
  qed

```

corollary *great_Picard_alt:*

assumes M : *open* M $z \in M$ **and** hol_f : f *holomorphic_on* $(M - \{z\})$
and non : $\bigwedge l. \neg (f \longrightarrow l) \text{ (at } z) \bigwedge l. \neg ((\text{inverse} \circ f) \longrightarrow l) \text{ (at } z)$
obtains a **where** $-\{a\} \subseteq f^{-1}(M - \{z\})$
unfolding *subset_iff_image_iff*
by (*metis* *great_Picard* [*OF* M hol_f] *non Compl_iff insertI1*)

corollary *great_Picard_infinite:*

assumes M : *open* M $z \in M$ **and** hol_f : f *holomorphic_on* $(M - \{z\})$
and non : $\bigwedge l. \neg (f \longrightarrow l) \text{ (at } z) \bigwedge l. \neg ((\text{inverse} \circ f) \longrightarrow l) \text{ (at } z)$
obtains a **where** $\bigwedge w. w \neq a \implies \text{infinite } \{x. x \in M - \{z\} \wedge f x = w\}$
proof –
have *False* **if** $a \neq b$ **and** ab : *finite* $\{x. x \in M - \{z\} \wedge f x = a\}$ *finite* $\{x. x \in M - \{z\} \wedge f x = b\}$ **for** $a \neq b$
proof –
have $finab$: *finite* $\{x. x \in M - \{z\} \wedge f x \in \{a, b\}\}$
using *finite_UnI* [*OF* ab] **unfolding** *mem_Collect_eq insert_iff empty_iff*
by (*simp add: conj_disj_distribL*)
obtain r **where** $0 < r$ **and** zrM : $\text{ball } z \ r \subseteq M$ **and** r : $\bigwedge x. \llbracket x \in M - \{z\}; f x \in \{a, b\} \rrbracket \implies x \notin \text{ball } z \ r$
proof –
obtain e **where** $e > 0$ **and** e : $\text{ball } z \ e \subseteq M$
using *assms openE* **by** *blast*
show *?thesis*
proof (*cases* $\{x \in M - \{z\}. f x \in \{a, b\}\} = \{\}$)
case *True*
then show *?thesis*
using $e < e > 0$ **that** **by** *fastforce*
next
case *False*
let $?r = \min e \ (\text{Min } (\text{dist } z^{-1} \{x \in M - \{z\}. f x \in \{a, b\}\}))$
show *?thesis*
proof
show $0 < ?r$
using *min_less_iff_conj Min_gr_iff finab False* $\langle 0 < e \rangle$ **by** *auto*
have $\text{ball } z \ ?r \subseteq \text{ball } z \ e$
by (*simp add: subset_ball*)
with e **show** $\text{ball } z \ ?r \subseteq M$ **by** *blast*
show $\bigwedge x. \llbracket x \in M - \{z\}; f x \in \{a, b\} \rrbracket \implies x \notin \text{ball } z \ ?r$
using *min_less_iff_conj Min_gr_iff finab False* $\langle 0 < e \rangle$ **by** *auto*
qed
qed
qed
have hol_fb : f *holomorphic_on* $(\text{ball } z \ r - \{z\})$
apply (*rule holomorphic_on_subset* [*OF* hol_f])
using zrM **by** *auto*
show *?thesis*
apply (*rule great_Picard* [*OF* *open_ball* $\langle a \neq b \rangle$ hol_fb])


```

    using non ⟨0 < r⟩ r z rM by auto
  qed
  with that show thesis
    by meson
  qed

theorem Casorati_Weierstrass:
  assumes open M z ∈ M f holomorphic_on (M - {z})
    and  $\bigwedge l. \neg (f \longrightarrow l) \text{ (at } z) \bigwedge l. \neg ((\text{inverse} \circ f) \longrightarrow l) \text{ (at } z)$ 
  shows  $\text{closure}(f^{-1}(M - \{z\})) = \text{UNIV}$ 
proof -
  obtain a where  $a: -\{a\} \subseteq f^{-1}(M - \{z\})$ 
    using great_Picard_alt [OF assms] .
  have  $\text{UNIV} = \text{closure}(-\{a\})$ 
    by (simp add: closure_interior)
  also have  $\dots \subseteq \text{closure}(f^{-1}(M - \{z\}))$ 
    by (simp add: a closure_mono)
  finally show ?thesis
    by blast
  qed
end

```

7 Moebius functions, Equivalents of Simply Connected Sets, Riemann Mapping Theorem

```

theory Riemann_Mapping
imports Great_Picard
begin

```

7.1 Moebius functions are biholomorphisms of the unit disc

definition *Moebius_function* :: $[\text{real}, \text{complex}, \text{complex}] \Rightarrow \text{complex}$ **where**
 $\text{Moebius_function} \equiv \lambda t w z. \exp(i * \text{of_real } t) * (z - w) / (1 - \text{cnj } w * z)$

lemma *Moebius_function_simple*:
 $\text{Moebius_function } 0 w z = (z - w) / (1 - \text{cnj } w * z)$
by (simp add: Moebius_function_def)

lemma *Moebius_function_eq_zero*:
 $\text{Moebius_function } t w w = 0$
by (simp add: Moebius_function_def)

lemma *Moebius_function_of_zero*:
 $\text{Moebius_function } t w 0 = - \exp(i * \text{of_real } t) * w$
by (simp add: Moebius_function_def)

lemma *Moebius_function_norm_lt_1*:

```

    assumes w1: norm w < 1 and z1: norm z < 1
    shows norm (Moebius_function t w z) < 1
  proof -
    have 1 - cnj w * z ≠ 0
      by (metis complex_cnj_cnj complex_mod_sqrt_Re_mult_cnj mult.commute
        mult_less_cancel_right1 norm_ge_zero norm_mult norm_one order.asym right_minus_eq
        w1 z1)
    then have VV: 1 - w * cnj z ≠ 0
      by (metis complex_cnj_cnj complex_cnj_mult complex_cnj_one right_minus_eq)
    then have 1 - norm (Moebius_function t w z) ^ 2 =
      ((1 - norm w ^ 2) / (norm (1 - cnj w * z) ^ 2)) * (1 - norm z ^ 2)
    apply (cases w)
    apply (cases z)
    apply (simp add: Moebius_function_def divide_simps norm_divide norm_mult)
    apply (simp add: complex_norm complex_diff complex_mult one_complex.code
      complex_cnj)
    apply (auto simp: algebra_simps power2_eq_square)
    done
    then have 1 - (cmod (Moebius_function t w z))^2 = (1 - cmod (w * w)) /
      (cmod (1 - cnj w * z))^2 * (1 - cmod (z * z))
    by (simp add: norm_mult power2_eq_square)
    moreover have 0 < 1 - cmod (z * z)
    by (metis (no_types) z1 diff_gt_0_iff_gt mult.left_neutral norm_mult_less)
    ultimately have 0 < 1 - norm (Moebius_function t w z) ^ 2
    using ‹1 - cnj w * z ≠ 0› w1 norm_mult_less by fastforce
    then show ?thesis
    using linorder_not_less by fastforce
  qed

```

```

lemma Moebius_function_holomorphic:
  assumes norm w < 1
  shows Moebius_function t w holomorphic_on ball 0 1
  proof -
    have *: 1 - z * w ≠ 0 if norm z < 1 for z
    proof -
      have norm (1::complex) ≠ norm (z * w)
      using assms that norm_mult_less by fastforce
      then show ?thesis by auto
    qed
    show ?thesis
    unfolding Moebius_function_def
    proof (intro holomorphic_intros)
      show ∧z. z ∈ ball 0 1 ⇒ 1 - cnj w * z ≠ 0
      by (metis * complex_cnj_cnj complex_cnj_mult complex_mod_sqrt_Re_mult_cnj mem_ball_0
        mult.commute mult_1 right_minus_eq)
    qed
  qed

```

```

lemma Moebius_function_compose:

```

```

assumes meq:  $-w1 = w2$  and  $\text{norm } w1 < 1$   $\text{norm } z < 1$ 
shows  $\text{Moebius\_function } 0 \ w1 \ (\text{Moebius\_function } 0 \ w2 \ z) = z$ 
proof -
  have  $\text{norm } w2 < 1$ 
    using assms by auto
  then have  $-w1 = z$  if  $\text{cnj } w2 * z = 1$ 
    by (metis assms(3) complex_mod_cnj_less_irrefl_mult.right_neutral_norm_mult_less_norm_one that)
  moreover have  $z=0$  if  $1 - \text{cnj } w2 * z = \text{cnj } w1 * (z - w2)$ 
  proof -
    have  $w2 * \text{cnj } w2 = 1$ 
      using that meq by (auto simp: algebra_simps)
    then show  $z = 0$ 
      using  $\langle \text{cmod } w2 < 1 \rangle$  complex_mod_sqrt_Re_mult_cnj by force
  qed
  moreover have  $z - w2 - w1 * (1 - \text{cnj } w2 * z) = z * (1 - \text{cnj } w2 * z - \text{cnj } w1 * (z - w2))$ 
    using meq by (fastforce simp: algebra_simps)
  ultimately
  show ?thesis
    by (simp add: Moebius_function_def divide_simps norm_divide norm_mult)
qed

```

lemma *ball_biholomorphism_exists*:

```

assumes  $a \in \text{ball } 0 \ 1$ 
obtains  $f \ g$  where  $f \ a = 0$ 
   $f \text{ holomorphic\_on } \text{ball } 0 \ 1$   $f \ ' \ \text{ball } 0 \ 1 \subseteq \text{ball } 0 \ 1$ 
   $g \text{ holomorphic\_on } \text{ball } 0 \ 1$   $g \ ' \ \text{ball } 0 \ 1 \subseteq \text{ball } 0 \ 1$ 
   $\bigwedge z. z \in \text{ball } 0 \ 1 \implies f \ (g \ z) = z$ 
   $\bigwedge z. z \in \text{ball } 0 \ 1 \implies g \ (f \ z) = z$ 

```

proof

```

show  $\text{Moebius\_function } 0 \ a \text{ holomorphic\_on } \text{ball } 0 \ 1$   $\text{Moebius\_function } 0 \ (-a)$ 
 $\text{holomorphic\_on } \text{ball } 0 \ 1$ 
  using Moebius_function_holomorphic assms mem_ball_0 by auto
show  $\text{Moebius\_function } 0 \ a \ a = 0$ 
  by (simp add: Moebius_function_eq_zero)
show  $\text{Moebius\_function } 0 \ a \ ' \ \text{ball } 0 \ 1 \subseteq \text{ball } 0 \ 1$ 
   $\text{Moebius\_function } 0 \ (-a) \ ' \ \text{ball } 0 \ 1 \subseteq \text{ball } 0 \ 1$ 
  using Moebius_function_norm_lt_1 assms by auto
show  $\text{Moebius\_function } 0 \ a \ (\text{Moebius\_function } 0 \ (-a) \ z) = z$ 
   $\text{Moebius\_function } 0 \ (-a) \ (\text{Moebius\_function } 0 \ a \ z) = z$  if  $z \in \text{ball } 0 \ 1$  for
 $z$ 
  using Moebius_function_compose assms that by auto
qed

```

7.2 A big chain of equivalents of simple connectedness for an open set

lemma *biholomorphic_to_disc_aux*:

```

assumes open S connected S 0 ∈ S and S01: S ⊆ ball 0 1
and prev: ⋀f. [f holomorphic_on S; ⋀z. z ∈ S ⟹ f z ≠ 0; inj_on f S]
    ⟹ ∃ g. g holomorphic_on S ∧ (∀ z ∈ S. f z = (g z)2)
shows ∃ f g. f holomorphic_on S ∧ g holomorphic_on ball 0 1 ∧
    (∀ z ∈ S. f z ∈ ball 0 1 ∧ g(f z) = z) ∧
    (∀ z ∈ ball 0 1. g z ∈ S ∧ f(g z) = z)

proof –
  define F where F ≡ {h. h holomorphic_on S ∧ h ‘ S ⊆ ball 0 1 ∧ h 0 = 0 ∧
    inj_on h S}
  have idF: id ∈ F
    using S01 by (auto simp: F_def)
  then have F ≠ {}
    by blast
  have imF_ne: ((λh. norm(deriv h 0)) ‘ F) ≠ {}
    using idF by auto
  have holF: ⋀h. h ∈ F ⟹ h holomorphic_on S
    by (auto simp: F_def)
  obtain f where f ∈ F and normf: ⋀h. h ∈ F ⟹ norm(deriv h 0) ≤ norm(deriv
    f 0)
  proof –
    obtain r where r > 0 and r: ball 0 r ⊆ S
      using ⟨open S⟩ ⟨0 ∈ S⟩ openE by auto
    have bdd: bdd_above ((λh. norm(deriv h 0)) ‘ F)
    proof (intro bdd_aboveI exI ballI, clarify)
      show norm (deriv f 0) ≤ 1 / r if f ∈ F for f
    proof –
      have r01: (*) (complex_of_real r) ‘ ball 0 1 ⊆ S
        using that ⟨r > 0⟩ by (auto simp: norm_mult r [THEN subsetD])
      then have f holomorphic_on (*) (complex_of_real r) ‘ ball 0 1
        using holomorphic_on_subset [OF holF] by (simp add: that)
      then have holf: f ∘ (λz. (r * z)) holomorphic_on (ball 0 1)
        by (intro holomorphic_intros holomorphic_on_compose)
      have f0: (f ∘ (*)) (complex_of_real r) 0 = 0
        using F_def that by auto
      have f ‘ S ⊆ ball 0 1
        using F_def that by blast
      with r01 have fr1: ⋀z. norm z < 1 ⟹ norm ((f ∘ (*))(of_real r)) z < 1
        by force
      have *: ((λw. f (r * w)) has_field_derivative deriv f (r * z) * r) (at z)
        if z ∈ ball 0 1 for z::complex
        using DERIV_chain' [where g=f] ⟨open S⟩
        by (meson DERIV_cmult_Id ⟨f ∈ F⟩ holF holomorphic_derivI im-
          age_subset_iff
            r01 that)
      have df0: ((λw. f (r * w)) has_field_derivative deriv f 0 * r) (at 0)
        using * [of 0] by simp
      have deq: deriv (λx. f (complex_of_real r * x)) 0 = deriv f 0 * com-
        plex_of_real r
        using DERIV_imp_deriv df0 by blast
    proof

```

```

    have norm (deriv (f ∘ (*)) (complex_of_real r)) 0 ≤ 1
      by (auto intro: Schwarz_Lemma [OF holf f0 fr1, of 0])
    with ⟨r > 0⟩ show ?thesis
      by (simp add: deq norm_mult divide_simps o_def)
  qed
  qed
  define l where l ≡ SUP h∈F. norm (deriv h 0)
  have eql: norm (deriv f 0) = l if le: l ≤ norm (deriv f 0) and f ∈ F for f
  proof (rule order_antisym [OF _ le])
    show cmod (deriv f 0) ≤ l
      using ⟨f ∈ F⟩ bdd cSUP_upper by (fastforce simp: l_def)
  qed
  obtain F where Fin: ∧n. F n ∈ F and Flim: (λn. norm (deriv (F n) 0))
    → l
  proof -
    have ∃f. f ∈ F ∧ |norm (deriv f 0) - l| < 1 / (Suc n) for n
    proof -
      obtain f where f ∈ F and f: l < norm (deriv f 0) + 1 / (Suc n)
        using cSup_least [OF imF_ne, of l - 1 / (Suc n)] by (fastforce simp:
l_def)
      then have |norm (deriv f 0) - l| < 1 / (Suc n)
        by (fastforce simp: abs_if_not_less eql)
      with ⟨f ∈ F⟩ show ?thesis
        by blast
    qed
    then obtain F where fF: ∧n. (F n) ∈ F
      and fless: ∧n. |norm (deriv (F n) 0) - l| < 1 / (Suc n)
      by metis
    have (λn. norm (deriv (F n) 0)) → l
    proof (rule metric_LIMSEQ_I)
      fix e::real
      assume e > 0
      then obtain N::nat where N: e > 1 / (Suc N)
        using nat_approx_posE by blast
      show ∃ N. ∀ n ≥ N. dist (norm (deriv (F n) 0)) l < e
      proof (intro exI allI impI)
        fix n assume N ≤ n
        have dist (norm (deriv (F n) 0)) l < 1 / (Suc n)
          using fless by (simp add: dist_norm)
        also have ... < e
          using N ⟨N ≤ n⟩ inverse_of_nat_le_le_less_trans by blast
        finally show dist (norm (deriv (F n) 0)) l < e .
      qed
    qed
  qed
  with fF show ?thesis
    using that by blast
  qed
  have ∧K. [compact K; K ⊆ S] ⇒ ∃ B. ∀ h∈F. ∀ z∈K. norm (h z) ≤ B
  by (rule_tac x=1 in exI) (force simp: F_def)

```

```

moreover have range  $\mathcal{F} \subseteq F$ 
using  $\langle \bigwedge n. \mathcal{F} \ n \in F \rangle$  by blast
ultimately obtain  $f$  and  $r :: \text{nat} \Rightarrow \text{nat}$ 
where  $\text{hol}f$ :  $f$  holomorphic_on  $S$  and  $r$ : strict_mono  $r$ 
and  $\text{lim}f$ :  $\bigwedge x. x \in S \implies (\lambda n. \mathcal{F} \ (r \ n) \ x) \longrightarrow f \ x$ 
and  $\text{ulim}f$ :  $\bigwedge K. \llbracket \text{compact } K; K \subseteq S \rrbracket \implies \text{uniform\_limit } K \ (\mathcal{F} \circ r) \ f$ 
sequentially
using Montel [of  $S \ F \ \mathcal{F}$ , OF  $\langle \text{open } S \rangle \ \text{hol}f$ ] by auto+
have  $\text{der}$ :  $\bigwedge n \ x. x \in S \implies ((\mathcal{F} \circ r) \ n \ \text{has\_field\_derivative } ((\lambda n. \text{deriv } (\mathcal{F} \ n)) \circ r) \ n \ x) \ (\text{at } x)$ 
using  $\langle \bigwedge n. \mathcal{F} \ n \in F \rangle \langle \text{open } S \rangle \ \text{hol}f$  holomorphic_derivI by fastforce
have  $\text{ulim}$ :  $\bigwedge x. x \in S \implies \exists d > 0. \text{cball } x \ d \subseteq S \wedge \text{uniform\_limit } (\text{cball } x \ d) \ (\mathcal{F} \circ r) \ f$  sequentially
by (meson  $\text{ulim}f \ \langle \text{open } S \rangle \ \text{compact\_cball\_open\_contains\_cball}$ )
obtain  $f' :: \text{complex} \Rightarrow \text{complex}$  where  $f'$ :  $(f \ \text{has\_field\_derivative } f' \ 0) \ (\text{at } 0)$ 
and  $\text{tof}'0$ :  $(\lambda n. ((\lambda n. \text{deriv } (\mathcal{F} \ n)) \circ r) \ n \ 0) \longrightarrow f' \ 0$ 
using has_complex_derivative_uniform_sequence [OF  $\langle \text{open } S \rangle \ \text{der} \ \text{ulim}$ ]  $\langle 0 \in S \rangle$  by metis
then have  $\text{der}f0$ :  $\text{deriv } f \ 0 = f' \ 0$ 
by (simp add: DERIV_imp_deriv)
have  $f$  field_differentiable  $(\text{at } 0)$ 
using field_differentiable_def  $f'$  by blast
have  $(\lambda x. (\text{norm } (\text{deriv } (\mathcal{F} \ (r \ x)) \ 0))) \longrightarrow \text{norm } (\text{deriv } f \ 0)$ 
using isCont_tendsto_compose [OF continuous_norm [OF continuous_ident]  $\text{tof}'0$ ]  $\text{der}f0$  by auto
with LIMSEQ_subseq_LIMSEQ [OF Flim  $r$ ] have  $\text{no\_df}0$ :  $\text{norm}(\text{deriv } f \ 0) = l$ 
by (force simp: o_def intro: tendsto_unique)
have  $\text{nonconst}f$ :  $\neg f$  constant_on  $S$ 
using  $\langle \text{open } S \rangle \ \langle 0 \in S \rangle \ \text{no\_df}0$  holomorphic_nonconstant [OF  $\text{hol}f$ ] eql [OF  $\_idF$ ]
by force
show ?thesis
proof
show  $f \in F$ 
unfolding  $F\_def$ 
proof (intro CollectI conjI  $\text{hol}f$ )
have  $\text{norm}(f \ z) \leq 1$  if  $z \in S$  for  $z$ 
proof (intro Lim_norm_ubound [OF  $\_limf$ ] always_eventually_allI that)
fix  $n$ 
have  $\mathcal{F} \ (r \ n) \in F$ 
by (simp add: Fin)
then show  $\text{norm } (\mathcal{F} \ (r \ n) \ z) \leq 1$ 
using that by (auto simp:  $F\_def$ )
qed simp
then have  $\text{fless}1$ :  $\text{norm}(f \ z) < 1$  if  $z \in S$  for  $z$ 
using maximum_modulus_principle [OF  $\text{hol}f \ \langle \text{open } S \rangle \ \langle \text{connected } S \rangle \ \langle \text{open } S \rangle$ ] nonconstf that
by fastforce

```

```

    then show  $f' S \subseteq \text{ball } 0 \ 1$ 
      by auto
    have  $(\lambda n. \mathcal{F} (r \ n) \ 0) \longrightarrow 0$ 
      using  $\mathcal{F}in$  by (auto simp:  $F\_def$ )
    then show  $f \ 0 = 0$ 
      using  $tendsto\_unique$  [ $OF \_ \limf$ ]  $\langle 0 \in S \rangle$   $trivial\_limit\_sequentially$  by
blast
  show  $inj\_on \ f \ S$ 
  proof (rule  $Hurwitz\_injective$  [ $OF \langle open \ S \rangle \langle connected \ S \rangle \_ \ hol f$ ])
    show  $\bigwedge n. (\mathcal{F} \circ r) \ n \ holomorphic\_on \ S$ 
      by (simp add:  $\mathcal{F}in \ hol F$ )
    show  $\bigwedge K. [\![compact \ K; K \subseteq S]\!] \implies uniform\_limit \ K (\mathcal{F} \circ r) \ f \ sequentially$ 
      by (metis  $ulimf$ )
    show  $\neg f \ constant\_on \ S$ 
      using  $nonconstf$  by auto
    show  $\bigwedge n. inj\_on ((\mathcal{F} \circ r) \ n) \ S$ 
      using  $\mathcal{F}in$  by (auto simp:  $F\_def$ )
  qed
  qed
  show  $\bigwedge h. h \in F \implies norm (deriv \ h \ 0) \leq norm (deriv \ f \ 0)$ 
    by (metis  $eql \ le\_cases \ no\_df0$ )
  qed
  qed
  have  $hol f: f \ holomorphic\_on \ S$  and  $inj f: inj\_on \ f \ S$  and  $f01: f' S \subseteq \text{ball } 0 \ 1$ 
    using  $\langle f \in F \rangle$  by (auto simp:  $F\_def$ )
  obtain  $g$  where  $hol g: g \ holomorphic\_on (f' S)$ 
    and  $der g: \bigwedge z. z \in S \implies deriv \ f \ z * deriv \ g (f \ z) = 1$ 
    and  $gf: \bigwedge z. z \in S \implies g(f \ z) = z$ 
    using  $holomorphic\_has\_inverse$  [ $OF \ hol f \langle open \ S \rangle \ inj f$ ] by metis
  have  $\text{ball } 0 \ 1 \subseteq f' S$ 
  proof
    fix  $a::complex$ 
    assume  $a: a \in \text{ball } 0 \ 1$ 
    have  $False$  if  $\bigwedge x. x \in S \implies f \ x \neq a$ 
    proof -
      obtain  $h \ k$  where  $h \ a = 0$ 
        and  $hol h: h \ holomorphic\_on \ \text{ball } 0 \ 1$  and  $h01: h' \ \text{ball } 0 \ 1 \subseteq \text{ball } 0 \ 1$ 
        and  $hol k: k \ holomorphic\_on \ \text{ball } 0 \ 1$  and  $k01: k' \ \text{ball } 0 \ 1 \subseteq \text{ball } 0 \ 1$ 
        and  $hk: \bigwedge z. z \in \text{ball } 0 \ 1 \implies h (k \ z) = z$ 
        and  $kh: \bigwedge z. z \in \text{ball } 0 \ 1 \implies k (h \ z) = z$ 
        using  $ball\_biholomorphism\_exists$  [ $OF \ a$ ] by blast
      have  $nf1: \bigwedge z. z \in S \implies norm(f \ z) < 1$ 
        using  $\langle f \in F \rangle$  by (auto simp:  $F\_def$ )
      have  $1: h \circ f \ holomorphic\_on \ S$ 
      using  $F\_def \langle f \in F \rangle \ hol h \ holomorphic\_on\_compose \ holomorphic\_on\_subset$ 
    by blast
      have  $2: \bigwedge z. z \in S \implies (h \circ f) \ z \neq 0$ 
        by (metis  $\langle h \ a = 0 \rangle \ a \ comp\_eq\_dest\_lhs \ nf1 \ kh \ mem\_ball\_0 \ that$ )
      have  $3: inj\_on (h \circ f) \ S$ 

```

```

    by (metis (no_types, lifting) F_def ‹f ∈ F› comp_inj_on inj_on_inverseI
    injf kh mem_Collect_eq inj_on_subset)
  obtain ψ where holψ: ψ holomorphic_on ((h ∘ f) ‘ S)
    and ψ2:  $\bigwedge z. z \in S \implies \psi(h(fz)) \wedge^2 = h(fz)$ 
  proof (rule exE [OF prev [OF 1 2 3]], safe)
    fix v
    assume holv: v holomorphic_on S and v2:  $(\forall z \in S. (h \circ f) z = (v z)^2)$ 
    show thesis
  proof
    show (v ∘ g ∘ k) holomorphic_on (h ∘ f) ‘ S
    proof (intro holomorphic_on_compose)
      show k holomorphic_on (h ∘ f) ‘ S
        using holomorphic_on_subset [OF holk] f01 h01 by force
      show g holomorphic_on k ‘ (h ∘ f) ‘ S
        using holomorphic_on_subset [OF holg] by (force simp: kh nf1)
      show v holomorphic_on g ‘ k ‘ (h ∘ f) ‘ S
        using holomorphic_on_subset [OF holv] by (force simp: gf kh nf1)
    qed
    show ((v ∘ g ∘ k) (h(fz)))2 = h(fz) if z ∈ S for z
      using v2 gf kh nf1 that by fastforce
    qed
  qed
  have normψ1: norm(ψ (h(fz))) < 1 if z ∈ S for z
  by (metis ψ2 h01 image_subset_iff mem_ball_0 nf1 norm_power power_less1_D
  that)
  then have ψ01: ψ (h(f0)) ∈ ball 0 1
    by (simp add: ‹0 ∈ S›)
  obtain p q where p0: p (ψ (h(f0))) = 0
    and holp: p holomorphic_on ball 0 1 and p01: p ‘ ball 0 1 ⊆ ball 0 1
    and holq: q holomorphic_on ball 0 1 and q01: q ‘ ball 0 1 ⊆ ball 0 1
    and pq:  $\bigwedge z. z \in \text{ball } 0 \ 1 \implies p(qz) = z$ 
    and qp:  $\bigwedge z. z \in \text{ball } 0 \ 1 \implies q(pz) = z$ 
    using ball_biholomorphism_exists [OF ψ01] by metis
  have p ∘ ψ ∘ h ∘ f ∈ F
    unfolding F_def
  proof (intro CollectI conjI holp)
    show p ∘ ψ ∘ h ∘ f holomorphic_on S
    proof (intro holomorphic_on_compose holp)
      show h holomorphic_on f ‘ S
        using holomorphic_on_subset [OF holh] f01 by fastforce
      show ψ holomorphic_on h ‘ f ‘ S
        using holomorphic_on_subset [OF holψ] by fastforce
      show p holomorphic_on ψ ‘ h ‘ f ‘ S
        using holomorphic_on_subset [OF holp] by (simp add: image_subset_iff
    normψ1)
    qed
  qed
  show (p ∘ ψ ∘ h ∘ f) ‘ S ⊆ ball 0 1
    using normψ1 p01 by fastforce
  show (p ∘ ψ ∘ h ∘ f) 0 = 0

```



```

    by (simp add: ⟨p (ψ (h (f 0))) = 0⟩)
  show inj_on (p ∘ ψ ∘ h ∘ f) S
    unfolding inj_on_def o_def
    by (metis ψ2 dist_0_norm gf kh mem_ball nf1 normψ1 qp)
qed
then have le_norm_df0: norm (deriv (p ∘ ψ ∘ h ∘ f) 0) ≤ norm (deriv f 0)
  by (rule normf)
have 1: k ∘ power2 ∘ q holomorphic_on ball 0 1
proof (intro holomorphic_on_compose holq)
  show power2 holomorphic_on q ‘ ball 0 1
    using holomorphic_on_subset holomorphic_on_power
    by (blast intro: holomorphic_on_ident)
  show k holomorphic_on power2 ‘ q ‘ ball 0 1
    using q01 holomorphic_on_subset [OF holk]
    by (force simp: norm_power abs_square_less_1)
qed
have 2: (k ∘ power2 ∘ q) 0 = 0
  using p0 F_def ⟨f ∈ F⟩ ψ01 ψ2 ⟨0 ∈ S⟩ kh qp by force
have 3: norm ((k ∘ power2 ∘ q) z) < 1 if norm z < 1 for z
proof -
  have norm ((power2 ∘ q) z) < 1
    using that q01 by (force simp: norm_power abs_square_less_1)
  with k01 show ?thesis
    by fastforce
qed
have False if c: ∀ z. norm z < 1 ⟶ (k ∘ power2 ∘ q) z = c * z and norm
c = 1 for c
proof -
  have c ≠ 0 using that by auto
  have norm (p(1/2)) < 1 norm (p(-1/2)) < 1
    using p01 by force+
  then have (k ∘ power2 ∘ q) (p(1/2)) = c * p(1/2) (k ∘ power2 ∘ q)
(p(-1/2)) = c * p(-1/2)
    using c by force+
  then have p (1/2) = p (- (1/2))
    by (auto simp: ⟨c ≠ 0⟩ qp o_def)
  then have q (p (1/2)) = q (p (- (1/2)))
    by simp
  then have 1/2 = - (1/2::complex)
    by (auto simp: qp)
  then show False
    by simp
qed
moreover
have False if norm (deriv (k ∘ power2 ∘ q) 0) ≠ 1 norm (deriv (k ∘ power2
∘ q) 0) ≤ 1
  and le: ⋀ξ. norm ξ < 1 ⟹ norm ((k ∘ power2 ∘ q) ξ) ≤ norm ξ
proof -
  have norm (deriv (k ∘ power2 ∘ q) 0) < 1

```

```

      using that by simp
      moreover have eq: deriv f 0 = deriv (k ∘ (λz. z ^ 2) ∘ q) 0 * deriv (p ∘
ψ ∘ h ∘ f) 0
      proof (intro DERIV_imp_deriv has_field_derivative_transform_within_open
[OF DERIV_chain])
        show (k ∘ power2 ∘ q has_field_derivative deriv (k ∘ power2 ∘ q) 0) (at
((p ∘ ψ ∘ h ∘ f) 0))
          using 1 holomorphic_derivI p0 by auto
          show (p ∘ ψ ∘ h ∘ f has_field_derivative deriv (p ∘ ψ ∘ h ∘ f) 0) (at 0)
            using ⟨p ∘ ψ ∘ h ∘ f ∈ F⟩ ⟨open S⟩ ⟨0 ∈ S⟩ holF holomorphic_derivI
by blast
          show ∧x. x ∈ S ⇒ (k ∘ power2 ∘ q ∘ (p ∘ ψ ∘ h ∘ f)) x = f x
            using ψ2 f01 kh normψ1 qp by auto
          qed (use assms in simp_all)
          ultimately have cmod (deriv (p ∘ ψ ∘ h ∘ f) 0) ≤ 0
            using le_norm_df0
          by (metis linorder_not_le mult.commute mult_less_cancel_left2 norm_mult)
          moreover have 1 ≤ norm (deriv f 0)
            using normf [of id] by (simp add: idF)
          ultimately show False
            by (simp add: eq)
          qed
          ultimately show ?thesis
            using Schwarz_Lemma [OF 1 2 3] norm_one by blast
          qed
          then show a ∈ f ' S
            by blast
          qed
          then have fS: f ' S = ball 0 1
            using F_def ⟨f ∈ F⟩ by blast
          then have ∀z∈ball 0 1. g z ∈ S ∧ f (g z) = z
            by (metis gf_imageE)
          with fS show ?thesis
            by (metis gf_holf_holg_image_eqI)
        qed

```

```

locale SC_Chain =
  fixes S :: complex set
  assumes openS: open S
begin

```

```

lemma winding_number_zero:
  assumes simply_connected S
  shows connected S ∧
    (∀γ z. path γ ∧ path_image γ ⊆ S ∧
      pathfinish γ = pathstart γ ∧ z ∉ S ⟶ winding_number γ z = 0)
  using assms
  by (auto simp: simply_connected_imp_connected simply_connected_imp_winding_number_zero)

```

lemma *contour_integral_zero*:

assumes *valid_path* g *path_image* $g \subseteq S$ *pathfinish* $g = \text{pathstart } g$ *f holomorphic_on* S

$\bigwedge \gamma z. \llbracket \text{path } \gamma; \text{path_image } \gamma \subseteq S; \text{pathfinish } \gamma = \text{pathstart } \gamma; z \notin S \rrbracket \implies \text{winding_number } \gamma z = 0$

shows $(f \text{ has_contour_integral } 0) \ g$

using *assms* **by** (*meson Cauchy_theorem_global openS valid_path_imp_path*)

lemma *global_primitive*:

assumes *connected* S **and** *holf*: $f \text{ holomorphic_on } S$

and *prev*: $\bigwedge \gamma f. \llbracket \text{valid_path } \gamma; \text{path_image } \gamma \subseteq S; \text{pathfinish } \gamma = \text{pathstart } \gamma; f \text{ holomorphic_on } S \rrbracket \implies (f \text{ has_contour_integral } 0) \ \gamma$

shows $\exists h. \forall z \in S. (h \text{ has_field_derivative } f z) \text{ (at } z)$

proof (*cases* $S = \{\}$)

case *True* **then show** *?thesis*

by *simp*

next

case *False*

then obtain a **where** $a \in S$

by *blast*

show *?thesis*

proof (*intro exI ballI*)

fix x **assume** $x \in S$

then obtain d **where** $d > 0$ **and** d : $\text{cball } x \ d \subseteq S$

using *openS open_contains_cball_eq* **by** *blast*

let $?g = \lambda z. (\text{SOME } g. \text{polynomial_function } g \wedge \text{path_image } g \subseteq S \wedge \text{pathstart } g = a \wedge \text{pathfinish } g = z)$

show $((\lambda z. \text{contour_integral } (?g \ z) \ f) \text{ has_field_derivative } f x) \text{ (at } x)$

proof (*simp add: has_field_derivative_def has_derivative_at2 bounded_linear_mult_right, rule Lim_transform*)

show $(\lambda y. \text{inverse}(\text{norm}(y - x)) *_{\mathbb{R}} (\text{contour_integral}(\text{linepath } x \ y) \ f - f x * (y - x))) \rightarrow x \ 0$

proof (*clarsimp simp add: Lim_at*)

fix $e::\text{real}$ **assume** $e > 0$

moreover have *continuous* (at x) f

using *openS* $\langle x \in S \rangle$ *holf continuous_on_eq_continuous_at holomorphic_on_imp_continuous_on* **by** *auto*

ultimately obtain $d1$ **where** $d1 > 0$

and $d1$: $\bigwedge x'. \text{dist } x' \ x < d1 \implies \text{dist } (f \ x') \ (f \ x) < e/2$

unfolding *continuous_at_eps_delta*

by (*metis less_divide_eq_numeral1(1) mult_zero_left*)

obtain $d2$ **where** $d2 > 0$ **and** $d2$: $\text{ball } x \ d2 \subseteq S$

using *openS* $\langle x \in S \rangle$ *open_contains_ball_eq* **by** *blast*

have $\text{inverse}(\text{norm}(y - x)) * \text{norm}(\text{contour_integral}(\text{linepath } x \ y) \ f - f x * (y - x)) < e$

if $0 < d1$ $0 < d2$ $y \neq x$ $\text{dist } y \ x < d1$ $\text{dist } y \ x < d2$ **for** y

proof –

```

      have f contour_integrable_on linepath x y
    proof (rule contour_integrable_continuous_linepath [OF continuous_on_subset])
      show continuous_on S f
        by (simp add: holf holomorphic_on_imp_continuous_on)
      have closed_segment x y  $\subseteq$  ball x d2
        by (meson dist_commute_lessI dist_in_closed_segment le_less_trans
mem_ball_subsetI that(5))
      with d2 show closed_segment x y  $\subseteq$  S
        by blast
    qed
    then obtain z where z: (f has_contour_integral z) (linepath x y)
      by (force simp: contour_integrable_on_def)
    have con: (( $\lambda w$ . f x) has_contour_integral f x * (y - x)) (linepath x y)
      using has_contour_integral_const_linepath [of f x y x] by metis
    have norm (z - f x * (y - x))  $\leq$  (e/2) * norm (y - x)
    proof (rule has_contour_integral_bound_linepath)
      show (( $\lambda w$ . f w - f x) has_contour_integral z - f x * (y - x)) (linepath
x y)
        by (rule has_contour_integral_diff [OF z con])
      show  $\bigwedge w. w \in \text{closed\_segment } x \ y \implies \text{norm } (f \ w - f \ x) \leq e/2$ 
        by (metis d1 dist_norm less_le_trans not_less not_less_iff_gr_or_eq
segment_bound1 that(4))
    qed (use  $\langle e > 0 \rangle$  in auto)
    with  $\langle e > 0 \rangle$  have inverse (norm (y - x)) * norm (z - f x * (y - x))
 $\leq e/2$ 
      by (simp add: field_split_simps)
    also have ...  $< e$ 
      using  $\langle e > 0 \rangle$  by simp
    finally show ?thesis
      by (simp add: contour_integral_unique [OF z])
    qed
    with  $\langle d1 > 0 \rangle \ \langle d2 > 0 \rangle$ 
    show  $\exists d > 0. \forall z. z \neq x \wedge \text{dist } z \ x < d \longrightarrow$ 
      inverse (norm (z - x)) * norm (contour_integral (linepath x z) f -
f x * (z - x))  $< e$ 
      by (rule_tac x=min d1 d2 in exI) auto
    qed
  next
    have *: (1 / norm (y - x)) *R (contour_integral (?g y) f -
      (contour_integral (?g x) f + f x * (y - x))) =
      (contour_integral (linepath x y) f - f x * (y - x)) /R norm (y - x)
    if  $0 < d \ y \neq x$  and  $yx: \text{dist } y \ x < d$  for y
    proof -
      have y  $\in S$ 
        by (metis subsetD d dist_commute less_eq_real_def mem_cball yx)
      have gxy: polynomial_function (?g x)  $\wedge$  path_image (?g x)  $\subseteq S \wedge$  pathstart
(?g x) = a  $\wedge$  pathfinish (?g x) = x
        polynomial_function (?g y)  $\wedge$  path_image (?g y)  $\subseteq S \wedge$  pathstart
(?g y) = a  $\wedge$  pathfinish (?g y) = y

```

```

      using someI_ex [OF connected_open_polynomial_connected [OF openS
⟨connected S⟩ ⟨a ∈ S⟩]] ⟨x ∈ S⟩ ⟨y ∈ S⟩
    by meson+
  then have vp: valid_path (?g x) valid_path (?g y)
    by (simp_all add: valid_path_polynomial_function)
    have f0: (f has_contour_integral 0) ((?g x) +++ linepath x y +++
reversepath (?g y))
  proof (rule prev)
    show valid_path ((?g x) +++ linepath x y +++ reversepath (?g y))
      using gxy vp by (auto simp: valid_path_join)
    have closed_segment x y ⊆ cball x d
      using yx by (auto simp: dist_commute dest!: dist_in_closed_segment)
    then have closed_segment x y ⊆ S
      using d by blast
    then show path_image ((?g x) +++ linepath x y +++ reversepath (?g
y)) ⊆ S
      using gxy by (auto simp: path_image_join)
    qed (use gxy holf in auto)
  then have fintry: f contour_integrable_on linepath x y
    using gxy(2) has_contour_integral_integrable vp by fastforce
  have fintgx: f contour_integrable_on (?g x) f contour_integrable_on (?g y)
    using openS contour_integrable_holomorphic_simple gxy holf vp by blast+
  show ?thesis
    apply (clarsimp simp add: divide_simps)
    using contour_integral_unique [OF f0]
  apply (simp add: fintry gxy contour_integrable_reversepath contour_integral_reversepath
fintgx vp)
    apply (simp add: algebra_simps)
    done
  qed
  show (λz. (1 / norm (z - x)) *R
(contour_integral (?g z) f - (contour_integral (?g x) f + f x * (z
- x))) -
(contour_integral (linepath x z) f - f x * (z - x)) /R norm (z - x))
    -x→ 0
    apply (rule tendsto_eventually)
    apply (simp add: eventually_at)
    apply (rule_tac x=d in exI)
    using ⟨d > 0⟩ * by simp
  qed
qed
qed
qed

lemma holomorphic_log:
  assumes connected S and holf: f holomorphic_on S and nz: ∧z. z ∈ S ⇒ f z
  ≠ 0
  and prev: ∧f. f holomorphic_on S ⇒ ∃ h. ∀ z ∈ S. (h has_field_derivative f z)
  (at z)
  shows ∃ g. g holomorphic_on S ∧ (∀ z ∈ S. f z = exp(g z))

```

```

proof –
  have ( $\lambda z. \text{deriv } f z / f z$ ) holomorphic_on S
    by (simp add: openS holf holomorphic_deriv holomorphic_on_divide nz)
  then obtain g where g:  $\bigwedge z. z \in S \implies (g \text{ has\_field\_derivative } \text{deriv } f z / f z)$ 
    (at z)
    using prev [of  $\lambda z. \text{deriv } f z / f z$ ] by metis
  have Df:  $\bigwedge x. x \in S \implies \text{DERIV } f x :> \text{deriv } f x$ 
    using holf holomorphic_derivI openS by force
  have hfd:  $\bigwedge x. x \in S \implies ((\lambda z. \text{exp } (g z) / f z) \text{ has\_field\_derivative } 0)$  (at x)
    by (rule derivative_eq_intros Df g nz| simp)+
  obtain c where c:  $\bigwedge x. x \in S \implies \text{exp } (g x) / f x = c$ 
  proof (rule DERIV_zero_connected_constant [OF ‹connected S› openS finite.emptyI])
    show continuous_on S ( $\lambda z. \text{exp } (g z) / f z$ )
      by (metis (full_types) openS g continuous_on_divide continuous_on_exp holf
holomorphic_on_imp_continuous_on holomorphic_on_open nz)
    then show  $\forall x \in S - \{ \}. ((\lambda z. \text{exp } (g z) / f z) \text{ has\_field\_derivative } 0)$  (at x)
      using hfd by (blast intro: DERIV_zero_connected_constant [OF ‹connected
S› openS finite.emptyI, of  $\lambda z. \text{exp } (g z) / f z$ ])
    qed auto
  show ?thesis
  proof (intro exI ballI conjI)
    have g holomorphic_on S
      using openS g holomorphic_on_open by blast
    then show ( $\lambda z. \text{Ln}(\text{inverse } c) + g z$ ) holomorphic_on S
      by (intro holomorphic_intros)
    fix z :: complex
    assume z  $\in S$ 
    then have  $\text{exp } (g z) / c = f z$ 
      by (metis c divide_divide_eq_right exp_not_eq_zero nonzero_mult_div_cancel_left)
    moreover have  $1 / c \neq 0$ 
      using ‹z  $\in S$ › c nz by fastforce
    ultimately show  $f z = \text{exp } (\text{Ln } (\text{inverse } c) + g z)$ 
      by (simp add: exp_add inverse_eq_divide)
    qed
  qed

```

lemma *holomorphic_sqrt*:

```

  assumes holf: f holomorphic_on S and nz:  $\bigwedge z. z \in S \implies f z \neq 0$ 
  and prev:  $\bigwedge f. \llbracket f \text{ holomorphic\_on } S; \forall z \in S. f z \neq 0 \rrbracket \implies \exists g. g \text{ holomorphic\_on } S \wedge (\forall z \in S. f z = \text{exp}(g z))$ 
  shows  $\exists g. g \text{ holomorphic\_on } S \wedge (\forall z \in S. f z = (g z)^2)$ 
proof –
  obtain g where holg: g holomorphic_on S and g:  $\bigwedge z. z \in S \implies f z = \text{exp } (g z)$ 
    (z)
    using prev [of f] holf nz by metis
  show ?thesis
  proof (intro exI ballI conjI)
    show ( $\lambda z. \text{exp}(g z / 2)$ ) holomorphic_on S
      by (intro holomorphic_intros) (auto simp: holg)

```

```

    show  $\bigwedge z. z \in S \implies f z = (\exp (g z / 2))^2$ 
    by (metis (no_types) g exp_double nonzero_mult_div_cancel_left times_divide_eq_right
    zero_neq_numeral)
  qed
qed

lemma biholomorphic_to_disc:
  assumes connected S and S:  $S \neq \{\}$   $S \neq \text{UNIV}$ 
  and prev:  $\bigwedge f. \llbracket f \text{ holomorphic\_on } S; \forall z \in S. f z \neq 0 \rrbracket \implies \exists g. g \text{ holomorphic\_on } S \wedge (\forall z \in S. f z = (g z)^2)$ 
  shows  $\exists f g. f \text{ holomorphic\_on } S \wedge g \text{ holomorphic\_on ball } 0 \ 1 \wedge$ 
     $(\forall z \in S. f z \in \text{ball } 0 \ 1 \wedge g(f z) = z) \wedge$ 
     $(\forall z \in \text{ball } 0 \ 1. g z \in S \wedge f(g z) = z)$ 
proof -
  obtain a b where  $a \in S$   $b \notin S$ 
  using S by blast
  then obtain  $\delta$  where  $\delta > 0$  and  $\delta: \text{ball } a \ \delta \subseteq S$ 
  using openS openE by blast
  obtain g where holg:  $g \text{ holomorphic\_on } S$  and eqg:  $\bigwedge z. z \in S \implies z - b = (g z)^2$ 
  proof (rule exE [OF prev [of  $\lambda z. z - b$ ]])
    show  $(\lambda z. z - b) \text{ holomorphic\_on } S$ 
    by (intro holomorphic_intros)
  qed (use  $\langle b \notin S \rangle$  in auto)
  have  $\neg g \text{ constant\_on } S$ 
  proof -
    have  $(a + \delta/2) \in \text{ball } a \ \delta$   $a + (\delta/2) \neq a$ 
    using  $\langle \delta > 0 \rangle$  by (simp_all add: dist_norm)
    then show ?thesis
    unfolding constant_on_def
    using eqg [of a] eqg [of  $a + \delta/2$ ]  $\langle a \in S \rangle \ \delta$ 
    by (metis diff_add_cancel subset_eq)
  qed
  then have open  $(g \text{ ` } \text{ball } a \ \delta)$ 
  using open_mapping_thm [of  $g \ S \ \text{ball } a \ \delta$ , OF holg openS  $\langle \text{connected } S \rangle$ ]  $\delta$  by
blast
  then obtain r where  $r > 0$  and  $r: \text{ball } (g a) \ r \subseteq (g \text{ ` } \text{ball } a \ \delta)$ 
  by (metis  $\langle 0 < \delta \rangle$  centre_in_ball imageI openE)
  have g_not_r:  $g z \notin \text{ball } (-(g a)) \ r$  if  $z \in S$  for  $z$ 
  proof
    assume  $g z \in \text{ball } (-(g a)) \ r$ 
    then have  $-g z \in \text{ball } (g a) \ r$ 
    by (metis add.inverse_inverse dist_minus mem_ball)
    with r have  $-g z \in (g \text{ ` } \text{ball } a \ \delta)$ 
    by blast
    then obtain w where  $w: -g z = g w$   $\text{dist } a \ w < \delta$ 
    by auto
    with  $\delta$  have  $w \in S$ 
    by force
  qed

```

```

then have  $w = z$ 
  by (metis diff_add_cancel eqg power_minus_Bit0 that  $w(1)$ )
then have  $g\ z = 0$ 
  using  $\langle -\ g\ z = g\ w \rangle$  by auto
with eqg that  $\langle b \notin S \rangle$  show False
  by force
qed
then have nz:  $\bigwedge z. z \in S \implies g\ z + g\ a \neq 0$ 
  by (metis  $\langle 0 < r \rangle$  add commute add_diff_cancel_left' centre_in_ball diff_0)
let ?f =  $\lambda z. (r/3) / (g\ z + g\ a) - (r/3) / (g\ a + g\ a)$ 
obtain h where holh:  $h$  holomorphic_on  $S$  and  $h\ a = 0$  and h01:  $h\ ' S \subseteq \text{ball}\ 0\ 1$  and inj_on h  $S$ 
proof
  show ?f holomorphic_on  $S$ 
    by (intro holomorphic_intros holg nz)
  have 3:  $\llbracket \text{norm}\ x \leq 1/3; \text{norm}\ y \leq 1/3 \rrbracket \implies \text{norm}(x - y) < 1$  for  $x\ y::\text{complex}$ 
    using norm_triangle_ineq4 [of  $x\ y$ ] by simp
  have norm  $((r/3) / (g\ z + g\ a) - (r/3) / (g\ a + g\ a)) < 1$  if  $z \in S$  for  $z$ 
    apply (rule 3)
    unfolding norm_divide
    using  $\langle r > 0 \rangle$  g_not_r [OF  $\langle z \in S \rangle$ ] g_not_r [OF  $\langle a \in S \rangle$ ]
    by (simp_all add: field_split_simps dist_commute dist_norm)
  then show ?f  $' S \subseteq \text{ball}\ 0\ 1$ 
    by auto
  show inj_on ?f  $S$ 
    using  $\langle r > 0 \rangle$  eqg apply (clarsimp simp: inj_on_def)
    by (metis diff_add_cancel)
qed auto
obtain k where holh:  $k$  holomorphic_on  $(h\ ' S)$ 
  and derk:  $\bigwedge z. z \in S \implies \text{deriv}\ h\ z * \text{deriv}\ k\ (h\ z) = 1$ 
  and kh:  $\bigwedge z. z \in S \implies k(h\ z) = z$ 
  using holomorphic_has_inverse [OF holh openS  $\langle \text{inj\_on}\ h\ S \rangle$ ] by metis

have 1: open  $(h\ ' S)$ 
  by (simp add:  $\langle \text{inj\_on}\ h\ S \rangle$  holh openS open_mapping_thm3)
have 2: connected  $(h\ ' S)$ 
  by (simp add: connected_continuous_image  $\langle \text{connected}\ S \rangle$  holh holomorphic_on_imp_continuous_on)
have 3:  $0 \in h\ ' S$ 
  using  $\langle a \in S \rangle$   $\langle h\ a = 0 \rangle$  by auto
have 4:  $\exists g. g$  holomorphic_on  $h\ ' S \wedge (\forall z \in h\ ' S. f\ z = (g\ z)^2)$ 
  if holh:  $f$  holomorphic_on  $h\ ' S$  and nz:  $\bigwedge z. z \in h\ ' S \implies f\ z \neq 0$  inj_on f  $(h\ ' S)$  for f
proof -
  obtain g where holg:  $g$  holomorphic_on  $S$  and eqg:  $\bigwedge z. z \in S \implies (f \circ h)\ z = (g\ z)^2$ 
  by (smt (verit) comp_def holh holh holomorphic_on_compose image_eqI nz(1) prev)
  show ?thesis
  proof (intro exI conjI)

```



```

    show  $g \circ k$  holomorphic_on  $h$  '  $S$ 
    by (smt (verit) holg holk holomorphic_on_compose holomorphic_on_subset
imageE image_subset_iff kh)
    show  $\forall z \in h$  '  $S. f z = ((g \circ k) z)^2$ 
    using eqg kh by auto
  qed
qed
obtain  $f g$  where  $f: f$  holomorphic_on  $h$  '  $S$  and  $g: g$  holomorphic_on ball 0 1
  and  $gf: \forall z \in h$  '  $S. f z \in \text{ball } 0 \ 1 \wedge g(f z) = z$  and  $fg: \forall z \in \text{ball } 0 \ 1. g z \in h$ 
  '  $S \wedge f(g z) = z$ 
  using biholomorphic_to_disc_aux [OF 1 2 3 h01 4] by blast
show ?thesis
proof (intro exI conjI)
  show  $f \circ h$  holomorphic_on  $S$ 
  by (simp add: f holh holomorphic_on_compose)
  show  $k \circ g$  holomorphic_on ball 0 1
  by (metis holomorphic_on_subset image_subset_iff fg holk g holomor-
phic_on_compose)
  qed (use fg gf kh in auto)
qed

lemma homeomorphic_to_disc:
  assumes  $S = \text{UNIV} \vee$ 
    ( $\exists f g. f$  holomorphic_on  $S \wedge g$  holomorphic_on ball 0 1  $\wedge$ 
      ( $\forall z \in S. f z \in \text{ball } 0 \ 1 \wedge g(f z) = z$ )  $\wedge$ 
      ( $\forall z \in \text{ball } 0 \ 1. g z \in S \wedge f(g z) = z$ )) (is  $\_ \vee ?P$ )
  shows  $S$  homeomorphic ball (0::complex) 1
  by (smt (verit, ccfv_SIG) holomorphic_on_imp_continuous_on homeomorphic_ball01_UNIV
    homeomorphic_minimal assms)

lemma homeomorphic_to_disc_imp_simply_connected:
  assumes  $S = \{\}$   $\vee S$  homeomorphic ball (0::complex) 1
  shows simply_connected  $S$ 
  using assms homeomorphic_simply_connected_eq convex_imp_simply_connected
  by auto

end

proposition
  assumes open  $S$ 
  shows simply_connected_eq_winding_number_zero:
    simply_connected  $S \iff$ 
      connected  $S \wedge$ 
      ( $\forall g z. \text{path } g \wedge \text{path\_image } g \subseteq S \wedge$ 
        pathfinish  $g = \text{pathstart } g \wedge (z \notin S)$ 
         $\longrightarrow \text{winding\_number } g z = 0$ ) (is ?wn0)
  and simply_connected_eq_contour_integral_zero:
    simply_connected  $S \iff$ 
      connected  $S \wedge$ 

```

```

      (∀ g f. valid_path g ∧ path_image g ⊆ S ∧
        pathfinish g = pathstart g ∧ f holomorphic_on S
        ⟶ (f has_contour_integral 0) g) (is ?ci0)
and simply_connected_eq_global_primitive:
  simply_connected S ⟷
    connected S ∧
    (∀ f. f holomorphic_on S ⟶
      (∃ h. ∀ z. z ∈ S ⟶ (h has_field_derivative f z) (at z))) (is ?gp)
and simply_connected_eq_holomorphic_log:
  simply_connected S ⟷
    connected S ∧
    (∀ f. f holomorphic_on S ∧ (∀ z ∈ S. f z ≠ 0)
      ⟶ (∃ g. g holomorphic_on S ∧ (∀ z ∈ S. f z = exp(g z)))) (is ?log)
and simply_connected_eq_holomorphic_sqrt:
  simply_connected S ⟷
    connected S ∧
    (∀ f. f holomorphic_on S ∧ (∀ z ∈ S. f z ≠ 0)
      ⟶ (∃ g. g holomorphic_on S ∧ (∀ z ∈ S. f z = (g z)2))) (is ?sqrt)
and simply_connected_eq_biholomorphic_to_disc:
  simply_connected S ⟷
    S = {} ∨ S = UNIV ∨
    (∃ f g. f holomorphic_on S ∧ g holomorphic_on ball 0 1 ∧
      (∀ z ∈ S. f z ∈ ball 0 1 ∧ g(f z) = z) ∧
      (∀ z ∈ ball 0 1. g z ∈ S ∧ f(g z) = z)) (is ?bih)
and simply_connected_eq_homeomorphic_to_disc:
  simply_connected S ⟷ S = {} ∨ S homeomorphic ball (0::complex) 1
(is ?disc)
proof -
  interpret SC_Chain
  using assms by (simp add: SC_Chain_def)
  have ?wn0 ∧ ?ci0 ∧ ?gp ∧ ?log ∧ ?sqrt ∧ ?bih ∧ ?disc
proof -
  have *: [α ⟹ β; β ⟹ γ; γ ⟹ δ; δ ⟹ ζ; ζ ⟹ η; η ⟹ ϑ; ϑ ⟹ ξ; ξ ⟹
α]
    ⟹ (α ⟷ β) ∧ (α ⟷ γ) ∧ (α ⟷ δ) ∧ (α ⟷ ζ) ∧
      (α ⟷ η) ∧ (α ⟷ ϑ) ∧ (α ⟷ ξ) for α β γ δ ζ η ϑ ξ
  by blast
show ?thesis
  apply (rule *)
  using winding_number_zero apply metis
  using contour_integral_zero apply metis
  using global_primitive apply metis
  using holomorphic_log apply metis
  using holomorphic_sqrt apply simp
  using biholomorphic_to_disc apply blast
  using homeomorphic_to_disc apply blast
  using homeomorphic_to_disc_imp_simply_connected apply blast
done
qed

```

```

    then show ?wn0 ?ci0 ?gp ?log ?sqrt ?bih ?disc
      by safe
qed

```

```

corollary contractible_eq_simply_connected_2d:
  fixes S :: complex set
  assumes open S
  shows contractible S  $\longleftrightarrow$  simply_connected S
proof
  show contractible S  $\implies$  simply_connected S
    by (simp add: contractible_imp_simply_connected)
  show simply_connected S  $\implies$  contractible S
    using assms convex_imp_contractible homeomorphic_contractible_eq
      simply_connected_eq_homeomorphic_to_disc by auto
qed

```

7.3 A further chain of equivalences about components of the complement of a simply connected set

(following 1.35 in Burckel'S book)

```

context SC_Chain
begin

```

```

lemma frontier_properties:
  assumes simply_connected S
  shows if bounded S then connected(frontier S)
        else  $\forall C \in \text{components}(\text{frontier } S). \neg \text{bounded } C$ 
proof -
  have S = {}  $\vee$  S homeomorphic ball (0::complex) 1
    using simply_connected_eq_homeomorphic_to_disc assms openS by blast
  then show ?thesis
  proof
    assume S = {}
    then show ?thesis
      by simp
  next
    assume S01: S homeomorphic ball (0::complex) 1
    then obtain g f
      where gim: g ' S = ball 0 1 and fg:  $\bigwedge x. x \in S \implies f(g\ x) = x$ 
        and fim: f ' ball 0 1 = S and gf:  $\bigwedge y. \text{cmod } y < 1 \implies g(f\ y) = y$ 
        and contg: continuous_on S g and conf: continuous_on (ball 0 1) f
      by (fastforce simp: homeomorphism_def homeomorphic_def)
    define D where D  $\equiv \lambda n. \text{ball } (0::\text{complex}) (1 - 1/(\text{of\_nat } n + 2))$ 
    define A where A  $\equiv \lambda n. \{z::\text{complex}. 1 - 1/(\text{of\_nat } n + 2) < \text{norm } z \wedge$ 
norm z < 1}
    define X where X  $\equiv \lambda n::\text{nat}. \text{closure}(f ' A\ n)$ 
    have D01: D n  $\subseteq$  ball 0 1 for n
      by (simp add: D_def ball_subset_ball_iff)
    have A01: A n  $\subseteq$  ball 0 1 for n

```

```

    by (auto simp: A_def)
  have cloX: closed (X n) for n
    by (simp add: X_def)
  have Xsubclo:  $X n \subseteq \text{closure } S$  for n
    unfolding X_def by (metis A01 closure_mono fim image_mono)
  have connected (A n) for n
    using connected_annulus [of _ 0::complex] by (simp add: A_def)
  then have connX: connected (X n) for n
    unfolding X_def
  by (metis A01 connected_continuous_image connected_imp_connected_closure
contf continuous_on_subset)
  have nestX:  $X n \subseteq X m$  if  $m \leq n$  for m n
  proof -
    have  $1 - 1 / (\text{real } m + 2) \leq 1 - 1 / (\text{real } n + 2)$ 
      using that by (auto simp: field_simps)
    then show ?thesis
      by (auto simp: X_def A_def intro!: closure_mono)
  qed
  have closure S - S  $\subseteq (\bigcap n. X n)$ 
  proof
    fix x
    assume  $x \in \text{closure } S - S$ 
    then have  $x \in \text{closure } S$   $x \notin S$  by auto
    show  $x \in (\bigcap n. X n)$ 
  proof
    fix n
    have ball 0 1 = closure (D n)  $\cup A n$ 
      by (auto simp: D_def A_def le_less_trans)
    with fim have Seq:  $S = f' (\text{closure } (D n)) \cup f' (A n)$ 
      by (simp add: image_Un)
    have continuous_on (closure (D n)) f
      by (simp add: D_def cball_subset_ball_iff continuous_on_subset [OF
contf])
    moreover have compact (closure (D n))
      by (simp add: D_def)
    ultimately have clo_fim: closed (f' closure (D n))
      using compact_continuous_image compact_imp_closed by blast
    have *:  $(f' \text{ cball } 0 (1 - 1 / (\text{real } n + 2))) \subseteq S$ 
      by (force simp: D_def Seq)
    show  $x \in X n$ 
      using Seq X_def  $\langle x \in \text{closure } S \rangle \langle x \notin S \rangle$  clo_fim by fastforce
  qed
  qed
  moreover have  $(\bigcap n. X n) \subseteq \text{closure } S - S$ 
  proof -
    have  $(\bigcap n. X n) \subseteq \text{closure } S$ 
      using Xsubclo by blast
    moreover have  $(\bigcap n. X n) \cap S \subseteq \{ \}$ 
    proof (clarify, clarsimp simp: X_def fim [symmetric])

```

```

fix x assume x [rule_format]:  $\forall n. f\ x \in \text{closure}\ (f\ 'A\ n)$  and  $\text{cmod}\ x < 1$ 
then obtain n where  $n: 1 / (1 - \text{norm}\ x) < \text{of\_nat}\ n$ 
  using reals_Archimedean2 by blast
with  $\langle \text{cmod}\ x < 1 \rangle$  gr0I have XX:  $1 / \text{of\_nat}\ n < 1 - \text{norm}\ x$  and  $n > 0$ 
  by (fastforce simp: field_split_simps algebra_simps)+
have  $f\ x \in f\ '(D\ n)$ 
  using n  $\langle \text{cmod}\ x < 1 \rangle$  by (auto simp: field_split_simps algebra_simps
D_def)
moreover have  $f\ 'D\ n \cap \text{closure}\ (f\ 'A\ n) = \{\}$ 
proof -
  have inj_on f (D n)
    unfolding inj_on_def using D01 by (metis gf mem_ball_0 subsetCE)
  then have op_fDn:  $\text{open}(f\ '(D\ n))$ 
  by (metis invariance_of_domain D_def Elementary_Metric_Spaces.open_ball

      continuous_on_subset [OF contf D01])
  have injf:  $\text{inj\_on}\ f\ (\text{ball}\ 0\ 1)$ 
    by (metis mem_ball_0 inj_on_def gf)
  have  $D\ n \cup A\ n \subseteq \text{ball}\ 0\ 1$ 
    using D01 A01 by simp
  moreover have  $D\ n \cap A\ n = \{\}$ 
    by (auto simp: D_def A_def)
  ultimately have  $f\ 'D\ n \cap f\ 'A\ n = \{\}$ 
    by (metis A01 D01 image_is_empty inj_on_image_Int injf)
  then show ?thesis
    by (simp add: open_Int_closure_eq_empty [OF op_fDn])
qed
ultimately show False
  using x [of n] by blast
qed
ultimately
show  $(\bigcap n. X\ n) \subseteq \text{closure}\ S - S$ 
  using closure_subset disjoint_iff_not_equal by blast
qed
ultimately have  $\text{closure}\ S - S = (\bigcap n. X\ n)$  by blast
then have frontierS:  $\text{frontier}\ S = (\bigcap n. X\ n)$ 
  by (simp add: frontier_def openS interior_open)
show ?thesis
proof (cases bounded S)
case True
  have bouX:  $\text{bounded}\ (X\ n)$  for n
    by (meson True Xsubclo bounded_closure bounded_subset)
  have compaX:  $\text{compact}\ (X\ n)$  for n
    by (simp add: bouX cloX compact_eq_bounded_closed)
  have connected  $(\bigcap n. X\ n)$ 
    by (metis nestX compaX connX connected_nest)
  then show ?thesis
    by (simp add: True  $\langle \text{frontier}\ S = (\bigcap n. X\ n) \rangle$ )
next

```

```

case False
have unboundedX:  $\neg \text{bounded}(X\ n)$  for n
proof
  assume bXn:  $\text{bounded}(X\ n)$ 
  have continuous_on (cball 0 ( $1 - 1 / (2 + \text{real } n)$ )) f
    by (simp add: cball_subset_ball_iff continuous_on_subset [OF contf])
  then have bounded (f ' cball 0 ( $1 - 1 / (2 + \text{real } n)$ )))
    by (simp add: compact_imp_bounded [OF compact_continuous_image])
  moreover have bounded (f ' A n)
    by (auto simp: X_def closure_subset_image_subset_iff bounded_subset
      [OF bXn])
  ultimately have bounded (f ' (cball 0 ( $1 - 1 / (2 + \text{real } n)$ ))  $\cup A\ n$ )
    by (simp add: image_Un)
  then have bounded (f ' ball 0 1)
    apply (rule bounded_subset)
    apply (auto simp: A_def algebra_simps)
  done
then show False
  using False by (simp add: fim [symmetric])
qed
have clo_INTX:  $\text{closed}(\bigcap (\text{range } X))$ 
  by (metis cloX closed_INT)
then have lcX: locally compact ( $\bigcap (\text{range } X)$ )
  by (metis closed_imp_locally_compact)
have False if C:  $C \in \text{components}(\text{frontier } S)$  and boC: bounded C for C
proof –
  have closed C
    by (metis C closed_components frontier_closed)
  then have compact C
    by (metis boC compact_eq_bounded_closed)
  have Cco:  $C \in \text{components}(\bigcap (\text{range } X))$ 
    by (metis frontierS C)
  obtain K where  $C \subseteq K$  compact K
    and Ksub:  $K \subseteq \bigcap (\text{range } X)$  and clo:  $\text{closed}(\bigcap (\text{range } X) - K)$ 
  proof (cases {k.  $C \subseteq k \wedge \text{compact } k \wedge \text{openin } (\text{top\_of\_set } (\bigcap (\text{range } X)))$ 
     $k\} = \{\}$ )
    case True
    then show ?thesis
      using Sura_Bura [OF lcX Cco  $\langle \text{compact } C \rangle$ ] boC
      by (simp add: True)
    next
    case False
    then obtain L where  $\text{compact } L$   $C \subseteq L$  and K:  $\text{openin } (\text{top\_of\_set } (\bigcap x. X\ x))\ L$ 
      by blast
    show ?thesis
    proof
      show  $L \subseteq \bigcap (\text{range } X)$ 
        by (metis K openin_imp_subset)

```

```

      show closed ( $\bigcap (\text{range } X) - L$ )
      by (metis closedin_diff closedin_self closedin_closed_trans [OF _
clo_INTX] K)
    qed (use ⟨compact L⟩ ⟨ $C \subseteq L$ ⟩ in auto)
  qed
  obtain U V where open U open V and compact (closure U)
    and V:  $\bigcap (\text{range } X) - K \subseteq V$  and U:  $K \subseteq U$   $U \cap V = \{\}$ 
    by (metis Diff_disjoint separation_normal_compact [OF ⟨compact K⟩
clo])
  then have  $U \cap (\bigcap (\text{range } X) - K) = \{\}$ 
    by blast
  have (closure U - U)  $\cap (\bigcap n. X\ n \cap \text{closure } U) \neq \{\}$ 
  proof (rule compact_imp_fip)
    show compact (closure U - U)
      by (metis ⟨compact (closure U)⟩ ⟨open U⟩ compact_diff)
    show  $\bigwedge T. T \in \text{range } (\lambda n. X\ n \cap \text{closure } U) \implies \text{closed } T$ 
      by clarify (metis cloX_closed_Int closed_closure)
    show (closure U - U)  $\cap \bigcap \mathcal{F} \neq \{\}$ 
      if finite  $\mathcal{F}$  and  $\mathcal{F}$ :  $\mathcal{F} \subseteq \text{range } (\lambda n. X\ n \cap \text{closure } U)$  for  $\mathcal{F}$ 
    proof
      assume empty: (closure U - U)  $\cap \bigcap \mathcal{F} = \{\}$ 
      obtain J where finite J and J:  $\mathcal{F} = (\lambda n. X\ n \cap \text{closure } U) \text{ `` } J$ 
        using finite_subset_image [OF ⟨finite  $\mathcal{F}$ ⟩  $\mathcal{F}$ ] by auto
      show False
      proof (cases J =  $\{\}$ )
        case True
        with J_empty have closed U
          by (simp add: closure_subset_eq)
        have  $C \neq \{\}$ 
          using C_in_components_nonempty by blast
        then have  $U \neq \{\}$ 
          using ⟨ $K \subseteq U$ ⟩ ⟨ $C \subseteq K$ ⟩ by blast
        moreover have  $U \neq \text{UNIV}$ 
          using ⟨compact (closure U)⟩ by auto
        ultimately show False
          using ⟨open U⟩ ⟨closed U⟩ clopen by blast
      next
        case False
        define j where j  $\equiv \text{Max } J$ 
        have j  $\in J$ 
          by (simp add: False ⟨finite J⟩ j_def)
        have jmax:  $\bigwedge m. m \in J \implies m \leq j$ 
          by (simp add: j_def ⟨finite J⟩)
        have  $\bigcap ((\lambda n. X\ n \cap \text{closure } U) \text{ `` } J) = X\ j \cap \text{closure } U$ 
          using False jmax nestX ⟨j  $\in J$ ⟩ by auto
        then have XU:  $X\ j \cap \text{closure } U = X\ j \cap U$ 
          using J_closure_subset_empty by fastforce
        then have openin (top_of_set (X j)) (X j  $\cap \text{closure } U$ )
          by (simp add: openin_open_Int ⟨open U⟩)

```

```

      moreover have closedin (top_of_set (X j)) (X j  $\cap$  closure U)
      by (simp add: closedin_closed_Int)
      moreover have X j  $\cap$  closure U  $\neq$  X j
      by (metis unboundedX  $\langle$ compact (closure U) $\rangle$  bounded_subset
compact_eq_bounded_closed inf.order_iff)
      moreover have X j  $\cap$  closure U  $\neq$  {}
      by (metis Cco Ksub UNIV_I  $\langle$ C  $\subseteq$  K $\rangle$   $\langle$ K  $\subseteq$  U $\rangle$  XU bot.extremum_uniqueI
in_components_maximal le_INF_iff le_inf_iff)
      ultimately show False
      using connX [of j] by (force simp: connected_clopen)
    qed
  qed
  qed
  moreover have ( $\bigcap n. X n \cap$  closure U) = ( $\bigcap n. X n$ )  $\cap$  closure U
  by blast
  moreover have  $x \in U$  if  $\bigwedge n. x \in X n$   $x \in$  closure U for x
  by (metis Diff_iff INT_I U V  $\langle$ open V $\rangle$  closure_iff_nhds_not_empty
order.refl subsetD that)
  ultimately show False
  by (auto simp: open_Int_closure_eq_empty [OF  $\langle$ open V $\rangle$ , of U])
  qed
  then show ?thesis
  by (auto simp: False)
  qed
  qed
  qed

```

```

lemma unbounded_complement_components:
  assumes C: C  $\in$  components ( $-$  S) and S: connected S
  and prev: if bounded S then connected(frontier S)
    else  $\forall C \in$  components(frontier S).  $\neg$  bounded C
  shows  $\neg$  bounded C
proof (cases bounded S)
  case True
  with prev have S  $\neq$  UNIV and confr: connected(frontier S)
  by auto
  obtain w where C_ccsw: C = connected_component_set ( $-$  S) w and w  $\notin$  S
  using C by (auto simp: components_def)
  show ?thesis
proof (cases S = {})
  case True with C show ?thesis by auto
next
  case False
  show ?thesis
proof
  assume bounded C
  then have outside C  $\neq$  {}
  using outside_bounded_nonempty by metis

```



```

    then obtain z where z:  $\neg$  bounded (connected_component_set (- C) z)
and z  $\notin$  C
    by (auto simp: outside_def)
    have clo_ccs: closed (connected_component_set (- S) x) for x
    by (simp add: closed_Compl closed_connected_component openS)
    have connected_component_set (- S) w = connected_component_set (- S)
z
    proof (rule joinable_connected_component_eq [OF confr])
    show frontier S  $\subseteq$  - S
    using openS by (auto simp: frontier_def interior_open)
    have False if connected_component_set (- S) w  $\cap$  frontier (- S) = {}
    proof -
    have C  $\cap$  frontier S = {}
    using that by (simp add: C_ccsw)
    moreover have closed C
    using C_ccsw clo_ccs by blast
    ultimately show False
    by (metis C  $\langle$ S  $\neq$  {} $\rangle$   $\langle$ S  $\neq$  UNIV $\rangle$  C_ccsw bot_eq_sup_iff connected_component_eq_UNIV frontier_Int_closed
    frontier_closed frontier_complement frontier_eq_empty frontier_of_components_subset in_components_maximal inf.orderE)
    qed
    then show connected_component_set (- S) w  $\cap$  frontier S  $\neq$  {}
    by auto
    have *:  $\llbracket$ frontier C  $\subseteq$  C; frontier C  $\subseteq$  F; frontier C  $\neq$  {} $\rrbracket \implies C \cap F \neq$ 
{} for C F::complex set
    by blast
    have connected_component_set (- S) z  $\cap$  frontier (- S)  $\neq$  {}
    proof (rule *)
    show frontier (connected_component_set (- S) z)  $\subseteq$  connected_component_set
(- S) z
    by (auto simp: closed_Compl closed_connected_component frontier_def
openS)
    show frontier (connected_component_set (- S) z)  $\subseteq$  frontier (- S)
    using frontier_of_connected_component_subset by fastforce
    have connected (closure S - S)
    by (metis confr frontier_def interior_open openS)
    moreover have  $\neg$  bounded (-S)
    by (simp add: True cobounded_imp_unbounded)
    moreover have bounded (connected_component_set (- S) w)
    using C_ccsw  $\langle$ bounded C $\rangle$  by auto
    ultimately have z  $\notin$  S
    using  $\langle$ w  $\notin$  S $\rangle$  openS
    by (metis ComplI Compl_eq_Diff_UNIV connected_UNIV closed_closure
closure_subset
    connected_component_eq_self connected_diff_open_from_closed
subset_UNIV)
    then have connected_component_set (- S) z  $\neq$  {}
    by (metis ComplI connected_component_eq_empty)

```

```

      then show frontier (connected_component_set (- S) z) ≠ {}
      by (metis False ⟨S ≠ UNIV⟩ connected_component_eq_UNIV frontier_complement frontier_eq_empty)
    qed
    then show connected_component_set (- S) z ∩ frontier S ≠ {}
    by auto
  qed
  then show False
  by (metis C_ccsw Compl_iff ⟨w ∉ S⟩ ⟨z ∉ C⟩ connected_component_eq_empty connected_component_idemp)
  qed
  qed
next
  case False
  obtain w where C_ccsw: C = connected_component_set (- S) w and w ∉ S
  using C by (auto simp: components_def)
  have frontier (connected_component_set (- S) w) ⊆ connected_component_set (- S) w
  by (simp add: closed_Compl closed_connected_component frontier_subset_eq openS)
  moreover have frontier (connected_component_set (- S) w) ⊆ frontier S
  using frontier_complement frontier_of_connected_component_subset by blast
  moreover have frontier (connected_component_set (- S) w) ≠ {}
  by (metis C C_ccsw False bounded_empty compl_top_eq connected_component_eq_UNIV double_compl frontier_not_empty in_components_nonempty)
  ultimately obtain z where zin: z ∈ frontier S and z: z ∈ connected_component_set (- S) w
  by blast
  have connected_component_set (frontier S) z ∈ components(frontier S)
  by (simp add: ⟨z ∈ frontier S⟩ componentsI)
  with prev False have ¬ bounded (connected_component_set (frontier S) z)
  by simp
  moreover have connected_component (- S) w = connected_component (- S) z
  using connected_component_eq [OF z] by force
  ultimately show ?thesis
  by (metis C_ccsw SC_Chain.openS SC_Chain_axioms bounded_subset closed_Compl connected_component_mono frontier_complement frontier_subset_eq)
  qed

```

lemma empty_inside:

```

  assumes connected S ∧ C. C ∈ components (- S) ⇒ ¬ bounded C
  shows inside S = {}
  using assms by (auto simp: components_def inside_def)

```

lemma empty_inside_imp_simply_connected:

```

  [[connected S; inside S = {}]] ⇒ simply_connected S
  by (metis ComplI inside_Un_outside openS outside_mono simply_connected_eq_winding_number_zero subsetCE sup_bot.left_neutral winding_number_zero_in_outside)

```

end

proposition

fixes $S :: \text{complex set}$

assumes $\text{open } S$

shows $\text{simply_connected_eq_frontier_properties}$:

$\text{simply_connected } S \longleftrightarrow$

$\text{connected } S \wedge$

$(\text{if bounded } S \text{ then } \text{connected}(\text{frontier } S)$

$\text{else } (\forall C \in \text{components}(\text{frontier } S). \neg \text{bounded } C)) \text{ (is ?fp)}$

and $\text{simply_connected_eq_unbounded_complement_components}$:

$\text{simply_connected } S \longleftrightarrow$

$\text{connected } S \wedge (\forall C \in \text{components}(- S). \neg \text{bounded } C) \text{ (is ?ucc)}$

and $\text{simply_connected_eq_empty_inside}$:

$\text{simply_connected } S \longleftrightarrow$

$\text{connected } S \wedge \text{inside } S = \{\} \text{ (is ?ei)}$

proof –

interpret SC_Chain

using assms by $(\text{simp add: } SC_Chain_def)$

have $?fp \wedge ?ucc \wedge ?ei$

using $\text{empty_inside empty_inside_imp_simply_connected_frontier_properties}$

$\text{unbounded_complement_components winding_number_zero}$ by blast

then show $?fp ?ucc ?ei$

by blast+

qed

lemma $\text{simply_connected_iff_simple}$:

fixes $S :: \text{complex set}$

assumes $\text{open } S \text{ bounded } S$

shows $\text{simply_connected } S \longleftrightarrow \text{connected } S \wedge \text{connected}(- S) \text{ (is ?lhs = ?rhs)}$

proof

show $?lhs \implies ?rhs$

by $(\text{metis DIM_complex assms cobounded_has_bounded_component double_complement dual_order.order_iff_strict}$

$\text{simply_connected_eq_unbounded_complement_components})$

show $?rhs \implies ?lhs$

by $(\text{simp add: assms connected_frontier_simple simply_connected_eq_frontier_properties})$

qed

lemma $\text{subset_simply_connected_imp_inside_subset}$:

fixes $A :: \text{complex set}$

assumes $\text{simply_connected } A \text{ open } A \text{ } B \subseteq A$

shows $\text{inside } B \subseteq A$

by $(\text{metis assms Diff_eq_empty_iff_inside_mono subset_empty simply_connected_eq_empty_inside})$

7.4 Further equivalences based on continuous logs and sqrts

context *SC_Chain*

begin

lemma *continuous_log*:

fixes $f :: \text{complex} \Rightarrow \text{complex}$

assumes $S: \text{simply_connected } S$

and $\text{contf}: \text{continuous_on } S \ f$ and $\text{nz}: \bigwedge z. z \in S \implies f \ z \neq 0$

shows $\exists g. \text{continuous_on } S \ g \wedge (\forall z \in S. f \ z = \exp(g \ z))$

proof –

consider $S = \{ \} \mid S \text{ homeomorphic ball } (0::\text{complex}) \ 1$

using *simply_connected_eq_homeomorphic_to_disc* [OF *openS*] S by *metis*

then show ?thesis

proof cases

case 1 then show ?thesis

by *simp*

next

case 2

then obtain $h \ k :: \text{complex} \Rightarrow \text{complex}$

where $kh: \bigwedge x. x \in S \implies k(h \ x) = x$ and $him: h \ ' \ S = \text{ball } 0 \ 1$

and $\text{conth}: \text{continuous_on } S \ h$

and $hk: \bigwedge y. y \in \text{ball } 0 \ 1 \implies h(k \ y) = y$ and $kim: k \ ' \ \text{ball } 0 \ 1 = S$

and $\text{contk}: \text{continuous_on } (\text{ball } 0 \ 1) \ k$

unfolding *homeomorphism_def* *homeomorphic_def* by *metis*

obtain g where $\text{contg}: \text{continuous_on } (\text{ball } 0 \ 1) \ g$

and $\text{expg}: \bigwedge z. z \in \text{ball } 0 \ 1 \implies (f \circ k) \ z = \exp \ (g \ z)$

proof (rule *continuous_logarithm_on_ball*)

show $\text{continuous_on } (\text{ball } 0 \ 1) \ (f \circ k)$

using contf $\text{continuous_on_compose}$ contk kim by *blast*

show $\bigwedge z. z \in \text{ball } 0 \ 1 \implies (f \circ k) \ z \neq 0$

using kim nz by *auto*

qed *auto*

then show ?thesis

by (*metis* *comp_apply* conth $\text{continuous_on_compose}$ him *imageI* kh)

qed

qed

lemma *continuous_sqrt*:

fixes $f :: \text{complex} \Rightarrow \text{complex}$

assumes $\text{contf}: \text{continuous_on } S \ f$ and $\text{nz}: \bigwedge z. z \in S \implies f \ z \neq 0$

and $\text{prev}: \bigwedge f::\text{complex} \Rightarrow \text{complex}.$

$\llbracket \text{continuous_on } S \ f; \bigwedge z. z \in S \implies f \ z \neq 0 \rrbracket$

$\implies \exists g. \text{continuous_on } S \ g \wedge (\forall z \in S. f \ z = \exp(g \ z))$

shows $\exists g. \text{continuous_on } S \ g \wedge (\forall z \in S. f \ z = (g \ z)^2)$

proof –

obtain g where $\text{contg}: \text{continuous_on } S \ g$ and $\text{geq}: \bigwedge z. z \in S \implies f \ z = \exp(g \ z)$

z)

using contf nz prev by *metis*

show ?thesis

```

proof (intro exI ballI conjI)
  show continuous_on S (λz. exp(g z/2))
    by (intro continuous_intros) (auto simp: contg)
  show  $\bigwedge z. z \in S \implies f z = (\exp(g z/2))^2$ 
    by (metis (no_types, lifting) divide_inverse exp_double geq mult.left_commute
mult.right_neutral right_inverse zero_neq numeral)
qed
qed

lemma continuous_sqrt_imp_simply_connected:
  assumes connected S
  and prev:  $\bigwedge f::\text{complex} \Rightarrow \text{complex}. \llbracket \text{continuous\_on } S f; \forall z \in S. f z \neq 0 \rrbracket$ 
     $\implies \exists g. \text{continuous\_on } S g \wedge (\forall z \in S. f z = (g z)^2)$ 
  shows simply_connected S
proof (clarsimp simp add: simply_connected_eq_holomorphic_sqrt [OF openS]
  ⟨connected S⟩)
  fix f
  assume f_holomorphic_on_S and nz:  $\forall z \in S. f z \neq 0$ 
  then obtain g where contg: continuous_on S g and geq:  $\bigwedge z. z \in S \implies f z =$ 
     $(g z)^2$ 
    by (metis holomorphic_on_imp_continuous_on prev)
  show  $\exists g. g \text{ holomorphic\_on } S \wedge (\forall z \in S. f z = (g z)^2)$ 
proof (intro exI ballI conjI)
  show g_holomorphic_on_S
  proof (clarsimp simp add: holomorphic_on_open [OF openS])
    fix z
    assume z ∈ S
    with nz geq have g z ≠ 0
      by auto
    obtain δ where 0 < δ  $\bigwedge w. \llbracket w \in S; \text{dist } w z < \delta \rrbracket \implies \text{dist } (g w) (g z) <$ 
      cmod (g z)
      using contg [unfolded continuous_on_iff] by (metis ⟨g z ≠ 0⟩ ⟨z ∈ S⟩
zero_less_norm_iff)
    then have δ:  $\bigwedge w. \llbracket w \in S; w \in \text{ball } z \delta \rrbracket \implies g w + g z \neq 0$ 
      by (metis add.commute add_cancel_right_left dist_commute dist_complex_def
mem_ball)
    norm_increases_online norm_not_less_zero norm_zero order_less_asym)
  have *:  $(\lambda x. (f x - f z) / (x - z) / (g x + g z)) - z \rightarrow \text{deriv } f z / (g z + g z)$ 
proof (intro tendsto_intros)
  show  $(\lambda x. (f x - f z) / (x - z)) - z \rightarrow \text{deriv } f z$ 
    using ⟨f holomorphic_on S⟩ ⟨z ∈ S⟩ has_field_derivative_iff holomor-
phic_derivI openS by blast
  show g - z → g z
    using ⟨z ∈ S⟩ contg continuous_on_eq_continuous_at isCont_def openS
by blast
  qed (simp add: ⟨g z ≠ 0⟩)
  then have (g has_field_derivative deriv f z / (g z + g z)) (at z)
    unfolding has_field_derivative_iff
  proof (rule Lim_transform_within_open)

```

```

show open (ball z  $\delta \cap S$ )
  by (simp add: openS open_Int)
show  $z \in \text{ball } z \delta \cap S$ 
  using  $\langle z \in S \rangle \langle 0 < \delta \rangle$  by simp
show  $\bigwedge x. \llbracket x \in \text{ball } z \delta \cap S; x \neq z \rrbracket$ 
   $\implies (f x - f z) / (x - z) / (g x + g z) = (g x - g z) / (x - z)$ 
  using  $\delta \langle z \in S \rangle$ 
  apply (simp add: geq_field_split_simps power2_eq_square)
  by (metis distrib_left mult_cancel_right)
qed
then show  $\exists f'. (g \text{ has\_field\_derivative } f') (at z) ..$ 
qed
qed (use geq in auto)
qed

end

```

proposition

```

fixes  $S :: \text{complex set}$ 
assumes open S
shows simply_connected_eq_continuous_log:
  simply_connected S  $\longleftrightarrow$ 
  connected S  $\wedge$ 
  ( $\forall f::\text{complex} \Rightarrow \text{complex}. \text{continuous\_on } S f \wedge (\forall z \in S. f z \neq 0)$ 
 $\longrightarrow (\exists g. \text{continuous\_on } S g \wedge (\forall z \in S. f z = \exp (g z))))$  (is ?log)
and simply_connected_eq_continuous_sqrt:
  simply_connected S  $\longleftrightarrow$ 
  connected S  $\wedge$ 
  ( $\forall f::\text{complex} \Rightarrow \text{complex}. \text{continuous\_on } S f \wedge (\forall z \in S. f z \neq 0)$ 
 $\longrightarrow (\exists g. \text{continuous\_on } S g \wedge (\forall z \in S. f z = (g z)^2)))$  (is ?sqrt)

```

proof –

```

interpret SC_Chain
  using assms by (simp add: SC_Chain_def)
show ?log ?sqrt
  using local.continuous_log winding_number_zero continuous_sqrt continu-
ous_sqrt_imp_simply_connected
  by auto
qed

```

7.5 More Borsukian results

lemma Borsukian_componentwise_eq:

```

fixes  $S :: 'a::\text{euclidean\_space set}$ 
assumes  $S: \text{locally\_connected } S \vee \text{compact } S$ 
shows Borsukian S  $\longleftrightarrow (\forall C \in \text{components } S. \text{Borsukian } C)$ 
proof –
  have *:  $\text{ANR}(-\{0::\text{complex}\})$ 
  by (simp add: ANR_delete open_Compl open_imp_ANR)
show ?thesis

```

using *cohomotopically_trivial_on_components* [*OF assms **] **by** (*auto simp: Borsukian_alt*)

qed

lemma *Borsukian_componentwise*:

fixes *S* :: 'a::euclidean_space set

assumes *locally_connected S* \vee *compact S* \wedge *C*. *C* \in *components S* \implies *Borsukian C*

shows *Borsukian S*

by (*metis Borsukian_componentwise_eq assms*)

lemma *simply_connected_eq_Borsukian*:

fixes *S* :: complex set

shows *open S* \implies (*simply_connected S* \longleftrightarrow *connected S* \wedge *Borsukian S*)

by (*auto simp: simply_connected_eq_continuous_log Borsukian_continuous_logarithm*)

lemma *Borsukian_eq_simply_connected*:

fixes *S* :: complex set

shows *open S* \implies *Borsukian S* \longleftrightarrow ($\forall C \in \text{components } S. \text{simply_connected } C$)

by (*meson Borsukian_componentwise_eq in_components_connected open_components open_imp_locally_connected simply_connected_eq_Borsukian*)

lemma *Borsukian_separation_open_closed*:

fixes *S* :: complex set

assumes *S*: *open S* \vee *closed S* **and** *bounded S*

shows *Borsukian S* \longleftrightarrow *connected*($- S$)

using *S*

proof

assume *open S*

show *?thesis*

unfolding *Borsukian_eq_simply_connected* [*OF* $\langle \text{open } S \rangle$]

by (*metis* $\langle \text{open } S \rangle$ $\langle \text{bounded } S \rangle$ *bounded_subset in_components_maximal non-separation_by_component_eq open_components_simply_connected_iff_simple*)

next

assume *closed S*

with $\langle \text{bounded } S \rangle$ **show** *?thesis*

by (*simp add: Borsukian_separation_compact compact_eq_bounded_closed*)

qed

7.6 Finally, the Riemann Mapping Theorem

theorem *Riemann_mapping_theorem*:

open S \wedge *simply_connected S* \longleftrightarrow

S = {} \vee *S* = *UNIV* \vee

($\exists f g. f \text{ holomorphic_on } S \wedge g \text{ holomorphic_on ball } 0 \ 1 \wedge$

($\forall z \in S. f z \in \text{ball } 0 \ 1 \wedge g(f z) = z$) \wedge

($\forall z \in \text{ball } 0 \ 1. g z \in S \wedge f(g z) = z$))

(*is* $_ = ?rhs$)

proof –

```

have simply_connected S  $\longleftrightarrow$  ?rhs if open S
  by (simp add: simply_connected_eq_biholomorphic_to_disc that)
moreover have open S if ?rhs
proof -
  { fix f g
    assume g: g holomorphic_on ball 0 1  $\forall z \in \text{ball } 0 \ 1. g \ z \in S \wedge f \ (g \ z) = z$ 
    and  $\forall z \in S. \text{cmod } (f \ z) < 1 \wedge g \ (f \ z) = z$ 
    then have S = g ` (ball 0 1)
      by force
    then have open S
      by (metis open_ball g inj_on_def open_mapping_thm3)
  }
  with that show open S by auto
qed
ultimately show ?thesis by metis
qed

```

7.7 Applications to Winding Numbers

```

lemma simply_connected_inside_simple_path:
  fixes p :: real  $\Rightarrow$  complex
  shows simple_path p  $\implies$  simply_connected(inside(path_image p))
  using Jordan_inside_outside connected_simple_path_image inside_simple_curve_imp_closed
  simply_connected_eq_frontier_properties
  by fastforce

```

```

lemma simply_connected_Int:
  fixes S :: complex set
  assumes open S open T simply_connected S simply_connected T connected (S
 $\cap$  T)
  shows simply_connected (S  $\cap$  T)
  using assms by (force simp: simply_connected_eq_winding_number_zero open_Int)

```

7.8 The winding number defines a continuous logarithm for the path itself

```

lemma winding_number_as_continuous_log:
  assumes path p and  $\zeta: \zeta \notin \text{path\_image } p$ 
  obtains q where path q
    pathfinish q = pathstart q =  $2 * \text{of\_real } \pi * i * \text{winding\_number } p \ \zeta$ 
     $\wedge t. t \in \{0..1\} \implies p \ t = \zeta + \exp(q \ t)$ 
proof -
  let ?q =  $\lambda t. 2 * \text{of\_real } \pi * i * \text{winding\_number}(\text{subpath } 0 \ t \ p) \ \zeta + \text{Ln}(\text{pathstart } p - \zeta)$ 
  show ?thesis
  proof
    have *: continuous (at t within {0..1}) ( $\lambda x. \text{winding\_number}(\text{subpath } 0 \ x \ p) \ \zeta$ )
    if t:  $t \in \{0..1\}$  for t

```



```

proof –
  let ?B = ball (p t) (norm(p t - ζ))
  have p t ≠ ζ
    using path_image_def that ζ by blast
  then have simply_connected ?B
    by (simp add: convex_imp_simply_connected)
  then have  $\bigwedge f::\text{complex} \Rightarrow \text{complex}. \text{continuous\_on } ?B f \wedge (\forall \zeta \in ?B. f \zeta \neq$ 
0)
     $\longrightarrow (\exists g. \text{continuous\_on } ?B g \wedge (\forall \zeta \in ?B. f \zeta = \exp(g \zeta)))$ 
    by (simp add: simply_connected_eq_continuous_log)
  moreover have continuous_on ?B ( $\lambda w. w - \zeta$ )
    by (intro continuous_intros)
  moreover have ( $\forall z \in ?B. z - \zeta \neq 0$ )
    by (auto simp: dist_norm)
  ultimately obtain g where contg: continuous_on ?B g
    and geg:  $\bigwedge z. z \in ?B \Longrightarrow z - \zeta = \exp(g z)$  by blast
  obtain d where 0 < d and d:
     $\bigwedge x. [x \in \{0..1\}; \text{dist } x t < d] \Longrightarrow \text{dist } (p x) (p t) < cmod (p t - \zeta)$ 
    using  $\langle \text{path } p \rangle t$  unfolding path_def continuous_on_iff
    by (metis  $\langle p t \neq \zeta \rangle$  right_minus_eq zero_less_norm_iff)
  have (( $\lambda x. \text{winding\_number } (\lambda w. \text{subpath } 0 x p w - \zeta) 0 -$ 
    winding_number ( $\lambda w. \text{subpath } 0 t p w - \zeta) 0$ )  $\longrightarrow 0$ )
    (at t within {0..1})
  proof (rule Lim_transform_within [OF _  $\langle d > 0 \rangle$ ])
    have continuous (at t within {0..1}) (g o p)
    proof (rule continuous_within_compose)
      show continuous (at t within {0..1}) p
        using  $\langle \text{path } p \rangle$  continuous_on_eq_continuous_within path_def that by
blast
      show continuous (at (p t) within p ‘ {0..1}) g
        by (metis (no_types, lifting) open_ball UNIV_I  $\langle p t \neq \zeta \rangle$  centre_in_ball
contg continuous_on_eq_continuous_at continuous_within_topological right_minus_eq
zero_less_norm_iff)
    qed
    with LIM_zero have (( $\lambda u. (g (\text{subpath } t u p 1) - g (\text{subpath } t u p 0)))$ 
 $\longrightarrow 0$ ) (at t within {0..1})
    by (auto simp: subpath_def continuous_within_o_def)
    then show (( $\lambda u. (g (\text{subpath } t u p 1) - g (\text{subpath } t u p 0)) / (2 * of\_real$ 
pi * i))  $\longrightarrow 0$ )
      (at t within {0..1})
      by (simp add: tendsto_divide_zero)
    show (g (subpath t u p 1) - g (subpath t u p 0)) / (2 * of_real pi * i) =
      winding_number ( $\lambda w. \text{subpath } 0 u p w - \zeta$ ) 0 - winding_number ( $\lambda w.$ 
subpath 0 t p w - ζ) 0
      if  $u \in \{0..1\}$  0 < dist u t dist u t < d for u
    proof –
      have closed_segment t u  $\subseteq \{0..1\}$ 
      using closed_segment_eq_real_ivl t that by auto
      then have  $\bigwedge r. [r \in \text{closed\_segment } t u] \Longrightarrow \text{dist } (p t) (p r) < cmod (p$ 

```

```

t - ζ)
  by (smt (verit, best) d dist_commute dist_in_closed_segment subsetD
    ‹dist u t < d›)
  then have piB: path_image(subpath t u p) ⊆ ?B
    by (auto simp: path_image_subpath_gen)
  have *: path (g ∘ subpath t u p)
  proof (rule path_continuous_image)
    show path (subpath t u p)
    using ‹path p› t that by auto
    show continuous_on (path_image (subpath t u p)) g
    using piB contg continuous_on_subset by blast
  qed
  have (g (subpath t u p 1) - g (subpath t u p 0)) / (2 * of_real pi * i)
    = winding_number (exp ∘ g ∘ subpath t u p) 0
    using winding_number_compose_exp [OF *]
    by (simp add: pathfinish_def pathstart_def o_assoc)
  also have ... = winding_number (λw. subpath t u p w - ζ) 0
  proof (rule winding_number_cong)
    have exp(g y) = y - ζ if y ∈ (subpath t u p) ‘{0..1}’ for y
    by (metis that geq_path_image_def piB subset_eq)
    then show ∧x. [0 ≤ x; x ≤ 1] ⇒ (exp ∘ g ∘ subpath t u p) x = subpath
      t u p x - ζ
    by auto
  qed
  also have ... = winding_number (λw. subpath 0 u p w - ζ) 0 -
    winding_number (λw. subpath 0 t p w - ζ) 0
    apply (simp add: winding_number_offset [symmetric])
    using winding_number_subpath_combine [OF ‹path p› ζ, of 0 t u] ‹t ∈
      {0..1}› ‹u ∈ {0..1}›
    by (simp add: add commute eq_diff_eq)
  finally show ?thesis .
  qed
  qed
  then show ?thesis
  by (subst winding_number_offset) (simp add: continuous_within LIM_zero_iff)
  qed
  show path ?q
  unfolding path_def
  by (intro continuous_intros) (simp add: continuous_on_eq_continuous_within
    *)

have ζ ≠ p 0
  by (metis ζ pathstart_def pathstart_in_path_image)
then show pathfinish ?q - pathstart ?q = 2 * of_real pi * i * winding_number
  p ζ
  by (simp add: pathfinish_def pathstart_def)
show p t = ζ + exp (?q t) if t ∈ {0..1} for t
proof -
  have path (subpath 0 t p)

```

```

      using ⟨path p⟩ that by auto
    moreover
    have  $\zeta \notin \text{path\_image } (\text{subpath } 0 \ t \ p)$ 
    using  $\zeta$  [unfolded path_image_def] that by (auto simp: path_image_subpath)
    ultimately show ?thesis
      using winding_number_exp_2pi [of subpath 0 t p  $\zeta$ ] ⟨ $\zeta \neq p \ 0$ ⟩
      by (auto simp: exp_add algebra_simps pathfinish_def pathstart_def sub-
path_def)
    qed
  qed
qed

```

7.9 Winding number equality is the same as path/loop homotopy in $\mathbb{C} - 0$

```

lemma winding_number_homotopic_loops_null_eq:
  assumes path p and  $\zeta: \zeta \notin \text{path\_image } p$ 
  shows winding_number p  $\zeta = 0 \iff (\exists a. \text{homotopic\_loops } (-\{\zeta\}) \ p \ (\lambda t. a))$ 
    (is ?lhs = ?rhs)
proof
  assume [simp]: ?lhs
  obtain q where path q
    and qeq: pathfinish q - pathstart q = 2 * of_real pi * i * winding_number
    p  $\zeta$ 
    and peq:  $\bigwedge t. t \in \{0..1\} \implies p \ t = \zeta + \exp(q \ t)$ 
  using winding_number_as_continuous_log [OF assms] by blast
  have *: homotopic_with_canon ( $\lambda r. \text{pathfinish } r = \text{pathstart } r$ )
    {0..1} (-{ $\zeta$ }) (( $\lambda w. \zeta + \exp w$ )  $\circ$  q) (( $\lambda w. \zeta + \exp w$ )  $\circ$  ( $\lambda t. 0$ ))
  proof (rule homotopic_with_compose_continuous_left)
    show homotopic_with_canon ( $\lambda f. \text{pathfinish } ((\lambda w. \zeta + \exp w) \circ f) = \text{pathstart } ((\lambda w. \zeta + \exp w) \circ f)$ )
      {0..1} UNIV q ( $\lambda t. 0$ )
    proof (rule homotopic_with_mono, simp_all add: pathfinish_def pathstart_def)
      have homotopic_loops UNIV q ( $\lambda t. 0$ )
      by (rule homotopic_loops_linear) (use qeq ⟨path q⟩ in ⟨auto simp: path_defs⟩)
      then have homotopic_with ( $\lambda r. r \ 1 = r \ 0$ ) (top_of_set {0..1}) euclidean q
        ( $\lambda t. 0$ )
      by (simp add: homotopic_loops_def pathfinish_def pathstart_def)
      then show homotopic_with ( $\lambda h. \exp (h \ 1) = \exp (h \ 0)$ ) (top_of_set {0..1})
        euclidean q ( $\lambda t. 0$ )
      by (rule homotopic_with_mono) simp
    qed
    show continuous_on UNIV ( $\lambda w. \zeta + \exp w$ )
    by (rule continuous_intros)+
  qed auto
  then have homotopic_with_canon ( $\lambda r. \text{pathfinish } r = \text{pathstart } r$ ) {0..1} (-{ $\zeta$ })
    p ( $\lambda x. \zeta + 1$ )
  by (rule homotopic_with_eq) (auto simp: o_def peq pathfinish_def pathstart_def)
  then have homotopic_loops (-{ $\zeta$ }) p ( $\lambda t. \zeta + 1$ )

```

```

    by (simp add: homotopic_loops_def)
  then show ?rhs ..
next
  assume ?rhs
  then obtain a where homotopic_loops  $(-\{\zeta\})$  p  $(\lambda t. a)$  ..
  then have winding_number p  $\zeta =$  winding_number  $(\lambda t. a)$   $\zeta$   $a \neq \zeta$ 
    using winding_number_homotopic_loops homotopic_loops_imp_subset by
  (force simp:)+
  then show ?lhs
    using winding_number_zero_const by auto
qed

```

```

lemma winding_number_homotopic_paths_null_explicit_eq:
  assumes path p and  $\zeta: \zeta \notin \text{path\_image } p$ 
  shows winding_number p  $\zeta = 0 \iff$  homotopic_paths  $(-\{\zeta\})$  p (linepath
    (pathstart p) (pathstart p))
    (is ?lhs = ?rhs)
proof
  assume ?lhs
  then show ?rhs
    using homotopic_loops_imp_homotopic_paths_null
    by (force simp: linepath_refl winding_number_homotopic_loops_null_eq [OF
  assms])
next
  assume ?rhs
  then show ?lhs
    by (metis  $\zeta$  pathstart_in_path_image winding_number_homotopic_paths wind-
    ing_number_trivial)
qed

```

```

lemma winding_number_homotopic_paths_null_eq:
  assumes path p and  $\zeta: \zeta \notin \text{path\_image } p$ 
  shows winding_number p  $\zeta = 0 \iff (\exists a. \text{homotopic\_paths } (-\{\zeta\})$  p  $(\lambda t. a))$ 
    (is ?lhs = ?rhs)
proof
  assume ?lhs
  then show ?rhs
    by (auto simp: winding_number_homotopic_paths_null_explicit_eq [OF assms]
  linepath_refl)
next
  assume ?rhs
  then show ?lhs
    by (metis  $\zeta$  homotopic_paths_imp_pathfinish pathfinish_def pathfinish_in_path_image
  winding_number_homotopic_paths winding_number_zero_const)
qed

```

```

proposition winding_number_homotopic_paths_eq:
  assumes path p and  $\zeta p: \zeta \notin \text{path\_image } p$ 
    and path q and  $\zeta q: \zeta \notin \text{path\_image } q$ 

```

```

    and qp: pathstart q = pathstart p pathfinish q = pathfinish p
    shows winding_number p  $\zeta$  = winding_number q  $\zeta \longleftrightarrow$  homotopic_paths
      ( $-\{\zeta\}$ ) p q
      (is ?lhs = ?rhs)
  proof
    assume ?lhs
    then have winding_number (p +++ reversepath q)  $\zeta$  = 0
    using assms by (simp add: winding_number_join winding_number_reversepath)
    moreover
    have path (p +++ reversepath q)  $\zeta \notin$  path_image (p +++ reversepath q)
    using assms by (auto simp: not_in_path_image_join)
    ultimately obtain a where homotopic_paths ( $-\{\zeta\}$ ) (p +++ reversepath q)
      (linepath a a)
    using winding_number_homotopic_paths_null_explicit_eq by blast
    then show ?rhs
    using homotopic_paths_imp_pathstart assms
    by (fastforce simp: dest: homotopic_paths_imp_homotopic_loops homotopic_paths_loop_parts)
  qed (simp add: winding_number_homotopic_paths)

lemma winding_number_homotopic_loops_eq:
  assumes path p and  $\zeta p$ :  $\zeta \notin$  path_image p
    and path q and  $\zeta q$ :  $\zeta \notin$  path_image q
    and loops: pathfinish p = pathstart p pathfinish q = pathstart q
  shows winding_number p  $\zeta$  = winding_number q  $\zeta \longleftrightarrow$  homotopic_loops
    ( $-\{\zeta\}$ ) p q
    (is ?lhs = ?rhs)
  proof
    assume L: ?lhs
    have pathstart p  $\neq \zeta$  pathstart q  $\neq \zeta$ 
    using  $\zeta p \zeta q$  by blast+
    moreover have path_connected ( $-\{\zeta\}$ )
    by (simp add: path_connected_punctured_universe)
    ultimately obtain r where path r and rim: path_image r  $\subseteq -\{\zeta\}$ 
      and pas: pathstart r = pathstart p and paf: pathfinish r =
pathstart q
    by (auto simp: path_connected_def)
    then have pathstart r  $\neq \zeta$  by blast
    have homotopic_loops ( $-\{\zeta\}$ ) p (r +++ q +++ reversepath r)
    proof (rule homotopic_paths_imp_homotopic_loops)
      have path (r +++ q +++ reversepath r)
      by (simp add:  $\langle$ path r $\rangle \langle$ path q $\rangle$  loops paf)
      moreover have  $\zeta \notin$  path_image (r +++ q +++ reversepath r)
      by (metis  $\zeta q$  not_in_path_image_join path_image_reversepath rim subset_Compl_singleton)
      moreover have homotopic_loops ( $-\{\zeta\}$ ) (r +++ q +++ reversepath r) q
      using  $\langle$ path q $\rangle \langle$ path r $\rangle \zeta q$  homotopic_loops_conjugate loops(2) paf rim by
blast
    ultimately show homotopic_paths ( $-\{\zeta\}$ ) p (r +++ q +++ reversepath r)
    using loops pathfinish_join pathfinish_reversepath pathstart_join

```

```

    by (metis L  $\zeta p$   $\langle \text{path } p \rangle$  pas winding_number_homotopic_loops winding_number_homotopic_paths_eq)
qed (use loops pas in auto)
moreover have homotopic_loops ( $-\{\zeta\}$ ) ( $r +++ q +++ \text{reversepath } r$ )  $q$ 
  using rim  $\zeta q$  by (auto simp: homotopic_loops_conjugate paf  $\langle \text{path } q \rangle \langle \text{path } r \rangle$ 
loops)
ultimately show ?rhs
  using homotopic_loops_trans by metis
qed (simp add: winding_number_homotopic_loops)

end
theory Complex_Singularities
  imports Conformal_Mappings
begin

```

7.10 Non-essential singular points

```

lemma at_to_0': NO_MATCH 0  $z \implies \text{at } z = \text{filtermap } (\lambda x. x + z) (\text{at } 0)$ 
  for  $z :: 'a :: \text{real\_normed\_vector}$ 
  by (rule at_to_0)

lemma nhds_to_0: nhds ( $x :: 'a :: \text{real\_normed\_vector}$ ) = filtermap ((+)  $x$ ) (nhds 0)
proof -
  have ( $\lambda xa. xa - - x$ ) = (+)  $x$ 
  by auto
  thus ?thesis
  using filtermap_nhds_shift[of  $-x$  0] by simp
qed

lemma nhds_to_0': NO_MATCH 0  $x \implies \text{nhds } (x :: 'a :: \text{real\_normed\_vector})$ 
= filtermap ((+)  $x$ ) (nhds 0)
  by (rule nhds_to_0)

definition
  is_pole :: ( $'a :: \text{topological\_space} \Rightarrow 'b :: \text{real\_normed\_vector}$ )  $\Rightarrow 'a \Rightarrow \text{bool}$ 
  where is_pole  $f a = (\text{LIM } x (\text{at } a). f x :> \text{at\_infinity})$ 

lemma is_pole_cong:
  assumes eventually ( $\lambda x. f x = g x$ ) ( $\text{at } a$ )  $a=b$ 
  shows is_pole  $f a \longleftrightarrow \text{is\_pole } g b$ 
  unfolding is_pole_def using assms by (intro filterlim_cong, auto)

lemma is_pole_transform:
  assumes is_pole  $f a$  eventually ( $\lambda x. f x = g x$ ) ( $\text{at } a$ )  $a=b$ 
  shows is_pole  $g b$ 
  using is_pole_cong assms by auto

lemma is_pole_shift_iff:

```

```

fixes f :: ('a::real_normed_vector  $\Rightarrow$  'b::real_normed_vector)
shows is_pole (f  $\circ$  (+) d) a = is_pole f (a + d)
by (metis add_diff_cancel_right' filterlim_shift_iff is_pole_def)

lemma is_pole_tendsto:
  fixes f :: ('a::topological_space  $\Rightarrow$  'b::real_normed_div_algebra)
  shows is_pole f x  $\implies$  ((inverse o f)  $\longrightarrow$  0) (at x)
  unfolding is_pole_def
  by (auto simp add: filterlim_inverse_at_iff[symmetric] comp_def filterlim_at)

lemma is_pole_shift_0:
  fixes f :: ('a::real_normed_vector  $\Rightarrow$  'b::real_normed_vector)
  shows is_pole f z  $\longleftrightarrow$  is_pole ( $\lambda x. f (z + x)$ ) 0
  unfolding is_pole_def by (subst at_to_0) (auto simp: filterlim_filtermap add_ac)

lemma is_pole_shift_0':
  fixes f :: ('a::real_normed_vector  $\Rightarrow$  'b::real_normed_vector)
  shows NO_MATCH 0 z  $\implies$  is_pole f z  $\longleftrightarrow$  is_pole ( $\lambda x. f (z + x)$ ) 0
  by (metis is_pole_shift_0)

lemma is_pole_compose_iff:
  assumes filtermap g (at x) = (at y)
  shows is_pole (f  $\circ$  g) x  $\longleftrightarrow$  is_pole f y
  unfolding is_pole_def filterlim_def filtermap_compose assms ..

lemma is_pole_inverse_holomorphic:
  assumes open s
  and f_holo: f holomorphic_on (s - {z})
  and pole: is_pole f z
  and non_z:  $\forall x \in s - \{z\}. f x \neq 0$ 
  shows ( $\lambda x. \text{if } x = z \text{ then } 0 \text{ else inverse } (f x)$ ) holomorphic_on s
proof -
  define g where g  $\equiv \lambda x. \text{if } x = z \text{ then } 0 \text{ else inverse } (f x)$ 
  have isCont g z unfolding isCont_def using is_pole_tendsto[OF pole]
  by (simp add: g_def cong: LIM_cong)
  moreover have continuous_on (s - {z}) f using f_holo holomorphic_on_imp_continuous_on
by auto
  hence continuous_on (s - {z}) (inverse o f) unfolding comp_def
  by (auto elim!: continuous_on_inverse simp add: non_z)
  hence continuous_on (s - {z}) g unfolding g_def
  using continuous_on_eq by fastforce
  ultimately have continuous_on s g using open_delete[OF <open s>] <open s>
  by (auto simp add: continuous_on_eq_continuous_at)
  moreover have (inverse o f) holomorphic_on (s - {z})
  unfolding comp_def using f_holo
  by (auto elim!: holomorphic_on_inverse simp add: non_z)
  hence g holomorphic_on (s - {z})
  using g_def holomorphic_transform by force
  ultimately show ?thesis unfolding g_def using <open s>

```

by (auto elim!: no_isolated_singularity)
qed

lemma not_is_pole_holomorphic:
assumes open A x \in A f holomorphic_on A
shows \neg is_pole f x
proof –
have continuous_on A f
by (intro holomorphic_on_imp_continuous_on) fact
with assms have isCont f x
by (simp add: continuous_on_eq_continuous_at)
hence $f - x \rightarrow f x$
by (simp add: isCont_def)
thus \neg is_pole f x
unfolding is_pole_def
using not_tendsto_and_filterlim_at_infinity[of at x f f x] by auto
qed

lemma is_pole_inverse_power: $n > 0 \implies$ is_pole $(\lambda z::\text{complex}. 1 / (z - a) ^ n)$ a
unfolding is_pole_def inverse_eq_divide [symmetric]
by (intro filterlim_compose[OF filterlim_inverse_at_infinity] tendsto_intros)
(auto simp: filterlim_at_eventually_at intro!: exI[of _ 1] tendsto_eq_intros)

lemma is_pole_cmult_iff [simp]:
assumes $c \neq 0$
shows is_pole $(\lambda z. c * f z :: 'a :: \text{real_normed_field}) z \longleftrightarrow$ is_pole f z
proof
assume is_pole $(\lambda z. c * f z) z$
with $\langle c \neq 0 \rangle$ have is_pole $(\lambda z. \text{inverse } c * (c * f z)) z$
unfolding is_pole_def
by (force intro: tendsto_mult_filterlim_at_infinity)
thus is_pole f z
using $\langle c \neq 0 \rangle$ by (simp add: field_simps)
next
assume is_pole f z
with $\langle c \neq 0 \rangle$ show is_pole $(\lambda z. c * f z) z$
by (auto intro!: tendsto_mult_filterlim_at_infinity simp: is_pole_def)
qed

lemma is_pole_uminus_iff [simp]: is_pole $(\lambda z. -f z :: 'a :: \text{real_normed_field}) z \longleftrightarrow$ is_pole f z
using is_pole_cmult_iff[of -1 f] by (simp del: is_pole_cmult_iff)

lemma is_pole_inverse: is_pole $(\lambda z::\text{complex}. 1 / (z - a)) a$
using is_pole_inverse_power[of 1 a] by simp

lemma is_pole_divide:
fixes f $:: 'a :: t2_space \Rightarrow 'b :: \text{real_normed_field}$


```

assumes isCont f z filterlim g (at 0) (at z) f z  $\neq$  0
shows is_pole ( $\lambda z. f z / g z$ ) z
proof -
  have filterlim ( $\lambda z. f z * \text{inverse } (g z)$ ) at_infinity (at z)
    using assms filterlim_compose filterlim_inverse_at_infinity isCont_def
    tendsto_mult_filterlim_at_infinity by blast
  thus ?thesis by (simp add: field_split_simps is_pole_def)
qed

```

```

lemma is_pole_basic:
  assumes f holomorphic_on A open A z  $\in$  A f z  $\neq$  0 n > 0
  shows is_pole ( $\lambda w. f w / (w - z) ^ n$ ) z
proof (rule is_pole_divide)
  have continuous_on A f by (rule holomorphic_on_imp_continuous_on) fact
  with assms show isCont f z by (auto simp: continuous_on_eq_continuous_at)
  have filterlim ( $\lambda w. (w - z) ^ n$ ) (nhds 0) (at z)
    using assms by (auto intro!: tendsto_eq_intros)
  thus filterlim ( $\lambda w. (w - z) ^ n$ ) (at 0) (at z)
    by (intro filterlim_atI tendsto_eq_intros)
    (use assms in  $\langle$ auto simp: eventually_at_filter $\rangle$ )
qed fact

```

```

lemma is_pole_basic':
  assumes f holomorphic_on A open A 0  $\in$  A f 0  $\neq$  0 n > 0
  shows is_pole ( $\lambda w. f w / w ^ n$ ) 0
  using is_pole_basic[of f A 0] assms by simp

```

```

lemma is_pole_compose:
  assumes is_pole f w g  $-z \rightarrow w$  eventually ( $\lambda z. g z \neq w$ ) (at z)
  shows is_pole ( $\lambda x. f (g x)$ ) z
  using assms(1) unfolding is_pole_def
  by (rule filterlim_compose) (use assms in  $\langle$ auto simp: filterlim_at $\rangle$ )

```

```

lemma is_pole_plus_const_iff:
  is_pole f z  $\longleftrightarrow$  is_pole ( $\lambda x. f x + c$ ) z
proof
  assume is_pole f z
  then have filterlim f at_infinity (at z) unfolding is_pole_def .
  moreover have (( $\lambda \_. c$ )  $\longrightarrow$  c) (at z) by auto
  ultimately have LIM x (at z). f x + c  $:$ > at_infinity
    using tendsto_add_filterlim_at_infinity'[of f at z] by auto
  then show is_pole ( $\lambda x. f x + c$ ) z unfolding is_pole_def .
next
  assume is_pole ( $\lambda x. f x + c$ ) z
  then have filterlim ( $\lambda x. f x + c$ ) at_infinity (at z)
    unfolding is_pole_def .
  moreover have (( $\lambda \_. -c$ )  $\longrightarrow$  -c) (at z) by auto
  ultimately have LIM x (at z). f x  $:$ > at_infinity
    using tendsto_add_filterlim_at_infinity'[of ( $\lambda x. f x + c$ )

```

```

      at z ( $\lambda \_. - c$ ) - c]
    by auto
  then show  $is\_pole\ f\ z$  unfolding  $is\_pole\_def$  .
qed

```

```

lemma  $is\_pole\_minus\_const\_iff$ :
   $is\_pole\ (\lambda x. f\ x - c)\ z \longleftrightarrow is\_pole\ f\ z$ 
  using  $is\_pole\_plus\_const\_iff$  [of  $f\ z - c$ ] by simp

```

```

lemma  $is\_pole\_alt$ :
   $is\_pole\ f\ x = (\forall B > 0. \exists U. open\ U \wedge x \in U \wedge (\forall y \in U. y \neq x \longrightarrow norm\ (f\ y) \geq B))$ 
  unfolding  $is\_pole\_def$ 
  unfolding  $filterlim\_at\_infinity$ [of 0,simplified] eventually_at_topological
  by auto

```

```

lemma  $is\_pole\_mult\_analytic\_nonzero1$ :
  assumes  $is\_pole\ g\ x$   $f$  analytic_on  $\{x\}$   $f\ x \neq 0$ 
  shows  $is\_pole\ (\lambda x. f\ x * g\ x)\ x$ 
  unfolding  $is\_pole\_def$ 
  proof (rule tendsto_mult_filterlim_at_infinity)
    show  $f\ -x \rightarrow f\ x$ 
    using assms by (simp add: analytic_at_imp_isCont isContD)
  qed (use assms in <auto simp:  $is\_pole\_def$ >)

```

```

lemma  $is\_pole\_mult\_analytic\_nonzero2$ :
  assumes  $is\_pole\ f\ x$   $g$  analytic_on  $\{x\}$   $g\ x \neq 0$ 
  shows  $is\_pole\ (\lambda x. f\ x * g\ x)\ x$ 
  proof -
    have  $g: g$  analytic_on  $\{x\}$ 
    using assms by auto
    show ?thesis
    using  $is\_pole\_mult\_analytic\_nonzero1$  [OF < $is\_pole\ f\ x$ >  $g$ ] < $g\ x \neq 0$ >
    by (simp add: mult.commute)
  qed

```

```

lemma  $is\_pole\_mult\_analytic\_nonzero1\_iff$ :
  assumes  $f$  analytic_on  $\{x\}$   $f\ x \neq 0$ 
  shows  $is\_pole\ (\lambda x. f\ x * g\ x)\ x \longleftrightarrow is\_pole\ g\ x$ 
  proof
    assume  $is\_pole\ g\ x$ 
    thus  $is\_pole\ (\lambda x. f\ x * g\ x)\ x$ 
    by (intro  $is\_pole\_mult\_analytic\_nonzero1$  assms)
  next
    assume  $is\_pole\ (\lambda x. f\ x * g\ x)\ x$ 
    hence  $is\_pole\ (\lambda x. inverse\ (f\ x) * (f\ x * g\ x))\ x$ 
    by (rule  $is\_pole\_mult\_analytic\_nonzero1$ )
    (use assms in <auto intro!: analytic_intros>)
    also have ?this  $\longleftrightarrow is\_pole\ g\ x$ 
    proof (rule  $is\_pole\_cong$ )

```

```

have eventually ( $\lambda x. f\ x \neq 0$ ) (at x)
  using assms by (simp add: analytic_at_neq_imp_eventually_neq)
thus eventually ( $\lambda x. \text{inverse } (f\ x) * (f\ x * g\ x) = g\ x$ ) (at x)
  by eventually_elim auto
qed auto
finally show is_pole g x .
qed

```

```

lemma is_pole_mult_analytic_nonzero2_iff:
  assumes g analytic_on {x} g x  $\neq 0$ 
  shows is_pole ( $\lambda x. f\ x * g\ x$ ) x  $\longleftrightarrow$  is_pole f x
  by (subst mult.commute, rule is_pole_mult_analytic_nonzero1_iff) (fact assms)+

```

```

lemma frequently_const_imp_not_is_pole:
  fixes z :: 'a::first_countable_topology
  assumes frequently ( $\lambda w. f\ w = c$ ) (at z)
  shows  $\neg$  is_pole f z
proof
  assume is_pole f z
  from assms have z islimpt {w. f w = c}
    by (simp add: islimpt_conv_frequently_at)
  then obtain g where g:  $\bigwedge n. g\ n \in \{w. f\ w = c\} - \{z\}$  g  $\longrightarrow$  z
    unfolding islimpt_sequential by blast
  then have (f  $\circ$  g)  $\longrightarrow$  c
    by (simp add: tendsto_eventually)
  moreover have filterlim g (at z) sequentially
    using g by (auto simp: filterlim_at)
  then have filterlim (f  $\circ$  g) at_infinity sequentially
    unfolding o_def
    using  $\langle$ is_pole f z $\rangle$  filterlim_compose is_pole_def by blast
  ultimately show False
    using not_tendsto_and_filterlim_at_infinity trivial_limit_sequentially by
  blast
qed

```

7.11 Isolated singularities

The proposition $\exists x. f - z \rightarrow x \vee \text{is_pole } f\ z$ can be interpreted as the complex function f has a non-essential singularity at z (i.e. the singularity is either removable or a pole).

definition not_essential:: $[\text{complex} \Rightarrow \text{complex}, \text{complex}] \Rightarrow \text{bool}$ **where**
 $\text{not_essential } f\ z = (\exists x. f - z \rightarrow x \vee \text{is_pole } f\ z)$

definition isolated_singularity_at:: $[\text{complex} \Rightarrow \text{complex}, \text{complex}] \Rightarrow \text{bool}$ **where**
 $\text{isolated_singularity_at } f\ z = (\exists r > 0. f \text{ analytic_on ball } z\ r - \{z\})$

```

lemma not_essential_cong:
  assumes eventually ( $\lambda x. f\ x = g\ x$ ) (at z) z = z'
  shows not_essential f z  $\longleftrightarrow$  not_essential g z'

```

unfolding *not_essential_def* **using** *assms filterlim_cong is_pole_cong* **by** *fastforce*

lemma *not_essential_compose_iff*:
assumes *filtermap g (at z) = at z'*
shows *not_essential (f ∘ g) z = not_essential f z'*
unfolding *not_essential_def filterlim_def filtermap_compose* *assms is_pole_compose_iff* [*OF assms*]
by *blast*

lemma *isolated_singularity_at_cong*:
assumes *eventually (λx. f x = g x) (at z) z = z'*
shows *isolated_singularity_at f z ⟷ isolated_singularity_at g z'*
proof –
have *isolated_singularity_at g z*
if *isolated_singularity_at f z eventually (λx. f x = g x) (at z)* **for** *f g*
proof –
from *that(1)* **obtain** *r* **where** *r: r > 0 f analytic_on ball z r - {z}*
by (*auto simp: isolated_singularity_at_def*)
from *that(2)* **obtain** *r'* **where** *r': r' > 0 ∀ x ∈ ball z r' - {z}. f x = g x*
unfolding *eventually_at_filter eventually_nhds_metric* **by** (*auto simp: dist_commute*)

have *f holomorphic_on ball z r - {z}*
using *r(2)* **by** (*subst (asm) analytic_on_open*) *auto*
hence *f holomorphic_on ball z (min r r') - {z}*
by (*rule holomorphic_on_subset*) *auto*
also have *?this ⟷ g holomorphic_on ball z (min r r') - {z}*
using *r'* **by** (*intro holomorphic_cong*) *auto*
also have *... ⟷ g analytic_on ball z (min r r') - {z}*
by (*subst analytic_on_open*) *auto*
finally show *?thesis*
unfolding *isolated_singularity_at_def*
by (*intro exI[of _ min r r']*) (*use ⟨r > 0⟩ ⟨r' > 0⟩ in auto*)
qed
from *this[of f g] this[of g f]* *assms* **show** *?thesis*
by (*auto simp: eq_commute*)
qed

lemma *removable_singularity*:
assumes *f holomorphic_on A - {x} open A*
assumes *f -x→ c*
shows *(λy. if y = x then c else f y) holomorphic_on A (is ?g holomorphic_on _)*
proof –
have *continuous_on A ?g*
unfolding *continuous_on_def*
proof
fix *y* **assume** *y: y ∈ A*

```

show (?g ⟶ ?g y) (at y within A)
proof (cases y = x)
  case False
  have continuous_on (A - {x}) f
    using assms(1) by (meson holomorphic_on_imp_continuous_on)
  hence (f ⟶ ?g y) (at y within A - {x})
    using False y by (auto simp: continuous_on_def)
  also have ?this ⟷ (?g ⟶ ?g y) (at y within A - {x})
    by (intro filterlim_cong refl) (auto simp: eventually_at_filter)
  also have at y within A - {x} = at y within A
    using y assms False
    by (intro at_within_nhd[of _ A - {x}]) auto
  finally show ?thesis .
next
case [simp]: True
have f -x→ c
  by fact
also have ?this ⟷ (?g ⟶ ?g y) (at y)
  by (simp add: LIM_equal)
finally show ?thesis
  by (meson Lim_at_imp_Lim_at_within)
qed
qed
moreover {
  have ?g holomorphic_on A - {x}
    using assms(1) holomorphic_transform by fastforce
}
ultimately show ?thesis
  using assms by (simp add: no_isolated_singularity)
qed

lemma removable_singularity':
  assumes isolated_singularity_at f z
  assumes f -z→ c
  shows (λy. if y = z then c else f y) analytic_on {z}
proof -
  from assms obtain r where r: r > 0 f analytic_on ball z r - {z}
    by (auto simp: isolated_singularity_at_def)
  have (λy. if y = z then c else f y) holomorphic_on ball z r
  proof (rule removable_singularity)
    show f holomorphic_on ball z r - {z}
      using r(2) by (subst (asm) analytic_on_open) auto
  qed (use assms in auto)
  thus ?thesis
    using r(1) unfolding analytic_at by (intro exI[of _ ball z r]) auto
qed

named_theorems singularity_intros introduction rules for singularities

```

```

lemma holomorphic_factor_unique:
  fixes f:: complex  $\Rightarrow$  complex and z::complex and r::real and m n::int
  assumes r>0 g z $\neq$ 0 h z $\neq$ 0
    and asm:  $\forall w \in \text{ball } z \ r - \{z\}. f \ w = g \ w * (w - z)^{\text{powi } n} \wedge g \ w \neq 0 \wedge f \ w = h \ w$ 
    *  $(w - z)^{\text{powi } m} \wedge h \ w \neq 0$ 
    and g_holo: g holomorphic_on ball z r and h_holo: h holomorphic_on ball z r
  shows n=m
proof -
  have [simp]: at z within ball z r  $\neq$  bot using <r>0>
    by (auto simp add: at_within_ball_bot_iff)
  have False when n>m
proof -
  have (h  $\longrightarrow$  0) (at z within ball z r)
    proof (rule Lim_transform_within[OF _ <r>0>, where f= $\lambda w. (w - z)^{\text{powi } (n - m)} * g \ w$ ])
    have  $\forall w \in \text{ball } z \ r - \{z\}. h \ w = (w - z)^{\text{powi } (n - m)} * g \ w$ 
      using <n>m> asm <r>0> by (simp add: field_simps power_int_diff) force
    then show  $\llbracket x' \in \text{ball } z \ r; 0 < \text{dist } x' \ z; \text{dist } x' \ z < r \rrbracket$ 
       $\implies (x' - z)^{\text{powi } (n - m)} * g \ x' = h \ x'$  for x' by auto
  next
  define F where F  $\equiv$  at z within ball z r
  define f' where f'  $\equiv \lambda x. (x - z)^{\text{powi } (n - m)}$ 
  have f' z=0 using <n>m> unfolding f'_def by auto
  moreover have continuous F f' unfolding f'_def F_def continuous_def
    using <n>m>
    by (auto simp add: Lim_ident_at intro!: tendsto_powr_complex_0 tendsto_eq_intros)
  ultimately have (f'  $\longrightarrow$  0) F unfolding F_def
    by (simp add: continuous_within)
  moreover have (g  $\longrightarrow$  g z) F
    unfolding F_def
    using <r>0> centre_in_ball continuous_on_def g_holo holomorphic_on_imp_continuous_on
by blast
  ultimately show  $((\lambda w. f' \ w * g \ w) \longrightarrow 0) \ F$  using tendsto_mult by fastforce
qed
  moreover have (h  $\longrightarrow$  h z) (at z within ball z r)
    using holomorphic_on_imp_continuous_on[OF h_holo]
    by (auto simp add: continuous_on_def <r>0>)
  ultimately have h z=0 by (auto intro!: tendsto_unique)
  thus False using <h z $\neq$ 0> by auto
qed
  moreover have False when m>n
proof -
  have (g  $\longrightarrow$  0) (at z within ball z r)
    proof (rule Lim_transform_within[OF _ <r>0>, where f= $\lambda w. (w - z)^{\text{powi } (m - n)} * h \ w$ ])
    have  $\forall w \in \text{ball } z \ r - \{z\}. g \ w = (w - z)^{\text{powi } (m - n)} * h \ w$  using <m>n> asm
      by (simp add: field_simps power_int_diff) force

```

```

then show  $\llbracket x' \in \text{ball } z \ r; \ 0 < \text{dist } x' \ z; \text{dist } x' \ z < r \rrbracket$ 
   $\implies (x' - z) \text{ powi } (m - n) * h \ x' = g \ x' \text{ for } x' \text{ by auto}$ 
next
  define  $F$  where  $F \equiv \text{at } z \text{ within ball } z \ r$ 
  define  $f'$  where  $f' \equiv \lambda x. (x - z) \text{ powi } (m - n)$ 
  have  $f' \ z = 0$  using  $\langle m > n \rangle$  unfolding  $f'_\text{def}$  by auto
  moreover have  $\text{continuous } F \ f'$  unfolding  $f'_\text{def}$   $F_\text{def}$   $\text{continuous\_def}$ 
    using  $\langle m > n \rangle$ 
  by ( $\text{auto simp: Lim\_ident\_at intro!: tendsto\_powr\_complex\_0 tendsto\_eq\_intros}$ )
  ultimately have  $(f' \longrightarrow 0) \ F$  unfolding  $F_\text{def}$ 
    by ( $\text{simp add: continuous\_within}$ )
  moreover have  $(h \longrightarrow h \ z) \ F$ 
    using  $\text{holomorphic\_on\_imp\_continuous\_on}[OF \ h\_holo, \text{unfolded continuous\_on\_def}] \ \langle r > 0 \rangle$ 
    unfolding  $F_\text{def}$  by auto
  ultimately show  $((\lambda w. f' \ w * h \ w) \longrightarrow 0) \ F$  using  $\text{tendsto\_mult}$  by
fastforce
qed
moreover have  $(g \longrightarrow g \ z) \ (\text{at } z \text{ within ball } z \ r)$ 
  using  $\text{holomorphic\_on\_imp\_continuous\_on}[OF \ g\_holo]$ 
  by ( $\text{auto simp add: continuous\_on\_def } \langle r > 0 \rangle$ )
ultimately have  $g \ z = 0$  by ( $\text{auto intro!: tendsto\_unique}$ )
thus  $\text{False}$  using  $\langle g \ z \neq 0 \rangle$  by auto
qed
ultimately show  $n = m$  by fastforce
qed

lemma  $\text{holomorphic\_factor\_puncture}$ :
  assumes  $f\_iso$ :  $\text{isolated\_singularity\_at } f \ z$ 
    and  $\text{not\_essential}$   $f \ z$  —  $f$  has either a removable singularity or a pole at  $z$ 
    and  $\text{non\_zero}$ :  $\exists \ F \ w \text{ in } (\text{at } z). \ f \ w \neq 0$  —  $f$  will not be constantly zero in a
neighbour of  $z$ 
  shows  $\exists ! n :: \text{int}. \ \exists \ g \ r. \ 0 < r \wedge g \text{ holomorphic\_on } \text{cball } z \ r \wedge g \ z \neq 0$ 
     $\wedge (\forall w \in \text{cball } z \ r - \{z\}. \ f \ w = g \ w * (w - z) \text{ powi } n \wedge g \ w \neq 0)$ 
proof —
  define  $P$  where  $P = (\lambda f \ n \ g \ r. \ 0 < r \wedge g \text{ holomorphic\_on } \text{cball } z \ r \wedge g \ z \neq 0$ 
     $\wedge (\forall w \in \text{cball } z \ r - \{z\}. \ f \ w = g \ w * (w - z) \text{ powi } n \wedge g \ w \neq 0))$ 
  have  $\text{imp\_unique}$ :  $\exists ! n :: \text{int}. \ \exists \ g \ r. \ P \ f \ n \ g \ r$  when  $\exists \ n \ g \ r. \ P \ f \ n \ g \ r$ 
  proof ( $\text{rule ex\_ex1I}[OF \ \text{that}]$ )
    fix  $n1 \ n2 :: \text{int}$ 
    assume  $g1\_asm$ :  $\exists \ g1 \ r1. \ P \ f \ n1 \ g1 \ r1$  and  $g2\_asm$ :  $\exists \ g2 \ r2. \ P \ f \ n2 \ g2 \ r2$ 
    define  $\text{fac}$  where  $\text{fac} \equiv \lambda n \ g \ r. \ \forall w \in \text{cball } z \ r - \{z\}. \ f \ w = g \ w * (w - z) \text{ powi } n$ 
 $\wedge g \ w \neq 0$ 
    obtain  $g1 \ r1$  where  $0 < r1$  and  $g1\_holo$ :  $g1 \text{ holomorphic\_on } \text{cball } z \ r1$  and
 $g1 \ z \neq 0$ 
    and  $\text{fac } n1 \ g1 \ r1$  using  $g1\_asm$  unfolding  $P_\text{def}$   $\text{fac\_def}$  by auto
    obtain  $g2 \ r2$  where  $0 < r2$  and  $g2\_holo$ :  $g2 \text{ holomorphic\_on } \text{cball } z \ r2$  and
 $g2 \ z \neq 0$ 
    and  $\text{fac } n2 \ g2 \ r2$  using  $g2\_asm$  unfolding  $P_\text{def}$   $\text{fac\_def}$  by auto

```

```

define r where r  $\equiv$  min r1 r2
have r>0 using <r1>0> <r2>0> unfolding r_def by auto
moreover have  $\forall w \in \text{ball } z \text{ } r - \{z\}. f \ w = g1 \ w * (w - z)^{\text{powi } n1} \wedge g1 \ w \neq 0$ 
 $\wedge f \ w = g2 \ w * (w - z)^{\text{powi } n2} \wedge g2 \ w \neq 0$ 
using <fac n1 g1 r1> <fac n2 g2 r2> unfolding fac_def r_def
by fastforce
ultimately show n1=n2
using g1_holo g2_holo <g1 z $\neq$ 0> <g2 z $\neq$ 0>
apply (elim holomorphic_factor_unique)
by (auto simp add: r_def)
qed

have P_exist:  $\exists \ n \ g \ r. P \ h \ n \ g \ r$  when
 $\exists z'. (h \longrightarrow z') \ (at \ z) \ \text{isolated\_singularity\_at } h \ z \ \exists_F w \ \text{in } (at \ z). \ h \ w \neq 0$ 
for h
proof -
from that(2) obtain r where r>0 and r: h analytic_on ball z r - {z}
unfolding isolated_singularity_at_def by auto
obtain z' where (h  $\longrightarrow$  z') (at z) using < $\exists z'. (h \longrightarrow z') \ (at \ z)$ > by auto
define h' where h'=( $\lambda x. \text{if } x=z \text{ then } z' \text{ else } h \ x$ )
have h' holomorphic_on ball z r
proof (rule no_isolated_singularity'[of {z}])
show  $\bigwedge w. w \in \{z\} \implies (h' \longrightarrow h' \ w) \ (at \ w \ \text{within } \text{ball } z \ r)$ 
by (simp add: LIM_cong Lim_at_imp_Lim_at_within <h - z  $\rightarrow$  z'> h'_def)
show h' holomorphic_on ball z r - {z}
using r analytic_imp_holomorphic h'_def holomorphic_transform by
fastforce
qed auto
have ?thesis when z'=0
proof -
have h' z=0 using that unfolding h'_def by auto
moreover have  $\neg \ h' \ \text{constant\_on } \text{ball } z \ r$ 
using < $\exists_F w \ \text{in } (at \ z). \ h \ w \neq 0$ > unfolding constant_on_def frequently_def
eventually_at h'_def
by (metis <0 < r> centre_in_ball dist_commute mem_ball that)
moreover note <h' holomorphic_on ball z r>
ultimately obtain g r1 n where 0 < n 0 < r1 ball z r1  $\subseteq$  ball z r and
g: g holomorphic_on ball z r1
 $\bigwedge w. w \in \text{ball } z \ r1 \implies h' \ w = (w - z)^{\wedge n} * g \ w$ 
 $\bigwedge w. w \in \text{ball } z \ r1 \implies g \ w \neq 0$ 
using holomorphic_factor_zero_nonconstant[of _ ball z r z thesis,simplified,
OF <h' holomorphic_on ball z r> <r>0> <h' z=0> < $\neg \ h' \ \text{constant\_on}$ 
ball z r>]
by (auto simp add: dist_commute)
define rr where rr=r1/2
have P h' n g rr
unfolding P_def rr_def
using <n>0> <r1>0> g by (auto simp add: powr_nat)
then have P h n g rr

```



```

      unfolding h'_def P_def by auto
    then show ?thesis unfolding P_def by blast
  qed
  moreover have ?thesis when z' ≠ 0
  proof -
    have h' z ≠ 0 using that unfolding h'_def by auto
    obtain r1 where r1 > 0 cball z r1 ⊆ ball z r ∀ x ∈ cball z r1. h' x ≠ 0
    proof -
      have isCont h' z h' z ≠ 0
      by (auto simp add: Lim_cong_within ⟨h - z → z'⟩ ⟨z' ≠ 0⟩ continuous_at
h'_def)
      then obtain r2 where r2: r2 > 0 ∀ x ∈ ball z r2. h' x ≠ 0
      using continuous_at_avoid[of z h' 0] unfolding ball_def by auto
      define r1 where r1 = min r2 r / 2
      have 0 < r1 cball z r1 ⊆ ball z r
      using ⟨r2 > 0⟩ ⟨r > 0⟩ unfolding r1_def by auto
      moreover have ∀ x ∈ cball z r1. h' x ≠ 0
      using r2 unfolding r1_def by simp
      ultimately show ?thesis using that by auto
    qed
    then have P h' 0 h' r1 using ⟨h' holomorphic_on ball z r⟩ unfolding P_def
  by auto
    then have P h 0 h' r1 unfolding P_def h'_def by auto
    then show ?thesis unfolding P_def by blast
  qed
  ultimately show ?thesis by auto
qed

have ?thesis when ∃ x. (f ⟶ x) (at z)
  apply (rule_tac imp_unique[unfolded P_def])
  using P_exist[OF that(1) f_iso non_zero] unfolding P_def .
moreover have ?thesis when is_pole f z
proof (rule imp_unique[unfolded P_def])
  obtain e where [simp]: e > 0 and e_holo: f holomorphic_on ball z e - {z}
and e_nz: ∀ x ∈ ball z e - {z}. f x ≠ 0
  proof -
    have ∀_F z in at z. f z ≠ 0
    using ⟨is_pole f z⟩ filterlim_at_infinity_imp_eventually_ne unfolding
is_pole_def
    by auto
    then obtain e1 where e1: e1 > 0 ∀ x ∈ ball z e1 - {z}. f x ≠ 0
    using that eventually_at[of λx. f x ≠ 0 z UNIV, simplified] by (auto simp
add: dist_commute)
    obtain e2 where e2: e2 > 0 f holomorphic_on ball z e2 - {z}
    using f_iso analytic_imp_holomorphic unfolding isolated_singularity_at_def
  by auto
    show ?thesis
    using e1 e2 by (force intro: that[of min e1 e2])
  qed

```

```

define  $h$  where  $h \equiv \lambda x. \text{inverse } (f\ x)$ 
have  $\exists n\ g\ r. P\ h\ n\ g\ r$ 
proof –
  have  $(\lambda x. \text{inverse } (f\ x)) \text{ analytic\_on ball } z\ e - \{z\}$ 
  by (metis e_holo e_nz open_ball analytic_on_open holomorphic_on_inverse
open_delete)
  moreover have  $h - z \rightarrow 0$ 
  using Lim_transform_within_open assms(2) h_def is_pole_tendsto that
by fastforce
  moreover have  $\exists_F w \text{ in } (at\ z). h\ w \neq 0$ 
  using non_zero by (simp add: h_def)
  ultimately show ?thesis
  using P_exist[of h]  $\langle e > 0 \rangle$ 
  unfolding isolated_singularity_at_def h_def
  by blast
qed
then obtain  $n\ g\ r$ 
where  $0 < r$  and
   $g\_holo: g \text{ holomorphic\_on cball } z\ r \text{ and } g\ z \neq 0 \text{ and}$ 
   $g\_fac: (\forall w \in \text{cball } z\ r - \{z\}. h\ w = g\ w * (w - z)^{\text{powi } n} \wedge g\ w \neq 0)$ 
  unfolding P_def by auto
have  $P\ f\ (-n)\ (\text{inverse } o\ g)\ r$ 
proof –
  have  $f\ w = \text{inverse } (g\ w) * (w - z)^{\text{powi } (-n)}$  when  $w \in \text{cball } z\ r - \{z\}$  for  $w$ 
  by (metis g_fac h_def inverse_inverse_eq inverse_mult_distrib power_int_minus
that)
  then show ?thesis
  unfolding P_def comp_def
  using  $\langle r > 0 \rangle$  g_holo g_fac  $\langle g\ z \neq 0 \rangle$  by (auto intro: holomorphic_intros)
qed
then show  $\exists x\ g\ r. 0 < r \wedge g \text{ holomorphic\_on cball } z\ r \wedge g\ z \neq 0$ 
   $\wedge (\forall w \in \text{cball } z\ r - \{z\}. f\ w = g\ w * (w - z)^{\text{powi } x} \wedge g\ w \neq 0)$ 
  unfolding P_def by blast
qed
ultimately show ?thesis
using  $\langle \text{not\_essential } f\ z \rangle$  unfolding not_essential_def by presburger
qed

lemma not_essential_transform:
  assumes not_essential  $g\ z$ 
  assumes  $\forall_F w \text{ in } (at\ z). g\ w = f\ w$ 
  shows not_essential  $f\ z$ 
  using assms unfolding not_essential_def
  by (simp add: filterlim_cong is_pole_cong)

lemma isolated_singularity_at_transform:
  assumes isolated_singularity_at  $g\ z$ 
  assumes  $\forall_F w \text{ in } (at\ z). g\ w = f\ w$ 

```

shows *isolated_singularity_at* $f\ z$
 using *assms isolated_singularity_at_cong* by blast

lemma *not_essential_powr*[*singularity_intros*]:
 assumes *LIM* $w\ (at\ z).$ $f\ w\ :\>\ (at\ x)$
 shows *not_essential* $(\lambda w. (f\ w)\ powi\ n)\ z$
 proof -
 define *fp* where $fp = (\lambda w. (f\ w)\ powi\ n)$
 have *?thesis* when $n > 0$
 proof -
 have $(\lambda w. (f\ w)\ ^{(nat\ n)}) -z \rightarrow x\ ^{nat\ n}$
 using *that assms unfolding filterlim_at* by (auto intro!: *tendsto_eq_intros*)
 then have $fp -z \rightarrow x\ ^{nat\ n}$ *unfolding fp_def*
 by (smt (verit) *LIM_cong power_int_def that*)
 then show *?thesis unfolding not_essential_def fp_def* by auto
 qed
 moreover have *?thesis* when $n = 0$
 proof -
 have $\forall_F\ x\ in\ at\ z. fp\ x = 1$
 using *that filterlim_at_within_not_equal[OF assms]* by (auto simp: *fp_def*)
 then have $fp -z \rightarrow 1$
 by (simp add: *tendsto_eventually*)
 then show *?thesis unfolding fp_def not_essential_def* by auto
 qed
 moreover have *?thesis* when $n < 0$
 proof (cases $x = 0$)
 case True
 have $(\lambda x. f\ x\ ^{nat\ (-n)}) -z \rightarrow 0$
 using *assms True that unfolding filterlim_at* by (auto intro!: *tendsto_eq_intros*)
 moreover have $\forall_F\ x\ in\ at\ z. f\ x\ ^{nat\ (-n)} \neq 0$
 by (smt (verit) *True assms eventually_at_topological filterlim_at power_eq_0_iff*)
 ultimately have *LIM* $w\ (at\ z).$ $inverse\ ((f\ w)\ ^{(nat\ (-n)})) :\>\ at_infinity$
 by (metis *filterlim_atI filterlim_compose filterlim_inverse_at_infinity*)
 then have *LIM* $w\ (at\ z).$ $fp\ w\ :\>\ at_infinity$
 proof (elim *filterlim_mono_eventually*)
 show $\forall_F\ x\ in\ at\ z. inverse\ (f\ x\ ^{nat\ (-n)}) = fp\ x$
 using *filterlim_at_within_not_equal[OF assms, of 0]* *unfolding fp_def*
 by (smt (verit) *eventuallyI power_int_def power_inverse that*)
 qed auto
 then show *?thesis unfolding fp_def not_essential_def is_pole_def* by auto
 next
 case False
 let $?xx = inverse\ (x\ ^{(nat\ (-n)}))$
 have $(\lambda w. inverse\ ((f\ w)\ ^{(nat\ (-n)}))) -z \rightarrow ?xx$
 using *assms False unfolding filterlim_at* by (auto intro!: *tendsto_eq_intros*)
 then have $fp -z \rightarrow ?xx$
 by (smt (verit, best) *LIM_cong fp_def power_int_def power_inverse that*)
 then show *?thesis unfolding fp_def not_essential_def* by auto
 qed

ultimately show ?thesis by linarith
qed

lemma isolated_singularity_at_powr[singularity_intros]:
 assumes isolated_singularity_at f z $\forall_F w$ in (at z). f w $\neq 0$
 shows isolated_singularity_at ($\lambda w. (f w)^{\text{powi } n}$) z
 proof -
 obtain r1 where r1 > 0 f analytic_on ball z r1 - {z}
 using assms(1) unfolding isolated_singularity_at_def by auto
 then have r1: f holomorphic_on ball z r1 - {z}
 using analytic_on_open[of ball z r1 - {z} f] by blast
 obtain r2 where r2 > 0 and r2: $\forall w. w \neq z \wedge \text{dist } w z < r2 \longrightarrow f w \neq 0$
 using assms(2) unfolding eventually_at by auto
 define r3 where r3 = min r1 r2
 have ($\lambda w. (f w)^{\text{powi } n}$) holomorphic_on ball z r3 - {z}
 by (intro holomorphic_on_power_int) (use r1 r2 in <auto simp: dist_commute r3_def>)
 moreover have r3 > 0 unfolding r3_def using <0 < r1> <0 < r2> by linarith
 ultimately show ?thesis
 by (meson open_ball analytic_on_open isolated_singularity_at_def open_delete)
 qed

lemma non_zero_neighbour:
 assumes f_iso: isolated_singularity_at f z
 and f_ness: not_essential f z
 and f_nconst: $\exists_F w$ in (at z). f w $\neq 0$
 shows $\forall_F w$ in (at z). f w $\neq 0$
 proof -
 obtain fn fp fr
 where [simp]: fp z $\neq 0$ and fr > 0
 and fr: fp holomorphic_on cball z fr
 $\bigwedge w. w \in \text{cball } z \text{ fr} - \{z\} \implies f w = fp w * (w - z)^{\text{powi } fn} \wedge fp w \neq 0$
 using holomorphic_factor_puncture[OF f_iso f_ness f_nconst] by auto
 have f w $\neq 0$ when w $\neq z$ dist w z < fr for w
 proof -
 have f w = fp w * (w - z)^{powi fn} fp w $\neq 0$
 using fr that by (auto simp add: dist_commute)
 moreover have (w - z)^{powi fn} $\neq 0$
 unfolding powr_eq_0_iff using <w \neq z> by auto
 ultimately show ?thesis by auto
 qed
 then show ?thesis using <fr > 0> unfolding eventually_at by auto
 qed

lemma non_zero_neighbour_pole:
 assumes is_pole f z
 shows $\forall_F w$ in (at z). f w $\neq 0$
 using assms filterlim_at_infinity_imp_eventually_ne[of f at z 0]
 unfolding is_pole_def by auto

```

lemma non_zero_neighbour_alt:
  assumes holo:  $f$  holomorphic_on  $S$ 
    and open  $S$  connected  $S$   $z \in S$   $\beta \in S$   $f \beta \neq 0$ 
    shows  $\forall_F w$  in  $(at\ z).$   $f w \neq 0 \wedge w \in S$ 
proof (cases  $f\ z = 0$ )
  case True
    from isolated_zeros[OF holo  $\langle$ open  $S\rangle$   $\langle$ connected  $S\rangle$   $\langle$  $z \in S\rangle$  True  $\langle$  $\beta \in S\rangle$   $\langle$  $f \beta \neq 0\rangle$ ]
    obtain  $r$  where  $0 < r$  ball  $z\ r \subseteq S$   $\forall w \in$  ball  $z\ r - \{z\}. f w \neq 0$  by metis
    then show ?thesis
      by (smt (verit) open_ball centre_in_ball eventually_at_topological_insertE
insert_Diff subsetD)
  next
    case False
    obtain  $r1$  where  $r1 > 0$   $\forall y. dist\ z\ y < r1 \longrightarrow f\ y \neq 0$ 
      using continuous_at_avoid[OF  $z\ f$ , OF _ False] assms continuous_on_eq_continuous_at
        holo holomorphic_on_imp_continuous_on by blast
    obtain  $r2$  where  $r2 > 0$  ball  $z\ r2 \subseteq S$ 
      using assms openE by blast
    show ?thesis unfolding eventually_at
      by (metis (no_types) dist_commute order.strict_trans linorder_less_linear
mem_ball  $r1\ r2$  subsetD)
  qed

lemma not_essential_times[singularity_intros]:
  assumes  $f\_ness$ : not_essential  $f\ z$  and  $g\_ness$ : not_essential  $g\ z$ 
  assumes  $f\_iso$ : isolated_singularity_at  $f\ z$  and  $g\_iso$ : isolated_singularity_at  $g\ z$ 
  shows not_essential  $(\lambda w. f\ w * g\ w)\ z$ 
proof -
  define  $fg$  where  $fg = (\lambda w. f\ w * g\ w)$ 
  have ?thesis when  $\neg ((\exists_F w$  in  $(at\ z).$   $f\ w \neq 0) \wedge (\exists_F w$  in  $(at\ z).$   $g\ w \neq 0))$ 
  proof -
    have  $\forall_F w$  in  $(at\ z).$   $fg\ w = 0$ 
      using  $fg\_def$  frequently_elim1 not_eventually_that by fastforce
    from tendsto_cong[OF this] have  $fg \rightarrow z \rightarrow 0$  by auto
    then show ?thesis unfolding not_essential_def  $fg\_def$  by auto
  qed
  moreover have ?thesis when  $f\_nconst$ :  $\exists_F w$  in  $(at\ z).$   $f\ w \neq 0$  and  $g\_nconst$ :
 $\exists_F w$  in  $(at\ z).$   $g\ w \neq 0$ 
  proof -
    obtain  $fn\ fp\ fr$  where [simp]:  $fp\ z \neq 0$  and  $fr > 0$ 
      and  $fr$ :  $fp$  holomorphic_on cball  $z\ fr$ 
       $\forall w \in$  cball  $z\ fr - \{z\}. f\ w = fp\ w * (w - z)^{powi\ fn} \wedge fp\ w \neq 0$ 
    using holomorphic_factor_puncture[OF  $f\_iso\ f\_ness\ f\_nconst$ ] by auto
    obtain  $gn\ gp\ gr$  where [simp]:  $gp\ z \neq 0$  and  $gr > 0$ 
      and  $gr$ :  $gp$  holomorphic_on cball  $z\ gr$ 
       $\forall w \in$  cball  $z\ gr - \{z\}. g\ w = gp\ w * (w - z)^{powi\ gn} \wedge gp\ w \neq 0$ 

```

```

using holomorphic_factor_puncture[OF g_iso g_ness g_nconst] by auto

define r1 where r1=(min fr gr)
have r1>0 unfolding r1_def using ‹fr>0› ‹gr>0› by auto
have fg_times: fg w = (fp w * gp w) * (w-z) powi (fn+gn) and fgp_nz: fp
w*gp w≠0
  when w∈ball z r1 - {z} for w
proof -
  have f w = fp w * (w-z) powi fn fp w≠0
    using fr that unfolding r1_def by auto
  moreover have g w = gp w * (w-z) powi gn gp w ≠ 0
    using gr that unfolding r1_def by auto
  ultimately show fg w = (fp w * gp w) * (w-z) powi (fn+gn) fp w*gp w≠0
    using that by (auto simp add: fg_def power_int_add)
qed

obtain [intro]: fp -z→fp z gp -z→gp z
  using fr(1) ‹fr>0› gr(1) ‹gr>0›
  by (metis centre_in_ball continuous_at continuous_on_interior
    holomorphic_on_imp_continuous_on interior_cball)
have ?thesis when fn+gn>0
proof -
  have (λw. (fp w * gp w) * (w-z) ^ (nat (fn+gn))) -z→0
    using that by (auto intro!:tendsto_eq_intros)
  then have fg -z→ 0
    using Lim_transform_within[OF _ ‹r1>0›]
  by (smt (verit, best) Diff_iff dist_commute fg_times mem_ball power_int_def
    singletonD that zero_less_dist_iff)
  then show ?thesis unfolding not_essential_def fg_def by auto
qed
moreover have ?thesis when fn+gn=0
proof -
  have (λw. fp w * gp w) -z→fp z*gp z
    using that by (auto intro!:tendsto_eq_intros)
  then have fg -z→ fp z*gp z
    apply (elim Lim_transform_within[OF _ ‹r1>0›])
    apply (subst fg_times)
    by (auto simp add: dist_commute that)
  then show ?thesis unfolding not_essential_def fg_def by auto
qed
moreover have ?thesis when fn+gn<0
proof -
  have LIM x at z. (x - z) ^ nat (- (fn + gn)) :> at 0
    using eventually_at_topological that
    by (force intro!: tendsto_eq_intros filterlim_atI)
  moreover have ∃ c. (λc. fp c * gp c) -z→ c ∧ 0 ≠ c
    using ‹fp -z→ fp z› ‹gp -z→ gp z› tendsto_mult by fastforce
  ultimately have LIM w (at z). fp w * gp w / (w-z) ^ nat (- (fn+gn)) :>
    at_infinity

```

```

    using filterlim_divide_at_infinity by blast
  then have is_pole fg z unfolding is_pole_def
    apply (elim filterlim_transform_within[OF _ _ <r1>0])
    using that
      by (simp_all add: dist_commute fg_times of_int_of_nat divide_simps
power_int_def del: minus_add_distrib)
    then show ?thesis unfolding not_essential_def fg_def by auto
  qed
  ultimately show ?thesis unfolding not_essential_def fg_def by fastforce
  qed
  ultimately show ?thesis by auto
  qed

lemma not_essential_inverse[singularity_intros]:
  assumes f_ness: not_essential f z
  assumes f_iso: isolated_singularity_at f z
  shows not_essential ( $\lambda w. \text{inverse } (f w)$ ) z
proof -
  define vf where vf = ( $\lambda w. \text{inverse } (f w)$ )
  have ?thesis when  $\neg(\exists_F w \text{ in } (at z). f w \neq 0)$ 
  proof -
    have  $\forall_F w \text{ in } (at z). f w = 0$ 
      using not_eventually that by fastforce
    then have vf  $\rightarrow 0$ 
      unfolding vf_def by (simp add: tendsto_eventually)
    then show ?thesis
      unfolding not_essential_def vf_def by auto
  qed
  moreover have ?thesis when is_pole f z
    by (metis (mono_tags, lifting) filterlim_at filterlim_inverse_at_iff is_pole_def
not_essential_def that)
  moreover have ?thesis when  $\exists x. f \rightarrow x$  and f_nconst:  $\exists_F w \text{ in } (at z). f w \neq 0$ 
  proof -
    from that obtain fz where fz:  $f \rightarrow fz$  by auto
    have ?thesis when fz = 0
      proof -
        have ( $\lambda w. \text{inverse } (f w)$ )  $\rightarrow 0$ 
          using fz that unfolding vf_def by auto
        moreover have  $\forall_F w \text{ in } at z. \text{inverse } (f w) \neq 0$ 
          using non_zero_neighbour[OF f_iso f_ness f_nconst]
          unfolding vf_def by auto
        ultimately show ?thesis unfolding not_essential_def vf_def
          using filterlim_atI filterlim_inverse_at_iff is_pole_def by blast
      qed
    moreover have ?thesis when fz  $\neq 0$ 
      using fz not_essential_def tendsto_inverse that by blast
    ultimately show ?thesis by auto
  qed
  qed

```

```

ultimately show ?thesis using f_ness unfolding not_essential_def by auto
qed

lemma isolated_singularity_at_inverse[singularity_intros]:
  assumes f_iso: isolated_singularity_at f z
    and f_ness: not_essential f z
  shows isolated_singularity_at ( $\lambda w. \text{inverse } (f w)$ ) z
proof -
  define vf where vf = ( $\lambda w. \text{inverse } (f w)$ )
  have ?thesis when  $\neg(\exists_F w \text{ in } (at z). f w \neq 0)$ 
  proof -
    have  $\forall_F w \text{ in } (at z). f w = 0$ 
    using that[unfolded frequently_def, simplified] by (auto elim: eventually_rev_mp)
    then have  $\forall_F w \text{ in } (at z). vf w = 0$ 
    unfolding vf_def by auto
    then obtain d1 where d1 > 0 and d1:  $\forall x. x \neq z \wedge \text{dist } x z < d1 \longrightarrow vf x =$ 
0
    unfolding eventually_at by auto
    then have vf holomorphic_on ball z d1 - {z}
    using holomorphic_transform[of  $\lambda_. 0$ ]
    by (metis Diff_iff dist_commute holomorphic_on_const insert_iff mem_ball)
    then have vf analytic_on ball z d1 - {z}
    by (simp add: analytic_on_open open_delete)
    then show ?thesis
    using  $\langle d1 > 0 \rangle$  unfolding isolated_singularity_at_def vf_def by auto
  qed
  moreover have ?thesis when f_nconst:  $\exists_F w \text{ in } (at z). f w \neq 0$ 
  proof -
    have  $\forall_F w \text{ in } at z. f w \neq 0$  using non_zero_neighbour[OF f_iso f_ness f_nconst].
    then obtain d1 where d1:  $d1 > 0 \forall x. x \neq z \wedge \text{dist } x z < d1 \longrightarrow f x \neq 0$ 
    unfolding eventually_at by auto
    obtain d2 where d2 > 0 and d2:  $f \text{ analytic\_on ball } z d2 - \{z\}$ 
    using f_iso unfolding isolated_singularity_at_def by auto
    define d3 where d3 = min d1 d2
    have d3 > 0 unfolding d3_def using  $\langle d1 > 0 \rangle \langle d2 > 0 \rangle$  by auto
    moreover
    have f analytic_on ball z d3 - {z}
    by (smt (verit, best) Diff_iff analytic_on_analytic_at d2 d3_def mem_ball)
    then have vf analytic_on ball z d3 - {z}
    unfolding vf_def
    by (intro analytic_on_inverse; simp add: d1(2) d3_def dist_commute)
    ultimately show ?thesis
    unfolding isolated_singularity_at_def vf_def by auto
  qed
  ultimately show ?thesis by auto
qed

```

```

lemma not_essential_divide[singularity_intros]:

```



```

assumes  $f\_ness$ :  $not\_essential\ f\ z$  and  $g\_ness$ :  $not\_essential\ g\ z$ 
assumes  $f\_iso$ :  $isolated\_singularity\_at\ f\ z$  and  $g\_iso$ :  $isolated\_singularity\_at\ g\ z$ 
shows  $not\_essential\ (\lambda w. f\ w / g\ w)\ z$ 
proof –
  have  $not\_essential\ (\lambda w. f\ w * inverse\ (g\ w))\ z$ 
    by ( $simp\ add: f\_iso\ f\_ness\ g\_iso\ g\_ness\ isolated\_singularity\_at\_inverse$ 
       $not\_essential\_inverse\ not\_essential\_times$ )
  then show  $?thesis$  by ( $simp\ add: field\_simps$ )
qed

```

lemma

```

assumes  $f\_iso$ :  $isolated\_singularity\_at\ f\ z$ 
and  $g\_iso$ :  $isolated\_singularity\_at\ g\ z$ 
shows  $isolated\_singularity\_at\_times[singularity\_intros]$ :
   $isolated\_singularity\_at\ (\lambda w. f\ w * g\ w)\ z$ 
and  $isolated\_singularity\_at\_add[singularity\_intros]$ :
   $isolated\_singularity\_at\ (\lambda w. f\ w + g\ w)\ z$ 
proof –
  obtain  $d1\ d2$  where  $d1 > 0\ d2 > 0$ 
    and  $d1$ :  $f\ analytic\_on\ ball\ z\ d1 - \{z\}$  and  $d2$ :  $g\ analytic\_on\ ball\ z\ d2 - \{z\}$ 
    using  $f\_iso\ g\_iso$  unfolding  $isolated\_singularity\_at\_def$  by  $auto$ 
  define  $d3$  where  $d3 = \min\ d1\ d2$ 
  have  $d3 > 0$  unfolding  $d3\_def$  using  $\langle d1 > 0 \rangle\ \langle d2 > 0 \rangle$  by  $auto$ 

  have  $fan$ :  $f\ analytic\_on\ ball\ z\ d3 - \{z\}$ 
    by ( $smt\ (verit, best)\ Diff\_iff\ analytic\_on\_analytic\_at\ d1\ d3\_def\ mem\_ball$ )
  have  $gan$ :  $g\ analytic\_on\ ball\ z\ d3 - \{z\}$ 
    by ( $smt\ (verit, best)\ Diff\_iff\ analytic\_on\_analytic\_at\ d2\ d3\_def\ mem\_ball$ )
  have  $(\lambda w. f\ w * g\ w)\ analytic\_on\ ball\ z\ d3 - \{z\}$ 
    using  $analytic\_on\_mult\ fan\ gan$  by  $blast$ 
  then show  $isolated\_singularity\_at\ (\lambda w. f\ w * g\ w)\ z$ 
    using  $\langle d3 > 0 \rangle$  unfolding  $isolated\_singularity\_at\_def$  by  $auto$ 
  have  $(\lambda w. f\ w + g\ w)\ analytic\_on\ ball\ z\ d3 - \{z\}$ 
    using  $analytic\_on\_add\ fan\ gan$  by  $blast$ 
  then show  $isolated\_singularity\_at\ (\lambda w. f\ w + g\ w)\ z$ 
    using  $\langle d3 > 0 \rangle$  unfolding  $isolated\_singularity\_at\_def$  by  $auto$ 
qed

```

lemma $isolated_singularity_at_uminus[singularity_intros]$:

```

assumes  $f\_iso$ :  $isolated\_singularity\_at\ f\ z$ 
shows  $isolated\_singularity\_at\ (\lambda w. -f\ w)\ z$ 
using  $assms$  unfolding  $isolated\_singularity\_at\_def$  using  $analytic\_on\_neg$  by  $blast$ 

```

lemma $isolated_singularity_at_id[singularity_intros]$:

```

 $isolated\_singularity\_at\ (\lambda w. w)\ z$ 
unfolding  $isolated\_singularity\_at\_def$  by ( $simp\ add: gt\_ex$ )

```

```

lemma isolated_singularity_at_minus[singularity_intros]:
  assumes isolated_singularity_at  $f\ z$  and isolated_singularity_at  $g\ z$ 
  shows isolated_singularity_at  $(\lambda w. f\ w - g\ w)\ z$ 
  unfolding diff_conv_add_uminus
  using assms isolated_singularity_at_add isolated_singularity_at_uminus by
blast

```

```

lemma isolated_singularity_at_divide[singularity_intros]:
  assumes isolated_singularity_at  $f\ z$ 
    and isolated_singularity_at  $g\ z$ 
    and not_essential  $g\ z$ 
  shows isolated_singularity_at  $(\lambda w. f\ w / g\ w)\ z$ 
  unfolding divide_inverse
  by (simp add: assms isolated_singularity_at_inverse isolated_singularity_at_times)

```

```

lemma isolated_singularity_at_const[singularity_intros]:
  isolated_singularity_at  $(\lambda w. c)\ z$ 
  unfolding isolated_singularity_at_def by (simp add: gt_ex)

```

```

lemma isolated_singularity_at_holomorphic:
  assumes  $f$  holomorphic_on  $s - \{z\}$  open  $s\ z \in s$ 
  shows isolated_singularity_at  $f\ z$ 
  using assms unfolding isolated_singularity_at_def
  by (metis analytic_on_holomorphic centre_in_ball insert_Diff openE open_delete
subset_insert_iff)

```

```

lemma isolated_singularity_at_altdef:
  isolated_singularity_at  $f\ z \longleftrightarrow$  eventually  $(\lambda z. f\ \text{analytic\_on}\ \{z\})\ (at\ z)$ 
proof
  assume isolated_singularity_at  $f\ z$ 
  then obtain  $r$  where  $r: r > 0\ f\ \text{analytic\_on}\ \text{ball}\ z\ r - \{z\}$ 
    unfolding isolated_singularity_at_def by blast
  have eventually  $(\lambda w. w \in \text{ball}\ z\ r - \{z\})\ (at\ z)$ 
    using  $r(1)$  by (intro eventually_at_in_open) auto
  thus eventually  $(\lambda z. f\ \text{analytic\_on}\ \{z\})\ (at\ z)$ 
    by eventually_elim (use  $r$  analytic_on_subset in auto)
next
  assume eventually  $(\lambda z. f\ \text{analytic\_on}\ \{z\})\ (at\ z)$ 
  then obtain  $A$  where  $A: \text{open}\ A\ z \in A \wedge w. w \in A - \{z\} \implies f\ \text{analytic\_on}\ \{w\}$ 
    unfolding eventually_at_topological by blast
  then show isolated_singularity_at  $f\ z$ 
    by (meson analytic_imp_holomorphic analytic_on_analytic_at isolated_singularity_at_holomorphic)
qed

```

```

lemma isolated_singularity_at_shift:
  assumes isolated_singularity_at  $(\lambda x. f\ (x + w))\ z$ 
  shows isolated_singularity_at  $f\ (z + w)$ 
proof –

```

```

from assms obtain r where r:  $r > 0$  and ana:  $(\lambda x. f (x + w))$  analytic_on
 $\text{ball } z \text{ } r - \{z\}$ 
  unfolding isolated_singularity_at_def by blast
  have  $((\lambda x. f (x + w)) \circ (\lambda x. x - w))$  analytic_on  $(\text{ball } (z + w) \text{ } r - \{z + w\})$ 
  by (rule analytic_on_compose_gen[OF _ ana])
    (auto simp: dist_norm algebra_simps intro!: analytic_intros)
  hence f analytic_on  $(\text{ball } (z + w) \text{ } r - \{z + w\})$ 
  by (simp add: o_def)
  thus ?thesis using r
  unfolding isolated_singularity_at_def by blast
qed

```

```

lemma isolated_singularity_at_shift_iff:
  isolated_singularity_at f  $(z + w) \longleftrightarrow \text{isolated\_singularity\_at } (\lambda x. f (x + w)) \text{ } z$ 
  using isolated_singularity_at_shift[of f w z]
    isolated_singularity_at_shift[of  $\lambda x. f (x + w) - w$  w + z]
  by (auto simp: algebra_simps)

```

```

lemma isolated_singularity_at_shift_0:
  NO_MATCH  $0 \text{ } z \implies \text{isolated\_singularity\_at } f \text{ } z \longleftrightarrow \text{isolated\_singularity\_at}$ 
 $(\lambda x. f (z + x)) \text{ } 0$ 
  using isolated_singularity_at_shift_iff[of f 0 z] by (simp add: add_ac)

```

```

lemma not_essential_shift:
  assumes not_essential  $(\lambda x. f (x + w)) \text{ } z$ 
  shows not_essential f  $(z + w)$ 
proof -
  from assms consider c where  $(\lambda x. f (x + w)) - z \rightarrow c \mid \text{is\_pole } (\lambda x. f (x + w)) \text{ } z$ 
  unfolding not_essential_def by blast
  thus ?thesis
proof cases
  case (1 c)
  hence  $f - z + w \rightarrow c$ 
  by (smt (verit, ccfv_SIG) LIM_cong add.assoc filterlim_at_to_0)
  thus ?thesis
  by (auto simp: not_essential_def)
next
  case 2
  hence is_pole f  $(z + w)$ 
  by (subst is_pole_shift_iff [symmetric]) (auto simp: o_def add_ac)
  thus ?thesis
  by (auto simp: not_essential_def)
qed
qed

```

```

lemma not_essential_shift_iff: not_essential f  $(z + w) \longleftrightarrow \text{not\_essential } (\lambda x.$ 
 $f (x + w)) \text{ } z$ 
  using not_essential_shift[of f w z]

```

$\text{not_essential_shift}[of \ \lambda x. f \ (x + w) - w \ w + z]$
by (auto simp: algebra_simps)

lemma *not_essential_shift_0*:
 $\text{NO_MATCH } 0 \ z \implies \text{not_essential } f \ z \longleftrightarrow \text{not_essential } (\lambda x. f \ (z + x)) \ 0$
using *not_essential_shift_iff*[of $f \ 0 \ z$] **by** (simp add: add_ac)

lemma *not_essential_holomorphic*:
assumes $f \text{ holomorphic_on } A \ x \in A \ \text{open } A$
shows $\text{not_essential } f \ x$
by (metis assms at_within_open continuous_on holomorphic_on_imp_continuous_on not_essential_def)

lemma *not_essential_analytic*:
assumes $f \text{ analytic_on } \{z\}$
shows $\text{not_essential } f \ z$
using *analytic_at* assms *not_essential_holomorphic* **by** blast

lemma *not_essential_id* [*singularity_intros*]: $\text{not_essential } (\lambda w. w) \ z$
by (simp add: not_essential_analytic)

lemma *is_pole_imp_not_essential* [*intro*]: $\text{is_pole } f \ z \implies \text{not_essential } f \ z$
by (auto simp: not_essential_def)

lemma *tendsto_imp_not_essential* [*intro*]: $f \ -z \rightarrow c \implies \text{not_essential } f \ z$
by (auto simp: not_essential_def)

lemma *eventually_not_pole*:
assumes $\text{isolated_singularity_at } f \ z$
shows $\text{eventually } (\lambda w. \neg \text{is_pole } f \ w) \ (\text{at } z)$
proof –
from assms **obtain** $r \ \text{where } r > 0 \ \text{and } r: f \text{ analytic_on ball } z \ r - \{z\}$
by (auto simp: isolated_singularity_at_def)
then have $\text{eventually } (\lambda w. w \in \text{ball } z \ r - \{z\}) \ (\text{at } z)$
by (intro eventually_at_in_open) auto
thus eventually $(\lambda w. \neg \text{is_pole } f \ w) \ (\text{at } z)$
by (metis (no_types, lifting) analytic_at analytic_on_analytic_at eventually_mono not_is_pole_holomorphic r)
qed

lemma *not_islimpt_poles*:
assumes $\text{isolated_singularity_at } f \ z$
shows $\neg z \text{ islimpt } \{w. \text{is_pole } f \ w\}$
using *eventually_not_pole* [*OF* assms]
by (auto simp: islimpt_conv_frequently_at frequently_def)

lemma *analytic_at_imp_no_pole*: $f \text{ analytic_on } \{z\} \implies \neg \text{is_pole } f \ z$
using *analytic_at* *not_is_pole_holomorphic* **by** blast

lemma *not_essential_const* [*singularity_intros*]: *not_essential* ($\lambda_.$ *c*) *z*
 by *blast*

lemma *not_essential_uminus* [*singularity_intros*]:

assumes *f_ness*: *not_essential* *f z*

assumes *f_iso*: *isolated_singularity_at* *f z*

shows *not_essential* ($\lambda w.$ $-f\ w$) *z*

proof –

have *not_essential* ($\lambda w.$ $-1 * f\ w$) *z*

by (*intro* *assms* *singularity_intros*)

thus ?*thesis* by *simp*

qed

lemma *isolated_singularity_at_analytic*:

assumes *f* *analytic_on* {*z*}

shows *isolated_singularity_at* *f z*

by (*meson* *Diff_subset* *analytic_at* *assms* *holomorphic_on_subset* *isolated_singularity_at_holomorphic*)

lemma *isolated_singularity_sum* [*singularity_intros*]:

assumes $\bigwedge x. x \in A \implies \text{isolated_singularity_at } (f\ x)\ z$

shows *isolated_singularity_at* ($\lambda w.$ $\sum_{x \in A} f\ x\ w$) *z*

using *assms* by (*induction* *A* *rule*: *infinite_finite_induct*) (*auto* *intro!*: *singularity_intros*)

lemma *isolated_singularity_prod* [*singularity_intros*]:

assumes $\bigwedge x. x \in A \implies \text{isolated_singularity_at } (f\ x)\ z$

shows *isolated_singularity_at* ($\lambda w.$ $\prod_{x \in A} f\ x\ w$) *z*

using *assms* by (*induction* *A* *rule*: *infinite_finite_induct*) (*auto* *intro!*: *singularity_intros*)

lemma *isolated_singularity_sum_list* [*singularity_intros*]:

assumes $\bigwedge f. f \in \text{set } fs \implies \text{isolated_singularity_at } f\ z$

shows *isolated_singularity_at* ($\lambda w.$ $\sum f \leftarrow fs. f\ w$) *z*

using *assms* by (*induction* *fs*) (*auto* *intro!*: *singularity_intros*)

lemma *isolated_singularity_prod_list* [*singularity_intros*]:

assumes $\bigwedge f. f \in \text{set } fs \implies \text{isolated_singularity_at } f\ z$

shows *isolated_singularity_at* ($\lambda w.$ $\prod f \leftarrow fs. f\ w$) *z*

using *assms* by (*induction* *fs*) (*auto* *intro!*: *singularity_intros*)

lemma *isolated_singularity_sum_mset* [*singularity_intros*]:

assumes $\bigwedge f. f \in \# F \implies \text{isolated_singularity_at } f\ z$

shows *isolated_singularity_at* ($\lambda w.$ $\sum f \in \# F. f\ w$) *z*

using *assms* by (*induction* *F*) (*auto* *intro!*: *singularity_intros*)

lemma *isolated_singularity_prod_mset* [*singularity_intros*]:

assumes $\bigwedge f. f \in \# F \implies \text{isolated_singularity_at } f\ z$

shows *isolated_singularity_at* ($\lambda w.$ $\prod f \in \# F. f\ w$) *z*

using *assms* by (*induction* *F*) (*auto* *intro!*: *singularity_intros*)

```

lemma analytic_nhd_imp_isolated_singularity:
  assumes f analytic_on A - {x} x ∈ A open A
  shows isolated_singularity_at f x
  unfolding isolated_singularity_at_def using assms
  using analytic_imp_holomorphic isolated_singularity_at_def isolated_singularity_at_holomorphic
  by blast

```

```

lemma isolated_singularity_at_iff_analytic_nhd:
  isolated_singularity_at f x  $\longleftrightarrow$  ( $\exists A. x \in A \wedge \text{open } A \wedge f \text{ analytic\_on } A - \{x\}$ )
  by (meson open_ball analytic_nhd_imp_isolated_singularity
    centre_in_ball isolated_singularity_at_def)

```

7.12 The order of non-essential singularities (i.e. removable singularities or poles)

```

definition zorder :: (complex  $\Rightarrow$  complex)  $\Rightarrow$  complex  $\Rightarrow$  int where
  zorder f z = (THE n. ( $\exists h r. r > 0 \wedge h \text{ holomorphic\_on cball } z r \wedge h z \neq 0$ 
     $\wedge (\forall w \in \text{cball } z r - \{z\}. f w = h w * (w - z)^{\text{powi } n}$ 
     $\wedge h w \neq 0)$ ))

```

```

definition zor_poly
  :: [complex  $\Rightarrow$  complex, complex]  $\Rightarrow$  complex  $\Rightarrow$  complex where
  zor_poly f z = (SOME h.  $\exists r. r > 0 \wedge h \text{ holomorphic\_on cball } z r \wedge h z \neq 0$ 
     $\wedge (\forall w \in \text{cball } z r - \{z\}. f w = h w * (w - z)^{\text{powi } n}$ 
     $\wedge h w \neq 0)$ )

```

```

lemma zorder_exist:
  fixes f:: complex  $\Rightarrow$  complex and z::complex
  defines n  $\equiv$  zorder f z and g  $\equiv$  zor_poly f z
  assumes f_iso: isolated_singularity_at f z
    and f_ness: not_essential f z
    and f_nconst:  $\exists_F w \text{ in } (\text{at } z). f w \neq 0$ 
  shows g z  $\neq 0 \wedge (\exists r. r > 0 \wedge g \text{ holomorphic\_on cball } z r$ 
     $\wedge (\forall w \in \text{cball } z r - \{z\}. f w = g w * (w - z)^{\text{powi } n} \wedge g w \neq 0)$ )
  proof -
    define P where P = ( $\lambda n g r. 0 < r \wedge g \text{ holomorphic\_on cball } z r \wedge g z \neq 0$ 
       $\wedge (\forall w \in \text{cball } z r - \{z\}. f w = g w * (w - z)^{\text{powi } n} \wedge g w \neq 0)$ )
    have  $\exists! k. \exists g r. P k g r$ 
    using holomorphic_factor_puncture[OF assms(3-)] unfolding P_def by auto
    then have  $\exists g r. P n g r$ 
    unfolding n_def P_def zorder_def by (rule theI')
    then have  $\exists r. P n g r$ 
    unfolding P_def zor_poly_def g_def n_def by (rule someI_ex)
    then obtain r1 where P n g r1
    by auto
    then show ?thesis
    unfolding P_def by auto
  qed

```

```

lemma zorder_shift:
  shows zorder f z = zorder ( $\lambda u. f (u + z)$ ) 0
  unfolding zorder_def
  apply (rule arg_cong [of concl: The])
  apply (auto simp: fun_eq_iff Ball_def dist_norm)
  subgoal for x h r
    apply (rule_tac x=h o (+)z in exI)
    apply (rule_tac x=r in exI)
    apply (intro conjI holomorphic_on_compose holomorphic_intros)
      apply (simp_all flip: cball_translation)
    apply (simp add: add commute)
  done
  subgoal for x h r
    apply (rule_tac x=h o ( $\lambda u. u - z$ ) in exI)
    apply (rule_tac x=r in exI)
    apply (intro conjI holomorphic_on_compose holomorphic_intros)
      apply (simp_all flip: cball_translation_subtract)
    by (metis diff_add_cancel eq_iff_diff_eq_0 norm_minus_commute)
  done

lemma zorder_shift': NO_MATCH 0 z  $\implies$  zorder f z = zorder ( $\lambda u. f (u + z)$ ) 0
  by (rule zorder_shift)

lemma
  fixes f:: complex  $\Rightarrow$  complex and z::complex
  assumes f_iso: isolated_singularity_at f z
    and f_ness: not_essential f z
    and f_nconst:  $\exists_F w$  in (at z). f w  $\neq$  0
  shows zorder_inverse: zorder ( $\lambda w. \text{inverse } (f w)$ ) z = - zorder f z
    and zor_poly_inverse:  $\forall_F w$  in (at z). zor_poly ( $\lambda w. \text{inverse } (f w)$ ) z w
      = inverse (zor_poly f z w)

proof -
  define vf where vf = ( $\lambda w. \text{inverse } (f w)$ )
  define fn vfn where
    fn = zorder f z and vfn = zorder vf z
  define fp vfp where
    fp = zor_poly f z and vfp = zor_poly vf z

  obtain fr where [simp]: fp z  $\neq$  0 and fr > 0
    and fr: fp holomorphic_on cball z fr
     $\forall w \in \text{cball } z \text{ fr} - \{z\}. f w = fp w * (w - z)^{\text{powi } fn} \wedge fp w \neq 0$ 
  using zorder_exist[OF f_iso f_ness f_nconst, folded fn_def fp_def]
  by auto
  have fr_inverse: vf w = (inverse (fp w)) * (w - z)^{\text{powi } (-fn)}
    and fr_nz: inverse (fp w)  $\neq$  0
    when w  $\in$  ball z fr - {z} for w
  proof -
    have f w = fp w * (w - z)^{\text{powi } fn} fp w  $\neq$  0

```

```

    using fr(2) that by auto
  then show  $vf\ w = (inverse\ (fp\ w)) * (w - z)\ powi\ (-fn)\ inverse\ (fp\ w) \neq 0$ 
    by (simp_all add: power_int_minus vf_def)
qed
obtain vfr where [simp]:  $vfp\ z \neq 0$  and  $vfr > 0$  and vfr:  $vfp\ holomorphic\_on\ cball\ z\ vfr$ 
  ( $\forall w \in cball\ z\ vfr - \{z\}. vf\ w = vfp\ w * (w - z)\ powi\ vfn \wedge vfp\ w \neq 0$ )
proof -
  have isolated_singularity_at vf z
    using isolated_singularity_at_inverse[OF f_iso f_ness] unfolding vf_def .
  moreover have not_essential vf z
    using not_essential_inverse[OF f_ness f_iso] unfolding vf_def .
  moreover have  $\exists_F w\ in\ at\ z. vf\ w \neq 0$ 
    using f_nconst unfolding vf_def by (auto elim: frequently_elim1)
  ultimately show ?thesis using zorder_exist[of vf z, folded vfn_def vfp_def]
that by auto
qed

define r1 where  $r1 = \min\ fr\ vfr$ 
have  $r1 > 0$  using  $\langle fr > 0 \rangle\ \langle vfr > 0 \rangle$  unfolding r1_def by simp
show  $vfn = -fn$ 
proof (rule holomorphic_factor_unique)
  have  $\S: \bigwedge w. \llbracket fp\ w = 0; dist\ z\ w < fr \rrbracket \implies False$ 
    using fr_nz by force
  then show  $\forall w \in ball\ z\ r1 - \{z\}. vf\ w = vfp\ w * (w - z)\ powi\ vfn \wedge vfp\ w \neq 0 \wedge vf\ w = inverse\ (fp\ w) * (w - z)\ powi\ (-fn) \wedge inverse\ (fp\ w) \neq 0$ 
    using fr_inverse r1_def vfr(2)
    by (smt (verit) Diff_iff inverse_nonzero_iff_nonzero mem_ball mem_cball)
  show  $vfp\ holomorphic\_on\ ball\ z\ r1$ 
    using r1_def vfr(1) by auto
  show  $(\lambda w. inverse\ (fp\ w))\ holomorphic\_on\ ball\ z\ r1$ 
    by (metis  $\S\ ball\_subset\_cball\ fr(1)\ holomorphic\_on\_inverse\ holomorphic\_on\_subset\ mem\_ball\ min.cobounded2\ min.commute\ r1\_def\ subset\_ball$ )
qed (use  $\langle r1 > 0 \rangle$  in auto)
have  $vfp\ w = inverse\ (fp\ w)$  when  $w \in ball\ z\ r1 - \{z\}$  for w
proof -
  have  $w \in ball\ z\ fr - \{z\} \wedge w \in cball\ z\ vfr - \{z\} \wedge w \neq z$ 
    using that unfolding r1_def by auto
  then show ?thesis
    by (metis  $\langle vfn = -fn \rangle\ power\_int\_not\_zero\ right\_minus\_eq\ fr\_inverse\ vfr(2)\ vector\_space\_over\_itself.scale\_right\_imp\_eq$ )
qed
then show  $\forall_F w\ in\ (at\ z). vfp\ w = inverse\ (fp\ w)$ 
  unfolding eventually_at by (metis DiffI dist_commute mem_ball singletonD  $\langle r1 > 0 \rangle$ )
qed

```



```

lemma zor_poly_shift:
  assumes iso1: isolated_singularity_at f z
    and ness1: not_essential f z
    and nzero1:  $\exists_F w \text{ in at } z. f w \neq 0$ 
  shows  $\forall_F w \text{ in nhds } z. \text{zor\_poly } f z w = \text{zor\_poly } (\lambda u. f (z + u)) 0 (w - z)$ 
proof -
  obtain r1 where r1>0 zor_poly f z z  $\neq 0$  and
    holo1: zor_poly f z holomorphic_on cball z r1 and
    rball1:  $\forall w \in \text{cball } z r1 - \{z\}. f w = \text{zor\_poly } f z w * (w - z) \text{ powi } (\text{zorder } f z) \wedge \text{zor\_poly } f z w \neq 0$ 
  using zorder_exist[OF iso1 ness1 nzero1] by blast

  define ff where ff =  $(\lambda u. f (z + u))$ 
  have isolated_singularity_at ff 0
    unfolding ff_def
    using iso1 isolated_singularity_at_shift_iff[of f 0 z]
    by (simp add: algebra_simps)
  moreover have not_essential ff 0
    unfolding ff_def
    using ness1 not_essential_shift_iff[of f 0 z]
    by (simp add: algebra_simps)
  moreover have  $\exists_F w \text{ in at } 0. ff w \neq 0$ 
    unfolding ff_def using nzero1
  by (smt (verit, ccfv_SIG) add.commute eventually_at_to_0 eventually_mono not_frequently)
  ultimately
  obtain r2 where r2>0 zor_poly ff 0 0  $\neq 0$ 
    and holo2: zor_poly ff 0 holomorphic_on cball 0 r2
    and rball2:  $\forall w \in \text{cball } 0 r2 - \{0\}. ff w = \text{zor\_poly } ff 0 w * w \text{ powi } (\text{zorder } ff 0) \wedge \text{zor\_poly } ff 0 w \neq 0$ 
  using zorder_exist[of ff 0] by auto

  define r where r = min r1 r2
  have r>0 using <r1>0 <r2>0 unfolding r_def by auto

  have zor_poly f z w = zor_poly ff 0 (w - z)
    if  $w \in \text{ball } z r - \{z\}$  for w
  proof -
    define n where n  $\equiv \text{zorder } f z$ 

    have f w = zor_poly f z w * (w - z) powi n
      using n_def r_def rball1 that by auto
    moreover have f w = zor_poly ff 0 (w - z) * (w - z) powi n
  proof -
    have  $w - z \in \text{cball } 0 r2 - \{0\}$ 
      using r_def that by (auto simp: dist_complex_def)
    then have ff (w - z) = zor_poly ff 0 (w - z) * (w - z) powi (zorder ff 0)

```

```

    using rball2 by blast
  moreover have of_int (zorder ff 0) = n
    unfolding n_def ff_def by (simp add: zorder_shift' add.commute)
  ultimately show ?thesis unfolding ff_def by auto
qed
ultimately have zor_poly f z w * (w-z) powi n = zor_poly ff 0 (w-z) *
(w-z) powi n
  by auto
moreover have (w-z) powi n ≠ 0
  using that by auto
ultimately show ?thesis
  using mult_cancel_right by blast
qed
then have  $\forall_F w \text{ in } \text{at } z. \text{zor\_poly } f \ z \ w = \text{zor\_poly } ff \ 0 \ (w-z)$ 
  unfolding eventually_at
  by (metis DiffI <0 < r> dist_commute mem_ball singletonD)
moreover have isCont (zor_poly f z) z
  using holo1[THEN holomorphic_on_imp_continuous_on]
  by (simp add: <0 < r1> continuous_on_interior)
moreover
have isCont (zor_poly ff 0) 0
  using <0 < r2> continuous_on_interior holo2 holomorphic_on_imp_continuous_on
  by fastforce
then have isCont ( $\lambda w. \text{zor\_poly } ff \ 0 \ (w-z)$ ) z
  unfolding isCont_iff by simp
ultimately show  $\forall_F w \text{ in } \text{nhds } z. \text{zor\_poly } f \ z \ w = \text{zor\_poly } ff \ 0 \ (w-z)$ 
  by (elim at_within_isCont_imp_nhds; auto)
qed

lemma
  fixes f g:: complex  $\Rightarrow$  complex and z::complex
  assumes f_iso: isolated_singularity_at f z and g_iso: isolated_singularity_at g
z
    and f_ess: not_essential f z and g_ess: not_essential g z
    and fg_nconst:  $\exists_F w \text{ in } (\text{at } z). f \ w * g \ w \neq 0$ 
  shows zor_order_times:  $\text{zor\_order } (\lambda w. f \ w * g \ w) \ z = \text{zor\_order } f \ z + \text{zor\_order } g \ z$  and
    zor_poly_times:  $\forall_F w \text{ in } (\text{at } z). \text{zor\_poly } (\lambda w. f \ w * g \ w) \ z \ w$ 
    =  $\text{zor\_poly } f \ z \ w * \text{zor\_poly } g \ z \ w$ 

proof -
  define fg where fg = ( $\lambda w. f \ w * g \ w$ )
  define fn gn fgn where
    fn = zor_order f z and gn = zor_order g z and fgn = zor_order fg z
  define fp gp fgp where
    fp = zor_poly f z and gp = zor_poly g z and fgp = zor_poly fg z
  have f_nconst:  $\exists_F w \text{ in } (\text{at } z). f \ w \neq 0$  and g_nconst:  $\exists_F w \text{ in } (\text{at } z). g \ w \neq 0$ 
    using fg_nconst by (auto elim!: frequently_elim1)
  obtain fr where [simp]: fp z ≠ 0 and fr > 0
    and fr: fp holomorphic_on cball z fr
     $\forall w \in \text{cball } z \text{ fr} - \{z\}. f \ w = fp \ w * (w-z) \text{ powi } fn \wedge fp \ w \neq 0$ 

```

```

    using zorder_exist[OF f_iso f_ness f_nconst,folded fp_def fn_def] by auto
  obtain gr where [simp]: gp z  $\neq$  0 and gr > 0
    and gr: gp holomorphic_on cball z gr
       $\forall w \in \text{cball } z \text{ gr} - \{z\}. \text{ gp } w = \text{gp } w * (w - z)^{\text{powi } \text{gn}} \wedge \text{gp } w \neq 0$ 
    using zorder_exist[OF g_iso g_ness g_nconst,folded gn_def gp_def] by auto
  define r1 where r1 = min fr gr
  have r1 > 0 unfolding r1_def using <fr>0> <gr>0> by auto
  have fg_times: fg w = (fp w * gp w) * (w - z)^{\text{powi } (fn + gn)} and fgp_nz: fp w * gp
w  $\neq$  0
    when w  $\in$  ball z r1 - {z} for w
  proof -
    have f w = fp w * (w - z)^{\text{powi } fn} fp w  $\neq$  0
      using fr(2) r1_def that by auto
    moreover have g w = gp w * (w - z)^{\text{powi } gn} gp w  $\neq$  0
      using gr(2) that unfolding r1_def by auto
    ultimately show fg w = (fp w * gp w) * (w - z)^{\text{powi } (fn + gn)} fp w * gp w  $\neq$  0
      using that unfolding fg_def by (auto simp add: power_int_add)
  qed

  obtain fgr where [simp]: fgp z  $\neq$  0 and fgr > 0
    and fgr: fgp holomorphic_on cball z fgr
       $\forall w \in \text{cball } z \text{ fgr} - \{z\}. \text{ fg } w = \text{fgp } w * (w - z)^{\text{powi } \text{fgn}} \wedge \text{fgp } w \neq 0$ 
  proof -
    have isolated_singularity_at fg z
      unfolding fg_def using isolated_singularity_at_times[OF f_iso g_iso] .
    moreover have not_essential fg z
      by (simp add: f_iso f_ness fg_def g_iso g_ness not_essential_times)
    moreover have  $\exists_F w \text{ in at } z. \text{ fg } w \neq 0$ 
      using fg_def fg_nconst by blast
    ultimately show ?thesis
      using that zorder_exist[of fg z] fgn_def fgp_def by fastforce
  qed
  define r2 where r2 = min fgr r1
  have r2 > 0 using <r1>0> <fgr>0> unfolding r2_def by simp
  show fgn = fn + gn
  proof (rule holomorphic_factor_unique)
    show  $\forall w \in \text{ball } z \text{ r2} - \{z\}. \text{ fg } w = \text{fgp } w * (w - z)^{\text{powi } \text{fgn}} \wedge \text{fgp } w \neq 0 \wedge$ 
 $\text{fg } w = \text{fp } w * \text{gp } w * (w - z)^{\text{powi } (fn + gn)} \wedge \text{fp } w * \text{gp } w \neq 0$ 
      using fg_times fgp_nz fgr(2) r2_def by fastforce
    next
      show fgp holomorphic_on ball z r2
        using fgr(1) r2_def by auto
    next
      show  $(\lambda w. \text{fp } w * \text{gp } w) \text{ holomorphic\_on ball } z \text{ r2}$ 
        by (metis ball_subset_cball fr(1) gr(1) holomorphic_on_mult holomor-
phic_on_subset
          min.cobounded1 min.cobounded2 r1_def r2_def subset_ball)
  qed (auto simp add: <0 < r2>)

```

```

have fgp w = fp w * gp w when w: w ∈ ball z r2 - {z} for w
proof -
  have w ∈ ball z r1 - {z} w ∈ cball z fgr - {z} w ≠ z
    using w unfolding r2_def by auto
  then show ?thesis
  by (metis ⟨fgn = fn + gn⟩ eq_iff_diff_eq_0 fg_times fgr(2) power_int_eq_0_iff
    mult_right_cancel)
qed
then show ∀F w in (at z). fgp w = fp w * gp w
  using ⟨r2 > 0⟩ unfolding eventually_at by (auto simp add: dist_commute)
qed

```

lemma

```

fixes f g:: complex ⇒ complex and z::complex
assumes f_iso: isolated_singularity_at f z and g_iso: isolated_singularity_at g
z
  and f_ess: not_essential f z and g_ess: not_essential g z
  and fg_nconst: ∃F w in (at z). f w * g w ≠ 0
shows zorder_divide: zorder (λw. f w / g w) z = zorder f z - zorder g z and
  zor_poly_divide: ∀F w in (at z). zor_poly (λw. f w / g w) z w
    = zor_poly f z w / zor_poly g z w

```

proof -

```

have f_nconst: ∃F w in (at z). f w ≠ 0 and g_nconst: ∃F w in (at z). g w ≠ 0
  using fg_nconst by (auto elim!: frequently_elim1)
define vg where vg = (λw. inverse (g w))
have 1: isolated_singularity_at vg z
  by (simp add: g_iso g_ess isolated_singularity_at_inverse vg_def)
moreover have 2: not_essential vg z
  by (simp add: g_iso g_ess not_essential_inverse vg_def)
moreover have 3: ∃F w in at z. f w * vg w ≠ 0
  using fg_nconst vg_def by auto
ultimately have zorder (λw. f w * vg w) z = zorder f z + zorder vg z
  using zorder_times[OF f_iso _ f_ess] by blast
then show zorder (λw. f w / g w) z = zorder f z - zorder g z
  using zorder_inverse[OF g_iso g_ess g_nconst, folded vg_def] unfolding
vg_def
  by (auto simp add: field_simps)
have ∀F w in at z. zor_poly (λw. f w * vg w) z w = zor_poly f z w * zor_poly
vg z w
  using zor_poly_times[OF f_iso _ f_ess, of vg] 1 2 3 by blast
then show ∀F w in (at z). zor_poly (λw. f w / g w) z w = zor_poly f z w /
zor_poly g z w
  using zor_poly_inverse[OF g_iso g_ess g_nconst, folded vg_def] unfolding
vg_def
  by eventually_elim (auto simp add: field_simps)
qed

```

lemma zorder_exist_zero:

```

fixes f:: complex ⇒ complex and z::complex

```

```

defines  $n \equiv \text{zorder } f \, z$  and  $g \equiv \text{zor\_poly } f \, z$ 
assumes  $\text{holo}: f \text{ holomorphic\_on } S$  and  $\text{open } S$  connected  $S \, z \in S$ 
and  $\text{non\_const}: \exists w \in S. f \, w \neq 0$ 
shows  $(\text{if } f \, z = 0 \text{ then } n > 0 \text{ else } n = 0) \wedge (\exists r. r > 0 \wedge \text{cball } z \, r \subseteq S \wedge g \text{ holomorphic\_on } \text{cball } z \, r$ 
 $\wedge (\forall w \in \text{cball } z \, r. f \, w = g \, w * (w - z)^{\wedge \text{nat } n} \wedge g \, w \neq 0))$ 
proof -
obtain  $r$  where  $g \, z \neq 0$  and  $r: r > 0 \text{ cball } z \, r \subseteq S \wedge g \text{ holomorphic\_on } \text{cball } z \, r$ 
 $(\forall w \in \text{cball } z \, r - \{z\}. f \, w = g \, w * (w - z)^{\text{powi } n} \wedge g \, w \neq 0)$ 
proof -
have  $g \, z \neq 0 \wedge (\exists r > 0. g \text{ holomorphic\_on } \text{cball } z \, r$ 
 $\wedge (\forall w \in \text{cball } z \, r - \{z\}. f \, w = g \, w * (w - z)^{\text{powi } n} \wedge g \, w \neq 0))$ 
proof (rule  $\text{zorder\_exist}[of \, f \, z, \text{folded } g\_def \, n\_def]$ )
show  $\text{isolated\_singularity\_at } f \, z$ 
using  $\langle \text{open } S \rangle \langle z \in S \rangle \text{holo holomorphic\_on\_imp\_analytic\_at isolated\_singularity\_at\_analytic}$ 

by force
show  $\text{not\_essential } f \, z$  unfolding  $\text{not\_essential\_def}$ 
using  $\langle \text{open } S \rangle \langle z \in S \rangle \text{at\_within\_open continuous\_on holo holomorphic\_on\_imp\_continuous\_on}$ 
by fastforce
have  $\forall_F w \text{ in at } z. f \, w \neq 0 \wedge w \in S$ 
using  $\text{assms}(4,5,6) \text{holo non\_const non\_zero\_neighbour\_alt}$  by blast
then show  $\exists_F w \text{ in at } z. f \, w \neq 0$ 
by (auto elim: eventually_frequentlyE)
qed
then obtain  $r1$  where  $g \, z \neq 0 \wedge r1 > 0$  and  $r1: g \text{ holomorphic\_on } \text{cball } z \, r1$ 
 $(\forall w \in \text{cball } z \, r1 - \{z\}. f \, w = g \, w * (w - z)^{\text{powi } n} \wedge g \, w \neq 0)$ 
by auto
obtain  $r2$  where  $r2: r2 > 0 \text{ cball } z \, r2 \subseteq S$ 
using  $\text{assms}(4,6) \text{open\_contains\_cball\_eq}$  by blast
define  $r3$  where  $r3 \equiv \min \, r1 \, r2$ 
have  $r3 > 0 \text{ cball } z \, r3 \subseteq S$  using  $\langle r1 > 0 \rangle \, r2$  unfolding  $r3\_def$  by auto
moreover have  $g \text{ holomorphic\_on } \text{cball } z \, r3$ 
using  $r1(1) \text{unfolding } r3\_def$  by auto
moreover have  $(\forall w \in \text{cball } z \, r3 - \{z\}. f \, w = g \, w * (w - z)^{\text{powi } n} \wedge g \, w \neq 0)$ 
using  $r1(2) \text{unfolding } r3\_def$  by auto
ultimately show  $?thesis$  using  $\text{that}[of \, r3] \langle g \, z \neq 0 \rangle$  by auto
qed

have  $fz\_lim: f - z \rightarrow f \, z$ 
by (metis  $\text{assms}(4,6) \text{at\_within\_open continuous\_on holo holomorphic\_on\_imp\_continuous\_on}$ )
have  $gz\_lim: g - z \rightarrow g \, z$ 
using  $r$ 
by (meson  $\text{Elementary\_Metric\_Spaces.open\_ball analytic\_at analytic\_at\_imp\_isCont}$ 

 $\text{ball\_subset\_cball centre\_in\_ball holomorphic\_on\_subset isContD}$ )
have  $\text{if\_0}: \text{if } f \, z = 0 \text{ then } n > 0 \text{ else } n = 0$ 
proof -

```

```

have ( $\lambda w. g \ w * (w-z) \text{ powi } n) -z \rightarrow f \ z$ 
  using  $\text{fz\_lim\_Lim\_transform\_within\_open}[\text{where } s=\text{ball } z \ r] \ r$  by fastforce
then have ( $\lambda w. (g \ w * (w-z) \text{ powi } n) / g \ w) -z \rightarrow f \ z/g \ z$ 
  using  $\text{gz\_lim } \langle g \ z \neq 0 \rangle \text{ tendsto\_divide}$  by blast
then have  $\text{powi\_tendsto}: (\lambda w. (w-z) \text{ powi } n) -z \rightarrow f \ z/g \ z$ 
  using  $\text{Lim\_transform\_within\_open}[\text{where } s=\text{ball } z \ r] \ r$  by fastforce

have ?thesis when  $n \geq 0$   $f \ z = 0$ 
proof -
  have ( $\lambda w. (w-z) \wedge \text{nat } n) -z \rightarrow f \ z/g \ z$ 
    using  $\text{Lim\_transform\_within}[OF \ \text{powi\_tendsto}, \text{where } d=r]$ 
    by ( $\text{meson } \text{power\_int\_def } r(1) \text{ that}(1)$ )
  then have *: ( $\lambda w. (w-z) \wedge \text{nat } n) -z \rightarrow 0$  using  $\langle f \ z = 0 \rangle$  by simp
  moreover have False when  $n = 0$ 
  proof -
    have ( $\lambda w. (w-z) \wedge \text{nat } n) -z \rightarrow 1$ 
      using  $\langle n = 0 \rangle$  by auto
    then show False using * using  $\text{LIM\_unique } \text{zero\_neq\_one}$  by blast
  qed
  ultimately show ?thesis using that by fastforce
qed
moreover have ?thesis when  $n \geq 0$   $f \ z \neq 0$ 
proof -
  have False when  $n > 0$ 
  proof -
    have ( $\lambda w. (w-z) \wedge \text{nat } n) -z \rightarrow f \ z/g \ z$ 
      using  $\text{Lim\_transform\_within}[OF \ \text{powi\_tendsto}, \text{where } d=r]$ 
      by ( $\text{meson } \langle 0 \leq n \rangle \text{ power\_int\_def } r(1)$ )
    moreover have ( $\lambda w. (w-z) \wedge \text{nat } n) -z \rightarrow 0$ 
      using  $\langle n > 0 \rangle$  by ( $\text{auto intro!: tendsto\_eq\_intros}$ )
    ultimately show False
      using  $\langle f \ z \neq 0 \rangle \langle g \ z \neq 0 \rangle \text{ LIM\_unique } \text{divide\_eq\_0\_iff}$  by blast
  qed
  then show ?thesis using that by force
qed
moreover have False when  $n < 0$ 
proof -
  have ( $\lambda w. \text{inverse } ((w-z) \wedge \text{nat } (-n))) -z \rightarrow f \ z/g \ z$ 
    by ( $\text{smt } (\text{verit}) \text{ LIM\_cong } \text{power\_int\_def } \text{power\_inverse } \text{powi\_tendsto}$  that)
  moreover
  have ( $\lambda w. ((w-z) \wedge \text{nat } (-n))) -z \rightarrow 0$ 
    using that by ( $\text{auto intro!: tendsto\_eq\_intros}$ )
  ultimately
  have ( $\lambda x. \text{inverse } ((x-z) \wedge \text{nat } (-n)) * (x-z) \wedge \text{nat } (-n)) -z \rightarrow 0$ 
    using  $\text{tendsto\_mult}$  by fastforce
  then have ( $\lambda x. 1::\text{complex}$ )  $-z \rightarrow 0$ 
    using  $\text{Lim\_transform\_within\_open}$  by fastforce
  then show False
    using  $\text{LIM\_const\_eq}$  by fastforce

```

```

qed
ultimately show ?thesis by fastforce
qed
moreover have  $f w = g w * (w - z)^{\wedge \text{nat } n} \wedge g w \neq 0$  when  $w \in \text{cball } z \ r$  for  $w$ 
proof (cases  $w = z$ )
case True
then have  $f - z \rightarrow f w$ 
using fz_lim by blast
then have  $(\lambda w. g w * (w - z)^{\wedge \text{nat } n}) - z \rightarrow f w$ 
proof (elim Lim_transform_within[OF  $\langle r > 0 \rangle$ ])
fix  $x$  assume  $0 < \text{dist } x \ z \ \text{dist } x \ z < r$ 
then have  $x \in \text{cball } z \ r - \{z\} \ x \neq z$ 
unfolding cball_def by (auto simp add: dist_commute)
then have  $f x = g x * (x - z)^{\text{powi } n}$ 
using r(4)[rule_format, of  $x$ ] by simp
also have  $\dots = g x * (x - z)^{\wedge \text{nat } n}$ 
by (smt (verit, best) if_0_int_nat_eq_power_int_of_nat)
finally show  $f x = g x * (x - z)^{\wedge \text{nat } n}$  .
qed
moreover have  $(\lambda w. g w * (w - z)^{\wedge \text{nat } n}) - z \rightarrow g w * (w - z)^{\wedge \text{nat } n}$ 
using True by (auto intro!: tendsto_eq_intros gz_lim)
ultimately have  $f w = g w * (w - z)^{\wedge \text{nat } n}$  using LIM_unique by blast
then show ?thesis using  $\langle g z \neq 0 \rangle$  True by auto
next
case False
then have  $f w = g w * (w - z)^{\text{powi } n} \ g w \neq 0$ 
using r(4) that by auto
then show ?thesis
by (smt (verit, best) False if_0_int_nat_eq_power_int_of_nat)
qed
ultimately show ?thesis using r by auto
qed

lemma zorder_exist_pole:
fixes  $f :: \text{complex} \Rightarrow \text{complex}$  and  $z :: \text{complex}$ 
defines  $n \equiv \text{zorder } f \ z$  and  $g \equiv \text{zor\_poly } f \ z$ 
assumes holo:  $f$  holomorphic_on  $S - \{z\}$  and open  $S \ z \in S$  and is_pole  $f \ z$ 
shows  $n < 0 \wedge g z \neq 0 \wedge (\exists r. r > 0 \wedge \text{cball } z \ r \subseteq S \wedge g \text{ holomorphic\_on } \text{cball } z \ r$ 
 $\wedge (\forall w \in \text{cball } z \ r - \{z\}. f w = g w / (w - z)^{\wedge \text{nat } (-n)} \wedge g w \neq 0)$ )
proof -
obtain  $r$  where  $g z \neq 0$  and  $r: r > 0 \ \text{cball } z \ r \subseteq S \ g \text{ holomorphic\_on } \text{cball } z \ r$ 
 $(\forall w \in \text{cball } z \ r - \{z\}. f w = g w * (w - z)^{\text{powi } n} \wedge g w \neq 0)$ 
proof -
have  $g z \neq 0 \wedge (\exists r > 0. g \text{ holomorphic\_on } \text{cball } z \ r$ 
 $\wedge (\forall w \in \text{cball } z \ r - \{z\}. f w = g w * (w - z)^{\text{powi } n} \wedge g w \neq 0))$ 
proof (rule zorder_exist[of  $f \ z$ , folded g_def n_def])
show isolated_singularity_at  $f \ z$  unfolding isolated_singularity_at_def
using holo assms(4,5)
by (metis analytic_on_holomorphic centre_in_ball insert_Diff openE)

```

```

open_delete subset_insert_iff)
  show not_essential f z unfolding not_essential_def
  using assms(4,6) at_within_open continuous_on_holo holomorphic_on_imp_continuous_on
  by fastforce
  from non_zero_neighbour_pole[OF ‹is_pole f z›] show  $\exists_F w$  in at z.  $f w \neq$ 
0
  by (auto elim: eventually_frequentlyE)
qed
then obtain r1 where  $g z \neq 0$   $r1 > 0$  and  $r1$ :  $g$  holomorphic_on cball z r1
  ( $\forall w \in \text{cball } z \text{ r1} - \{z\}. f w = g w * (w - z)^{\text{powi } n} \wedge g w \neq 0$ )
  by auto
obtain r2 where  $r2 > 0$  cball z r2  $\subseteq S$ 
  using assms(4,5) open_contains_cball_eq by metis
define r3 where  $r3 = \min r1 r2$ 
have  $r3 > 0$  cball z r3  $\subseteq S$  using ‹ $r1 > 0$ › r2 unfolding r3_def by auto
moreover have  $g$  holomorphic_on cball z r3
  using r1(1) unfolding r3_def by auto
moreover have ( $\forall w \in \text{cball } z \text{ r3} - \{z\}. f w = g w * (w - z)^{\text{powi } n} \wedge g w \neq 0$ )
  using r1(2) unfolding r3_def by auto
ultimately show ?thesis
  using that[of r3] ‹ $g z \neq 0$ › by auto
qed

have  $n < 0$ 
proof (rule ccontr)
  assume  $\neg n < 0$ 
  define c where  $c = (\text{if } n = 0 \text{ then } g z \text{ else } 0)$ 
  have [simp]:  $g - z \rightarrow g z$ 
  using r
  by (metis centre_in_ball continuous_on_interior holomorphic_on_imp_continuous_on
    interior_cball isContD)
  have  $\forall x \in \text{ball } z r. x \neq z \rightarrow f x = g x * (x - z)^{\wedge \text{nat } n}$ 
  by (simp add: ‹ $\neg n < 0$ › linorder_not_le power_int_def r)
  then have  $\forall_F x$  in at z.  $f x = g x * (x - z)^{\wedge \text{nat } n}$ 
  using centre_in_ball eventually_at_topological r(1) by blast
  moreover have ( $\lambda x. g x * (x - z)^{\wedge \text{nat } n} - z \rightarrow c$ )
  proof (cases  $n = 0$ )
    case True
    then show ?thesis unfolding c_def by simp
  next
    case False
    then have ( $\lambda x. (x - z)^{\wedge \text{nat } n} - z \rightarrow 0$ ) using ‹ $\neg n < 0$ ›
    by (auto intro!: tendsto_eq_intros)
    from tendsto_mult[OF _ this, of g g z, simplified]
    show ?thesis unfolding c_def using False by simp
  qed
  ultimately have  $f - z \rightarrow c$  using tendsto_cong by fast
then show False using ‹is_pole f z› at_neq_bot not_tendsto_and_filterlim_at_infinity
  unfolding is_pole_def by blast

```



```

qed
moreover have  $\forall w \in \text{cball } z \ r - \{z\}. f \ w = g \ w / (w - z) \wedge^{\text{nat}} (-n) \wedge g \ w \neq 0$ 
  using  $r(4) \langle n < 0 \rangle$ 
  by (smt (verit) inverse_eq_divide mult.right_neutral power_int_def power_inverse
times_divide_eq_right)
ultimately show ?thesis
  using  $r \langle g \ z \neq 0 \rangle$  by auto
qed

```

lemma *zorder_eqI*:

```

assumes open  $S \ z \in S \ g \text{ holomorphic\_on } S \ g \ z \neq 0$ 
assumes fg_eq:  $\bigwedge w. \llbracket w \in S; w \neq z \rrbracket \implies f \ w = g \ w * (w - z) \text{ powi } n$ 
shows  $\text{zorder } f \ z = n$ 

```

proof –

```

have continuous_on  $S \ g$  by (rule holomorphic_on_imp_continuous_on) fact
moreover have open  $(-\{0::\text{complex}\})$  by auto
ultimately have open  $((g - '(-\{0\})) \cap S)$ 
  unfolding continuous_on_open_vimage[OF  $\langle \text{open } S \rangle$ ] by blast
moreover from assms have  $z \in (g - '(-\{0\})) \cap S$  by auto
ultimately obtain  $r$  where  $r: r > 0 \ \text{cball } z \ r \subseteq S \cap (g - '(-\{0\}))$ 
  unfolding open_contains_cball by blast

```

```

let ?gg =  $(\lambda w. g \ w * (w - z) \text{ powi } n)$ 
define  $P$  where  $P = (\lambda n \ g \ r. 0 < r \wedge g \text{ holomorphic\_on } \text{cball } z \ r \wedge g \ z \neq 0$ 
   $\wedge (\forall w \in \text{cball } z \ r - \{z\}. f \ w = g \ w * (w - z) \text{ powi } n \wedge g \ w \neq 0))$ 
have  $P \ n \ g \ r$ 
  unfolding  $P\_def$  using  $r$  assms(3,4,5) by auto
then have  $\exists g \ r. P \ n \ g \ r$  by auto
moreover have unique:  $\exists! n. \exists g \ r. P \ n \ g \ r$  unfolding  $P\_def$ 
proof (rule holomorphic_factor_puncture)
  have  $\text{ball } z \ r - \{z\} \subseteq S$  using  $r$  using ball_subset_cball by blast
  then have ?gg holomorphic_on  $\text{ball } z \ r - \{z\}$ 
    using  $\langle g \text{ holomorphic\_on } S \rangle \ r$  by (auto intro!: holomorphic_intros)
  then have  $f \text{ holomorphic\_on } \text{ball } z \ r - \{z\}$ 
    by (smt (verit, best) DiffD2  $\langle \text{ball } z \ r - \{z\} \subseteq S \rangle \ fg\_eq \text{ holomorphic\_cong}$ 
singleton_iff subset_iff)
  then show isolated_singularity_at  $f \ z$  unfolding isolated_singularity_at_def
    using analytic_on_open open_delete  $r(1)$  by blast
next
have not_essential ?gg  $z$ 
proof (intro singularity_intros)
  show not_essential  $g \ z$ 
    by (meson  $\langle \text{continuous\_on } S \ g \rangle$  assms continuous_on_eq_continuous_at
isCont_def not_essential_def)
  show  $\forall_F w \text{ in at } z. w - z \neq 0$  by (simp add: eventually_at_filter)
  then show  $\text{LIM } w \text{ at } z. w - z :> \text{at } 0$ 
    unfolding filterlim_at by (auto intro: tendsto_eq_intros)
  show isolated_singularity_at  $g \ z$ 
    by (meson Diff_subset open_ball analytic_on_holomorphic

```

```

      assms holomorphic_on_subset isolated_singularity_at_def openE)
qed
moreover
have  $\forall_F w \text{ in at } z. g w * (w-z)^{\text{powi } n} = f w$ 
  unfolding eventually_at_topological using assms fg_eq by force
ultimately show not_essential f z
  using not_essential_transform by blast
show  $\exists_F w \text{ in at } z. f w \neq 0$  unfolding frequently_at
proof (intro strip)
  fix d::real assume  $0 < d$ 
  define z' where  $z' \equiv z + \min d r / 2$ 
  have  $z' \neq z \wedge \text{dist } z' z < d$ 
    unfolding z'_def using  $\langle d > 0 \rangle \langle r > 0 \rangle$  by (auto simp add: dist_norm)
  moreover have  $f z' \neq 0$ 
  proof (subst fg_eq[OF _  $\langle z' \neq z \rangle$ ])
    have  $z' \in \text{cball } z r$ 
      unfolding z'_def using  $\langle r > 0 \rangle \langle d > 0 \rangle$  by (auto simp add: dist_norm)
    then show  $z' \in S$  using  $r(2)$  by blast
    show  $g z' * (z' - z)^{\text{powi } n} \neq 0$ 
      using P_def  $\langle P n g r \rangle \langle z' \in \text{cball } z r \rangle \langle z' \neq z \rangle$  by auto
  qed
qed
ultimately show  $\exists x \in \text{UNIV}. x \neq z \wedge \text{dist } x z < d \wedge f x \neq 0$  by auto
qed
qed
ultimately have  $(THE n. \exists g r. P n g r) = n$ 
  by (rule_tac the1_equality)
then show ?thesis unfolding zorder_def P_def by blast
qed

lemma simple_zeroI:
  assumes open S  $z \in S$   $g \text{ holomorphic\_on } S$   $g z \neq 0$ 
  assumes  $\bigwedge w. w \in S \implies f w = g w * (w-z)$ 
  shows  $\text{zorder } f z = 1$ 
  using assms zorder_eqI by force

lemma higher_deriv_power:
  shows  $(\text{deriv } \sim j) (\lambda w. (w-z)^\wedge n) w =$ 
    pochhammer (of_nat (Suc n - j)) j *  $(w-z)^\wedge (n-j)$ 
proof (induction j arbitrary: w)
  case 0
  thus ?case by auto
next
  case (Suc j w)
  have  $(\text{deriv } \sim \text{Suc } j) (\lambda w. (w-z)^\wedge n) w = \text{deriv } ((\text{deriv } \sim j) (\lambda w. (w-z)^\wedge$ 
 $n)) w$ 
  by simp
  also have  $(\text{deriv } \sim j) (\lambda w. (w-z)^\wedge n) =$ 
     $(\lambda w. \text{pochhammer (of\_nat (Suc } n - j)) j * (w-z)^\wedge (n-j))$ 
  using Suc by (intro Suc.IH ext)

```

```

also {
  have (... has_field_derivative of_nat (n - j) *
    pochhammer (of_nat (Suc n - j)) j * (w-z) ^ (n - Suc j)) (at w)
    using Suc.premis by (auto intro!: derivative_eq_intros)
  also have of_nat (n - j) * pochhammer (of_nat (Suc n - j)) j =
    pochhammer (of_nat (Suc n - Suc j)) (Suc j)
    by (cases Suc j ≤ n, subst pochhammer_rec)
    (use Suc.premis in ⟨simp_all add: algebra_simps Suc_diff_le pochhammer_0_left⟩)
  finally have deriv (λw. pochhammer (of_nat (Suc n - j)) j * (w-z) ^ (n - j)) w =
    ... * (w-z) ^ (n - Suc j)
    by (rule DERIV_imp_deriv)
}
finally show ?case .
qed

```

lemma *zorder_zero_eqI*:

```

assumes f_holo: f holomorphic_on S and open S z ∈ S
assumes zero: ⋀ i. i < nat n ⇒ (deriv ^^ i) f z = 0
assumes nz: (deriv ^^ nat n) f z ≠ 0 and n ≥ 0
shows zorder f z = n
proof -
  obtain r where [simp]: r > 0 and ball z r ⊆ S
    using ⟨open S⟩ ⟨z ∈ S⟩ openE by blast
  have nz': ∃ w ∈ ball z r. f w ≠ 0
  proof (rule ccontr)
    assume ¬ (∃ w ∈ ball z r. f w ≠ 0)
    then have eventually (λu. f u = 0) (nhds z)
      using open_ball ⟨0 < r⟩ centre_in_ball eventually_nhds by blast
    then have (deriv ^^ nat n) f z = (deriv ^^ nat n) (λ_. 0) z
      by (intro higher_deriv_cong_ev) auto
    also have (deriv ^^ nat n) (λ_. 0) z = 0
      by (induction n) simp_all
    finally show False using nz by contradiction
  qed
qed

```

```

define zn g where zn = zorder f z and g = zor_poly f z
obtain e where e_if: if f z = 0 then 0 < zn else zn = 0 and
  [simp]: e > 0 and cball z e ⊆ ball z r and
  g_holo: g holomorphic_on cball z e and
  e_fac: (∀ w ∈ cball z e. f w = g w * (w-z) ^ nat zn ∧ g w ≠ 0)
proof -
  have f_holomorphic_on ball z r
    using f_holo ⟨ball z r ⊆ S⟩ by auto
  from that zorder_exist_zero[of f ball z r z, simplified, OF this nz', folded zn_def g_def]
  show thesis by blast
qed

```

```

then obtain zn ≥ 0 g z ≠ 0
  by (metis centre_in_cball less_le_not_le order_refl)

define A where A ≡ (λi. of_nat (i choose (nat zn)) * fact (nat zn) * (deriv ~
(i - nat zn)) g z)
have deriv_A: (deriv ~ i) f z = (if zn ≤ int i then A i else 0) for i
proof -
  have eventually (λw. w ∈ ball z e) (nhds z)
    using ⟨cball z e ⊆ ball z r⟩ ⟨e>0⟩ by (intro eventually_nhds_in_open) auto
  hence eventually (λw. f w = (w-z) ^ (nat zn) * g w) (nhds z)
    using e_fac eventually_mono by fastforce
  hence (deriv ~ i) f z = (deriv ~ i) (λw. (w-z) ^ nat zn * g w) z
    by (intro higher_deriv_cong_ev) auto
  also have ... = (∑ j=0..i. of_nat (i choose j) *
    (deriv ~ j) (λw. (w-z) ^ nat zn) z * (deriv ~ (i - j)) g z)
    using g_holo ⟨e>0⟩
    by (intro higher_deriv_mult[of _ ball z e]) (auto intro!: holomorphic_intros)
  also have ... = (∑ j=0..i. if j = nat zn then
    of_nat (i choose nat zn) * fact (nat zn) * (deriv ~ (i - nat zn))
g z else 0)
  proof (intro sum.cong_refl, goal_cases)
    case (1 j)
    have (deriv ~ j) (λw. (w-z) ^ nat zn) z =
      pochhammer (of_nat (Suc (nat zn) - j)) j * 0 ^ (nat zn - j)
    by (subst higher_deriv_power) auto
    also have ... = (if j = nat zn then fact j else 0)
    by (auto simp: not_less pochhammer_0_left pochhammer_fact)
    also have of_nat (i choose j) * ... * (deriv ~ (i - j)) g z =
      (if j = nat zn then of_nat (i choose (nat zn)) * fact (nat zn)
        * (deriv ~ (i - nat zn)) g z else 0)
    by simp
    finally show ?case .
  qed
  also have ... = (if i ≥ zn then A i else 0)
  by (auto simp: A_def)
  finally show (deriv ~ i) f z = ... .
qed

have False when n < zn
  using deriv_A[of nat n] that ⟨n ≥ 0⟩ by (simp add: nz)
moreover have n ≤ zn
proof -
  have g z ≠ 0
    by (simp add: ⟨g z ≠ 0⟩)
  then have (deriv ~ nat zn) f z ≠ 0
    using deriv_A[of nat zn] by (auto simp add: A_def)
  then have nat zn ≥ nat n using zero[of nat zn] by linarith
  moreover have zn ≥ 0 using e_if by (auto split: if_splits)
  ultimately show ?thesis using nat_le_eq_zle by blast

```

```

qed
ultimately show ?thesis unfolding zn_def by fastforce
qed

lemma
  assumes eventually ( $\lambda z. f z = g z$ ) (at z)  $z = z'$ 
  shows zorder_cong:  $zorder f z = zorder g z'$  and zor_poly_cong:  $zor\_poly f z = zor\_poly g z'$ 
proof -
  define P where  $P = (\lambda ff\ n\ h\ r. 0 < r \wedge h\ holomorphic\_on\ cball\ z\ r \wedge h\ z \neq 0 \wedge (\forall w \in cball\ z\ r - \{z\}. ff\ w = h\ w * (w - z)^{powi\ n} \wedge h\ w \neq 0))$ 
  have  $(\exists r. P\ f\ n\ h\ r) = (\exists r. P\ g\ n\ h\ r)$  for  $n\ h$ 
  proof -
    have *:  $\exists r. P\ g\ n\ h\ r$  if  $\exists r. P\ f\ n\ h\ r$  and eventually ( $\lambda x. f x = g x$ ) (at z)
  for  $f\ g$ 
  proof -
    from that(1) obtain r1 where  $r1\_P: P\ f\ n\ h\ r1$  by auto
    from that(2) obtain r2 where  $r2 > 0$  and  $r2\_dist: \forall x. x \neq z \wedge dist\ x\ z \leq r2 \longrightarrow f x = g x$ 
    unfolding eventually_at_le by auto
    define r where  $r = \min\ r1\ r2$ 
    have  $r > 0\ h\ z \neq 0$  using  $r1\_P\ \langle r2 > 0 \rangle$  unfolding r_def P_def by auto
    moreover have  $h\ holomorphic\_on\ cball\ z\ r$ 
    using  $r1\_P$  unfolding P_def r_def by auto
    moreover have  $g\ w = h\ w * (w - z)^{powi\ n} \wedge h\ w \neq 0$  when  $w \in cball\ z\ r - \{z\}$  for  $w$ 
    proof -
      have  $f\ w = h\ w * (w - z)^{powi\ n} \wedge h\ w \neq 0$ 
      using  $r1\_P$  that unfolding P_def r_def by auto
      moreover have  $f\ w = g\ w$ 
      using  $r2\_dist$  that by (simp add: dist_commute r_def)
      ultimately show ?thesis by simp
    qed
    qed
    ultimately show ?thesis unfolding P_def by auto
  qed
  from assms have eq': eventually ( $\lambda z. g z = f z$ ) (at z)
  by (simp add: eq_commute)
  show ?thesis
  using * assms(1) eq' by blast
qed
then show  $zorder f z = zorder g z'\ zor\_poly f z = zor\_poly g z'$ 
  using  $\langle z = z' \rangle$  unfolding P_def zorder_def zor_poly_def by auto
qed

```

```

lemma zorder_times_analytic':
  assumes isolated_singularity_at  $f\ z$  not_essential  $f\ z$ 
  assumes  $g$  analytic_on  $\{z\}$  frequently ( $\lambda z. f z * g z \neq 0$ ) (at z)
  shows  $zorder (\lambda x. f x * g x)\ z = zorder f z + zorder g z$ 
  using assms isolated_singularity_at_analytic not_essential_analytic zorder_times

```

by blast

lemma zorder_cmult:

assumes $c \neq 0$

shows $\text{zorder } (\lambda z. c * f z) z = \text{zorder } f z$

proof -

define P where

$P = (\lambda f n h r. 0 < r \wedge h \text{ holomorphic_on } \text{cball } z r \wedge$
 $h z \neq 0 \wedge (\forall w \in \text{cball } z r - \{z\}. f w = h w * (w - z)^{\text{powi } n \wedge h w$
 $\neq 0}))$

have *: $P (\lambda x. c * f x) n (\lambda x. c * h x) r$

if $P f n h r c \neq 0$ for $f n h r c$

using that unfolding P_def by (auto intro!: holomorphic_intros)

have $(\exists h r. P (\lambda x. c * f x) n h r) \longleftrightarrow (\exists h r. P f n h r)$ for n

using *[of $f n _ c$] *[of $\lambda x. c * f x n _ \text{inverse } c$] $\langle c \neq 0 \rangle$

by (fastforce simp: field_simps)

hence $(THE n. \exists h r. P (\lambda x. c * f x) n h r) = (THE n. \exists h r. P f n h r)$

by simp

thus ?thesis

by (simp add: zorder_def P_def)

qed

lemma zorder_uminus [simp]: $\text{zorder } (\lambda z. -f z) z = \text{zorder } f z$

using zorder_cmult[of $-1 f$] by simp

lemma zorder_nonzero_div_power:

assumes sz : $\text{open } S z \in S f \text{ holomorphic_on } S f z \neq 0$ and $n > 0$

shows $\text{zorder } (\lambda w. f w / (w - z)^{\wedge n}) z = -n$

by (intro zorder_eqI [OF sz]) (simp add: inverse_eq_divide power_int_minus)

lemma zor_poly_eq:

assumes isolated_singularity_at $f z$ not_essential $f z \exists_F w \text{ in at } z. f w \neq 0$

shows eventually $(\lambda w. \text{zor_poly } f z w = f w * (w - z)^{\text{powi } - \text{zorder } f z})$ (at z)

proof -

obtain r where $r: r > 0$

$(\forall w \in \text{cball } z r - \{z\}. f w = \text{zor_poly } f z w * (w - z)^{\text{powi } (\text{zorder } f z)})$

using zorder_exist[OF $assms$] by blast

then have *: $\forall w \in \text{ball } z r - \{z\}. \text{zor_poly } f z w = f w * (w - z)^{\text{powi } - \text{zorder } f z}$

z

by (auto simp: field_simps power_int_minus)

have eventually $(\lambda w. w \in \text{ball } z r - \{z\})$ (at z)

using r eventually_at_ball'[of $r z UNIV$] by auto

thus ?thesis by eventually_elim (insert *, auto)

qed

lemma zor_poly_zero_eq:

assumes $f \text{ holomorphic_on } S \text{ open } S \text{ connected } S z \in S \exists w \in S. f w \neq 0$

shows eventually $(\lambda w. \text{zor_poly } f z w = f w / (w - z)^{\wedge \text{nat } (\text{zorder } f z)})$ (at z)

proof -

```

obtain  $r$  where  $r: r > 0$ 
   $(\forall w \in \text{cball } z \ r - \{z\}. f \ w = \text{zor\_poly } f \ z \ w * (w - z) ^{\text{nat } (\text{zorder } f \ z)})$ 
  using  $\text{zorder\_exist\_zero}[OF \ \text{assms}]$  by  $\text{auto}$ 
then have  $*$ :  $\forall w \in \text{ball } z \ r - \{z\}. \text{zor\_poly } f \ z \ w = f \ w / (w - z) ^{\text{nat } (\text{zorder } f \ z)}$ 
by  $(\text{auto simp: field\_simps powr\_minus})$ 
have  $\text{eventually } (\lambda w. w \in \text{ball } z \ r - \{z\}) \ (\text{at } z)$ 
  using  $r \ \text{eventually\_at\_ball}'[\text{of } r \ z \ \text{UNIV}]$  by  $\text{auto}$ 
thus  $?thesis$  by  $\text{eventually\_elim } (\text{insert } *, \text{auto})$ 
qed

```

```

lemma  $\text{zor\_poly\_pole\_eq}$ :
  assumes  $f\_iso$ :  $\text{isolated\_singularity\_at } f \ z \ \text{is\_pole } f \ z$ 
  shows  $\text{eventually } (\lambda w. \text{zor\_poly } f \ z \ w = f \ w * (w - z) ^{\text{nat } (- \text{zorder } f \ z)}) \ (\text{at } z)$ 
proof -
  obtain  $e$  where  $[simp]$ :  $e > 0$  and  $f\_holo$ :  $f \ \text{holomorphic\_on } \text{ball } z \ e - \{z\}$ 
  using  $f\_iso \ \text{analytic\_imp\_holomorphic unfolding isolated\_singularity\_at\_def}$ 
by  $\text{blast}$ 
  obtain  $r$  where  $r: r > 0$ 
     $(\forall w \in \text{cball } z \ r - \{z\}. f \ w = \text{zor\_poly } f \ z \ w / (w - z) ^{\text{nat } (- \text{zorder } f \ z)})$ 
    using  $\text{zorder\_exist\_pole}[OF \ f\_holo, \text{simplified}, OF \ \langle \text{is\_pole } f \ z \rangle]$  by  $\text{auto}$ 
  then have  $*$ :  $\forall w \in \text{ball } z \ r - \{z\}. \text{zor\_poly } f \ z \ w = f \ w * (w - z) ^{\text{nat } (- \text{zorder } f \ z)}$ 
  by  $(\text{auto simp: field\_simps})$ 
  have  $\text{eventually } (\lambda w. w \in \text{ball } z \ r - \{z\}) \ (\text{at } z)$ 
    using  $r \ \text{eventually\_at\_ball}'[\text{of } r \ z \ \text{UNIV}]$  by  $\text{auto}$ 
  thus  $?thesis$  by  $\text{eventually\_elim } (\text{insert } *, \text{auto})$ 
qed

```

```

lemma  $\text{zor\_poly\_eqI}$ :
  fixes  $f :: \text{complex} \Rightarrow \text{complex}$  and  $z0 :: \text{complex}$ 
  defines  $n \equiv \text{zorder } f \ z0$ 
  assumes  $\text{isolated\_singularity\_at } f \ z0 \ \text{not\_essential } f \ z0 \ \exists_F \ w \ \text{in } \text{at } z0. f \ w \neq 0$ 
  assumes  $\text{lim}$ :  $((\lambda x. f \ (g \ x) * (g \ x - z0) ^{\text{powi } - n}) \longrightarrow c) \ F$ 
  assumes  $g$ :  $\text{filterlim } g \ (\text{at } z0) \ F$  and  $F \neq \text{bot}$ 
  shows  $\text{zor\_poly } f \ z0 \ z0 = c$ 
proof -
  from  $\text{zorder\_exist}[OF \ \text{assms}(2-4)]$  obtain  $r$  where
     $r: r > 0 \ \text{zor\_poly } f \ z0 \ \text{holomorphic\_on } \text{cball } z0 \ r$ 
     $\bigwedge w. w \in \text{cball } z0 \ r - \{z0\} \implies f \ w = \text{zor\_poly } f \ z0 \ w * (w - z0) ^{\text{powi } n}$ 
  unfolding  $n\_def$  by  $\text{blast}$ 
  from  $r(1)$  have  $\text{eventually } (\lambda w. w \in \text{ball } z0 \ r \wedge w \neq z0) \ (\text{at } z0)$ 
    using  $\text{eventually\_at\_ball}'[\text{of } r \ z0 \ \text{UNIV}]$  by  $\text{auto}$ 
  hence  $\text{eventually } (\lambda w. \text{zor\_poly } f \ z0 \ w = f \ w * (w - z0) ^{\text{powi } - n}) \ (\text{at } z0)$ 
    by  $\text{eventually\_elim } (\text{insert } r, \text{auto simp: field\_simps power\_int\_minus})$ 
  moreover have  $\text{continuous\_on } (\text{ball } z0 \ r) \ (\text{zor\_poly } f \ z0)$ 
    using  $r$  by  $(\text{intro holomorphic\_on\_imp\_continuous\_on}) \ \text{auto}$ 
  with  $r$  have  $\text{isCont } (\text{zor\_poly } f \ z0) \ z0$ 
    by  $(\text{auto simp: continuous\_on\_eq\_continuous\_at})$ 

```

hence $(\text{zor_poly } f \text{ } z0 \longrightarrow \text{zor_poly } f \text{ } z0 \text{ } z0) \text{ (at } z0)$
 unfolding *isCont_def* .
 ultimately have $((\lambda w. f \text{ } w * (w - z0) \text{ powi } - n) \longrightarrow \text{zor_poly } f \text{ } z0 \text{ } z0) \text{ (at } z0)$
 by (*blast intro: Lim_transform_eventually*)
 hence $((\lambda x. f \text{ } (g \text{ } x) * (g \text{ } x - z0) \text{ powi } - n) \longrightarrow \text{zor_poly } f \text{ } z0 \text{ } z0) \text{ } F$
 by (*rule filterlim_compose[OF _ g]*)
 from *tendsto_unique[OF <F ≠ bot> this lim]* show ?thesis .
 qed

lemma *zor_poly_zero_eqI*:

fixes $f :: \text{complex} \Rightarrow \text{complex}$ and $z0 :: \text{complex}$
 defines $n \equiv \text{zorder } f \text{ } z0$
 assumes $f \text{ holomorphic_on } A \text{ open } A \text{ connected } A \text{ } z0 \in A \exists z \in A. f \text{ } z \neq 0$
 assumes $\text{lim}: ((\lambda x. f \text{ } (g \text{ } x) / (g \text{ } x - z0) \wedge^{\text{nat } n}) \longrightarrow c) \text{ } F$
 assumes $g: \text{filterlim } g \text{ (at } z0) \text{ } F$ and $F \neq \text{bot}$
 shows $\text{zor_poly } f \text{ } z0 \text{ } z0 = c$
 proof –
 from *zorder_exist_zero[OF assms(2-6)]* obtain r where
 $r: r > 0 \text{ cball } z0 \text{ } r \subseteq A \text{ } \text{zor_poly } f \text{ } z0 \text{ holomorphic_on } \text{cball } z0 \text{ } r$
 $\bigwedge w. w \in \text{cball } z0 \text{ } r \implies f \text{ } w = \text{zor_poly } f \text{ } z0 \text{ } w * (w - z0) \wedge^{\text{nat } n}$
 unfolding *n_def* by *blast*
 from $r(1)$ have *eventually* $(\lambda w. w \in \text{ball } z0 \text{ } r \wedge w \neq z0) \text{ (at } z0)$
 using *eventually_at_ball'[of r z0 UNIV]* by *auto*
 hence *eventually* $(\lambda w. \text{zor_poly } f \text{ } z0 \text{ } w = f \text{ } w / (w - z0) \wedge^{\text{nat } n}) \text{ (at } z0)$
 by *eventually_elim* (*insert r, auto simp: field_simps*)
 moreover have *continuous_on* $(\text{ball } z0 \text{ } r) (\text{zor_poly } f \text{ } z0)$
 using r by (*intro holomorphic_on_imp_continuous_on*) *auto*
 with $r(1,2)$ have *isCont* $(\text{zor_poly } f \text{ } z0) \text{ } z0$
 by (*auto simp: continuous_on_eq_continuous_at*)
 hence $(\text{zor_poly } f \text{ } z0 \longrightarrow \text{zor_poly } f \text{ } z0 \text{ } z0) \text{ (at } z0)$
 unfolding *isCont_def* .
 ultimately have $((\lambda w. f \text{ } w / (w - z0) \wedge^{\text{nat } n}) \longrightarrow \text{zor_poly } f \text{ } z0 \text{ } z0) \text{ (at } z0)$
 by (*blast intro: Lim_transform_eventually*)
 hence $((\lambda x. f \text{ } (g \text{ } x) / (g \text{ } x - z0) \wedge^{\text{nat } n}) \longrightarrow \text{zor_poly } f \text{ } z0 \text{ } z0) \text{ } F$
 by (*rule filterlim_compose[OF _ g]*)
 from *tendsto_unique[OF <F ≠ bot> this lim]* show ?thesis .
 qed

lemma *zor_poly_pole_eqI*:

fixes $f :: \text{complex} \Rightarrow \text{complex}$ and $z0 :: \text{complex}$
 defines $n \equiv \text{zorder } f \text{ } z0$
 assumes $f \text{ iso: isolated_singularity_at } f \text{ } z0$ and $\text{is_pole } f \text{ } z0$
 assumes $\text{lim}: ((\lambda x. f \text{ } (g \text{ } x) * (g \text{ } x - z0) \wedge^{\text{nat } (-n)}) \longrightarrow c) \text{ } F$
 assumes $g: \text{filterlim } g \text{ (at } z0) \text{ } F$ and $F \neq \text{bot}$
 shows $\text{zor_poly } f \text{ } z0 \text{ } z0 = c$
 proof –
 obtain r where $r: r > 0 \text{ } \text{zor_poly } f \text{ } z0 \text{ holomorphic_on } \text{cball } z0 \text{ } r$
 proof –


```

have  $\exists_F w$  in at  $z0$ .  $f w \neq 0$ 
  using non_zero_neighbour_pole[OF  $\langle is\_pole f z0 \rangle$ ]
  by (auto elim: eventually_frequentlyE)
moreover have not_essential  $f z0$ 
  unfolding not_essential_def using  $\langle is\_pole f z0 \rangle$  by simp
ultimately show ?thesis
  using that zorder_exist[OF  $f\_iso, folded n\_def$ ] by auto
qed
from  $r(1)$  have eventually  $(\lambda w. w \in ball\ z0\ r \wedge w \neq z0)$  (at  $z0$ )
  using eventually_at_ball'[of  $r\ z0\ UNIV$ ] by auto
have eventually  $(\lambda w. zor\_poly\ f\ z0\ w = f w * (w - z0) ^ nat\ (-n))$  (at  $z0$ )
  using zor_poly_pole_eq[OF  $f\_iso\ \langle is\_pole f z0 \rangle$ ] unfolding  $n\_def$  .
moreover have continuous_on  $(ball\ z0\ r)$   $(zor\_poly\ f\ z0)$ 
  using  $r$  by (intro holomorphic_on_imp_continuous_on) auto
with  $r(1,2)$  have isCont  $(zor\_poly\ f\ z0)\ z0$ 
  by (auto simp: continuous_on_eq_continuous_at)
hence  $(zor\_poly\ f\ z0 \longrightarrow zor\_poly\ f\ z0\ z0)$  (at  $z0$ )
  unfolding isCont_def .
ultimately have  $((\lambda w. f w * (w - z0) ^ nat\ (-n)) \longrightarrow zor\_poly\ f\ z0\ z0)$  (at  $z0$ )
  by (blast intro: Lim_transform_eventually)
hence  $((\lambda x. f (g\ x) * (g\ x - z0) ^ nat\ (-n)) \longrightarrow zor\_poly\ f\ z0\ z0)\ F$ 
  by (rule filterlim_compose[OF  $g$ ])
from tendsto_unique[OF  $\langle F \neq bot \rangle$  this lim] show ?thesis .
qed

```

lemma

```

assumes is_pole  $f (x :: complex)$  open  $A\ x \in A$ 
assumes  $\bigwedge y. y \in A - \{x\} \implies (f\ has\_field\_derivative\ f'\ y)$  (at  $y$ )
shows is_pole_deriv': is_pole  $f'\ x$ 
  and zorder_deriv':  $zorder\ f'\ x = zorder\ f\ x - 1$ 
proof -
  have holo:  $f$  holomorphic_on  $A - \{x\}$ 
    using assms by (subst holomorphic_on_open) auto
  obtain  $r$  where  $r: r > 0\ ball\ x\ r \subseteq A$ 
    using assms(2,3) openE by blast
  moreover have open  $(ball\ x\ r - \{x\})$ 
    by auto
  ultimately have isolated_singularity_at  $f\ x$ 
    by (auto simp: isolated_singularity_at_def analytic_on_open
      intro!: exI[of  $r$ ] holomorphic_on_subset[OF holo])
  hence ev:  $\forall_F w$  in at  $x$ .  $zor\_poly\ f\ x\ w = f w * (w - x) ^ nat\ (-zorder\ f\ x)$ 
    using  $\langle is\_pole\ f\ x \rangle$  zor_poly_pole_eq by blast

```

define P where $P = zor_poly\ f\ x$

define n where $n = nat\ (-zorder\ f\ x)$

```

obtain  $r$  where  $r: r > 0\ cball\ x\ r \subseteq A\ P$  holomorphic_on  $cball\ x\ r$   $zorder\ f\ x$ 
 $< 0\ P\ x \neq 0$ 

```

```

   $\forall w \in \text{cball } x \ r - \{x\}. f \ w = P \ w / (w-x) ^ n \wedge P \ w \neq 0$ 
  using P_def assms holo n_def zorder_exist_pole by blast
have n:  $n > 0$ 
  using r(4) by (auto simp: n_def)

have [derivative_intros]: (P has_field_derivative deriv P w) (at w)
  if  $w \in \text{ball } x \ r$  for w
  using that by (intro holomorphic_derivI[OF holomorphic_on_subset[OF r(3),
of ball x r]]) auto

define D where  $D = (\lambda w. (\text{deriv } P \ w * (w-x) - \text{of\_nat } n * P \ w) / (w-x) ^ (n+1))$ 
define n' where  $n' = n - 1$ 
have n':  $n = \text{Suc } n'$ 
  using n by (simp add: n'_def)

have eventually ( $\lambda w. w \in \text{ball } x \ r$ ) (nhds x)
  using  $\langle r > 0 \rangle$  by (intro eventually_nhds_in_open) auto
hence ev'': eventually ( $\lambda w. w \in \text{ball } x \ r - \{x\}$ ) (at x)
  by (auto simp: eventually_at_filter elim: eventually_mono)

{
  fix w assume w:  $w \in \text{ball } x \ r - \{x\}$ 
  have ev': eventually ( $\lambda w. w \in \text{ball } x \ r - \{x\}$ ) (nhds w)
    using w by (intro eventually_nhds_in_open) auto

  have §:  $(\text{deriv } P \ w * (w-x) ^ n - P \ w * (n * (w-x) ^ (n-1))) / ((w-x) ^ n * (w-x) ^ n) = D \ w$ 
    using w n' by (simp add: divide_simps D_def) (simp add: algebra_simps)
  have  $((\lambda w. P \ w / (w-x) ^ n) \text{ has\_field\_derivative } D \ w)$  (at w)
    by (rule derivative_eq_intros refl | use w § in force)+
  also have  $?this \longleftrightarrow (f \text{ has\_field\_derivative } D \ w)$  (at w)
    using r by (intro has_field_derivative_cong_ev refl eventually_mono[OF ev'']) auto
  finally have  $(f \text{ has\_field\_derivative } D \ w)$  (at w) .
  moreover have  $(f \text{ has\_field\_derivative } f' \ w)$  (at w)
    using w r by (intro assms) auto
  ultimately have  $D \ w = f' \ w$ 
    using DERIV_unique by blast
} note D_eq = this

have is_pole D x
  unfolding D_def using n  $\langle r > 0 \rangle$   $\langle P \ x \neq 0 \rangle$ 
  by (intro is_pole_basic[where A = ball x r] holomorphic_intros holomorphic_on_subset[OF r(3)]) auto
also have  $?this \longleftrightarrow \text{is\_pole } f' \ x$ 
  by (intro is_pole_cong eventually_mono[OF ev'']) D_eq auto
finally show is_pole f' x .

```

```

have zorder f' x = -int (Suc n)
proof (rule zorder_eqI)
  show open (ball x r) x ∈ ball x r
  using ‹r > 0› by auto
  show f' w = (deriv P w * (w-x) - of_nat n * P w) * (w-x) powi (- int (Suc
n))
    if w ∈ ball x r w ≠ x for w
  using that D_eq[of w] n by (auto simp: D_def power_int_diff power_int_minus
powr_nat' divide_simps)
qed (use r n in ‹auto intro!: holomorphic_intros›)
thus zorder f' x = zorder f x - 1
  using n by (simp add: n_def)
qed

```

lemma

```

assumes is_pole f (x :: complex) isolated_singularity_at f x
shows is_pole_deriv: is_pole (deriv f) x
  and zorder_deriv: zorder (deriv f) x = zorder f x - 1
proof -
  from assms(2) obtain r where r: r > 0 f analytic_on ball x r - {x}
  by (auto simp: isolated_singularity_at_def)
  hence holo: f holomorphic_on ball x r - {x}
  by (subst (asm) analytic_on_open) auto
  have *: x ∈ ball x r open (ball x r) open (ball x r - {x})
  using ‹r > 0› by auto
  show is_pole (deriv f) x zorder (deriv f) x = zorder f x - 1
  by (meson * assms(1) holo holomorphic_derivI is_pole_deriv' zorder_deriv')+
qed

```

lemma removable_singularity_deriv':

```

assumes f -x→ c x ∈ A open (A :: complex set)
assumes ∧y. y ∈ A - {x} ⇒ (f has_field_derivative f' y) (at y)
shows ∃ c. f' -x→ c

```

proof -

```

have holo: f holomorphic_on A - {x}
  using assms by (subst holomorphic_on_open) auto

```

```

define g where g = (λy. if y = x then c else f y)

```

```

have deriv_g_eq: deriv g y = f' y if y ∈ A - {x} for y

```

proof -

```

have ev: eventually (λy. y ∈ A - {x}) (nhds y)
  using that assms by (intro eventually_nhds_in_open) auto
have (f has_field_derivative f' y) (at y)
  using assms that by auto

```

```

also have ?this ⟷ (g has_field_derivative f' y) (at y)

```

```

  by (intro has_field_derivative_cong_ev refl eventually_mono[OF ev]) (auto
simp: g_def)

```

```

finally show ?thesis

```

```

  by (intro DERIV_imp_deriv assms)

```

qed

```

have  $g$  holomorphic_on  $A$ 
  unfolding  $g\_def$  using  $assms$   $assms(1)$  holo
  by (intro removable_singularity) auto
hence  $deriv\ g$  holomorphic_on  $A$ 
  by (intro holomorphic_deriv  $assms$ )
hence continuous_on  $A$  ( $deriv\ g$ )
  by (meson holomorphic_on_imp_continuous_on)
hence ( $deriv\ g \longrightarrow deriv\ g\ x$ ) (at  $x$  within  $A$ )
  using  $assms$  by (auto simp: continuous_on_def)
also have  $?thesis \longleftrightarrow (f' \longrightarrow deriv\ g\ x)$  (at  $x$  within  $A$ )
  by (intro filterlim_cong refl) (auto simp: eventually_at_filter deriv_g_eq)
finally have  $f' -x\rightarrow deriv\ g\ x$ 
  using  $\langle open\ A \rangle \langle x \in A \rangle$  by (meson tendsto_within_open)
thus  $?thesis$ 
  by blast

```

qed

```

lemma removable_singularity_deriv:
  assumes  $f -x\rightarrow c$  isolated_singularity_at  $f\ x$ 
  shows  $\exists c. deriv\ f -x\rightarrow c$ 
proof -
  from  $assms(2)$  obtain  $r$  where  $r: r > 0$   $f$  analytic_on ball  $x\ r - \{x\}$ 
  by (auto simp: isolated_singularity_at_def)
  hence holo:  $f$  holomorphic_on ball  $x\ r - \{x\}$ 
  using analytic_imp_holomorphic by blast
  show  $?thesis$ 
  using  $assms(1)$ 
  proof (rule removable_singularity_deriv')
    show  $x \in ball\ x\ r$  open (ball  $x\ r$ )
    using  $\langle r > 0 \rangle$  by auto
  qed (auto intro!: holomorphic_derivI[OF holo])

```

qed

```

lemma not_essential_deriv':
  assumes  $not\_essential\ f\ x\ x \in A$  open  $A$ 
  assumes  $\bigwedge y. y \in A - \{x\} \implies (f\ has\_field\_derivative\ f'\ y)$  (at  $y$ )
  shows  $not\_essential\ f'\ x$ 
proof -
  have holo:  $f$  holomorphic_on  $A - \{x\}$ 
  using  $assms$  by (subst holomorphic_on_open) auto
  from  $assms$  consider  $is\_pole\ f\ x \mid c$  where  $f -x\rightarrow c$ 
  by (auto simp: not_essential_def)
  thus  $?thesis$ 
  proof cases
    case 1
    thus  $?thesis$ 
    using  $assms\ is\_pole\_deriv'$  by blast

```

```

next
  case (2 c)
  thus ?thesis
    by (meson assms removable_singularity_deriv' tendsto_imp_not_essential)
qed
qed

lemma not_essential_deriv[singularity_intros]:
  assumes not_essential f x isolated_singularity_at f x
  shows not_essential (deriv f) x
proof -
  from assms(2) obtain r where r:  $r > 0$   $f$  analytic_on ball x r - {x}
  by (auto simp: isolated_singularity_at_def)
  hence holo:  $f$  holomorphic_on ball x r - {x}
  by (subst (asm) analytic_on_open) auto
  show ?thesis
    using assms(1)
  proof (rule not_essential_deriv')
    show  $x \in \text{ball } x \ r \ \text{open } (\text{ball } x \ r)$ 
    using  $\langle r > 0 \rangle$  by auto
  qed (auto intro!: holomorphic_derivI[OF holo])
qed

lemma not_essential_frequently_0_imp_tendsto_0:
  fixes f :: complex  $\Rightarrow$  complex
  assumes sing: isolated_singularity_at f z not_essential f z
  assumes freq: frequently ( $\lambda z. f \ z = 0$ ) (at z)
  shows  $f \ -z \rightarrow 0$ 
proof -
  from freq obtain g :: nat  $\Rightarrow$  complex where g: filterlim g (at z) at_top  $\wedge$  n. f
  (g n) = 0
  using frequently_atE by blast
  have eventually ( $\lambda x. f \ (g \ x) = 0$ ) sequentially
  using g by auto
  hence fg: ( $\lambda x. f \ (g \ x)$ )  $\longrightarrow 0$ 
  by (simp add: tendsto_eventually)

  from assms(2) consider c where  $f \ -z \rightarrow c \mid \text{is\_pole } f \ z$ 
  unfolding not_essential_def by blast
  thus ?thesis
  proof cases
    case (1 c)
    have ( $\lambda x. f \ (g \ x)$ )  $\longrightarrow c$ 
    by (rule filterlim_compose[OF 1 g(1)])
    with fg have c = 0
    using LIMSEQ_unique by blast
    with 1 show ?thesis by simp
  next
    case 2

```

```

    have filterlim ( $\lambda x. f (g x)$ ) at_infinity sequentially
      using 2 filterlim_compose g(1) is_pole_def by blast
    with fg have False
      by (meson not_tendsto_and_filterlim_at_infinity sequentially_bot)
    thus ?thesis ..
  qed
qed

lemma not_essential_frequently_0_imp_eventually_0:
  fixes f :: complex  $\Rightarrow$  complex
  assumes sing: isolated_singularity_at f z not_essential f z
  assumes freq: frequently ( $\lambda z. f z = 0$ ) (at z)
  shows eventually ( $\lambda z. f z = 0$ ) (at z)
proof -
  from sing obtain r where r:  $r > 0$  and f_analytic_on_ball z r - {z}
    by (auto simp: isolated_singularity_at_def)
  hence holo: f_holomorphic_on_ball z r - {z}
    by (subst (asm) analytic_on_open) auto
  have eventually ( $\lambda w. w \in \text{ball } z \text{ } r - \{z\}$ ) (at z)
    using r by (intro eventually_at_in_open) auto
  from freq and this have frequently ( $\lambda w. f w = 0 \wedge w \in \text{ball } z \text{ } r - \{z\}$ ) (at z)
    using frequently_eventually_frequently by blast
  hence frequently ( $\lambda w. w \in \{w \in \text{ball } z \text{ } r - \{z\}. f w = 0\}$ ) (at z)
    by (simp add: conj_commute)
  hence limpt: z_islimpt { $w \in \text{ball } z \text{ } r - \{z\}. f w = 0$ }
    using islimpt_conv_frequently_at by blast

  define g where  $g = (\lambda w. \text{if } w = z \text{ then } 0 \text{ else } f w)$ 
  have f_z_tendsto_0
    by (intro not_essential_frequently_0_imp_tendsto_0 assms)
  hence g_holo: g_holomorphic_on_ball z r
    unfolding g_def by (intro removable_singularity holo) auto

  have g_eq_0:  $g w = 0$  if  $w \in \text{ball } z \text{ } r$  for w
  proof (rule analytic_continuation[where f = g])
    show open (ball z r) connected (ball z r)
      using r by auto
    show z_islimpt { $w \in \text{ball } z \text{ } r - \{z\}. f w = 0$ }
      by fact
    show  $g w = 0$  if  $w \in \{w \in \text{ball } z \text{ } r - \{z\}. f w = 0\}$  for w
      using that by (auto simp: g_def)
  qed (use r that g_holo in auto)

  have eventually ( $\lambda w. w \in \text{ball } z \text{ } r - \{z\}$ ) (at z)
    using r by (intro eventually_at_in_open) auto
  thus eventually ( $\lambda w. f w = 0$ ) (at z)
    by (metis freq non_zero_neighbour not_eventually_not_frequently sing)
qed

```

```

lemma pole_imp_not_constant:
  fixes f :: 'a :: {perfect_space}  $\Rightarrow$  _
  assumes is_pole f x open A x  $\in$  A A  $\subseteq$  insert x B
  shows  $\neg$ f constant_on B
proof
  assume *: f constant_on B
  then obtain c where c:  $\forall x \in B. f x = c$ 
    by (auto simp: constant_on_def)
  have eventually  $(\lambda y. y \in A - \{x\})$  (at x)
    using assms by (intro eventually_at_in_open) auto
  hence eventually  $(\lambda y. f y = c)$  (at x)
    by eventually_elim (use c assms in auto)
  hence **: f  $\rightarrow$  c
    by (simp add: tendsto_eventually)
  show False
    using **  $\langle$ is_pole f x $\rangle$  at_neq_bot is_pole_def
      not_tendsto_and_filterlim_at_infinity by blast
qed

lemma neg_zorder_imp_is_pole:
  assumes iso: isolated_singularity_at f z and f_ess: not_essential f z
    and zorder f z < 0 and fre_nz:  $\exists_F w$  in at z. f w  $\neq$  0
  shows is_pole f z
proof -
  define P where P = zor_poly f z
  define n where n = zorder f z
  have n<0 unfolding n_def by (simp add: assms(3))
  define nn where nn = nat (-n)

  obtain r where r: P z  $\neq$  0 r>0 and r_holo: P holomorphic_on cball z r and
    w_Pn:  $(\forall w \in \text{cball } z \text{ r} - \{z\}. f w = P w * (w-z)^{\text{powi } n} \wedge P w \neq 0)$ 
    using zorder_exist[OF iso f_ess fre_nz, folded P_def n_def] by auto

  have is_pole  $(\lambda w. P w * (w-z)^{\text{powi } n}) z$ 
    unfolding is_pole_def
  proof (rule tendsto_mult_filterlim_at_infinity)
    show P  $\rightarrow$  z  $\rightarrow$  P z
      by (metis  $\langle$ r>0 $\rangle$  r_holo centre_in_ball continuous_on_interior
        holomorphic_on_imp_continuous_on interior_cball isContD)
    show P z  $\neq$  0 by (simp add:  $\langle$ P z  $\neq$  0 $\rangle$ )

  have LIM x at z. inverse  $((x - z) ^ \text{nat } (-n))$  :> at_infinity
    apply (subst filterlim_inverse_at_iff[symmetric])
    using  $\langle$ n<0 $\rangle$ 
    by (auto intro!: tendsto_eq_intros filterlim_atI
      simp add: eventually_at_filter)
  then show LIM x at z.  $(x - z)^{\text{powi } n}$  :> at_infinity
    proof (elim filterlim_mono_eventually)

```

```

    have inverse  $((x - z) \wedge^{\text{nat}} (-n)) = (x - z) \text{ powi } n$ 
    if  $x \neq z$  for  $x$ 
    by (metis  $\langle n < 0 \rangle$  linorder_not_le power_int_def power_inverse)
    then show  $\forall_F x \text{ in at } z. \text{ inverse } ((x - z) \wedge^{\text{nat}} (-n))$ 
       $= (x - z) \text{ powi } n$ 
    by (simp add: eventually_at_filter)
  qed auto
qed
moreover have  $\forall_F w \text{ in at } z. f w = P w * (w - z) \text{ powi } n$ 
  unfolding eventually_at_le
  using  $w \text{ Pn } \langle r > 0 \rangle$  by (force simp add: dist_commute)
ultimately show ?thesis using is_pole_cong by fast
qed

lemma is_pole_divide_zorder:
  fixes  $f g :: \text{complex} \Rightarrow \text{complex}$  and  $z :: \text{complex}$ 
  assumes  $f\_iso: \text{isolated\_singularity\_at } f z$  and  $g\_iso: \text{isolated\_singularity\_at } g z$ 
  and  $f\_ness: \text{not\_essential } f z$  and  $g\_ness: \text{not\_essential } g z$ 
  and  $fg\_nconst: \exists_F w \text{ in } (\text{at } z). f w * g w \neq 0$ 
  and  $z\_less: \text{zorder } f z < \text{zorder } g z$ 
  shows  $\text{is\_pole } (\lambda z. f z / g z) z$ 
proof -
  define  $fn \ gn \ fg$  where  $fn = \text{zorder } f z$  and  $gn = \text{zorder } g z$ 
  and  $fg = (\lambda w. f w / g w)$ 

  have  $\text{isolated\_singularity\_at } fg z$ 
  unfolding  $fg\_def$  using  $f\_iso \ g\_iso \ g\_ness$ 
  by (auto intro: singularity_intros)
  moreover have  $\text{not\_essential } fg z$ 
  unfolding  $fg\_def$  using  $f\_iso \ g\_iso \ g\_ness \ f\_ness$ 
  by (auto intro: singularity_intros)
  moreover have  $\text{zorder } fg z < 0$ 
proof -
  have  $\text{zorder } fg z = fn - gn$ 
  using  $\text{zorder\_divide}[OF \ f\_iso \ g\_iso \ f\_ness \ g\_ness \ fg\_nconst]$ 
  by (simp add:  $fg\_def \ fn\_def \ gn\_def$ )
  then show ?thesis
  using  $z\_less$  by (simp add:  $fn\_def \ gn\_def$ )
qed
moreover have  $\exists_F w \text{ in at } z. fg w \neq 0$ 
  using  $fg\_nconst$  unfolding  $fg\_def$  by force
ultimately show  $\text{is\_pole } fg z$ 
  using  $\text{neg\_zorder\_imp\_is\_pole}$  by auto
qed

lemma isolated_pole_imp_nzero_times:
  assumes  $f\_iso: \text{isolated\_singularity\_at } f z$ 
  and  $\text{is\_pole } f z$ 

```



```

  shows  $\exists_F w \text{ in } (at\ z). \text{deriv } f\ w * f\ w \neq 0$ 
proof (rule ccontr)
  assume  $\neg (\exists_F w \text{ in } at\ z. \text{deriv } f\ w * f\ w \neq 0)$ 
  then have  $\forall_F x \text{ in } at\ z. \text{deriv } f\ x * f\ x = 0$ 
    unfolding not_frequently by simp
  moreover have  $\forall_F w \text{ in } at\ z. f\ w \neq 0$ 
    using non_zero_neighbour_pole[OF  $\langle is\_pole\ f\ z \rangle$ ] .
  moreover have  $\forall_F w \text{ in } at\ z. \text{deriv } f\ w \neq 0$ 
    using is_pole_deriv[OF  $\langle is\_pole\ f\ z \rangle$  f_iso, THEN non_zero_neighbour_pole]
    .
  ultimately have  $\forall_F w \text{ in } at\ z. False$ 
    by eventually_elim auto
  then show False by auto
qed

lemma isolated_pole_imp_neg_zorder:
  assumes isolated_singularity_at f z and is_pole f z
  shows  $zorder\ f\ z < 0$ 
  using analytic_imp_holomorphic assms centre_in_ball isolated_singularity_at_def
  zorder_exist_pole by blast

lemma isolated_singularity_at_deriv[singularity_intros]:
  assumes isolated_singularity_at f x
  shows isolated_singularity_at (deriv f) x
  by (meson analytic_deriv assms isolated_singularity_at_def)

lemma zorder_deriv_minus_1:
  fixes f g:: complex  $\Rightarrow$  complex and z::complex
  assumes f_iso: isolated_singularity_at f z
    and f_ess: not_essential f z
    and f_nconst:  $\exists_F w \text{ in } at\ z. f\ w \neq 0$ 
    and f_ord:  $zorder\ f\ z \neq 0$ 
  shows  $zorder\ (deriv\ f)\ z = zorder\ f\ z - 1$ 
proof -
  define P where  $P = zor\_poly\ f\ z$ 
  define n where  $n = zorder\ f\ z$ 
  have  $n \neq 0$  unfolding n_def using f_ord by auto

  obtain r where  $P\ z \neq 0$   $r > 0$  and P_holo:  $P$  holomorphic_on cball z r
    and  $(\forall w \in cball\ z\ r - \{z\}. f\ w = P\ w * (w - z)^{powi\ n} \wedge P\ w \neq 0)$ 
    using zorder_exist[OF f_iso f_ess f_nconst, folded P_def n_def] by auto
  from this(4)
  have f_eq:  $(\forall w \in cball\ z\ r - \{z\}. f\ w = P\ w * (w - z)^{powi\ n} \wedge P\ w \neq 0)$ 
    using complex_powr_of_int f_ord n_def by presburger

  define D where  $D = (\lambda w. (deriv\ P\ w * (w - z) + of\_int\ n * P\ w)$ 

```

$$* (w-z) \text{ powi } (n - 1))$$

```

have deriv_f_eq: deriv f w = D w if w ∈ ball z r - {z} for w
proof -
  have ev': eventually (λw. w ∈ ball z r - {z}) (nhds w)
    using that by (intro eventually_nhds_in_open) auto

  define wz where wz = w - z

  have wz ≠ 0 unfolding wz_def using that by auto
  moreover have (P has_field_derivative deriv P w) (at w)
    by (meson DiffD1 Elementary_Metric_Spaces.open_ball P_holo
      ball_subset_cball holomorphic_derivI holomorphic_on_subset that)
  ultimately have ((λw. P w * (w-z) powi n) has_field_derivative D w) (at w)
    unfolding D_def using that
    apply (auto intro!: derivative_eq_intros)
    by (auto simp: algebra_simps simp flip:power_int_add_1' wz_def)
  also have ?this ⟷ (f has_field_derivative D w) (at w)
    using f_eq
    by (intro has_field_derivative_cong_ev refl eventually_mono[OF ev']) auto
  ultimately have (f has_field_derivative D w) (at w) by simp
  moreover have (f has_field_derivative deriv f w) (at w)
    by (metis DERIV_imp_deriv calculation)
  ultimately show ?thesis using DERIV_imp_deriv by blast
qed

show zorder (deriv f) z = n - 1
proof (rule zorder_eqI)
  show open (ball z r) z ∈ ball z r
    using ⟨r > 0⟩ by auto
  define g where g=(λw. (deriv P w * (w-z) + of_int n * P w))
  show g holomorphic_on ball z r
    unfolding g_def using P_holo
    by (auto intro!:holomorphic_intros)
  show g z ≠ 0
    unfolding g_def using ⟨P z ≠ 0⟩ ⟨n≠0⟩ by auto
  show deriv f w = (deriv P w * (w-z) + of_int n * P w) * (w-z) powi (n -
1)
    if w ∈ ball z r w ≠ z for w
    using D_def deriv_f_eq that by blast
qed
qed

```

```

lemma deriv_divide_is_pole: — Generalises  $\llbracket \text{is\_pole } ?f \text{ ?}x; \text{ isolated\_singularity\_at } ?f \text{ ?}x \rrbracket \implies \text{zorder } (\text{deriv } ?f) \text{ ?}x = \text{zorder } ?f \text{ ?}x - 1$ 
fixes f g:: complex ⇒ complex and z::complex
assumes f_iso: isolated_singularity_at f z
and f_ness: not_essential f z

```

```

    and fg_nconst:  $\exists_F w \text{ in } (at\ z). \text{deriv } f\ w * f\ w \neq 0$ 
    and f_ord:  $\text{zorder } f\ z \neq 0$ 
    shows is_pole  $(\lambda z. \text{deriv } f\ z / f\ z)\ z$ 
  proof (rule neg_zorder_imp_is_pole)
    define ff where  $ff = (\lambda w. \text{deriv } f\ w / f\ w)$ 
    show isolated_singularity_at ff z
      using f_iso f_ness unfolding ff_def
      by (auto intro: singularity_intros)
    show not_essential ff z
      unfolding ff_def using f_ness f_iso by (auto intro: singularity_intros)

    have  $\text{zorder } ff\ z = \text{zorder } (\text{deriv } f)\ z - \text{zorder } f\ z$ 
      unfolding ff_def using f_iso f_ness fg_nconst
    using isolated_singularity_at_deriv not_essential_deriv zorder_divide by blast
    moreover have  $\text{zorder } (\text{deriv } f)\ z = \text{zorder } f\ z - 1$ 
      using f_iso f_ness f_ord fg_nconst frequently_elim1 zorder_deriv_minus_1
    by fastforce
    ultimately show  $\text{zorder } ff\ z < 0$  by auto

    show  $\exists_F w \text{ in } at\ z. ff\ w \neq 0$ 
      unfolding ff_def using fg_nconst by auto
  qed

lemma is_pole_deriv_divide_is_pole:
  fixes f g::complex  $\Rightarrow$  complex and z::complex
  assumes f_iso: isolated_singularity_at f z
    and is_pole f z
  shows is_pole  $(\lambda z. \text{deriv } f\ z / f\ z)\ z$ 
  proof (rule deriv_divide_is_pole[OF f_iso])
    show not_essential f z
      using <is_pole f z> unfolding not_essential_def by auto
    show  $\exists_F w \text{ in } at\ z. \text{deriv } f\ w * f\ w \neq 0$ 
      using assms f_iso isolated_pole_imp_nzero_times by blast
    show  $\text{zorder } f\ z \neq 0$ 
      using isolated_pole_imp_neg_zorder assms by fastforce
  qed

```

7.13 Isolated points

definition *isolated_points_of* :: complex set \Rightarrow complex set **where**
isolated_points_of A = $\{z \in A. \text{eventually } (\lambda w. w \notin A) (at\ z)\}$

lemma *isolated_points_of_altdef*: $\text{isolated_points_of } A = \{z \in A. \neg z \text{ islimpt } A\}$
unfolding *isolated_points_of_def* *islimpt_def* *eventually_at_filter* *eventually_nhds*
by blast

lemma *isolated_points_of_empty* [simp]: $\text{isolated_points_of } \{\} = \{\}$
and *isolated_points_of_UNIV* [simp]: $\text{isolated_points_of } UNIV = \{\}$
by (auto simp: isolated_points_of_def)

lemma *isolated_points_of_open_is_empty* [simp]: $\text{open } A \implies \text{isolated_points_of } A = \{\}$
unfolding *isolated_points_of_altdef*
by (simp add: interior_limit_point interior_open)

lemma *isolated_points_of_subset*: $\text{isolated_points_of } A \subseteq A$
by (auto simp: isolated_points_of_def)

lemma *isolated_points_of_discrete*:
assumes *discrete A*
shows $\text{isolated_points_of } A = A$
using *assms* **by** (auto simp: isolated_points_of_def discrete_altdef)

lemmas *uniform_discreteI1* = *uniformI1*
lemmas *uniform_discreteI2* = *uniformI2*

lemma *zorder_zero_eqI'*:
assumes *f analytic_on {z}*
assumes $\bigwedge i. i < \text{nat } n \implies (\text{deriv } \text{``} i) f z = 0$
assumes $(\text{deriv } \text{``} \text{nat } n) f z \neq 0 \text{ and } n \geq 0$
shows $\text{zorder } f z = n$
proof –
from *assms(1)* **obtain** *A* **where** $\text{open } A \ z \in A \ f \text{ holomorphic_on } A$
using *analytic_at* **by** *blast*
thus *?thesis*
using *zorder_zero_eqI* [of *f A z n*] *assms* **by** *blast*
qed

7.14 Isolated zeros

definition *isolated_zero* :: $('a::\text{topological_space} \Rightarrow 'b::\text{real_normed_div_algebra}) \Rightarrow 'a \Rightarrow \text{bool}$ **where**
 $\text{isolated_zero } f a \longleftrightarrow f -a \rightarrow 0 \wedge \text{eventually } (\lambda x. f x \neq 0) \text{ (at } a)$

lemma *isolated_zero_shift*:
fixes $z :: 'a :: \text{real_normed_vector}$
shows $\text{isolated_zero } f z \longleftrightarrow \text{isolated_zero } (\lambda w. f (z + w)) 0$
unfolding *isolated_zero_def*
by (simp add: at_to_0' eventually_filtermap filterlim_filtermap add_ac)

lemma *isolated_zero_shift'*:
fixes $z :: 'a :: \text{real_normed_vector}$
assumes *NO_MATCH 0 z*
shows $\text{isolated_zero } f z \longleftrightarrow \text{isolated_zero } (\lambda w. f (z + w)) 0$
by (rule *isolated_zero_shift*)

lemma *isolated_zero_imp_not_essential* [intro]:
 $\text{isolated_zero } f z \implies \text{not_essential } f z$

unfolding *isolated_zero_def not_essential_def*
using *tendsto_nhds_iff* **by** *blast*

lemma *pole_is_not_zero*:
fixes *f* :: 'a::perfect_space \Rightarrow 'b::real_normed_field
assumes *is_pole f z*
shows \neg *isolated_zero f z*
proof
assume *isolated_zero f z*
then have *filterlim f (nhds 0) (at z)*
unfolding *isolated_zero_def* **using** *tendsto_nhds_iff* **by** *blast*
moreover have *filterlim f at_infinity (at z)*
using \langle *is_pole f z* \rangle **unfolding** *is_pole_def* .
ultimately show *False*
using *not_tendsto_and_filterlim_at_infinity[OF at_neq_bot]*
by *auto*
qed

lemma *isolated_zero_imp_pole_inverse*:
fixes *f* :: $_ \Rightarrow$ 'b::{real_normed_div_algebra, division_ring}
assumes *isolated_zero f z*
shows *is_pole* ($\lambda z. \text{inverse } (f z)$) *z*
proof –
from *assms* **have** *ev*: *eventually* ($\lambda z. f z \neq 0$) (*at z*)
by (*auto simp: isolated_zero_def*)
have *filterlim f (nhds 0) (at z)*
using *assms* **by** (*simp add: isolated_zero_def*)
with *ev* **have** *filterlim f (at 0) (at z)*
using *filterlim_atI* **by** *blast*
also have $?this \longleftrightarrow \text{filterlim } (\lambda z. \text{inverse } (\text{inverse } (f z))) \text{ (at 0) (at z)}$
by (*rule filterlim_cong*) (*use ev in* \langle *auto elim!: eventually_mono* \rangle)
finally have *filterlim* ($\lambda z. \text{inverse } (f z)$) *at_infinity* (*at z*)
by (*subst filterlim_inverse_at_iff [symmetric]*)
thus *?thesis*
by (*simp add: is_pole_def*)
qed

lemma *is_pole_imp_isolated_zero_inverse*:
fixes *f* :: $_ \Rightarrow$ 'b::{real_normed_div_algebra, division_ring}
assumes *is_pole f z*
shows *isolated_zero* ($\lambda z. \text{inverse } (f z)$) *z*
proof –
from *assms* **have** *ev*: *eventually* ($\lambda z. f z \neq 0$) (*at z*)
by (*simp add: non_zero_neighbour_pole*)
have *filterlim f at_infinity* (*at z*)
using *assms* **by** (*simp add: is_pole_def*)
also have $?this \longleftrightarrow \text{filterlim } (\lambda z. \text{inverse } (\text{inverse } (f z))) \text{ at_infinity (at z)}$
by (*rule filterlim_cong*) (*use ev in* \langle *auto elim!: eventually_mono* \rangle)
finally have *filterlim* ($\lambda z. \text{inverse } (f z)$) (*at 0*) (*at z*)

```

  by (subst (asm) filterlim_inverse_at_iff [symmetric]) auto
  hence filterlim ( $\lambda z. \text{inverse } (f z)$ ) (nhds 0) (at z)
  using filterlim_at by blast
  moreover have eventually ( $\lambda z. \text{inverse } (f z) \neq 0$ ) (at z)
  using ev by eventually_elim auto
  ultimately show ?thesis
  by (simp add: isolated_zero_def)
qed

```

```

lemma is_pole_inverse_iff: is_pole ( $\lambda z. \text{inverse } (f z)$ )  $z \longleftrightarrow \text{isolated\_zero } f z$ 
  using is_pole_imp_isolated_zero_inverse isolated_zero_imp_pole_inverse by
  fastforce

```

```

lemma isolated_zero_inverse_iff: isolated_zero ( $\lambda z. \text{inverse } (f z)$ )  $z \longleftrightarrow \text{is\_pole } f z$ 
  using is_pole_imp_isolated_zero_inverse isolated_zero_imp_pole_inverse by
  fastforce

```

```

lemma zero_isolated_zero:
  fixes  $f :: 'a :: \{t2\_space, \text{perfect\_space}\} \Rightarrow \_$ 
  assumes isolated_zero  $f z$  isCont  $f z$ 
  shows  $f z = 0$ 
proof (rule tendsto_unique)
  show  $f -z \rightarrow f z$ 
  using assms(2) by (rule isContD)
  show  $f -z \rightarrow 0$ 
  using assms(1) by (simp add: isolated_zero_def)
qed auto

```

```

lemma zero_isolated_zero_analytic:
  assumes isolated_zero  $f z$   $f$  analytic_on  $\{z\}$ 
  shows  $f z = 0$ 
  using assms(1) analytic_at_imp_isCont[OF assms(2)] by (rule zero_isolated_zero)

```

```

lemma isolated_zero_analytic_iff:
  assumes  $f$  analytic_on  $\{z\}$ 
  shows isolated_zero  $f z \longleftrightarrow f z = 0 \wedge \text{eventually } (\lambda z. f z \neq 0) \text{ (at } z)$ 
proof safe
  assume  $f z = 0$  eventually ( $\lambda z. f z \neq 0$ ) (at z)
  with assms show isolated_zero  $f z$ 
  unfolding isolated_zero_def by (metis analytic_at_imp_isCont isCont_def)
qed (use zero_isolated_zero_analytic[OF _ assms] in ⟨auto simp: isolated_zero_def⟩)

```

```

lemma non_isolated_zero_imp_eventually_zero:
  assumes  $f$  analytic_on  $\{z\}$   $f z = 0$   $\neg \text{isolated\_zero } f z$ 
  shows eventually ( $\lambda z. f z = 0$ ) (at z)
proof (rule not_essential_frequently_0_imp_eventually_0)
  from assms(1) show isolated_singularity_at  $f z$  not_essential  $f z$ 
  by (simp_all add: isolated_singularity_at_analytic not_essential_analytic)

```

```

from assms(1,2) have  $f - z \rightarrow 0$ 
  by (metis analytic_at_imp_isCont continuous_within)
thus frequently ( $\lambda z. f z = 0$ ) (at  $z$ )
  using assms(2,3) by (auto simp: isolated_zero_def frequently_def)
qed

```

```

lemma non_isolated_zero_imp_eventually_zero':
  assumes  $f$  analytic_on  $\{z\}$   $f z = 0$   $\neg$ isolated_zero  $f z$ 
  shows eventually ( $\lambda z. f z = 0$ ) (nhds  $z$ )
  using non_isolated_zero_imp_eventually_zero[OF assms] assms(2)
  using eventually_nhds_conv_at by blast

```

```

end
theory Complex_Residues
  imports Complex_Singularities
begin

```

7.15 Definition of residues

Wenda Li and LC Paulson (2016). A Formal Proof of Cauchy's Residue Theorem. Interactive Theorem Proving

```

definition residue :: (complex  $\Rightarrow$  complex)  $\Rightarrow$  complex  $\Rightarrow$  complex where
  residue  $f z = (SOME\ int. \exists e > 0. \forall \varepsilon > 0. \varepsilon < e \longrightarrow$ 
     $\longrightarrow (f\ has\_contour\_integral\ 2 * \pi * i * int)\ (circlepath\ z\ \varepsilon))$ 

```

```

lemma residue_cong:

```

```

  assumes eq: eventually ( $\lambda z. f z = g z$ ) (at  $z$ ) and  $z = z'$ 
  shows residue  $f z = residue\ g\ z'$ 

```

```

proof -

```

```

  from assms have eq': eventually ( $\lambda z. g z = f z$ ) (at  $z$ )
    by (simp add: eq_commute)

```

```

  let  $?P = \lambda f\ c\ e. (\forall \varepsilon > 0. \varepsilon < e \longrightarrow$ 
     $(f\ has\_contour\_integral\ of\_real\ (2 * \pi) * i * c)\ (circlepath\ z\ \varepsilon))$ 

```

```

  have residue  $f z = residue\ g\ z$  unfolding residue_def

```

```

  proof (rule Eps_cong)

```

```

    fix  $c :: complex$ 

```

```

    have  $\exists e > 0. ?P\ g\ c\ e$ 

```

```

    if  $\exists e > 0. ?P\ f\ c\ e$  and eventually ( $\lambda z. f z = g z$ ) (at  $z$ ) for  $f\ g$ 

```

```

    proof -

```

```

      from that(1) obtain  $e$  where  $e > 0\ ?P\ f\ c\ e$ 

```

```

      by blast

```

```

      from that(2) obtain  $e'$  where  $e': e' > 0 \wedge z'. z' \neq z \implies dist\ z'\ z < e' \implies$ 
 $f\ z' = g\ z'$ 

```

```

      unfolding eventually_at by blast

```

```

      have  $?P\ g\ c\ (\min\ e\ e')$ 

```

```

      proof (intro allI exI impI, goal_cases)

```

```

        case (1  $\varepsilon$ )

```

```

        hence  $(f\ has\_contour\_integral\ of\_real\ (2 * \pi) * i * c)\ (circlepath\ z\ \varepsilon)$ 

```

```

        using e(2) by auto

```

```

    thus ?case
  proof (rule has_contour_integral_eq)
    fix z' assume z' ∈ path_image (circlepath z ε)
    hence dist z' z < e' and z' ≠ z
      using 1 by (auto simp: dist_commute)
    with e'(2)[of z'] show f z' = g z' by simp
  qed
qed
moreover from e and e' have min e e' > 0 by auto
ultimately show ?thesis by blast
qed
from this[OF _ eq] and this[OF _ eq]
  show (∃ e > 0. ?P f c e) ⟷ (∃ e > 0. ?P g c e)
  by blast
qed
with assms show ?thesis by simp
qed

lemma residue_shift_0: residue f z = residue (λx. f (z + x)) 0
proof -
  define Q where
    Q = (λr f z ε. (f has_contour_integral complex_of_real (2 * pi) * i * r)
      (circlepath z ε))
  define P where
    P = (λr f z. ∃ e > 0. ∀ ε > 0. ε < e ⟶ Q r f z ε)
  have path_eq: circlepath (z - w) ε = (+) (-w) ∘ circlepath z ε for z w ε
    by (simp add: circlepath_def o_def part_circlepath_def algebra_simps)
  have *: P r f z if P r (λx. f (x + w)) (z - w) for r w f z
    using that by (auto simp: P_def Q_def path_eq has_contour_integral_translate)
  have (SOME r. P r f z) = (SOME r. P r (λx. f (z + x)) 0)
    using *[of _ f z z] *[of _ λx. f (z + x) -z]
    by (intro arg_cong[where f = Eps] ext iffI) (simp_all add: add_ac)
  thus ?thesis
    by (simp add: residue_def P_def Q_def)
qed

lemma residue_shift_0': NO_MATCH 0 z ⟹ residue f z = residue (λx. f (z + x)) 0
  by (rule residue_shift_0)

lemma contour_integral_circlepath_eq:
  assumes open s and f_holo: f holomorphic_on (s - {z}) and 0 < e1 e1 ≤ e2
  and e2_cball: cball z e2 ⊆ s
  shows
    f contour_integrable_on circlepath z e1
    f contour_integrable_on circlepath z e2
    contour_integral (circlepath z e2) f = contour_integral (circlepath z e1) f
proof -
  define l where l ≡ linepath (z + e2) (z + e1)

```



```

have [simp]: valid_path l pathstart l=z+e2 pathfinish l=z+e1 unfolding l_def
by auto
have e2>0 using ⟨e1>0⟩ ⟨e1≤e2⟩ by auto
have z_l_img: z ∉ path_image l
proof
  assume z ∈ path_image l
  then have e2 ≤ cmod (e2 - e1)
    using segment_furthest_le[of z z+e2 z+e1 z+e2,simplified] ⟨e1>0⟩ ⟨e2>0⟩
unfolding l_def
  by (auto simp add: closed_segment_commute)
thus False using ⟨e2>0⟩ ⟨e1>0⟩ ⟨e1≤e2⟩
  apply (subst (asm) norm_of_real)
  by auto
qed
define g where g ≡ circlepath z e2 +++ l +++ reversepath (circlepath z e1)
+++ reversepath l
show [simp]: f contour_integrable_on circlepath z e2 f contour_integrable_on
(circlepath z e1)
proof -
  show f contour_integrable_on circlepath z e2
  apply (intro contour_integrable_continuous_circlepath[OF
    continuous_on_subset[OF holomorphic_on_imp_continuous_on[OF
f_holo]]])
  using ⟨e2>0⟩ e2_cball by auto
  show f contour_integrable_on (circlepath z e1)
  apply (intro contour_integrable_continuous_circlepath[OF
    continuous_on_subset[OF holomorphic_on_imp_continuous_on[OF
f_holo]]])
  using ⟨e1>0⟩ ⟨e1≤e2⟩ e2_cball by auto
qed
have [simp]: f contour_integrable_on l
proof -
  have closed_segment (z + e2) (z + e1) ⊆ cball z e2 using ⟨e2>0⟩ ⟨e1>0⟩
  ⟨e1≤e2⟩
  by (intro closed_segment_subset, auto simp add: dist_norm)
  hence closed_segment (z + e2) (z + e1) ⊆ s - {z} using z_l_img e2_cball
unfolding l_def
  by auto
  then show f contour_integrable_on l unfolding l_def
  apply (intro contour_integrable_continuous_linepath[OF
    continuous_on_subset[OF holomorphic_on_imp_continuous_on[OF
f_holo]]])
  by auto
qed
let ?ig = λg. contour_integral g f
have (f has_contour_integral 0) g
proof (rule Cauchy_theorem_global[OF f_holo])
  show open (s - {z}) using ⟨open s⟩ by auto
  show valid_path g unfolding g_def l_def by auto

```

```

    show pathfinish g = pathstart g unfolding g_def l_def by auto
  next
    have path_img:path_image g  $\subseteq$  cball z e2
    proof -
      have closed_segment (z + e2) (z + e1)  $\subseteq$  cball z e2 using  $\langle e2 > 0 \rangle$   $\langle e1 > 0 \rangle$ 
     $\langle e1 \leq e2 \rangle$ 
      by (intro closed_segment_subset, auto simp add: dist_norm)
      moreover have sphere z |e1|  $\subseteq$  cball z e2 using  $\langle e2 > 0 \rangle$   $\langle e1 \leq e2 \rangle$   $\langle e1 > 0 \rangle$ 
    by auto
      ultimately show ?thesis unfolding g_def l_def using  $\langle e2 > 0 \rangle$ 
      by (simp add: path_image_join closed_segment_commute)
    qed
  show path_image g  $\subseteq$  s - {z}
  proof -
    have  $z \notin$  path_image g using z_l_img
    unfolding g_def l_def by (auto simp add: path_image_join closed_segment_commute)
    moreover note  $\langle$  cball z e2  $\subseteq$  s  $\rangle$  and path_img
    ultimately show ?thesis by auto
  qed
  show winding_number g w = 0 when  $w \notin$  s - {z} for w
  proof -
    have winding_number g w = 0 when  $w \notin$  s using that e2_cball
    apply (intro winding_number_zero_outside[OF _ _ _ path_img])
    by (auto simp add: g_def l_def)
    moreover have winding_number g z = 0
    proof -
      let ?Wz =  $\lambda g. \text{winding\_number } g \ z$ 
      have ?Wz g = ?Wz (circlepath z e2) + ?Wz l + ?Wz (reversepath
    (circlepath z e1))
      + ?Wz (reversepath l)
      using  $\langle e2 > 0 \rangle$   $\langle e1 > 0 \rangle$  z_l_img unfolding g_def l_def
      by (subst winding_number_join, auto simp add: path_image_join
    closed_segment_commute) +
      also have ... = ?Wz (circlepath z e2) + ?Wz (reversepath (circlepath
    z e1))
      using z_l_img
      apply (subst (2) winding_number_reversepath)
      by (auto simp add: l_def closed_segment_commute)
      also have ... = 0
    proof -
      have ?Wz (circlepath z e2) = 1 using  $\langle e2 > 0 \rangle$ 
      by (auto intro: winding_number_circlepath_centre)
      moreover have ?Wz (reversepath (circlepath z e1)) = -1 using
     $\langle e1 > 0 \rangle$ 
      apply (subst winding_number_reversepath)
      by (auto intro: winding_number_circlepath_centre)
      ultimately show ?thesis by auto
    qed
  qed
  finally show ?thesis .

```

```

      qed
      ultimately show ?thesis using that by auto
    qed
  qed
  then have 0 = ?ig g using contour_integral_unique by simp
  also have ... = ?ig (circlepath z e2) + ?ig l + ?ig (reversepath (circlepath z e1))
    + ?ig (reversepath l)
    unfolding g_def
    by (auto simp add: contour_integrable_reversepath_eq)
  also have ... = ?ig (circlepath z e2) - ?ig (circlepath z e1)
    by (auto simp add: contour_integral_reversepath)
  finally show contour_integral (circlepath z e2) f = contour_integral (circlepath
z e1) f
    by simp
  qed

lemma base_residue:
  assumes open s z ∈ s r > 0 and f_holo: f holomorphic_on (s - {z})
    and r_cball: cball z r ⊆ s
  shows (f has_contour_integral 2 * pi * i * (residue f z)) (circlepath z r)
proof -
  obtain e where e > 0 and e_cball: cball z e ⊆ s
    using open_contains_cball[of s] ⟨open s⟩ ⟨z ∈ s⟩ by auto
  define c where c ≡ 2 * pi * i
  define i where i ≡ contour_integral (circlepath z e) f / c
  have (f has_contour_integral c * i) (circlepath z ε) when ε > 0 ε < e for ε
  proof -
    have contour_integral (circlepath z e) f = contour_integral (circlepath z ε) f
      f contour_integrable_on circlepath z ε
      f contour_integrable_on circlepath z e
    using ⟨ε < e⟩
    by (intro contour_integral_circlepath_eq[OF ⟨open s⟩ f_holo ⟨ε > 0⟩ _
e_cball], auto) +
    then show ?thesis unfolding i_def c_def
      by (auto intro: has_contour_integral_integral)
  qed
  then have ∃ e > 0. ∀ ε > 0. ε < e ⟶ (f has_contour_integral c * (residue f z))
(circlepath z ε)
    unfolding residue_def c_def
    apply (rule_tac someI[of _ i], intro exI[where x = e])
    by (auto simp add: ⟨e > 0⟩ c_def)
  then obtain e' where e' > 0
    and e'_def: ∀ ε > 0. ε < e' ⟶ (f has_contour_integral c * (residue f z))
(circlepath z ε)
    by auto
  let ?int = λ e. contour_integral (circlepath z e) f
  define ε where ε ≡ Min {r, e'} / 2
  have ε > 0 ε ≤ r ε < e' using ⟨r > 0⟩ ⟨e' > 0⟩ unfolding ε_def by auto
  have (f has_contour_integral c * (residue f z)) (circlepath z ε)

```

```

    using e'_def[rule_format,OF ‹ $\varepsilon > 0$ › ‹ $\varepsilon < e'$ ›] .
  then show ?thesis unfolding c_def
    using contour_integral_circlepath_eq[OF ‹open s› f_holo ‹ $\varepsilon > 0$ › ‹ $\varepsilon \leq r$ › r_cball]
    by (auto elim: has_contour_integral_eqpath[of _ _ circlepath z  $\varepsilon$  circlepath z
r])
qed

```

lemma *residue_holo*:

```

  assumes open s z  $\in$  s and f_holo: f holomorphic_on s
  shows residue f z = 0
proof -
  define c where c  $\equiv$  2 * pi * i
  obtain e where e > 0 and e_cball: cball z e  $\subseteq$  s using ‹open s› ‹z  $\in$  s›
  using open_contains_cball_eq by blast
  have (f has_contour_integral c * residue f z) (circlepath z e)
    using f_holo
    by (auto intro: base_residue[OF ‹open s› ‹z  $\in$  s› ‹e > 0› _ e_cball, folded c_def])
  moreover have (f has_contour_integral 0) (circlepath z e)
    using f_holo e_cball ‹e > 0›
    by (auto intro: Cauchy_theorem_convex_simple[of _ cball z e])
  ultimately have c * residue f z = 0
    using has_contour_integral_unique by blast
  thus ?thesis unfolding c_def by auto
qed

```

lemma *residue_const*: residue ($\lambda _.$ c) z = 0

```

  by (intro residue_holo[of UNIV::complex set], auto intro: holomorphic_intros)

```

lemma *residue_add*:

```

  assumes open s z  $\in$  s and f_holo: f holomorphic_on s - {z}
    and g_holo: g holomorphic_on s - {z}
  shows residue ( $\lambda z.$  f z + g z) z = residue f z + residue g z
proof -
  define c where c  $\equiv$  2 * pi * i
  define fg where fg  $\equiv$  ( $\lambda z.$  f z + g z)
  obtain e where e > 0 and e_cball: cball z e  $\subseteq$  s using ‹open s› ‹z  $\in$  s›
  using open_contains_cball_eq by blast
  have (fg has_contour_integral c * residue fg z) (circlepath z e)
    unfolding fg_def using f_holo g_holo
    apply (intro base_residue[OF ‹open s› ‹z  $\in$  s› ‹e > 0› _ e_cball, folded c_def])
    by (auto intro: holomorphic_intros)
  moreover have (fg has_contour_integral c * residue f z + c * residue g z) (circlepath
z e)
    unfolding fg_def using f_holo g_holo
    by (auto intro: has_contour_integral_add base_residue[OF ‹open s› ‹z  $\in$  s›
‹e > 0› _ e_cball, folded c_def])
  ultimately have c * (residue f z + residue g z) = c * residue fg z
    using has_contour_integral_unique by (auto simp add: distrib_left)
  thus ?thesis unfolding fg_def

```

by (auto simp add: c_def)
qed

lemma residue_lm1:

assumes open s z \in s and f_holo: f holomorphic_on s - {z}
shows residue ($\lambda z. c * (f z)$) z = c * residue f z
proof (cases c=0)
case True
thus ?thesis using residue_const by auto
next
case False
define c' where c' \equiv 2 * pi * i
define f' where f' \equiv ($\lambda z. c * (f z)$)
obtain e where e > 0 and e_cball: cball z e \subseteq s using <open s> <z \in s>
using open_contains_cball_eq by blast
have (f' has_contour_integral c' * residue f' z) (circlepath z e)
unfolding f'_def using f_holo
apply (intro base_residue[OF <open s> <z \in s> <e > 0> _ e_cball, folded c'_def])
by (auto intro: holomorphic_intros)
moreover have (f' has_contour_integral c * (c' * residue f z)) (circlepath z e)
unfolding f'_def using f_holo
by (auto intro: has_contour_integral_lm1
base_residue[OF <open s> <z \in s> <e > 0> _ e_cball, folded c'_def])
ultimately have c' * residue f' z = c * (c' * residue f z)
using has_contour_integral_unique by auto
thus ?thesis unfolding f'_def c'_def using False
by (auto simp add: field_simps)
qed

lemma residue_rm1:

assumes open s z \in s and f_holo: f holomorphic_on s - {z}
shows residue ($\lambda z. (f z) * c$) z = residue f z * c
using residue_lm1[OF assms, of c] by (auto simp add: algebra_simps)

lemma residue_div:

assumes open s z \in s and f_holo: f holomorphic_on s - {z}
shows residue ($\lambda z. (f z) / c$) z = residue f z / c
using residue_lm1[OF assms, of 1/c] by (auto simp add: algebra_simps)

lemma residue_neg:

assumes open s z \in s and f_holo: f holomorphic_on s - {z}
shows residue ($\lambda z. - (f z)$) z = - residue f z
using residue_lm1[OF assms, of -1] by auto

lemma residue_diff:

assumes open s z \in s and f_holo: f holomorphic_on s - {z}
and g_holo: g holomorphic_on s - {z}
shows residue ($\lambda z. f z - g z$) z = residue f z - residue g z
using residue_add[OF assms(1,2,3), of $\lambda z. - g z$] residue_neg[OF assms(1,2,4)]

by (auto intro:holomorphic_intros g_holo)

lemma residue_simple:

assumes open s $z \in s$ and f_holo: f holomorphic_on s

shows residue $(\lambda w. f w / (w - z)) z = f z$

proof -

define c where $c \equiv 2 * \pi * i$

define f' where $f' \equiv \lambda w. f w / (w - z)$

obtain e where $e > 0$ and e_cball: cball z e $\subseteq s$ using <open s> <z ∈ s>

using open_contains_cball_eq by blast

have (f' has_contour_integral c * f z) (circlepath z e)

unfolding f'_def c_def using <e > 0> f_holo e_cball

by (auto intro!: Cauchy_integral_circlepath_simple holomorphic_intros)

moreover have (f' has_contour_integral c * residue f' z) (circlepath z e)

unfolding f'_def using f_holo

apply (intro base_residue[OF <open s> <z ∈ s> <e > 0> _ e_cball_folded c_def])

by (auto intro!:holomorphic_intros)

ultimately have $c * f z = c * \text{residue } f' z$

using has_contour_integral_unique by blast

thus ?thesis unfolding c_def f'_def by auto

qed

lemma residue_simple':

assumes s: open s $z \in s$ and holo: f holomorphic_on (s - {z})

and lim: $((\lambda w. f w * (w - z)) \longrightarrow c) (at z)$

shows residue f z = c

proof -

define g where $g = (\lambda w. \text{if } w = z \text{ then } c \text{ else } f w * (w - z))$

from holo have $(\lambda w. f w * (w - z)) \text{ holomorphic_on } (s - \{z\})$ (is ?P)

by (force intro: holomorphic_intros)

also have $?P \longleftrightarrow g \text{ holomorphic_on } (s - \{z\})$

by (intro holomorphic_cong refl) (simp_all add: g_def)

finally have *: g holomorphic_on (s - {z}) .

note lim

also have $(\lambda w. f w * (w - z)) -z \rightarrow c \longleftrightarrow g -z \rightarrow g z$

by (intro filterlim_cong refl) (simp_all add: g_def [abs_def] eventually_at_filter)

finally have **: $g -z \rightarrow g z$.

have g_holo: g holomorphic_on s

by (rule no_isolated_singularity'[where K = {z}])

(insert assms * **, simp_all add: at_within_open_NO_MATCH)

from s and this have residue $(\lambda w. g w / (w - z)) z = g z$

by (rule residue_simple)

also have $\forall_F za \text{ in } at z. g za / (za - z) = f za$

unfolding eventually_at by (auto intro!: exI[of _ 1] simp: field_simps g_def)

hence residue $(\lambda w. g w / (w - z)) z = \text{residue } f z$

by (intro residue_cong refl)

finally show ?thesis

by (simp add: g_def)
qed

lemma residue_holomorphic_over_power:

assumes open A z0 ∈ A f holomorphic_on A
shows residue (λz. f z / (z - z0) ^ Suc n) z0 = (deriv ^^ n) f z0 / fact n
proof -
let ?f = λz. f z / (z - z0) ^ Suc n
from assms(1,2) obtain r where r: r > 0 cball z0 r ⊆ A
by (auto simp: open_contains_cball)
have (?f has_contour_integral 2 * pi * i * residue ?f z0) (circlepath z0 r)
using r assms by (intro base_residue[of A]) (auto intro!: holomorphic_intros)
moreover have (?f has_contour_integral 2 * pi * i / fact n * (deriv ^^ n) f
z0) (circlepath z0 r)
using assms r
by (intro Cauchy_has_contour_integral_higher_derivative_circlepath)
(auto intro!: holomorphic_on_subset[OF assms(3)] holomorphic_on_imp_continuous_on)
ultimately have 2 * pi * i * residue ?f z0 = 2 * pi * i / fact n * (deriv ^^ n)
f z0
by (rule has_contour_integral_unique)
thus ?thesis by (simp add: field_simps)
qed

lemma residue_holomorphic_over_power':

assumes open A 0 ∈ A f holomorphic_on A
shows residue (λz. f z / z ^ Suc n) 0 = (deriv ^^ n) f 0 / fact n
using residue_holomorphic_over_power[OF assms] by simp

theorem residue_fps_expansion_over_power_at_0:

assumes f has_fps_expansion F
shows residue (λz. f z / z ^ Suc n) 0 = fps_nth F n
proof -
from has_fps_expansion_imp_holomorphic[OF assms] obtain s
where open s 0 ∈ s f holomorphic_on s ∧ z. z ∈ s ⇒ f z = eval_fps F z
by auto
with assms have residue (λz. f z / (z - 0) ^ Suc n) 0 = (deriv ^^ n) f 0 / fact
n
unfolding has_fps_expansion_def
by (intro residue_holomorphic_over_power[of s]) (auto simp: zero_ereal_def)
also from assms have ... = fps_nth F n
by (subst fps_nth_fps_expansion) auto
finally show ?thesis by simp
qed

lemma residue_pole_order:

fixes f::complex ⇒ complex and z::complex
defines n ≡ nat (− zorder f z) and h ≡ zor_poly f z
assumes f_iso:isolated_singularity_at f z
and pole:is_pole f z

```

shows residue f z = ((deriv  $\wedge$  (n - 1)) h z / fact (n-1))
proof -
  define g where g  $\equiv$   $\lambda x$ . if x=z then 0 else inverse (f x)
  obtain e where [simp]:e>0 and f_holo:f holomorphic_on ball z e - {z}
  using f_iso analytic_imp_holomorphic unfolding isolated_singularity_at_def
  by blast
  obtain r where 0 < n 0 < r and r_cball:cball z r  $\subseteq$  ball z e and h_holo: h
  holomorphic_on cball z r
  and h_divide:( $\forall w \in$  cball z r. ( $w \neq z \longrightarrow f w = h w / (w - z) ^ n$ )  $\wedge$   $h w \neq 0$ )
  proof -
    obtain r where r:zorder f z < 0 h z  $\neq$  0 r>0 cball z r  $\subseteq$  ball z e h holomor-
    phic_on cball z r
    ( $\forall w \in$  cball z r - {z}.  $f w = h w / (w - z) ^ n \wedge h w \neq 0$ )
    using zorder_exist_pole[OF f_holo,simplified,OF  $\langle$ is_pole f z $\rangle$ ,folded n_def
    h_def] by auto
    have n>0 using  $\langle$ zorder f z < 0 $\rangle$  unfolding n_def by simp
    moreover have ( $\forall w \in$  cball z r. ( $w \neq z \longrightarrow f w = h w / (w - z) ^ n$ )  $\wedge$   $h w \neq$ 
    0)
      using  $\langle$ h z $\neq$ 0 $\rangle$  r(6) by blast
    ultimately show ?thesis using r(3,4,5) that by blast
  qed
  have r_nonzero: $\bigwedge w$ .  $w \in$  ball z r - {z}  $\implies f w \neq 0$ 
  using h_divide by simp
  define c where c  $\equiv$  2 * pi * i
  define der_f where der_f  $\equiv$  ((deriv  $\wedge$  (n - 1)) h z / fact (n-1))
  define h' where h'  $\equiv$   $\lambda u$ . h u / (u - z) ^ n
  have (h' has_contour_integral c / fact (n - 1) * (deriv  $\wedge$  (n - 1)) h z)
  (circlepath z r)
    unfolding h'_def
  proof (rule Cauchy_has_contour_integral_higher_derivative_circlepath[of z r
  h z n-1,
    folded c_def Suc_pred'[OF  $\langle$ n>0 $\rangle$ ]])
    show continuous_on (cball z r) h using holomorphic_on_imp_continuous_on
    h_holo by simp
    show h holomorphic_on ball z r using h_holo by auto
    show z  $\in$  ball z r using  $\langle$ r>0 $\rangle$  by auto
  qed
  then have (h' has_contour_integral c * der_f) (circlepath z r) unfolding
  der_f_def by auto
  then have (f has_contour_integral c * der_f) (circlepath z r)
  proof (elim has_contour_integral_eq)
    fix x assume x  $\in$  path_image (circlepath z r)
    hence x $\in$  cball z r - {z} using  $\langle$ r>0 $\rangle$  by auto
    then show h' x = f x using h_divide unfolding h'_def by auto
  qed
  moreover have (f has_contour_integral c * residue f z) (circlepath z r)
  using base_residue[of  $\langle$ ball z e $\rangle$  z,simplified,OF  $\langle$ r>0 $\rangle$  f_holo r_cball,folded
  c_def]
  unfolding c_def by simp

```


ultimately have $c * \text{der_}f = c * \text{residue } f z$ **using** *has_contour_integral_unique*
by *blast*
 hence $\text{der_}f = \text{residue } f z$ **unfolding** *c_def* **by** *auto*
 thus *?thesis* **unfolding** *der_f_def* **by** *auto*
qed

lemma *residue_simple_pole*:
 assumes *isolated_singularity_at* $f z0$
 assumes *is_pole* $f z0$ *zorder* $f z0 = -1$
 shows $\text{residue } f z0 = \text{zor_poly } f z0 z0$
using *assms* **by** (*subst residue_pole_order*) *simp_all*

lemma *residue_simple_pole_limit*:
 assumes *isolated_singularity_at* $f z0$
 assumes *is_pole* $f z0$ *zorder* $f z0 = -1$
 assumes $((\lambda x. f (g x) * (g x - z0)) \longrightarrow c) F$
 assumes *filterlim* $g (at z0) F$ $F \neq \text{bot}$
 shows $\text{residue } f z0 = c$

proof –
 have $\text{residue } f z0 = \text{zor_poly } f z0 z0$
by (*rule residue_simple_pole assms*) +
 also have $\dots = c$
apply (*rule zor_poly_pole_eqI*)
using *assms* **by** *auto*
 finally **show** *?thesis* .
qed

lemma
 assumes *f_holo*: f *holomorphic_on* s **and** *g_holo*: g *holomorphic_on* s
and *open s connected s* $z \in s$
 assumes *g_deriv*: (g *has_field_derivative* g') ($at z$)
 assumes $f z \neq 0$ $g z = 0$ $g' \neq 0$
 shows *porder_simple_pole_deriv*: *zorder* $(\lambda w. f w / g w) z = -1$
and *residue_simple_pole_deriv*: $\text{residue } (\lambda w. f w / g w) z = f z / g'$
proof –
 have [*simp*]: *isolated_singularity_at* $f z$ *isolated_singularity_at* $g z$
using *isolated_singularity_at_holomorphic* [*OF* $\langle \text{open } s \rangle \langle z \in s \rangle$] *f_holo g_holo*
by (*meson Diff_subset holomorphic_on_subset*) +
 have [*simp*]: *not_essential* $f z$ *not_essential* $g z$
unfolding *not_essential_def* **using** *f_holo g_holo assms* (3,5)
by (*meson continuous_on_eq_continuous_at continuous_within holomorphic_on_imp_continuous_on*) +
 have *g_nconst*: $\exists_F w$ *in* $at z.$ $g w \neq 0$
proof (*rule ccontr*)
 assume $\neg (\exists_F w$ *in* $at z.$ $g w \neq 0$)
 then have $\forall_F w$ *in* $nhds z.$ $g w = 0$
unfolding *eventually_at eventually_nhds frequently_at* **using** $\langle g z = 0 \rangle$
by (*metis open_ball UNIV_I centre_in_ball dist_commute mem_ball*)
 then have *deriv* $g z = \text{deriv } (\lambda_. 0) z$
by (*intro deriv_cong_ev*) *auto*

```

    then have  $\text{deriv } g \, z = 0$  by auto
    then have  $g' = 0$  using  $g\_deriv \, \text{DERIV\_imp\_deriv}$  by blast
    then show  $\text{False}$  using  $\langle g' \neq 0 \rangle$  by auto
  qed

  have  $\text{zorder } (\lambda w. f \, w / g \, w) \, z = \text{zorder } f \, z - \text{zorder } g \, z$ 
  proof -
    have  $\forall_F w \text{ in at } z. f \, w \neq 0 \wedge w \in s$ 
    apply (rule non_zero_neighbour_alt)
    using  $\text{assms}$  by auto
    with  $g\_nconst$  have  $\exists_F w \text{ in at } z. f \, w * g \, w \neq 0$ 
    by (elim frequently_rev_mp eventually_rev_mp, auto)
    then show  $?thesis$  using  $\text{zorder\_divide}[of f \, z \, g]$  by auto
  qed

  moreover have  $\text{zorder } f \, z = 0$ 
  apply (rule  $\text{zorder\_zero\_eqI}[OF f\_holo \langle \text{open } s \rangle \langle z \in s \rangle]$ )
  using  $\langle f \, z \neq 0 \rangle$  by auto
  moreover have  $\text{zorder } g \, z = 1$ 
  apply (rule  $\text{zorder\_zero\_eqI}[OF g\_holo \langle \text{open } s \rangle \langle z \in s \rangle]$ )
  subgoal using  $\text{assms}(8)$  by auto
  subgoal using  $\text{DERIV\_imp\_deriv } \text{assms}(9) \, g\_deriv$  by auto
  subgoal by simp
  done
  ultimately show  $\text{zorder } (\lambda w. f \, w / g \, w) \, z = -1$  by auto

  show  $\text{residue } (\lambda w. f \, w / g \, w) \, z = f \, z / g'$ 
  proof (rule  $\text{residue\_simple\_pole\_limit}[\text{where } g=id \text{ and } F=at \, z, \text{simplified}]$ )
    show  $\text{zorder } (\lambda w. f \, w / g \, w) \, z = -1$  by fact
    show  $\text{isolated\_singularity\_at } (\lambda w. f \, w / g \, w) \, z$ 
    by (auto intro: singularity_intros)
    show  $\text{is\_pole } (\lambda w. f \, w / g \, w) \, z$ 
    proof (rule  $\text{is\_pole\_divide}$ )
      have  $\forall_F x \text{ in at } z. g \, x \neq 0$ 
      apply (rule non_zero_neighbour)
      using  $g\_nconst$  by auto
      moreover have  $g - z \rightarrow 0$ 
      using  $\text{DERIV\_isCont } \text{assms}(8) \, \text{continuous\_at } g\_deriv$  by force
      ultimately show  $\text{filterlim } g \, (\text{at } 0) \, (\text{at } z)$  unfolding  $\text{filterlim\_at}$  by simp
      show  $\text{isCont } f \, z$ 
      using  $\text{assms}(3,5) \, \text{continuous\_on\_eq\_continuous\_at } f\_holo \, \text{holomorphic\_on\_imp\_continuous\_on}$ 
      by auto
      show  $f \, z \neq 0$  by fact
    qed
    show  $\text{filterlim } id \, (\text{at } z) \, (\text{at } z)$  by (simp add:  $\text{filterlim\_iff}$ )
    have  $((\lambda w. (f \, w * (w - z)) / g \, w) \longrightarrow f \, z / g') \, (\text{at } z)$ 
    proof (rule  $\text{lhospital\_complex\_simple}$ )
      show  $((\lambda w. f \, w * (w - z)) \text{ has\_field\_derivative } f \, z) \, (\text{at } z)$ 
      using  $\text{assms}$  by (auto intro!:  $\text{derivative\_eq\_intros } \text{holomorphic\_derivI}[OF$ 

```

```

f_holo])
  show (g has_field_derivative g') (at z) by fact
qed (insert assms, auto)
then show (( $\lambda w. (f w / g w) * (w - z)$ )  $\longrightarrow$   $f z / g'$ ) (at z)
  by (simp add: field_split_simps)
qed
qed

```

7.16 Poles and residues of some well-known functions

```

lemma is_pole_Gamma: is_pole Gamma (-of_nat n)
  unfolding is_pole_def using Gamma_poles .

```

```

lemma Gamma_residue:
  residue Gamma (-of_nat n) =  $(-1)^n / \text{fact } n$ 
proof (rule residue_simple')
  show open (- ( $\mathbb{Z}_{\leq 0}$  -  $\{-\text{of\_nat } n\}$ ) :: complex set)
    by (intro open_Compl closed_subset_Ints) auto
  show Gamma_holomorphic_on (- ( $\mathbb{Z}_{\leq 0}$  -  $\{-\text{of\_nat } n\}$ ) -  $\{-\text{of\_nat } n\}$ )
    by (rule holomorphic_Gamma) auto
  show ( $\lambda w. \text{Gamma } w * (w - (-\text{of\_nat } n))$ )  $\rightarrow (-\text{of\_nat } n) \rightarrow (-1)^n / \text{fact } n$ 
    using Gamma_residues[of n] by simp
qed auto

end

```

8 The Residue Theorem, the Argument Principle and Rouché's Theorem

```

theory Residue_Theorem
  imports Complex_Residues HOL-Library.Landau_Symbols
begin

```

Several theorems that could be moved up, IF there were a previous theory importing both Landau Symbols and Elementary Metric Spaces

```

lemma continuous_bounded_at_infinity_imp_bounded:
  fixes f :: real  $\Rightarrow$  'a :: real_normed_field
  assumes f  $\in O[at\_bot](\lambda_. 1)$ 
  assumes f  $\in O[at\_top](\lambda_. 1)$ 
  assumes cf: continuous_on UNIV f
  shows bounded (range f)
proof -
  obtain c1 c2
    where eventually ( $\lambda x. \text{norm } (f x) \leq c1$ ) at_bot eventually ( $\lambda x. \text{norm } (f x) \leq c2$ ) at_top
  using assms by (auto elim!: landau_o.bigE)
  then obtain x1 x2 where x1:  $\bigwedge x. x \leq x1 \implies \text{norm } (f x) \leq c1$  and x2:  $\bigwedge x. x \geq x2 \implies \text{norm } (f x) \leq c2$ 

```

```

  by (auto simp: eventually_at_bot_linorder eventually_at_top_linorder)
  have compact (f ' {x1..x2})
  by (intro compact_continuous_image continuous_on_subset[OF cf]) auto
  hence bounded (f ' {x1..x2})
  by (rule compact_imp_bounded)
  then obtain c3 where c3:  $\bigwedge x. x \in \{x1..x2\} \implies \text{norm } (f x) \leq c3$ 
  unfolding bounded_iff by fast
  have  $\text{norm } (f x) \leq \text{Max } \{c1, c2, c3\}$  for x
  by (cases  $x \leq x1$ ; cases  $x \geq x2$ ) (use x1 x2 c3 in (auto simp: le_max_iff_disj))
  thus ?thesis
  unfolding bounded_iff by blast
qed

```

lemma holomorphic_on_extend:

```

  assumes f holomorphic_on S - {ξ} ξ ∈ interior S f ∈ O[at ξ](λ_. 1)
  shows (∃ g. g holomorphic_on S ∧ (∀ z ∈ S - {ξ}. g z = f z))
  by (subst holomorphic_on_extend_bounded) (insert assms, auto elim!: landau_o.bigE)

```

lemma removable_singularities:

```

  assumes finite X X ⊆ interior S f holomorphic_on (S - X)
  assumes  $\bigwedge z. z \in X \implies f \in O[at z](\lambda_. 1)$ 
  shows ∃ g. g holomorphic_on S ∧ (∀ z ∈ S - X. g z = f z)
  using assms

```

proof (induction arbitrary: f rule: finite_induct)

```

  case empty
  thus ?case by auto

```

next

```

  case (insert z0 X f)
  from insert.prem1 and insert.hyps have z0: z0 ∈ interior (S - X)
  by (auto simp: interior_diff finite_imp_closed)
  hence ∃ g. g holomorphic_on (S - X) ∧ (∀ z ∈ S - X - {z0}. g z = f z)
  using insert.prem1 insert.hyps by (intro holomorphic_on_extend) auto
  then obtain g where g: g holomorphic_on (S - X) ∀ z ∈ S - X - {z0}. g z =
f z by blast

```

```

  have ∃ h. h holomorphic_on S ∧ (∀ z ∈ S - X. h z = g z)

```

proof (rule insert.IH)

```

  fix z0' assume z0': z0' ∈ X

```

```

  hence eventually (λz. z ∈ interior S - (X - {z0'}) - {z0'}) (nhds z0')

```

```

  using insert.prem1 insert.hyps

```

```

  by (intro eventually_nhds_in_open open_Diff finite_imp_closed) auto

```

```

  hence ev: eventually (λz. z ∈ S - X - {z0'}) (at z0')

```

```

  unfolding eventually_at_filter

```

```

  by eventually_elim (insert z0' insert.hyps interior_subset[of S], auto)

```

```

  have g ∈ Θ[at z0'](f)

```

```

  by (intro bigthetaI_cong eventually_mono[OF ev]) (insert g, auto)

```

```

  also have f ∈ O[at z0'](λ_. 1)

```

```

  using z0' by (intro insert.prem1) auto

```

```

  finally show g ∈ ...

```

```

qed (use insert.prem1 g in auto)

```

then obtain h where h holomorphic_on $S \setminus \{z \in S \mid z = X\}$. $h \ z = g \ z$ by blast
 with g have h holomorphic_on $S \setminus \{z \in S \mid z = \text{insert } z0 \ X\}$. $h \ z = f \ z$ by auto
 thus ?case by blast
 qed

lemma continuous_imp_bigo_1:
 assumes continuous (at x within A) f
 shows $f \in O[at \ x \ \text{within} \ A](\lambda _. 1)$
 proof (rule bigoI_tendsto)
 from assms show $((\lambda x. f \ x / 1) \longrightarrow f \ x)$ (at x within A)
 by (auto simp: continuous_within)
 qed auto

lemma taylor_bigo_linear:
 assumes f field_differentiable at $x0$ within A
 shows $(\lambda x. f \ x - f \ x0) \in O[at \ x0 \ \text{within} \ A](\lambda x. x - x0)$
 proof -
 from assms obtain f' where $(f \ \text{has_field_derivative} \ f')$ (at $x0$ within A)
 by (auto simp: field_differentiable_def)
 hence $((\lambda x. (f \ x - f \ x0) / (x - x0)) \longrightarrow f')$ (at $x0$ within A)
 by (auto simp: has_field_derivative_iff)
 thus ?thesis by (intro bigoI_tendsto[where $c = f'$]) (auto simp: eventually_at_filter)
 qed

8.1 Cauchy's residue theorem

lemma get_integrable_path:
 assumes open S connected (S -pts) finite pts f holomorphic_on (S -pts) $a \in S$ -pts
 $b \in S$ -pts
 obtains g where valid_path g pathstart $g = a$ pathfinish $g = b$
 path_image $g \subseteq S$ -pts f contour_integrable_on g using assms
 proof (induct arbitrary: S thesis a rule: finite_induct[OF <finite pts>])
 case 1
 obtain g where valid_path g path_image $g \subseteq S$ pathstart $g = a$ pathfinish $g = b$
 using connected_open_polynomial_connected[OF <open S >, of $a \ b$] <connected
 ($S - \{a\}\)$ >
 valid_path_polynomial_function 1.premis(6) 1.premis(7) by auto
 moreover have f contour_integrable_on g
 using contour_integrable_holomorphic_simple[OF __ <open S > <valid_path g >
 <path_image $g \subseteq S$ >, of f]
 < f holomorphic_on $S - \{a\}$ >
 by auto
 ultimately show ?case using 1(1)[of g] by auto
 next
 case idt: (2 p pts)
 obtain e where $e > 0$ and $e: \forall w \in \text{ball } a \ e. w \in S \wedge (w \neq a \longrightarrow w \notin \text{insert } p \ \text{pts})$
 using finite_ball_avoid[OF <open S > <finite (insert p pts)>, of a]
 < $a \in S - \text{insert } p \ \text{pts}$ >

```

    by auto
  define a' where a'  $\equiv$  a + e/2
  have a'  $\in$  S - {p} - pts using e[rule_format, of a + e/2]  $\langle e > 0 \rangle$ 
    by (auto simp add: dist_complex_def a'_def)
  then obtain g' where g'[simp]: valid_path g' pathstart g' = a' pathfinish g' = b
    path_image g'  $\subseteq$  S - {p} - pts f contour_integrable_on g'
    using idt.hyps(3)[of a' S - {p}] idt.prem1 idt.hyps(1)
    by (metis Diff_insert2 open_delete)
  define g where g  $\equiv$  linepath a a' +++ g'
  have valid_path g unfolding g_def by (auto intro: valid_path_join)
  moreover have pathstart g = a and pathfinish g = b unfolding g_def by auto
  moreover have path_image g  $\subseteq$  S - insert p pts
    unfolding g_def
  proof (rule subset_path_image_join)
    have closed_segment a a'  $\subseteq$  ball a e using  $\langle e > 0 \rangle$ 
    by (auto dest!: segment_bound1 simp: a'_def dist_complex_def norm_minus_commute)
    then show path_image (linepath a a')  $\subseteq$  S - insert p pts using e idt(9)
    by auto
  next
    show path_image g'  $\subseteq$  S - insert p pts using g'(4) by blast
  qed
  moreover have f contour_integrable_on g
  proof -
    have closed_segment a a'  $\subseteq$  ball a e using  $\langle e > 0 \rangle$ 
    by (auto dest!: segment_bound1 simp: a'_def dist_complex_def norm_minus_commute)
    then have closed_segment a a'  $\subseteq$  S - insert p pts
      using e idt.prem1(6) by auto
    then have continuous_on (closed_segment a a') f
      using holomorphic_on_imp_continuous_on holomorphic_on_subset idt.prem1(5)
    by presburger
    then show ?thesis
      using contour_integrable_continuous_linepath by (simp add: g_def)
  qed
  ultimately show ?case using idt.prem1(1)[of g] by auto
qed

lemma Cauchy_theorem_aux:
  assumes open S connected (S - pts) finite pts pts  $\subseteq$  S f holomorphic_on S - pts
    valid_path g pathfinish g = pathstart g path_image g  $\subseteq$  S - pts
     $\forall z. (z \notin S) \longrightarrow \text{winding\_number } g \ z = 0$ 
     $\forall p \in S. h \ p > 0 \wedge (\forall w \in \text{cball } p (h \ p). w \in S \wedge (w \neq p \longrightarrow w \notin \text{pts}))$ 
  shows contour_integral g f =  $(\sum p \in \text{pts}. \text{winding\_number } g \ p * \text{contour\_integral } (\text{circlepath } p (h \ p)) f)$ 
    using assms
  proof (induct arbitrary: S g rule: finite_induct[OF  $\langle \text{finite pts} \rangle$ ])
    case 1
    then show ?case by (simp add: Cauchy_theorem_global contour_integral_unique)
  next
    case (2 p pts)

```

```

note  $fin[simp] = \langle finite (insert\ p\ pts) \rangle$ 
and  $connected = \langle connected\ (S - insert\ p\ pts) \rangle$ 
and  $valid[simp] = \langle valid\_path\ g \rangle$ 
and  $g\_loop[simp] = \langle pathfinish\ g = pathstart\ g \rangle$ 
and  $holo[simp] = \langle f\ holomorphic\_on\ S - insert\ p\ pts \rangle$ 
and  $path\_img = \langle path\_image\ g \subseteq S - insert\ p\ pts \rangle$ 
and  $winding = \langle \forall z. z \notin S \longrightarrow winding\_number\ g\ z = 0 \rangle$ 
and  $h = \langle \forall pa \in S. 0 < h\ pa \wedge (\forall w \in cball\ pa\ (h\ pa). w \in S \wedge (w \neq pa \longrightarrow w \notin insert\ p\ pts)) \rangle$ 
have  $h\ p > 0$  and  $p \in S$ 
and  $h\ p: \forall w \in cball\ p\ (h\ p). w \in S \wedge (w \neq p \longrightarrow w \notin insert\ p\ pts)$ 
using  $h\ \langle insert\ p\ pts \subseteq S \rangle$  by auto
obtain  $pg$  where  $pg[simp]: valid\_path\ pg\ pathstart\ pg = pathstart\ g\ pathfinish$ 
 $pg = p + h\ p$ 
 $path\_image\ pg \subseteq S - insert\ p\ pts$  f  $contour\_integrable\_on\ pg$ 
proof -
have  $p + h\ p \in cball\ p\ (h\ p)$  using  $h[rule\_format, of\ p]$ 
by  $(simp\ add: \langle p \in S \rangle\ dist\_norm)$ 
then have  $p + h\ p \in S - insert\ p\ pts$  using  $h[rule\_format, of\ p]\ \langle insert\ p\ pts \subseteq S \rangle$ 
by fastforce
moreover have  $pathstart\ g \in S - insert\ p\ pts$  using  $path\_img$  by auto
ultimately show ?thesis
using  $get\_integrable\_path[OF\ \langle open\ S \rangle\ connected\ fin\ holo, of\ pathstart\ g\ p + h\ p]$  that
by blast
qed
obtain  $n::int$  where  $n = winding\_number\ g\ p$ 
using  $integer\_winding\_number[OF\ \_g\_loop, of\ p]\ valid\ path\_img$ 
by  $(metis\ DiffD2\ Ints\_cases\ insertI1\ subset\_eq\ valid\_path\_imp\_path)$ 
define  $p\_circ$  where  $p\_circ \equiv circlepath\ p\ (h\ p)$ 
define  $p\_circ\_pt$  where  $p\_circ\_pt \equiv linepath\ (p + h\ p)\ (p + h\ p)$ 
define  $n\_circ$  where  $n\_circ \equiv \lambda n. ((+++)\ p\_circ\ \widehat{\sim}\ n)\ p\_circ\_pt$ 
define  $cp$  where  $cp \equiv if\ n \geq 0\ then\ reversepath\ (n\_circ\ (nat\ n))\ else\ n\_circ\ (nat\ (-\ n))$ 

have  $n\_circ: valid\_path\ (n\_circ\ k)$ 
 $winding\_number\ (n\_circ\ k)\ p = k$ 
 $pathstart\ (n\_circ\ k) = p + h\ p$ 
 $pathfinish\ (n\_circ\ k) = p + h\ p$ 
 $path\_image\ (n\_circ\ k) = (if\ k = 0\ then\ \{p + h\ p\}\ else\ sphere\ p\ (h\ p))$ 
 $p \notin path\_image\ (n\_circ\ k)$ 
 $\bigwedge p'. p' \notin S - pts \implies winding\_number\ (n\_circ\ k)\ p' = 0 \wedge p' \notin path\_image\ (n\_circ\ k)$ 
 $f\ contour\_integrable\_on\ (n\_circ\ k)$ 
 $contour\_integral\ (n\_circ\ k)\ f = k * contour\_integral\ p\_circ\ f$ 
for  $k$ 
proof (induct k)
case 0
show  $valid\_path\ (n\_circ\ 0)$ 

```

```

    and path_image (n_circ 0) = (if 0=0 then {p + h p} else sphere p (h p))
    and winding_number (n_circ 0) p = of_nat 0
    and pathstart (n_circ 0) = p + h p
    and pathfinish (n_circ 0) = p + h p
    and p ∉ path_image (n_circ 0)
    unfolding n_circ_def p_circ_pt_def using ⟨h p > 0⟩
    by (auto simp add: dist_norm)
  show winding_number (n_circ 0) p'=0 ∧ p'∉path_image (n_circ 0) when
p'∉S- pts for p'
    unfolding n_circ_def p_circ_pt_def
    apply (auto intro!:winding_number_trivial)
    by (metis Diff_iff pathfinish_in_path_image pg(3) pg(4) subsetCE sub-
set_insertI that)+
  show f contour_integrable_on (n_circ 0)
    unfolding n_circ_def p_circ_pt_def
    by (auto intro!:contour_integrable_continuous_linepath simp add:continuous_on_sing)
  show contour_integral (n_circ 0) f = of_nat 0 * contour_integral p_circ f
    unfolding n_circ_def p_circ_pt_def by auto
next
case (Suc k)
have n_Suc:n_circ (Suc k) = p_circ +++ n_circ k unfolding n_circ_def
by auto
  have pcirc:p ∉ path_image p_circ valid_path p_circ pathfinish p_circ =
pathstart (n_circ k)
  using Suc(3) unfolding p_circ_def using ⟨h p > 0⟩ by (auto simp add:
p_circ_def)
  have pcirc_image:path_image p_circ ⊆ S - insert p pts
  proof -
    have path_image p_circ ⊆ cball p (h p) using ⟨0 < h p⟩ p_circ_def by
auto
    then show ?thesis using h_p pcirc(1) by auto
  qed
have pcirc_integrable:f contour_integrable_on p_circ
  by (auto simp add:p_circ_def intro!: pcirc_image[unfolded p_circ_def]
contour_integrable_continuous_circlepath holomorphic_on_imp_continuous_on
holomorphic_on_subset[OF holo])
show valid_path (n_circ (Suc k))
  using valid_path_join[OF pcirc(2) Suc(1) pcirc(3)] unfolding n_circ_def
by auto
  show path_image (n_circ (Suc k))
    = (if Suc k = 0 then {p + complex_of_real (h p)} else sphere p (h p))
  proof -
    have path_image p_circ = sphere p (h p)
    unfolding p_circ_def using ⟨0 < h p⟩ by auto
    then show ?thesis unfolding n_Suc using Suc.hyps(5) ⟨h p>0⟩
    by (auto simp add: path_image_join[OF pcirc(3)] dist_norm)
  qed
then show p ∉ path_image (n_circ (Suc k)) using ⟨h p>0⟩ by auto
show winding_number (n_circ (Suc k)) p = of_nat (Suc k)

```



```

proof –
  have winding_number p_circ p = 1
  by (simp add: ⟨h p > 0⟩ p_circ_def winding_number_circlepath_centre)
  moreover have p ∉ path_image (n_circ k) using Suc(5) ⟨h p > 0⟩ by
auto
  then have winding_number (p_circ +++ n_circ k) p
    = winding_number p_circ p + winding_number (n_circ k) p
  using valid_path_imp_path Suc.hyps(1) Suc.hyps(2) pcirc
  apply (intro winding_number_join)
  by auto
  ultimately show ?thesis using Suc(2) unfolding n_circ_def
  by auto
qed
show pathstart (n_circ (Suc k)) = p + h p
  by (simp add: n_circ_def p_circ_def)
show pathfinish (n_circ (Suc k)) = p + h p
  using Suc(4) unfolding n_circ_def by auto
show winding_number (n_circ (Suc k)) p' = 0 ∧ p' ∉ path_image (n_circ
(Suc k)) when p' ∉ S-pts for p'
  proof –
    have p' ∉ path_image p_circ using ⟨p ∈ S⟩ h p_circ_def that using
pcirc_image by blast
    moreover have p' ∉ path_image (n_circ k)
    using Suc.hyps(7) that by blast
    moreover have winding_number p_circ p' = 0
    proof –
      have path_image p_circ ⊆ cball p (h p)
      using h unfolding p_circ_def using ⟨p ∈ S⟩ by fastforce
      moreover have p' ∉ cball p (h p) using ⟨p ∈ S⟩ h that 2.hyps(2) by
fastforce
      ultimately show ?thesis
      unfolding p_circ_def
      by (intro winding_number_zero_outside) auto
    qed
    ultimately show ?thesis
    unfolding n_Suc using Suc.hyps pcirc
    by (metis add.right_neutral not_in_path_image_join that valid_path_imp_path
winding_number_join)
  qed
show f contour_integrable_on (n_circ (Suc k))
  unfolding n_Suc
  by (rule contour_integrable_joinI[OF pcirc_integrable Suc(8) pcirc(2)
Suc(1)])
show contour_integral (n_circ (Suc k)) f = (Suc k) * contour_integral
p_circ f
  by (simp add: Rings.ring_distrib(2) Suc.hyps n_Suc pcirc pcirc_integrable)
qed
have cp[simp]: pathstart cp = p + h p pathfinish cp = p + h p
  valid_path cp path_image cp ⊆ S – insert p pts

```

```

    winding_number cp p = - n
     $\wedge p'. p' \notin S - pts \implies \text{winding\_number } cp \ p' = 0 \wedge p' \notin \text{path\_image } cp$ 
    f contour_integrable_on cp
    contour_integral cp f = - n * contour_integral p_circ f
  proof -
    show pathstart cp = p + h p and pathfinish cp = p + h p and valid_path cp
      using n_circ unfolding cp_def by auto
    next
      have sphere p (h p)  $\subseteq S - \text{insert } p \text{ pts}$ 
        using h[rule_format, of p]  $\langle \text{insert } p \text{ pts} \subseteq S \rangle$  by force
      moreover have p + complex_of_real (h p)  $\in S - \text{insert } p \text{ pts}$ 
        using pg(3) pg(4) by (metis pathfinish_in_path_image subsetCE)
      ultimately show path_image cp  $\subseteq S - \text{insert } p \text{ pts}$  unfolding cp_def
        using n_circ(5) by auto
    next
      show winding_number cp p = - n
        unfolding cp_def using winding_number_reversepath n_circ  $\langle h \ p > 0 \rangle$ 
        by (auto simp: valid_path_imp_path)
    next
      show winding_number cp  $p' = 0 \wedge p' \notin \text{path\_image } cp$  when  $p' \notin S - pts$  for
        p'
      proof -
        have winding_number (reversepath (n_circ (nat n)))  $p' = 0$ 
          using n_circ that
        by (metis add.inverse_neutral valid_path_imp_path winding_number_reversepath)
        then show ?thesis
          using cp_def n_circ(7) that by force
      qed
    next
      show f contour_integrable_on cp unfolding cp_def
        using contour_integrable_reversepath_eq n_circ(1,8) by auto
    next
      show contour_integral cp f = - n * contour_integral p_circ f
        unfolding cp_def using contour_integral_reversepath[OF n_circ(1)]
        n_circ(9)
        by auto
      qed
    define g' where  $g' \equiv g +++ pg +++ cp +++ (\text{reversepath } pg)$ 
    have contour_integral g' f =  $(\sum p \in pts. \text{winding\_number } g' \ p * \text{contour\_integral } (\text{circlepath } p \ (h \ p)) \ f)$ 
    proof (rule 2.hyps(3)[of  $S - \{p\}$  g', OF __  $\langle \text{finite } pts \rangle$  ])
      show connected  $(S - \{p\} - pts)$  using connected by (metis Diff_insert2)
      show open  $(S - \{p\})$  using  $\langle \text{open } S \rangle$  by auto
      show  $pts \subseteq S - \{p\}$  using  $\langle \text{insert } p \text{ pts} \subseteq S \rangle \langle p \notin pts \rangle$  by blast
      show f holomorphic_on  $S - \{p\} - pts$  using holo  $\langle p \notin pts \rangle$  by (metis
        Diff_insert2)
      show valid_path g'
        unfolding g'_def cp_def using n_circ valid pg g_loop
        by (auto intro!: valid_path_join)
    qed

```

```

show pathfinish g' = pathstart g'
  unfolding g'_def cp_def using pg(2) by simp
show path_image g'  $\subseteq S - \{p\} - pts$ 
  proof -
    define s' where s'  $\equiv S - \{p\} - pts$ 
    have s':s' = S - insert p pts unfolding s'_def by auto
    then show ?thesis using path_img pg(4) cp(4)
      by (simp add: g'_def s'_def subset_path_image_join)
    qed
  note path_join_imp[simp]
  show  $\forall z. z \notin S - \{p\} \longrightarrow \text{winding\_number } g' z = 0$ 
    proof clarify
      fix z assume z:  $z \notin S - \{p\}$ 
      have z_notin_cp:  $z \notin \text{path\_image } cp$ 
        using cp(6) cp_def n_circ(6) z by auto
      have z_notin_pg:  $z \notin \text{path\_image } pg$ 
        by (metis Diff_iff Diff_insert2 pg(4) subsetD z)
      have winding_number (g +++ pg +++ cp +++ reversepath pg) z =
        winding_number g z
        + winding_number (pg +++ cp +++ (reversepath pg)) z
      proof (rule winding_number_join)
        show path g using ⟨valid_path g⟩ by (simp add: valid_path_imp_path)
        show  $z \notin \text{path\_image } g$  using z path_img by auto
        show path (pg +++ cp +++ reversepath pg) using pg(3) cp
          by (simp add: valid_path_imp_path)
        next
          have path_image (pg +++ cp +++ reversepath pg)  $\subseteq S - \text{insert } p \text{ pts}$ 
            using pg(4) cp(4) by (auto simp: subset_path_image_join)
          then show  $z \notin \text{path\_image } (pg +++ cp +++ reversepath pg)$  using
            z by auto
        next
          show pathfinish g = pathstart (pg +++ cp +++ reversepath pg) using
            g_loop by auto
        qed
      also have ... = winding_number g z + (winding_number pg z
        + winding_number (cp +++ (reversepath pg)) z)
      proof (subst add_left_cancel, rule winding_number_join)
        show path pg and path (cp +++ reversepath pg)
          and pathfinish pg = pathstart (cp +++ reversepath pg)
          by (auto simp add: valid_path_imp_path)
        show  $z \notin \text{path\_image } pg$  using pg(4) z by blast
        show  $z \notin \text{path\_image } (cp +++ reversepath pg)$  using z
          by (metis Diff_iff ⟨ $z \notin \text{path\_image } pg$ ⟩ contra_subsetD cp(4) insertI1
            not_in_path_image_join path_image_reversepath singletonD)
        qed
      also have ... = winding_number g z + (winding_number pg z
        + (winding_number cp z + winding_number (reversepath pg) z))
      by (simp add: valid_path_imp_path winding_number_join z_notin_cp
        z_notin_pg)

```

```

    also have ... = winding_number g z + winding_number cp z
      by (simp add: valid_path_imp_path winding_number_reversepath
z_notin_pg)
    finally have winding_number g' z = winding_number g z + wind-
ing_number cp z
      unfolding g'_def .
    moreover have winding_number g z + winding_number cp z = 0
      using winding z ⟨n=winding_number g p⟩ by auto
    ultimately show winding_number g' z = 0 unfolding g'_def by auto
  qed
  show ∀ pa ∈ S - {p}. 0 < h pa ∧ (∀ w ∈ cball pa (h pa). w ∈ S - {p} ∧ (w
≠ pa ⟶ w ∉ pts))
    using h by fastforce
  qed
  moreover have contour_integral g' f = contour_integral g f
    - winding_number g p * contour_integral p_circ f
  proof -
    have *: f contour_integrable_on g f contour_integrable_on pg f contour_integrable_on
cp
      by (auto simp add: open_Diff[OF ⟨open S⟩, OF finite_imp_closed[OF fin]]
intro!: contour_integrable_holomorphic_simple[OF holo__ path_img])
    have contour_integral g' f = contour_integral g f + contour_integral pg f
      + contour_integral cp f + contour_integral (reversepath pg) f
      using * by (simp add: g'_def contour_integrable_reversepath_eq)
    also have ... = contour_integral g f + contour_integral cp f
      using contour_integral_reversepath
      by (auto simp add: contour_integrable_reversepath)
    also have ... = contour_integral g f - winding_number g p * contour_integral
p_circ f
      using ⟨n=winding_number g p⟩ by auto
    finally show ?thesis .
  qed
  moreover have winding_number g' p' = winding_number g p' when p' ∈ pts for
p'
  proof -
    obtain [simp]: p' ∉ path_image g p' ∉ path_image pg p' ∉ path_image cp
      using 2.prem8 that by (metis Diff_iff Diff_insert2 ⟨p' ∈ pts⟩ cp(4)
pg(4) subsetD)
    have winding_number g' p' = winding_number g p' + winding_number pg p'
      + winding_number (cp +++ reversepath pg) p'
      by (simp add: g'_def not_in_path_image_join valid_path_imp_path wind-
ing_number_join)
    also have ... = winding_number g p' using that
      by (simp add: valid_path_imp_path winding_number_join winding_number_reversepath)
    finally show ?thesis .
  qed
  ultimately show ?case unfolding p_circ_def
    apply (subst (asm) sum.cong[OF refl,
of pts _ λp. winding_number g p * contour_integral (circlepath p (h p)) f])

```

by (auto simp: sum.insert[OF ‹finite pts› ‹p∉pts›] algebra_simps)
qed

lemma *Cauchy_theorem_singularities:*

assumes *open S connected S finite pts and*
holo: f holomorphic_on S-pts and
valid_path g and
loop:pathfinish g = pathstart g and
path_image g ⊆ S-pts and
homo:∀ z. (z ∉ S) ⟶ winding_number g z = 0 and
avoid:∀ p∈S. h p>0 ∧ (∀ w∈cball p (h p). w∈S ∧ (w≠p ⟶ w ∉ pts))
shows *contour_integral g f = (∑ p∈pts. winding_number g p * contour_integral*
(circlepath p (h p)) f)
(is ?L=?R)
proof –
define *circ* **where** *circ ≡ λp. winding_number g p * contour_integral (circlepath*
p (h p)) f
define *pts1* **where** *pts1 ≡ pts ∩ S*
define *pts2* **where** *pts2 ≡ pts - pts1*
have *pts=pts1 ∪ pts2 pts1 ∩ pts2 = {} pts2 ∩ S={} pts1⊆S*
unfolding *pts1_def pts2_def* **by** *auto*
have *contour_integral g f = (∑ p∈pts1. circ p)* **unfolding** *circ_def*
proof (*rule Cauchy_theorem_aux[OF ‹open S› _ _ ‹pts1⊆S› _ ‹valid_path*
g› loop _ homo])
have *finite pts1* **unfolding** *pts1_def* **using** ‹finite pts› **by** *auto*
then show *connected (S - pts1)*
using ‹open S› ‹connected S› *connected_open_delete_finite[of S]* **by** *auto*
next
show *finite pts1* **using** ‹pts = pts1 ∪ pts2› *assms(3)* **by** *auto*
show *f holomorphic_on S - pts1* **by** (*metis Diff_Int2 Int_absorb holo*
pts1_def)
show *path_image g ⊆ S - pts1* **using** *assms(7) pts1_def* **by** *auto*
show *∀ p∈S. 0 < h p ∧ (∀ w∈cball p (h p). w ∈ S ∧ (w ≠ p ⟶ w ∉ pts1))*
by (*simp add: avoid pts1_def*)
qed
moreover have *sum circ pts2 = 0*
by (*metis ‹pts2 ∩ S = {}› circ_def disjoint_iff_not_equal homo mult_zero_left*
sum.neutral)
moreover have *?R=sum circ pts1 + sum circ pts2*
unfolding *circ_def*
using *sum.union_disjoint[OF _ _ ‹pts1 ∩ pts2 = {}›] ‹finite pts› ‹pts=pts1*
∪ pts2›
by *blast*
ultimately show *?thesis*
by *simp*
qed

theorem *Residue_theorem:*

fixes *S pts::complex set and f::complex ⇒ complex*

```

    and  $g::\text{real} \Rightarrow \text{complex}$ 
  assumes open  $S$  connected  $S$  finite pts and
    holo: $f$  holomorphic_on  $S$ -pts and
    valid_path  $g$  and
    loop:pathfinish  $g = \text{pathstart } g$  and
    path_image  $g \subseteq S$ -pts and
    homo: $\forall z. (z \notin S) \longrightarrow \text{winding\_number } g \ z = 0$ 
  shows contour_integral  $g \ f = 2 * \pi * i * (\sum p \in \text{pts}. \text{winding\_number } g \ p * \text{residue } f \ p)$ 
proof -
  define  $c$  where  $c \equiv 2 * \pi * i$ 
  obtain  $h$  where avoid: $\forall p \in S. h \ p > 0 \wedge (\forall w \in \text{cball } p \ (h \ p). w \in S \wedge (w \neq p \longrightarrow w \notin \text{pts}))$ 
  using finite_cball_avoid[OF open  $S$  finite pts] by metis
  have contour_integral  $g \ f$ 
    =  $(\sum p \in \text{pts}. \text{winding\_number } g \ p * \text{contour\_integral } (\text{circlepath } p \ (h \ p)) \ f)$ 
  using Cauchy_theorem_singularities[OF assms avoid] .
  also have ... =  $(\sum p \in \text{pts}. c * \text{winding\_number } g \ p * \text{residue } f \ p)$ 
  proof (intro sum.cong)
    show pts = pts by simp
  next
    fix  $x$  assume  $x \in \text{pts}$ 
    show winding_number  $g \ x * \text{contour\_integral } (\text{circlepath } x \ (h \ x)) \ f$ 
      =  $c * \text{winding\_number } g \ x * \text{residue } f \ x$ 
    proof (cases  $x \in S$ )
      case False
      then have winding_number  $g \ x = 0$  using homo by auto
      thus ?thesis by auto
    next
      case True
      have contour_integral  $(\text{circlepath } x \ (h \ x)) \ f = c * \text{residue } f \ x$ 
        using  $\langle x \in \text{pts} \rangle \langle \text{finite pts} \rangle$  avoid[rule_format, OF True]
      apply (intro base_residue[of  $S - (\text{pts} - \{x\})$ , THEN contour_integral_unique, folded  $c\_def$ ])
        by (auto intro:holomorphic_on_subset[OF holo] open_Diff[OF open  $S$  finite_imp_closed])
      then show ?thesis by auto
    qed
  qed
  also have ... =  $c * (\sum p \in \text{pts}. \text{winding\_number } g \ p * \text{residue } f \ p)$ 
    by (simp add: sum_distrib_left algebra_simps)
  finally show ?thesis unfolding  $c\_def$  .
qed

```

8.2 The argument principle

theorem argument_principle:

fixes $f::\text{complex} \Rightarrow \text{complex}$ and poles $S::\text{complex set}$

defines $pz \equiv \{w \in S. f \ w = 0 \vee w \in \text{poles}\}$ — pz is the set of poles and zeros

```

assumes open  $S$  connected  $S$  and
   $f\_holo$ :  $f$  holomorphic_on  $S - poles$  and
   $h\_holo$ :  $h$  holomorphic_on  $S$  and
  valid_path  $g$  and
  loop: pathfinish  $g = pathstart\ g$  and
  path_img: path_image  $g \subseteq S - pz$  and
  homo:  $\forall z. (z \notin S) \longrightarrow winding\_number\ g\ z = 0$  and
  finite: finite  $pz$  and
  poles:  $\forall p \in S \cap poles. is\_pole\ f\ p$ 
shows contour_integral  $g\ (\lambda x. deriv\ f\ x * h\ x / f\ x) = 2 * pi * i *$ 
   $(\sum_{p \in pz. winding\_number\ g\ p * h\ p * zorder\ f\ p})$ 
  (is ?L=?R)
proof -
  define  $c$  where  $c \equiv 2 * complex\_of\_real\ pi * i$ 
  define  $ff$  where  $ff \equiv (\lambda x. deriv\ f\ x * h\ x / f\ x)$ 
  define  $cont$  where  $cont \equiv \lambda ff\ p\ e. (ff\ has\_contour\_integral\ c * zorder\ f\ p * h\ p$ 
   $)\ (circlepath\ p\ e)$ 
  define  $avoid$  where  $avoid \equiv \lambda p\ e. \forall w \in cball\ p\ e. w \in S \wedge (w \neq p \longrightarrow w \notin pz)$ 

  have  $\exists e > 0. avoid\ p\ e \wedge (p \in pz \longrightarrow cont\ ff\ p\ e)$  when  $p \in S$  for  $p$ 
  proof -
    obtain  $e1$  where  $e1 > 0$  and  $e1\_avoid$ :  $avoid\ p\ e1$ 
    using finite_cball_avoid[OF <open  $S$ > finite] < $p \in S$ > unfolding avoid_def by
  auto
  have  $\exists e2 > 0. cball\ p\ e2 \subseteq ball\ p\ e1 \wedge cont\ ff\ p\ e2$  when  $p \in pz$ 
  proof -
    define  $po$  where  $po \equiv zorder\ f\ p$ 
    define  $pp$  where  $pp \equiv zor\_poly\ f\ p$ 
    define  $f'$  where  $f' \equiv \lambda w. pp\ w * (w - p)\ powi\ po$ 
    define  $ff'$  where  $ff' \equiv (\lambda x. deriv\ f'\ x * h\ x / f'\ x)$ 
    obtain  $r$  where  $pp\ p \neq 0\ r > 0$  and
       $r < e1$  and
       $pp\_holo$ :  $pp$  holomorphic_on  $cball\ p\ r$  and
       $pp\_po$ :  $(\forall w \in cball\ p\ r - \{p\}. f\ w = pp\ w * (w - p)\ powi\ po \wedge pp\ w \neq 0)$ 
    proof -
      have isolated_singularity_at  $f\ p$ 
      proof -
        have  $ball\ p\ e1 - \{p\} \subseteq S - poles$ 
        using avoid_def e1_avoid pz_def by fastforce
        then have  $f$  holomorphic_on  $ball\ p\ e1 - \{p\}$ 
        by (intro holomorphic_on_subset[OF  $f\_holo$ ])
        then show ?thesis unfolding isolated_singularity_at_def
        using < $e1 > 0$ > analytic_on_open open_delete by blast
      qed
    moreover have not_essential  $f\ p$ 
    proof (cases is_pole  $f\ p$ )
      case True
      then show ?thesis unfolding not_essential_def by auto
    next

```

```

case False
then have  $p \in S - \text{poles}$  using  $\langle p \in S \rangle$  poles unfolding  $pz\_def$  by auto
moreover have  $\text{open } (S - \text{poles})$ 
proof –
  have  $\text{closed } (S \cap \text{poles})$ 
    using finite by (simp add: pz_def finite_imp_closed rev_finite_subset
subset_eq)
  then show ?thesis
    by (metis Diff_Compl Diff_Diff_Int Diff_eq open S open_Diff)
  qed
ultimately have  $\text{isCont } f \text{ } p$ 
  using holomorphic_on_imp_continuous_on [OF f_holo] continuous_on_eq_continuous_at
  by auto
then show ?thesis unfolding  $\text{isCont\_def not\_essential\_def}$  by auto
qed
moreover have  $\exists_F w \text{ in at } p. f w \neq 0$ 
proof (rule ccontr)
  assume  $\neg (\exists_F w \text{ in at } p. f w \neq 0)$ 
  then have  $\forall_F w \text{ in at } p. f w = 0$  unfolding frequently_def by auto
  then obtain  $r1$  where  $r1 > 0$  and  $r1 : \forall w \in \text{ball } p \text{ } r1 - \{p\}. f w = 0$ 
    unfolding eventually_at by (auto simp add: dist_commute)
  obtain  $r2$  where  $r2 > 0$  and  $r2 : \text{ball } p \text{ } r2 \subseteq S$ 
    using  $\langle p \in S \rangle \langle \text{open } S \rangle \text{openE}$  by blast
  define  $rr$  where  $rr = \min r1 \ r2$ 

  from  $r1 \ r2$ 
  have  $\text{ball } p \text{ } rr - \{p\} \subseteq \{w \in S \cap \text{ball } p \text{ } rr - \{p\}. f w = 0\}$ 
    unfolding  $rr\_def$  by auto
  moreover have  $\text{infinite } (\text{ball } p \text{ } rr - \{p\})$ 
    using  $\langle r1 > 0 \rangle \langle r2 > 0 \rangle \text{finite\_imp\_not\_open}$ 
    unfolding  $rr\_def$  by fastforce
  ultimately have  $\text{infinite } \{w \in S \cap \text{ball } p \text{ } rr - \{p\}. f w = 0\}$  using infinite_super by blast
  then have  $\text{infinite } pz$ 
    unfolding  $pz\_def$  by (smt (verit) infinite_super Collect_mono_iff DiffE Int_iff)
  then show False using  $\langle \text{finite } pz \rangle$  by auto
qed
ultimately obtain  $r$  where  $pp \text{ } p \neq 0$  and  $r : r > 0 \text{ } pp \text{ holomorphic\_on } cball$ 
 $p \text{ } r$ 
   $(\forall w \in cball \text{ } p \text{ } r - \{p\}. f w = pp \text{ } w * (w - p) \text{ powi } po \wedge pp \text{ } w \neq 0)$ 
  using zorder_exist [of f p, folded po_def pp_def] by auto
  define  $r1$  where  $r1 = \min r \text{ } e1 / 2$ 
  have  $r1 < e1$  unfolding  $r1\_def$  using  $\langle e1 > 0 \rangle \langle r > 0 \rangle$  by auto
  moreover have  $r1 > 0 \text{ } pp \text{ holomorphic\_on } cball \text{ } p \text{ } r1$ 
     $(\forall w \in cball \text{ } p \text{ } r1 - \{p\}. f w = pp \text{ } w * (w - p) \text{ powi } po \wedge pp \text{ } w \neq 0)$ 
    unfolding  $r1\_def$  using  $\langle e1 > 0 \rangle \text{ } r$  by auto
  ultimately show ?thesis using that  $\langle pp \text{ } p \neq 0 \rangle$  by auto

```



```

qed

define e2 where e2  $\equiv$  r/2
have e2>0 using <r>0> unfolding e2_def by auto
define anal where anal  $\equiv$   $\lambda w. \text{deriv } pp \ w * h \ w / pp \ w$ 
define prin where prin  $\equiv$   $\lambda w. po * h \ w / (w - p)$ 
have (( $\lambda w. \text{prin } w + anal \ w$ ) has_contour_integral c * po * h p) (circlepath
p e2)
proof (rule has_contour_integral_add[of _ _ _ 0,simplified])
  have ball p r  $\subseteq$  S
  using <r<e1> avoid_def ball_subset_cball e1_avoid by (simp add:
subset_eq)
  then have cball p e2  $\subseteq$  S
  using <r>0> unfolding e2_def by auto
  then have ( $\lambda w. po * h \ w$ ) holomorphic_on cball p e2
  using h_holo by (auto intro!: holomorphic_intros)
  then show (prin has_contour_integral c * po * h p) (circlepath p e2)
  using Cauchy_integral_circlepath_simple[folded c_def, of  $\lambda w. po * h \ w$ ]
  <e2>0>
  unfolding prin_def by (auto simp add: mult.assoc)
  have anal holomorphic_on ball p r unfolding anal_def
  using pp_holo h_holo pp_po <ball p r  $\subseteq$  S> <pp  $\neq$  0>
  by (auto intro!: holomorphic_intros)
  then show (anal has_contour_integral 0) (circlepath p e2)
  using e2_def <r>0>
  by (auto elim!: Cauchy_theorem_disc_simple)
qed
then have cont ff' p e2 unfolding cont_def po_def
proof (elim has_contour_integral_eq)
  fix w assume w  $\in$  path_image (circlepath p e2)
  then have w $\in$ ball p r and w $\neq$ p unfolding e2_def using <r>0> by auto
  define wp where wp  $\equiv$  w-p
  have wp $\neq$ 0 and pp w  $\neq$  0
  unfolding wp_def using <w $\neq$ p> <w $\in$ ball p r> pp_po by auto
  moreover have der_f':deriv f' w = po * pp w * (w-p) powi (po - 1) +
deriv pp w * (w-p) powi po
  proof (rule DERIV_imp_deriv)
    have (pp has_field_derivative (deriv pp w)) (at w)
    using DERIV_deriv_iff_has_field_derivative pp_holo <w $\neq$ p>
    by (meson open_ball <w  $\in$  ball p r> ball_subset_cball holomorphic_derivI
holomorphic_on_subset)
    then show (f' has_field_derivative of_int po * pp w * (w - p) powi (po
- 1)
+ deriv pp w * (w - p) powi po) (at w)
    unfolding f'_def using <w $\neq$ p>
    by (auto intro!: derivative_eq_intros DERIV_cong[OF has_field_derivative_powr_of_int])
  qed
ultimately show prin w + anal w = ff' w
  unfolding f'_def ff'_def prin_def anal_def

```

```

    apply (simp add: field_simps flip: wp_def)
    by (metis (no_types, lifting) mult.commute power_int_minus_mult)
qed
then have cont ff p e2 unfolding cont_def
proof (elim has_contour_integral_eq)
  fix w assume w ∈ path_image (circlepath p e2)
  then have w ∈ ball p r and w ≠ p unfolding e2_def using ‹r > 0› by auto
  have deriv f' w = deriv f w
  proof (rule complex_derivative_transform_within_open[where s = ball p r
    - {p}])
    show f' holomorphic_on ball p r - {p} unfolding f'_def using pp_holo
      by (auto intro!: holomorphic_intros)
    next
    have ball p e1 - {p} ⊆ S - poles
      using ball_subset_cball e1_avoid[unfolded avoid_def] unfolding pz_def
      by auto
    then have ball p r - {p} ⊆ S - poles
      using ‹r < e1› by force
    then show f holomorphic_on ball p r - {p} using f_holo
      by auto
    next
    show open (ball p r - {p}) by auto
    show w ∈ ball p r - {p} using ‹w ∈ ball p r› ‹w ≠ p› by auto
  next
    fix x assume x ∈ ball p r - {p}
    then show f' x = f x
      using pp_po unfolding f'_def by auto
  qed
  moreover have f' w = f w
    using ‹w ∈ ball p r› ball_subset_cball subset_iff pp_po ‹w ≠ p›
    unfolding f'_def by auto
  ultimately show ff' w = ff w
    unfolding ff'_def ff_def by simp
qed
moreover have cball p e2 ⊆ ball p e1
  using ‹0 < r› ‹r < e1› e2_def by auto
ultimately show ?thesis using ‹e2 > 0› by auto
qed
then obtain e2 where e2: p ∈ pz ⟶ e2 > 0 ∧ cball p e2 ⊆ ball p e1 ∧ cont ff
p e2
  by auto
define e4 where e4 ≡ if p ∈ pz then e2 else e1
have e4 > 0 using e2 ‹e1 > 0› unfolding e4_def by auto
moreover have avoid p e4 using e2 ‹e1 > 0› e1_avoid unfolding e4_def
avoid_def by auto
moreover have p ∈ pz ⟶ cont ff p e4
  by (auto simp add: e2 e4_def)
ultimately show ?thesis by auto
qed

```

```

then obtain get_e where get_e:  $\forall p \in S. \text{get\_e } p > 0 \wedge \text{avoid } p (\text{get\_e } p)$ 
   $\wedge (p \in pz \longrightarrow \text{cont\_ff } p (\text{get\_e } p))$ 
by metis
define ci where ci  $\equiv \lambda p. \text{contour\_integral } (\text{circlepath } p (\text{get\_e } p)) \text{ ff}$ 
define w where w  $\equiv \lambda p. \text{winding\_number } g \ p$ 
have contour_integral g ff =  $(\sum p \in pz. w \ p * ci \ p)$  unfolding ci_def w_def
proof (rule Cauchy_theorem_singularities[OF  $\langle \text{open } S \rangle \langle \text{connected } S \rangle$  finite _
 $\langle \text{valid\_path } g \rangle$  loop
  path_img homo])
  have open (S - pz) using open_Diff[OF _ finite_imp_closed[OF finite]]  $\langle \text{open } S \rangle$  by auto
  then show ff holomorphic_on S - pz unfolding ff_def using f_holo h_holo
    by (auto intro!: holomorphic_intros simp add: pz_def)
  next
    show  $\forall p \in S. 0 < \text{get\_e } p \wedge (\forall w \in \text{cball } p (\text{get\_e } p). w \in S \wedge (w \neq p \longrightarrow w \notin pz))$ 
    using get_e using avoid_def by blast
  qed
also have ... =  $(\sum p \in pz. c * w \ p * h \ p * \text{zorder } f \ p)$ 
proof (rule sum.cong[of pz pz, simplified])
  fix p assume p  $\in$  pz
  show w p * ci p = c * w p * h p * (zorder f p)
  proof (cases p  $\in$  S)
    assume p  $\in$  S
    have ci p = c * h p * (zorder f p)
      unfolding ci_def
    using  $\langle p \in S \rangle \langle p \in pz \rangle$  cont_def contour_integral_unique get_e by fastforce
    thus ?thesis by auto
  next
    assume p  $\notin$  S
    then have w p = 0 using homo unfolding w_def by auto
    then show ?thesis by auto
  qed
qed
also have ... = c *  $(\sum p \in pz. w \ p * h \ p * \text{zorder } f \ p)$ 
  unfolding sum_distrib_left by (simp add: algebra_simps)
  finally have contour_integral g ff = c *  $(\sum p \in pz. w \ p * h \ p * \text{of\_int } (\text{zorder } f \ p))$  .
  then show ?thesis unfolding ff_def c_def w_def by simp
qed

```

8.3 Coefficient asymptotics for generating functions

For a formal power series that has a meromorphic continuation on some disc in the complex plane, we can use the Residue Theorem to extract precise asymptotic information from the residues at the poles. This can be used to derive the asymptotic behaviour of the coefficients ($a_n \sim \dots$). If the function is meromorphic on the entire complex plane, one can even derive a

full asymptotic expansion.

We will first show the relationship between the coefficients and the sum over the residues with an explicit remainder term (the contour integral along the circle used in the Residue theorem).

theorem

fixes $f :: \text{complex} \Rightarrow \text{complex}$ **and** $n :: \text{nat}$ **and** $r :: \text{real}$
defines $g \equiv (\lambda w. f w / w \wedge \text{Suc } n)$ **and** $\gamma \equiv \text{circlepath } 0 \ r$
assumes $\text{open } A \text{ connected } A \text{ cball } 0 \ r \subseteq A \ r > 0$
assumes $f \text{ holomorphic_on } A - S \ S \subseteq \text{ball } 0 \ r \text{ finite } S \ 0 \notin S$
shows $\text{fps_coeff_conv_residues:}$
 $(\text{deriv } \wedge^n) f \ 0 / \text{fact } n =$
 $\text{contour_integral } \gamma \ g / (2 * \pi * i) - (\sum z \in S. \text{residue } g \ z) \text{ (is ?thesis1)}$
and $\text{fps_coeff_residues_bound:}$
 $(\bigwedge z. \text{norm } z = r \implies z \notin k \implies \text{norm } (f z) \leq C) \implies C \geq 0 \implies \text{finite}$
 $k \implies$
 $\text{norm } ((\text{deriv } \wedge^n) f \ 0 / \text{fact } n + (\sum z \in S. \text{residue } g \ z)) \leq C / r \wedge n$

proof –

have $\text{holo: } g \text{ holomorphic_on } A - \text{insert } 0 \ S$
unfolding g_def **using** assms **by** $(\text{auto intro! : holomorphic_intros})$
have $\text{contour_integral } \gamma \ g = 2 * \pi * i * (\sum z \in \text{insert } 0 \ S. \text{winding_number } \gamma \ z * \text{residue } g \ z)$
proof $(\text{rule Residue_theorem})$
show $g \text{ holomorphic_on } A - \text{insert } 0 \ S$ **by** fact
from assms **show** $\forall z. z \notin A \longrightarrow \text{winding_number } \gamma \ z = 0$
unfolding γ_def **by** $(\text{intro allI impI winding_number_zero_outside[of cball } 0 \ r]) \text{ auto}$
qed $(\text{insert assms, auto simp: } \gamma_def)$
also have $\text{winding_number } \gamma \ z = 1$ **if** $z \in \text{insert } 0 \ S$ **for** z
unfolding γ_def **using** assms **that by** $(\text{intro winding_number_circlepath}) \text{ auto}$
hence $(\sum z \in \text{insert } 0 \ S. \text{winding_number } \gamma \ z * \text{residue } g \ z) = (\sum z \in \text{insert } 0 \ S. \text{residue } g \ z)$
by $(\text{intro sum.cong}) \text{ simp_all}$
also have $\dots = \text{residue } g \ 0 + (\sum z \in S. \text{residue } g \ z)$
using $\langle 0 \notin S \rangle$ **and** $\langle \text{finite } S \rangle$ **by** $(\text{subst sum.insert}) \text{ auto}$
also from $\langle r > 0 \rangle$ **have** $0 \in \text{cball } 0 \ r$ **by** simp
with assms **have** $0 \in A - S$ **by** blast
with assms **have** $\text{residue } g \ 0 = (\text{deriv } \wedge^n) f \ 0 / \text{fact } n$
unfolding g_def **by** $(\text{subst residue_holomorphic_over_power'[of } A - S])$
 $(\text{auto simp: finite_imp_closed})$
finally show $?thesis1$
by $(\text{simp add: field_sims})$

assume $C: \bigwedge z. \text{norm } z = r \implies z \notin k \implies \text{norm } (f z) \leq C \ C \geq 0$ **and** $k: \text{finite } k$

have $(\text{deriv } \wedge^n) f \ 0 / \text{fact } n + (\sum z \in S. \text{residue } g \ z) = \text{contour_integral } \gamma \ g / (2 * \pi * i)$
using $\langle ?thesis1 \rangle$ **by** $(\text{simp add: algebra_sims})$
also have $\text{norm } \dots = \text{norm } (\text{contour_integral } \gamma \ g) / (2 * \pi)$

```

  by (simp add: norm_divide norm_mult)
  also have norm (contour_integral  $\gamma$   $g$ )  $\leq C / r^{\text{Suc } n} * (2 * \pi * r)$ 
  proof (rule has_contour_integral_bound_circlepath_strong)
    from ‹open  $A$ › and ‹finite  $S$ › have open ( $A - \text{insert } 0 S$ )
    by (blast intro: finite_imp_closed)
    with assms show (g has_contour_integral contour_integral  $\gamma$   $g$ ) (circlepath 0
  r)
    unfolding  $\gamma\_def$ 
    by (intro has_contour_integral_integral contour_integrable_holomorphic_simple
  [OF holo]) auto
  next
    fix z assume z: norm ( $z - 0$ ) =  $r$  z  $\notin k$ 
    hence norm ( $g z$ ) = norm ( $f z$ ) /  $r^{\text{Suc } n}$ 
    by (simp add: norm_divide g_def norm_mult norm_power)
    also have ...  $\leq C / r^{\text{Suc } n}$ 
    using k and ‹ $r > 0$ › and z by (intro divide_right_mono C zero_le_power)
  auto
    finally show norm ( $g z$ )  $\leq C / r^{\text{Suc } n}$  .
  qed (insert C(2) k ‹ $r > 0$ ›, auto)
  also from ‹ $r > 0$ › have  $C / r^{\text{Suc } n} * (2 * \pi * r) / (2 * \pi) = C / r^n$ 
  by simp
  finally show norm ((deriv  $\sim^n$ )  $f$  0 / fact  $n + (\sum_{z \in S}. \text{residue } g z)) \leq \dots$ 
  by - (simp_all add: divide_right_mono)
qed

```

Since the circle is fixed, we can get an upper bound on the values of the generating function on the circle and therefore show that the integral is $O(r^{-n})$.

corollary *fps_coeff_residues_bigo*:

```

  fixes f :: complex  $\Rightarrow$  complex and r :: real
  assumes open A connected A cball 0 r  $\subseteq$  A r > 0
  assumes f_holomorphic_on A - S S  $\subseteq$  ball 0 r finite S 0  $\notin$  S
  assumes g: eventually ( $\lambda n. g n = -(\sum_{z \in S}. \text{residue } (\lambda z. f z / z^{\text{Suc } n}) z)$ )
  sequentially
    (is eventually ( $\lambda n. \_ = -?g' n$ )  $\_$ )
    shows ( $\lambda n. (\text{deriv } \sim^n) f 0 / \text{fact } n - g n$ )  $\in O(\lambda n. 1 / r^n)$  (is ( $\lambda n. ?c n$ 
  -  $\_$ )  $\in O(\_)$ )
  proof -
    from assms have compact (f ` sphere 0 r)
    by (intro compact_continuous_image holomorphic_on_imp_continuous_on
  holomorphic_on_subset[OF ‹f holomorphic_on A - S›]) auto
    hence bounded (f ` sphere 0 r) by (rule compact_imp_bounded)
    then obtain C where C:  $\bigwedge z. z \in \text{sphere } 0 r \implies \text{norm } (f z) \leq C$ 
    by (auto simp: bounded_iff sphere_def)
    have 0  $\leq$  norm (f (of_real r)) by simp
    also from C[of of_real r] and ‹r > 0› have ...  $\leq C$  by simp
    finally have C_nonneg:  $C \geq 0$  .

    have ( $\lambda n. ?c n + ?g' n$ )  $\in O(\lambda n. \text{of\_real } (1 / r^n))$ 

```

```

proof (intro bigoI[of _ C] always_eventually allI )
  fix n :: nat
  from assms and C and C_nonneg have norm (?c n + ?g' n) ≤ C / r ^ n
    by (intro fps_coeff_residues_bound[where A = A and k = {}]) auto
  also have ... = C * norm (complex_of_real (1 / r ^ n))
    using ⟨r > 0⟩ by (simp add: norm_divide norm_power)
  finally show norm (?c n + ?g' n) ≤ ... .
qed
also have ?this ⟷ (λn. ?c n - g n) ∈ O(λn. of_real (1 / r ^ n))
  by (intro landau_o.big.in_cong eventually_mono[OF g]) simp_all
  finally show ?thesis .
qed

corollary fps_coeff_residues_bigo':
  fixes f :: complex ⇒ complex and r :: real
  assumes exp: f has_fps_expansion F
  assumes open A connected A cball 0 r ⊆ A r > 0
  assumes f_holomorphic_on A - S S ⊆ ball 0 r finite S 0 ∉ S
  assumes eventually (λn. g n = -(∑ z∈S. residue (λz. f z / z ^ Suc n) z))
    sequentially
    (is eventually (λn. _ = -?g' n) _)
  shows (λn. fps_nth F n - g n) ∈ O(λn. 1 / r ^ n) (is (λn. ?c n - _) ∈
    O(_))
proof -
  have fps_nth F = (λn. (deriv ~ n) f 0 / fact n)
    using fps_nth_fps_expansion[OF exp] by (intro ext) simp_all
  with fps_coeff_residues_bigo[OF assms(2-)] show ?thesis by simp
qed

```

8.4 Rouché's theorem

```

theorem Rouché_theorem:
  fixes f g :: complex ⇒ complex and s :: complex set
  defines fg ≡ (λp. f p + g p)
  defines zeros_fg ≡ {p ∈ s. fg p = 0} and zeros_f ≡ {p ∈ s. f p = 0}
  assumes
    open s and connected s and
    finite zeros_fg and
    finite zeros_f and
    f_holo: f holomorphic_on s and
    g_holo: g holomorphic_on s and
    valid_path γ and
    loop: pathfinish γ = pathstart γ and
    path_img: path_image γ ⊆ s and
    path_less: ∀ z ∈ path_image γ. cmod(f z) > cmod(g z) and
    homo: ∀ z. (z ∉ s) ⟶ winding_number γ z = 0
  shows (∑ p ∈ zeros_fg. winding_number γ p * zorder fg p)
    = (∑ p ∈ zeros_f. winding_number γ p * zorder f p)
proof -

```

```

have path_fg:path_image  $\gamma \subseteq s - \text{zeros\_fg}$ 
proof -
  have False when  $z \in \text{path\_image } \gamma$  and  $f z + g z = 0$  for  $z$ 
  proof -
    have  $\text{cmod } (f z) > \text{cmod } (g z)$  using  $\langle z \in \text{path\_image } \gamma \rangle \text{ path\_less}$  by auto
    moreover have  $f z = - g z$  using  $\langle f z + g z = 0 \rangle$  by (simp add:
eq_neg_iff_add_eq_0)
    then have  $\text{cmod } (f z) = \text{cmod } (g z)$  by auto
    ultimately show False by auto
  qed
  then show ?thesis unfolding zeros_fg_def fg_def using path_img by auto
qed
have path_f:path_image  $\gamma \subseteq s - \text{zeros\_f}$ 
proof -
  have False when  $z \in \text{path\_image } \gamma$  and  $f z = 0$  for  $z$ 
  proof -
    have  $\text{cmod } (g z) < \text{cmod } (f z)$  using  $\langle z \in \text{path\_image } \gamma \rangle \text{ path\_less}$  by auto
    then have  $\text{cmod } (g z) < 0$  using  $\langle f z = 0 \rangle$  by auto
    then show False by auto
  qed
  then show ?thesis unfolding zeros_f_def using path_img by auto
qed
define w where  $w \equiv \lambda p. \text{winding\_number } \gamma p$ 
define c where  $c \equiv 2 * \text{complex\_of\_real } \pi * i$ 
define h where  $h \equiv \lambda p. g p / f p + 1$ 
obtain spikes
  where finite_spikes and spikes:  $\forall x \in \{0..1\} - \text{spikes}. \gamma \text{ differentiable at } x$ 
  using  $\langle \text{valid\_path } \gamma \rangle$ 
by (auto simp: valid_path_def piecewise_C1_differentiable_on_def C1_differentiable_on_eq)
have h_contour:  $((\lambda x. \text{deriv } h x / h x) \text{ has\_contour\_integral } 0) \gamma$ 
proof -
  have outside_img:  $0 \in \text{outside } (\text{path\_image } (h \circ \gamma))$ 
  proof -
    have  $h p \in \text{ball } 1 1$  when  $p \in \text{path\_image } \gamma$  for  $p$ 
    proof -
      have  $\text{cmod } (g p / f p) < 1$ 
      by (smt (verit) divide_less_eq_1_pos norm_divide norm_ge_zero
path_less that)
    then show ?thesis
      unfolding h_def by (auto simp add: dist_complex_def)
    qed
    then have  $\text{path\_image } (h \circ \gamma) \subseteq \text{ball } 1 1$ 
    by (simp add: image_subset_iff path_image_compose)
    moreover have  $(0::\text{complex}) \notin \text{ball } 1 1$  by (simp add: dist_norm)
    ultimately show ?thesis
      using convex_in_outside[of ball 1 1 0] outside_mono by blast
  qed
  have valid_h:  $\text{valid\_path } (h \circ \gamma)$ 
proof (rule valid_path_compose_holomorphic[OF  $\langle \text{valid\_path } \gamma \rangle$  _ path_f])

```

```

    show  $h$  holomorphic_on  $s - \text{zeros\_f}$ 
      unfolding  $h\_def$  using  $f\_holo$   $g\_holo$ 
      by (auto intro!: holomorphic_intros simp add: zeros_f_def)
  next
    show open  $(s - \text{zeros\_f})$  using  $\langle \text{finite zeros\_f} \rangle$   $\langle \text{open } s \rangle$  finite_imp_closed
      by auto
  qed
  have  $((\lambda z. 1/z)$  has_contour_integral 0)  $(h \circ \gamma)$ 
  proof -
    have  $0 \notin \text{path\_image } (h \circ \gamma)$  using outside_img by (simp add: outside_def)
    then have  $((\lambda z. 1/z)$  has_contour_integral  $c * \text{winding\_number } (h \circ \gamma)$  0)
       $(h \circ \gamma)$ 
      using has_contour_integral_winding_number[of  $h \circ \gamma$  0, simplified] valid_h
      unfolding  $c\_def$  by auto
    moreover have  $\text{winding\_number } (h \circ \gamma)$  0 = 0
  proof -
    have  $0 \in \text{outside } (\text{path\_image } (h \circ \gamma))$  using outside_img .
    moreover have  $\text{path } (h \circ \gamma)$ 
      using valid_h by (simp add: valid_path_imp_path)
    moreover have  $\text{path\_finish } (h \circ \gamma) = \text{path\_start } (h \circ \gamma)$ 
      by (simp add: loop_pathfinish_compose_pathstart_compose)
    ultimately show ?thesis using winding_number_zero_in_outside by auto
  qed
  ultimately show ?thesis by auto
  qed
  moreover have  $\text{vector\_derivative } (h \circ \gamma)$  (at  $x$ ) =  $\text{vector\_derivative } \gamma$  (at  $x$ )
  * deriv  $h$   $(\gamma x)$ 
    when  $x \in \{0..1\} - \text{spikes}$  for  $x$ 
  proof (rule vector_derivative_chain_at_general)
    show  $\gamma$  differentiable at  $x$  using that  $\langle \text{valid\_path } \gamma \rangle$  spikes by auto
  next
    define  $der$  where  $der \equiv \lambda p. (\text{deriv } g p * f p - g p * \text{deriv } f p) / (f p * f p)$ 
    define  $t$  where  $t \equiv \gamma x$ 
    have  $f t \neq 0$  unfolding zeros_f_def  $t\_def$ 
      by (metis DiffD1 image_eqI norm_not_less_zero norm_zero path_defs(4)
        path_less that)
    moreover have  $t \in s$ 
      using contra_subsetD  $\text{path\_image\_def}$   $\text{path\_fg } t\_def$  that by fastforce
    ultimately have  $(h$  has_field_derivative  $der t)$  (at  $t$ )
      unfolding  $h\_def$   $der\_def$  using  $g\_holo$   $f\_holo$   $\langle \text{open } s \rangle$ 
      by (auto intro!: holomorphic_derivI derivative_eq_intros)
    then show  $h$  field_differentiable at  $(\gamma x)$ 
      unfolding  $t\_def$  field_differentiable_def by blast
  qed
  then have  $((/)$  1 has_contour_integral 0)  $(h \circ \gamma)$ 
    =  $((\lambda x. \text{deriv } h x / h x)$  has_contour_integral 0)  $\gamma$ 
    unfolding has_contour_integral
  by (force intro!: has_integral_spike_eq[OF negligible_finite, OF  $\langle \text{finite spikes} \rangle$ ])
  ultimately show ?thesis by auto

```



```

qed
then have contour_integral  $\gamma$  ( $\lambda x. \text{deriv } h \ x / h \ x$ ) = 0
  using contour_integral_unique by simp
moreover have contour_integral  $\gamma$  ( $\lambda x. \text{deriv } fg \ x / fg \ x$ ) = contour_integral  $\gamma$ 
( $\lambda x. \text{deriv } f \ x / f \ x$ )
  + contour_integral  $\gamma$  ( $\lambda p. \text{deriv } h \ p / h \ p$ )
proof -
  have ( $\lambda p. \text{deriv } f \ p / f \ p$ ) contour_integrable_on  $\gamma$ 
  proof (rule contour_integrable_holomorphic_simple[OF _ _ <valid_path  $\gamma$ >
path_f])
    show open (s - zeros_f)
      using finite_imp_closed[OF <finite zeros_f>] <open s> by auto
    then show ( $\lambda p. \text{deriv } f \ p / f \ p$ ) holomorphic_on s - zeros_f
      using f_holo
      by (auto intro!: holomorphic_intros simp add:zeros_f_def)
  qed
  moreover have ( $\lambda p. \text{deriv } h \ p / h \ p$ ) contour_integrable_on  $\gamma$ 
    using h_contour
    by (simp add: has_contour_integral_integrable)
  ultimately have contour_integral  $\gamma$  ( $\lambda x. \text{deriv } f \ x / f \ x + \text{deriv } h \ x / h \ x$ ) =
    contour_integral  $\gamma$  ( $\lambda p. \text{deriv } f \ p / f \ p$ ) + contour_integral  $\gamma$ 
( $\lambda p. \text{deriv } h \ p / h \ p$ )
    using contour_integral_add[of ( $\lambda p. \text{deriv } f \ p / f \ p$ )  $\gamma$  ( $\lambda p. \text{deriv } h \ p / h \ p$ )]
    by auto
  moreover have deriv fg p / fg p = deriv f p / f p + deriv h p / h p
    when p  $\in$  path_image  $\gamma$  for p
  proof -
    have fg p  $\neq$  0 and f p  $\neq$  0
      using path_f path_fg that unfolding zeros_f_def zeros_fg_def by auto
    have h p  $\neq$  0
    proof (rule ccontr)
      assume  $\neg h \ p \neq 0$ 
      then have cmod (g p / f p) = 1
        by (simp add: add_eq_0_iff2 h_def)
      then show False
        by (smt (verit) divide_eq_1_iff norm_divide path_less that)
    qed
    have der_fg: deriv fg p = deriv f p + deriv g p unfolding fg_def
      using f_holo g_holo holomorphic_on_imp_differentiable_at[OF _ <open
s>] path_img that
      by auto
    have der_h: deriv h p = (deriv g p * f p - g p * deriv f p) / (f p * f p)
    proof -
      define der where der  $\equiv$   $\lambda p. (\text{deriv } g \ p * f \ p - g \ p * \text{deriv } f \ p) / (f \ p * f \ p)$ 
      have p  $\in$  s using path_img that by auto
      then have (h has_field_derivative der p) (at p)
        unfolding h_def der_def using g_holo f_holo <open s> <f p  $\neq$  0>
        by (auto intro!: derivative_eq_intros holomorphic_derivI)
      then show ?thesis unfolding der_def using DERIV_imp_deriv by auto
    qed
  qed

```

```

qed
show ?thesis
  using ⟨h p ≠ 0⟩ ⟨f p ≠ 0⟩ ⟨fg p ≠ 0⟩
  unfolding der_fg der_h
  apply (simp add: divide_simps h_def fg_def)
  by (simp add: mult.commute mult.left_commute ring_class.ring_distrib(1))
qed
then have contour_integral γ (λp. deriv fg p / fg p)
      = contour_integral γ (λp. deriv f p / f p + deriv h p / h p)
  by (elim contour_integral_eq)
ultimately show ?thesis by auto
qed
moreover have contour_integral γ (λx. deriv fg x / fg x) = c * (∑ p ∈ zeros_fg.
w p * zorder fg p)
proof -
  have fg holomorphic_on s
    unfolding fg_def using f_holo g_holo holomorphic_on_add by auto
  moreover
  have path_image γ ⊆ s - {p ∈ s. fg p = 0}
    using path_fg unfolding zeros_fg_def .
  moreover
  have finite {p ∈ s. fg p = 0}
    using ⟨finite zeros_fg⟩ unfolding zeros_fg_def .
  ultimately show ?thesis
    unfolding c_def zeros_fg_def w_def
    using argument_principle[OF ⟨open s⟩ ⟨connected s⟩ _ _ ⟨valid_path γ⟩ loop
_ homo, of _ {} λ_. 1]
    by simp
qed
moreover have contour_integral γ (λx. deriv f x / f x) = c * (∑ p ∈ zeros_f. w
p * zorder f p)
  unfolding c_def zeros_f_def w_def
proof (rule argument_principle[OF ⟨open s⟩ ⟨connected s⟩ _ _ ⟨valid_path γ⟩
loop _ homo
, of _ {} λ_. 1, simplified])
  show f holomorphic_on s
    using f_holo g_holo holomorphic_on_add by auto
  show path_image γ ⊆ s - {p ∈ s. f p = 0}
    using path_f unfolding zeros_f_def .
  show finite {p ∈ s. f p = 0}
    using ⟨finite zeros_f⟩ unfolding zeros_f_def .
qed
ultimately have c * (∑ p ∈ zeros_fg. w p * (zorder fg p)) = c * (∑ p ∈ zeros_f.
w p * (zorder f p))
  by auto
then show ?thesis unfolding c_def using w_def by auto
qed
end

```

```

theory Laurent_Convergence
imports HOL-Computational_Algebra.Formal_Laurent_Series HOL-Library.Landau_Symbols
        Residue_Theorem

begin

definition fls_conv_radius :: complex fls  $\Rightarrow$  ereal where
  fls_conv_radius f = fps_conv_radius (fls_regpart f)

definition eval_fls :: complex fls  $\Rightarrow$  complex  $\Rightarrow$  complex where
  eval_fls F z = eval_fps (fls_base_factor_to_fps F) z * z powi fls_subdegree F

definition
  has_laurent_expansion :: (complex  $\Rightarrow$  complex)  $\Rightarrow$  complex fls  $\Rightarrow$  bool
  (infixl <has' laurent' expansion> 60)
where (f has_laurent_expansion F)  $\longleftrightarrow$ 
  fls_conv_radius F > 0  $\wedge$  eventually ( $\lambda z.$  eval_fls F z = f z) (at 0)

lemma has_laurent_expansion_schematicI:
  f has_laurent_expansion F  $\Longrightarrow$  F = G  $\Longrightarrow$  f has_laurent_expansion G
by simp

lemma has_laurent_expansion_cong:
  assumes eventually ( $\lambda x.$  f x = g x) (at 0) F = G
  shows (f has_laurent_expansion F)  $\longleftrightarrow$  (g has_laurent_expansion G)
proof -
  have eventually ( $\lambda z.$  eval_fls F z = g z) (at 0)
  if eventually ( $\lambda z.$  eval_fls F z = f z) (at 0) eventually ( $\lambda x.$  f x = g x) (at 0)
for f g
  using that by eventually_elim auto
  from this[of f g] this[of g f] show ?thesis
  using assms by (auto simp: eq_commute has_laurent_expansion_def)
qed

lemma has_laurent_expansion_cong':
  assumes eventually ( $\lambda x.$  f x = g x) (at z) F = G z = z'
  shows (( $\lambda x.$  f (z + x)) has_laurent_expansion F)  $\longleftrightarrow$  (( $\lambda x.$  g (z' + x))
  has_laurent_expansion G)
  by (intro has_laurent_expansion_cong)
  (use assms in <auto simp: at_to_0' eventually_filtermap add_ac>)

lemma fls_conv_radius_altdef:
  fls_conv_radius F = fps_conv_radius (fls_base_factor_to_fps F)
proof -
  have conv_radius ( $\lambda n.$  fls_nth F (int n)) = conv_radius ( $\lambda n.$  fls_nth F (int n
  + fls_subdegree F))
  proof (cases fls_subdegree F  $\geq$  0)
  case True

```

```

hence conv_radius ( $\lambda n. \text{fls\_nth } F \text{ (int } n + \text{fls\_subdegree } F)$ ) =
  conv_radius ( $\lambda n. \text{fls\_nth } F \text{ (int (n + nat (fls\_subdegree } F))$ )))
by auto
thus ?thesis
by (subst (asm) conv_radius_shift) auto
next
case False
hence conv_radius ( $\lambda n. \text{fls\_nth } F \text{ (int } n)$ ) =
  conv_radius ( $\lambda n. \text{fls\_nth } F \text{ (fls\_subdegree } F + \text{int (n + nat (-fls\_subdegree } F))$ )))
by auto
thus ?thesis
by (subst (asm) conv_radius_shift) (auto simp: add_ac)
qed
thus ?thesis
by (simp add: fls_conv_radius_def fps_conv_radius_def)
qed

```

```

lemma eval_fps_of_nat [simp]: eval_fps (of_nat n) z = of_nat n
and eval_fps_of_int [simp]: eval_fps (of_int m) z = of_int m
by (simp_all flip: fps_of_nat fps_of_int)

```

```

lemma fps_conv_radius_of_nat [simp]: fps_conv_radius (of_nat n) =  $\infty$ 
and fps_conv_radius_of_int [simp]: fps_conv_radius (of_int m) =  $\infty$ 
by (simp_all flip: fps_of_nat fps_of_int)

```

```

lemma fps_conv_radius_fls_regpart: fps_conv_radius (fls_regpart F) = fls_conv_radius
F
by (simp add: fls_conv_radius_def)

```

```

lemma fls_conv_radius_0 [simp]: fls_conv_radius 0 =  $\infty$ 
and fls_conv_radius_1 [simp]: fls_conv_radius 1 =  $\infty$ 
and fls_conv_radius_const [simp]: fls_conv_radius (fls_const c) =  $\infty$ 
and fls_conv_radius_numeral [simp]: fls_conv_radius (numeral num) =  $\infty$ 
and fls_conv_radius_of_nat [simp]: fls_conv_radius (of_nat n) =  $\infty$ 
and fls_conv_radius_of_int [simp]: fls_conv_radius (of_int m) =  $\infty$ 
and fls_conv_radius_X [simp]: fls_conv_radius fls_X =  $\infty$ 
and fls_conv_radius_X_inv [simp]: fls_conv_radius fls_X_inv =  $\infty$ 
and fls_conv_radius_X_intpow [simp]: fls_conv_radius (fls_X_intpow m) =
 $\infty$ 
by (simp_all add: fls_conv_radius_def fls_X_intpow_regpart)

```

```

lemma fls_conv_radius_shift [simp]: fls_conv_radius (fls_shift n F) = fls_conv_radius
F
unfolding fls_conv_radius_altdef by (subst fls_base_factor_to_fps_shift) (rule
refl)

```

```

lemma fls_conv_radius_fps_to_fls [simp]: fls_conv_radius (fps_to_fls F) = fps_conv_radius
F

```

```

    by (simp add: fls_conv_radius_def)

lemma fls_conv_radius_deriv [simp]: fls_conv_radius (fls_deriv F) ≥ fls_conv_radius
F
proof -
  have fls_conv_radius (fls_deriv F) = fps_conv_radius (fls_regpart (fls_deriv
F))
    by (simp add: fls_conv_radius_def)
  also have fls_regpart (fls_deriv F) = fps_deriv (fls_regpart F)
    by (intro fps_ext) (auto simp: add_ac)
  also have fps_conv_radius ... ≥ fls_conv_radius F
    by (simp add: fls_conv_radius_def fps_conv_radius_deriv)
  finally show ?thesis .
qed

lemma fls_conv_radius_uminus [simp]: fls_conv_radius (-F) = fls_conv_radius
F
  by (simp add: fls_conv_radius_def)

lemma fls_conv_radius_add: fls_conv_radius (F + G) ≥ min (fls_conv_radius
F) (fls_conv_radius G)
  by (simp add: fls_conv_radius_def fps_conv_radius_add)

lemma fls_conv_radius_diff: fls_conv_radius (F - G) ≥ min (fls_conv_radius
F) (fls_conv_radius G)
  by (simp add: fls_conv_radius_def fps_conv_radius_diff)

lemma fls_conv_radius_mult: fls_conv_radius (F * G) ≥ min (fls_conv_radius
F) (fls_conv_radius G)
proof (cases F = 0 ∨ G = 0)
  case False
    hence [simp]: F ≠ 0 G ≠ 0
      by auto
    have fls_conv_radius (F * G) = fps_conv_radius (fls_regpart (fls_shift (fls_subdegree
F + fls_subdegree G) (F * G)))
      by (simp add: fls_conv_radius_altdef)
    also have fls_regpart (fls_shift (fls_subdegree F + fls_subdegree G) (F * G))
      =
        fls_base_factor_to_fps F * fls_base_factor_to_fps G
      by (simp add: fls_times_def)
    also have fps_conv_radius ... ≥ min (fls_conv_radius F) (fls_conv_radius G)
      unfolding fls_conv_radius_altdef by (rule fps_conv_radius_mult)
    finally show ?thesis .
qed auto

lemma fps_conv_radius_add_ge:
  fps_conv_radius F ≥ r ⟹ fps_conv_radius G ≥ r ⟹ fps_conv_radius (F +
G) ≥ r
  using fps_conv_radius_add[of F G] by (simp add: min_def split: if_splits)

```

lemma *fps_conv_radius_diff_ge*:

fps_conv_radius F ≥ r ⇒ fps_conv_radius G ≥ r ⇒ fps_conv_radius (F - G) ≥ r
using *fps_conv_radius_diff*[of F G] **by** (*simp add: min_def split: if_splits*)

lemma *fps_conv_radius_mult_ge*:

*fps_conv_radius F ≥ r ⇒ fps_conv_radius G ≥ r ⇒ fps_conv_radius (F * G) ≥ r*
using *fps_conv_radius_mult*[of F G] **by** (*simp add: min_def split: if_splits*)

lemma *fls_conv_radius_add_ge*:

fls_conv_radius F ≥ r ⇒ fls_conv_radius G ≥ r ⇒ fls_conv_radius (F + G) ≥ r
using *fls_conv_radius_add*[of F G] **by** (*simp add: min_def split: if_splits*)

lemma *fls_conv_radius_diff_ge*:

fls_conv_radius F ≥ r ⇒ fls_conv_radius G ≥ r ⇒ fls_conv_radius (F - G) ≥ r
using *fls_conv_radius_diff*[of F G] **by** (*simp add: min_def split: if_splits*)

lemma *fls_conv_radius_mult_ge*:

*fls_conv_radius F ≥ r ⇒ fls_conv_radius G ≥ r ⇒ fls_conv_radius (F * G) ≥ r*
using *fls_conv_radius_mult*[of F G] **by** (*simp add: min_def split: if_splits*)

lemma *fls_conv_radius_power*: *fls_conv_radius (F ^ n) ≥ fls_conv_radius F*
by (*induction n*) (*auto intro!: fls_conv_radius_mult_ge*)

lemma *eval_fls_0* [*simp*]: *eval_fls 0 z = 0*

and *eval_fls_1* [*simp*]: *eval_fls 1 z = 1*

and *eval_fls_const* [*simp*]: *eval_fls (fls_const c) z = c*

and *eval_fls_numeral* [*simp*]: *eval_fls (numeral num) z = numeral num*

and *eval_fls_of_nat* [*simp*]: *eval_fls (of_nat n) z = of_nat n*

and *eval_fls_of_int* [*simp*]: *eval_fls (of_int m) z = of_int m*

and *eval_fls_X* [*simp*]: *eval_fls fls_X z = z*

and *eval_fls_X_intpow* [*simp*]: *eval_fls (fls_X_intpow m) z = z powi m*

by (*simp_all add: eval_fls_def*)

lemma *eval_fls_at_0*: *eval_fls F 0 = (if fls_subdegree F ≥ 0 then fls_nth F 0 else 0)*

by (*cases fls_subdegree F = 0*)

(*simp_all add: eval_fls_def fls_regpart_def eval_fps_at_0*)

lemma *eval_fps_to_fls*:

assumes *norm z < fps_conv_radius F*

shows *eval_fls (fps_to_fls F) z = eval_fps F z*

proof (*cases F = 0*)

case [*simp*]: *False*

```

have eval_fps F z = eval_fps (unit_factor F * normalize F) z
  by (metis unit_factor_mult_normalize)
also have ... = eval_fps (unit_factor F * fps_X ^ subdegree F) z
  by simp
also have ... = eval_fps (unit_factor F) z * z ^ subdegree F
  using assms by (subst eval_fps_mult) auto
also have ... = eval_fls (fps_to_fls F) z
  unfolding eval_fls_def fls_base_factor_to_fps_to_fls fls_subdegree_fls_to_fps
    power_int_of_nat ..
finally show ?thesis ..
qed auto

```

```

lemma eval_fls_shift:
  assumes [simp]: z ≠ 0
  shows eval_fls (fls_shift n F) z = eval_fls F z * z powi -n
proof (cases F = 0)
  case [simp]: False
  show ?thesis
  unfolding eval_fls_def
  by (subst fls_base_factor_to_fps_shift, subst fls_shift_subdegree[OF ‹F ≠ 0›],
    subst power_int_diff)
    (auto simp: power_int_minus divide_simps)
qed auto

```

```

lemma eval_fls_add:
  assumes ereal (norm z) < fls_conv_radius F ereal (norm z) < fls_conv_radius
    G z ≠ 0
  shows eval_fls (F + G) z = eval_fls F z + eval_fls G z
  using assms
proof (induction fls_subdegree F fls_subdegree G arbitrary: F G rule: linorder_wlog)
  case (sym F G)
  show ?case
    using sym(1)[of G F] sym(2-) by (simp add: add_ac)
next
  case (le F G)
  show ?case
  proof (cases F = 0 ∨ G = 0)
    case False
    hence [simp]: F ≠ 0 G ≠ 0
      by auto
    note [simp] = ‹z ≠ 0›
    define F' G' where F' = fls_base_factor_to_fps F G' = fls_base_factor_to_fps
      G
    define m n where m = fls_subdegree F n = fls_subdegree G
    have m ≤ n
      using le by (auto simp: m_n_def)
    have conv1: ereal (cmod z) < fps_conv_radius F' ereal (cmod z) < fps_conv_radius
      G'
    using assms le by (simp_all add: F'_G'_def fls_conv_radius_altdef)

```

```

have conv2: ereal (cmod z) < fps_conv_radius (G' * fps_X ^ nat (n - m))
using conv1 by (intro less_le_trans[OF fps_conv_radius_mult]) auto
have conv3: ereal (cmod z) < fps_conv_radius (F' + G' * fps_X ^ nat (n - m))
using conv1 conv2 by (intro less_le_trans[OF fps_conv_radius_add])
auto

have eval_fls F z + eval_fls G z = eval_fps F' z * z powi m + eval_fps G' z
* z powi n
unfolding eval_fls_def m_n_def[symmetric] F'_G'_def[symmetric]
by (simp add: power_int_add algebra_simps)
also have ... = (eval_fps F' z + eval_fps G' z * z powi (n - m)) * z powi m
by (simp add: algebra_simps power_int_diff)
also have eval_fps G' z * z powi (n - m) = eval_fps (G' * fps_X ^ nat (n - m)) z
using assms ⟨m ≤ n⟩ conv1 by (subst eval_fps_mult) (auto simp: power_int_def)
also have eval_fps F' z + ... = eval_fps (F' + G' * fps_X ^ nat (n - m)) z
using conv1 conv2 by (subst eval_fps_add) auto
also have ... = eval_fls (fps_to_fls (F' + G' * fps_X ^ nat (n - m))) z
using conv3 by (subst eval_fps_to_fls) auto
also have ... * z powi m = eval_fls (fls_shift (-m) (fps_to_fls (F' + G' *
fps_X ^ nat (n - m)))) z
by (subst eval_fls_shift) auto
also have fls_shift (-m) (fps_to_fls (F' + G' * fps_X ^ nat (n - m))) = F
+ G
using ⟨m ≤ n⟩
by (simp add: fls_times_fps_to_fls fps_to_fls_power fls_X_power_conv_shift_1
fls_shifted_times_simps F'_G'_def m_n_def)
finally show ?thesis ..
qed auto
qed

```

lemma eval_fls_minus:

```

assumes ereal (norm z) < fls_conv_radius F
shows eval_fls (-F) z = -eval_fls F z
using assms by (simp add: eval_fls_def eval_fps_minus fls_conv_radius_altdef)

```

lemma eval_fls_diff:

```

assumes ereal (norm z) < fls_conv_radius F ereal (norm z) < fls_conv_radius G
and [simp]: z ≠ 0
shows eval_fls (F - G) z = eval_fls F z - eval_fls G z
proof -
have eval_fls (F + (-G)) z = eval_fls F z - eval_fls G z
using assms by (subst eval_fls_add) (auto simp: eval_fls_minus)
thus ?thesis
by simp
qed

```



```

lemma eval_fls_mult:
  assumes ereal (norm z) < fls_conv_radius F ereal (norm z) < fls_conv_radius
  G z ≠ 0
  shows eval_fls (F * G) z = eval_fls F z * eval_fls G z
proof (cases F = 0 ∨ G = 0)
  case False
  hence [simp]: F ≠ 0 G ≠ 0
  by auto
  note [simp] = ⟨z ≠ 0⟩
  define F' G' where F' = fls_base_factor_to_fps F G' = fls_base_factor_to_fps
  G
  define m n where m = fls_subdegree F n = fls_subdegree G
  have eval_fls F z * eval_fls G z = (eval_fps F' z * eval_fps G' z) * z powi (m
  + n)
  unfolding eval_fls_def m_n_def[symmetric] F'_G'_def[symmetric]
  by (simp add: power_int_add algebra_simps)
  also have ... = eval_fps (F' * G') z * z powi (m + n)
  using assms by (subst eval_fps_mult) (auto simp: F'_G'_def fls_conv_radius_altdef)
  also have ... = eval_fls (F * G) z
  by (simp add: eval_fls_def F'_G'_def m_n_def) (simp add: fls_times_def)
  finally show ?thesis ..
qed auto

```

```

lemma eval_fls_power:
  assumes ereal (norm z) < fls_conv_radius F z ≠ 0
  shows eval_fls (F ^ n) z = eval_fls F z ^ n
proof (induction n)
  case (Suc n)
  have eval_fls (F ^ Suc n) z = eval_fls (F * F ^ n) z
  by simp
  also have ... = eval_fls F z * eval_fls (F ^ n) z
  using assms by (subst eval_fls_mult) (auto intro!: less_le_trans[OF fls_conv_radius_power])
  finally show ?case
  using Suc by simp
qed auto

```

```

lemma eval_fls_eq:
  assumes N ≤ fls_subdegree F fls_subdegree F ≥ 0 ∨ z ≠ 0
  assumes (λn. fls_nth F (int n + N) * z powi (int n + N)) sums S
  shows eval_fls F z = S
proof (cases z = 0)
  case [simp]: True
  have (λn. fls_nth F (int n + N) * z powi (int n + N)) =
    (λn. if n ∈ (if N ≤ 0 then {nat (-N)} else {})) then fls_nth F (int n + N)
  else 0)
  by (auto simp: fun_eq_iff split: if_splits)
  also have ... sums (∑ n ∈ (if N ≤ 0 then {nat (-N)} else {}). fls_nth F (int n
  + N))
  by (rule sums_If_finite_set) auto

```

```

also have ... = fls_nth F 0
using assms by auto
also have ... = eval_fls F z
using assms by (auto simp: eval_fls_def eval_fps_at_0 power_int_0_left_if)
finally show ?thesis
using assms by (simp add: sums_iff)
next
case [simp]: False
define N' where N' = fls_subdegree F
define d where d = nat (N' - N)

have ( $\lambda n. \text{fls\_nth } F \text{ (int } n + N) * z^{\text{powi (int } n + N)}$ ) sums S
by fact
also have ?this  $\longleftrightarrow$  ( $\lambda n. \text{fls\_nth } F \text{ (int (n+d) + N) * z^{\text{powi (int (n+d) + N)}}$ ) sums S
by (rule sums_zero_iff_shift [symmetric]) (use assms in <auto simp: d_def N'_def>)
also have ( $\lambda n. \text{int (n+d) + N} = (\lambda n. \text{int } n + N')$ )
using assms by (auto simp: N'_def d_def)
finally have ( $\lambda n. \text{fls\_nth } F \text{ (int } n + N') * z^{\text{powi (int } n + N')}$ ) sums S .
hence ( $\lambda n. z^{\text{powi (-N')}} * (\text{fls\_nth } F \text{ (int } n + N') * z^{\text{powi (int } n + N')})$ )
sums ( $z^{\text{powi (-N')}} * S$ )
by (intro sums_mult)
hence ( $\lambda n. \text{fls\_nth } F \text{ (int } n + N') * z^n$ ) sums ( $z^{\text{powi (-N')}} * S$ )
by (simp add: power_int_add power_int_minus field_simps)
thus ?thesis
by (simp add: eval_fls_def eval_fps_def sums_iff power_int_minus N'_def)
qed

lemma norm_summable_fls:
 $\text{norm } z < \text{fls\_conv\_radius } f \implies \text{summable } (\lambda n. \text{norm } (\text{fls\_nth } f \text{ } n * z^n))$ 
using norm_summable_fps[of z fls_regpart f] by (simp add: fls_conv_radius_def)

lemma norm_summable_fls':
 $\text{norm } z < \text{fls\_conv\_radius } f \implies \text{summable } (\lambda n. \text{norm } (\text{fls\_nth } f \text{ (n + fls\_subdegree } f) * z^n))$ 
using norm_summable_fps[of z fls_base_factor_to_fps f] by (simp add: fls_conv_radius_altdef)

lemma summable_fls:
 $\text{norm } z < \text{fls\_conv\_radius } f \implies \text{summable } (\lambda n. \text{fls\_nth } f \text{ } n * z^n)$ 
by (rule summable_norm_cancel[OF norm_summable_fls])

theorem sums_eval_fls:
fixes f
defines n  $\equiv \text{fls\_subdegree } f$ 
assumes  $\text{norm } z < \text{fls\_conv\_radius } f$  and  $z \neq 0 \vee n \geq 0$ 
shows ( $\lambda k. \text{fls\_nth } f \text{ (int } k + n) * z^{\text{powi (int } k + n)}$ ) sums eval_fls f z
proof (cases z = 0)
case [simp]: False

```

```

have ( $\lambda k. \text{fps\_nth } (\text{fls\_base\_factor\_to\_fps } f) \ k * z^k * z^{\text{powi } n}$ ) sums
  ( $\text{eval\_fps } (\text{fls\_base\_factor\_to\_fps } f) \ z * z^{\text{powi } n}$ )
using assms(2) by (intro sums_eval_fps sums_mult2) (auto simp: fls_conv_radius_altdef)
thus ?thesis
  by (simp add: power_int_add n_def eval_fls_def mult_ac)
next
case [simp]: True
with assms have  $n \geq 0$ 
  by auto
have ( $\lambda k. \text{fls\_nth } f \ (int \ k + n) * z^{\text{powi } (int \ k + n)}$ ) sums
  ( $\sum_{k \in \{if \ n \leq 0 \text{ then } \{nat \ (-n)\} \text{ else } \{\}\}}. \text{fls\_nth } f \ (int \ k + n) * z^{\text{powi } (int \ k + n)}$ )
  by (intro sums_finite) (auto split: if_splits)
also have  $\dots = \text{eval\_fls } f \ z$ 
  using  $\langle n \geq 0 \rangle$  by (auto simp: eval_fls_at_0 n_def not_le)
finally show ?thesis .
qed

```

```

lemma holomorphic_on_eval_fls:
  fixes f
  defines  $n \equiv \text{fls\_subdegree } f$ 
  assumes  $A \subseteq \text{eball } 0 \ (\text{fls\_conv\_radius } f) - (if \ n \geq 0 \text{ then } \{\} \text{ else } \{0\})$ 
  shows  $\text{eval\_fls } f \text{ holomorphic\_on } A$ 
proof (cases  $n \geq 0$ )
case True
have  $\text{eval\_fls } f = (\lambda z. \text{eval\_fps } (\text{fls\_base\_factor\_to\_fps } f) \ z * z^{\text{nat } n})$ 
  using True by (simp add: fun_eq_iff eval_fls_def power_int_def n_def)
moreover have  $\dots \text{ holomorphic\_on } A$ 
  using True assms(2) by (intro holomorphic_intros) (auto simp: fls_conv_radius_altdef)
ultimately show ?thesis
  by simp
next
case False
show ?thesis using assms
  unfolding eval_fls_def by (intro holomorphic_intros) (auto simp: fls_conv_radius_altdef)
qed

```

```

lemma holomorphic_on_eval_fls' [holomorphic_intros]:
  assumes  $g \text{ holomorphic\_on } A$ 
  assumes  $g \ ' \ A \subseteq \text{eball } 0 \ (\text{fls\_conv\_radius } f) - (if \ \text{fls\_subdegree } f \geq 0 \text{ then } \{\} \text{ else } \{0\})$ 
  shows  $(\lambda x. \text{eval\_fls } f \ (g \ x)) \text{ holomorphic\_on } A$ 
  by (meson assms holomorphic_on_compose holomorphic_on_eval_fls holomorphic_transform o_def)

```

```

lemma continuous_on_eval_fls:
  fixes f
  defines  $n \equiv \text{fls\_subdegree } f$ 
  assumes  $A \subseteq \text{eball } 0 \ (\text{fls\_conv\_radius } f) - (if \ n \geq 0 \text{ then } \{\} \text{ else } \{0\})$ 

```

shows $\text{continuous_on } A \text{ (eval_fls } f)$
using $\text{assms holomorphic_on_eval_fls holomorphic_on_imp_continuous_on}$
by blast

lemma $\text{continuous_on_eval_fls' [continuous_intros]}$:
fixes f
defines $n \equiv \text{fls_subdegree } f$
assumes $g \text{ ' } A \subseteq \text{eball } 0 \text{ (fls_conv_radius } f) - (\text{if } n \geq 0 \text{ then } \{\} \text{ else } \{0\})$
assumes $\text{continuous_on } A \text{ } g$
shows $\text{continuous_on } A \text{ } (\lambda x. \text{eval_fls } f \text{ (} g \text{ } x))$
by $(\text{metis assms continuous_on_compose2 continuous_on_eval_fls order.refl})$

lemmas $\text{has_field_derivative_eval_fps' [derivative_intros]} =$
 $\text{DERIV_chain2[OF has_field_derivative_eval_fps]}$

lemma $\text{has_field_derivative_eval_fls}$:
assumes $z \in \text{eball } 0 \text{ (fls_conv_radius } f) - \{0\}$
shows $(\text{eval_fls } f \text{ has_field_derivative eval_fls (fls_deriv } f) \text{ } z) \text{ (at } z \text{ within } A)$
proof –
define g **where** $g = \text{fls_base_factor_to_fps } f$
define n **where** $n = \text{fls_subdegree } f$
have $[\text{simp}]: \text{fps_conv_radius } g = \text{fls_conv_radius } f$
by $(\text{simp add: fls_conv_radius_altdef } g_def)$
have $\text{conv1: fps_conv_radius (fps_deriv } g * \text{fps_X}) \geq \text{fls_conv_radius } f$
by $(\text{intro fps_conv_radius_mult_ge order.trans[OF fps_conv_radius_deriv]})$
 auto
have $\text{conv2: fps_conv_radius (of_int } n * g) \geq \text{fls_conv_radius } f$
by $(\text{intro fps_conv_radius_mult_ge}) \text{ auto}$
have $\text{conv3: fps_conv_radius (fps_deriv } g * \text{fps_X} + \text{of_int } n * g) \geq \text{fls_conv_radius } f$
by $(\text{intro fps_conv_radius_add_ge conv1 conv2})$

have $[\text{simp}]: \text{fps_conv_radius } g = \text{fls_conv_radius } f$
by $(\text{simp add: } g_def \text{ fls_conv_radius_altdef})$
have $((\lambda z. \text{eval_fps } g \text{ } z * z^{\text{powi fls_subdegree } f}) \text{ has_field_derivative}$
 $(\text{eval_fps (fps_deriv } g) \text{ } z * z^{\text{powi } n} + \text{of_int } n * z^{\text{powi } (n-1)} * \text{eval_fps}$
 $g \text{ } z))$
 $(\text{at } z \text{ within } A)$
using $\text{assms by (auto intro!: derivative_eq_intros simp: n_def)}$
also have $(\lambda z. \text{eval_fps } g \text{ } z * z^{\text{powi fls_subdegree } f}) = \text{eval_fls } f$
by $(\text{simp add: eval_fls_def } g_def \text{ fun_eq_iff})$
also have $\text{eval_fps (fps_deriv } g) \text{ } z * z^{\text{powi } n} + \text{of_int } n * z^{\text{powi } (n-1)} * \text{eval_fps}$
 $\text{eval_fps } g \text{ } z =$
 $(z * \text{eval_fps (fps_deriv } g) \text{ } z + \text{of_int } n * \text{eval_fps } g \text{ } z) * z^{\text{powi } (n-1)}$
 $1)$
using $\text{assms by (auto simp: power_int_diff field_simps)}$
also have $(z * \text{eval_fps (fps_deriv } g) \text{ } z + \text{of_int } n * \text{eval_fps } g \text{ } z) =$
 $\text{eval_fps (fps_deriv } g * \text{fps_X} + \text{of_int } n * g) \text{ } z$

```

    using conv1 conv2 assms fps_conv_radius_deriv[of g]
    by (subst eval_fps_add) (auto simp: eval_fps_mult)
  also have ... = eval_fls (fps_to_fls (fps_deriv g * fps_X + of_int n * g)) z
    using conv3 assms by (subst eval_fps_to_fls) auto
  also have ... * z powi (n - 1) = eval_fls (fls_shift (1 - n) (fps_to_fls
(fps_deriv g * fps_X + of_int n * g))) z
    using assms by (subst eval_fls_shift) auto
  also have fls_shift (1 - n) (fps_to_fls (fps_deriv g * fps_X + of_int n * g))
= fls_deriv f
    by (intro fls_eqI) (auto simp: g_def n_def algebra_simps eq_commute[of _
fls_subdegree f])
  finally show ?thesis .
qed

```

lemma *eval_fls_deriv*:

```

  assumes  $z \in \text{eball } 0 \text{ (fls\_conv\_radius } F) - \{0\}$ 
  shows  $\text{eval\_fls (fls\_deriv } F) z = \text{deriv (eval\_fls } F) z$ 
  by (metis DERIV_imp_deriv assms has_field_derivative_eval_fls)

```

lemma *analytic_on_eval_fls*:

```

  assumes  $A \subseteq \text{eball } 0 \text{ (fls\_conv\_radius } f) - (\text{if fls\_subdegree } f \geq 0 \text{ then } \{ \} \text{ else } \{0\})$ 
  shows  $\text{eval\_fls } f \text{ analytic\_on } A$ 
proof (rule analytic_on_subset [OF __ assms])
  show  $\text{eval\_fls } f \text{ analytic\_on eball } 0 \text{ (fls\_conv\_radius } f) - (\text{if fls\_subdegree } f \geq 0 \text{ then } \{ \} \text{ else } \{0\})$ 
    using holomorphic_on_eval_fls[OF order_refl]
    by (subst analytic_on_open) auto
qed

```

lemma *analytic_on_eval_fls' [analytic_intros]*:

```

  assumes  $g \text{ analytic\_on } A$ 
  assumes  $g ' A \subseteq \text{eball } 0 \text{ (fls\_conv\_radius } f) - (\text{if fls\_subdegree } f \geq 0 \text{ then } \{ \} \text{ else } \{0\})$ 
  shows  $(\lambda x. \text{eval\_fls } f (g x)) \text{ analytic\_on } A$ 
proof -
  have  $\text{eval\_fls } f \circ g \text{ analytic\_on } A$ 
    by (intro analytic_on_compose[OF assms(1) analytic_on_eval_fls]) (use assms
in auto)
  thus ?thesis
    by (simp add: o_def)
qed

```

lemma *continuous_eval_fls [continuous_intros]*:

```

  assumes  $z \in \text{eball } 0 \text{ (fls\_conv\_radius } F) - (\text{if fls\_subdegree } F \geq 0 \text{ then } \{ \} \text{ else } \{0\})$ 
  shows  $\text{continuous (at } z \text{ within } A) (\text{eval\_fls } F)$ 
proof -
  have  $\text{isCont (eval\_fls } F) z$ 

```

```

    using continuous_on_eval_fls[OF order.refl] assms
  by (subst (asm) continuous_on_eq_continuous_at) auto
thus ?thesis
  using continuous_at_imp_continuous_at_within by blast
qed

```

named_theorems laurent_expansion_intros

```

lemma has_laurent_expansion_imp_asympt_equiv_0:
  assumes F: f has_laurent_expansion F
  defines n  $\equiv$  fls_subdegree F
  shows f  $\sim$ [at 0] ( $\lambda z. fls\_nth F n * z^{powi n}$ )
proof (cases F = 0)
  case True
  thus ?thesis using assms
    by (auto simp: has_laurent_expansion_def)
next
  case [simp]: False
  define G where G = fls_base_factor_to_fps F
  have fls_conv_radius F > 0
    using F by (auto simp: has_laurent_expansion_def)
  hence isCont (eval_fps G) 0
    by (intro continuous_intros) (auto simp: G_def fps_conv_radius_fls_regpart
zero_ereal_def)
  hence lim: eval_fps G  $\rightarrow$  0  $\rightarrow$  eval_fps G 0
    by (meson isContD)
  have [simp]: fps_nth G 0  $\neq$  0
    by (auto simp: G_def)

  have f  $\sim$ [at 0] eval_fls F
    using F by (intro asympt_equiv_refl_ev) (auto simp: has_laurent_expansion_def
eq_commute)
  also have ... = ( $\lambda z. eval_fps G z * z^{powi n}$ )
    by (intro ext) (simp_all add: eval_fls_def G_def n_def)
  also have ...  $\sim$ [at 0] ( $\lambda z. fps_nth G 0 * z^{powi n}$ ) using lim
    by (intro asympt_equiv_intros tendsto_imp_asympt_equiv_const) (auto simp:
eval_fps_at_0)
  also have fps_nth G 0 = fls_nth F n
    by (simp add: G_def n_def)
  finally show ?thesis
    by simp
qed

```

```

lemma has_laurent_expansion_imp_asympt_equiv:
  assumes F: ( $\lambda w. f (z + w)$ ) has_laurent_expansion F
  defines n  $\equiv$  fls_subdegree F

```

```

shows  $f \sim[at\ z] (\lambda w. fls\_nth\ F\ n * (w - z)^{powi\ n})$ 
using  $has\_laurent\_expansion\_imp\_asympt\_equiv\_0[OF\ assms(1)]$  unfolding
 $n\_def$ 
by ( $simp\ add: at\_to\_0[of\ z]\ asympt\_equiv\_filtermap\_iff\ add\_ac$ )

```

```

lemmas  $[tendsto\_intros\ del] = tendsto\_power\_int$ 

```

```

lemma  $has\_laurent\_expansion\_imp\_tendsto\_0$ :
  assumes  $F: f\ has\_laurent\_expansion\ F$  and  $fls\_subdegree\ F \geq 0$ 
  shows  $f \rightarrow 0 \rightarrow fls\_nth\ F\ 0$ 
proof ( $rule\ asympt\_equiv\_tendsto\_transfer$ )
  show  $(\lambda z. fls\_nth\ F\ (fls\_subdegree\ F) * z^{powi\ fls\_subdegree\ F}) \sim[at\ 0] f$ 
  by ( $rule\ asympt\_equiv\_symI, rule\ has\_laurent\_expansion\_imp\_asympt\_equiv\_0$ )
fact
  show  $(\lambda z. fls\_nth\ F\ (fls\_subdegree\ F) * z^{powi\ fls\_subdegree\ F}) \rightarrow 0 \rightarrow fls\_nth\ F\ 0$ 
  by ( $rule\ tendsto\_eq\_intros\ refl \mid use\ assms(2)\ in\ simp$ ) +
    ( $use\ assms(2)\ in\ \langle auto\ simp: power\_int\_0\_left\_if \rangle$ )
qed

```

```

lemma  $has\_laurent\_expansion\_imp\_tendsto$ :
  assumes  $F: (\lambda w. f\ (z + w))\ has\_laurent\_expansion\ F$  and  $fls\_subdegree\ F \geq 0$ 
  shows  $f \rightarrow z \rightarrow fls\_nth\ F\ 0$ 
using  $has\_laurent\_expansion\_imp\_tendsto\_0[OF\ assms]$ 
by ( $simp\ add: at\_to\_0[of\ z]\ filterlim\_filtermap\ add\_ac$ )

```

```

lemma  $has\_laurent\_expansion\_imp\_filterlim\_infinity\_0$ :
  assumes  $F: f\ has\_laurent\_expansion\ F$  and  $fls\_subdegree\ F < 0$ 
  shows  $filterlim\ f\ at\_infinity\ (at\ 0)$ 
proof ( $rule\ asympt\_equiv\_at\_infinity\_transfer$ )
  have  $[simp]: F \neq 0$ 
  using  $assms(2)$  by  $auto$ 
  show  $(\lambda z. fls\_nth\ F\ (fls\_subdegree\ F) * z^{powi\ fls\_subdegree\ F}) \sim[at\ 0] f$ 
  by ( $rule\ asympt\_equiv\_symI, rule\ has\_laurent\_expansion\_imp\_asympt\_equiv\_0$ )
fact
  show  $filterlim\ (\lambda z. fls\_nth\ F\ (fls\_subdegree\ F) * z^{powi\ fls\_subdegree\ F})\ at\_infinity\ (at\ 0)$ 
  by ( $rule\ tendsto\_mult\_filterlim\_at\_infinity\ tendsto\_const$ 
     $filterlim\_power\_int\_neg\_at\_infinity \mid use\ assms(2)\ in\ simp$ ) +
    ( $auto\ simp: eventually\_at\_filter$ )
qed

```

```

lemma  $has\_laurent\_expansion\_imp\_neg\_fls\_subdegree$ :
  assumes  $F: f\ has\_laurent\_expansion\ F$ 
  and  $infy: filterlim\ f\ at\_infinity\ (at\ 0)$ 
  shows  $fls\_subdegree\ F < 0$ 
proof ( $rule\ ccontr$ )

```

```

assume asm:  $\neg \text{fls\_subdegree } F < 0$ 
define ff where ff =  $(\lambda z. \text{fls\_nth } F (\text{fls\_subdegree } F) * z \text{ powi } \text{fls\_subdegree } F)$ 

have (ff  $\longrightarrow$  (if fls_subdegree F = 0 then fls_nth F 0 else 0)) (at 0)
  using asm unfolding ff_def
  by (auto intro!: tendsto_eq_intros)
moreover have filterlim ff at_infinity (at 0)
proof (rule asympt_equiv_at_infinity_transfer)
  show f  $\sim$ [at 0] ff unfolding ff_def
  using has_laurent_expansion_imp_asympt_equiv_0[OF F] unfolding ff_def
  .
  show filterlim f at_infinity (at 0) by fact
qed
ultimately show False
  using not_tendsto_and_filterlim_at_infinity[of at (0::complex)] by auto
qed

lemma has_laurent_expansion_imp_filterlim_infinity:
  assumes F:  $(\lambda w. f (z + w)) \text{ has\_laurent\_expansion } F$  and fls_subdegree F < 0
  shows filterlim f at_infinity (at z)
  using has_laurent_expansion_imp_filterlim_infinity_0[OF assms]
  by (simp add: at_to_0[of z] filterlim_filtermap_add_ac)

lemma has_laurent_expansion_imp_is_pole_0:
  assumes F: f has_laurent_expansion F and fls_subdegree F < 0
  shows is_pole f 0
  using has_laurent_expansion_imp_filterlim_infinity_0[OF assms]
  by (simp add: is_pole_def)

lemma is_pole_0_imp_neg_fls_subdegree:
  assumes F: f has_laurent_expansion F and is_pole f 0
  shows fls_subdegree F < 0
  using F assms(2) has_laurent_expansion_imp_neg_fls_subdegree is_pole_def
  by blast

lemma has_laurent_expansion_imp_is_pole:
  assumes F:  $(\lambda x. f (z + x)) \text{ has\_laurent\_expansion } F$  and fls_subdegree F < 0
  shows is_pole f z
  using has_laurent_expansion_imp_is_pole_0[OF assms]
  by (simp add: is_pole_shift_0')

lemma is_pole_imp_neg_fls_subdegree:
  assumes F:  $(\lambda x. f (z + x)) \text{ has\_laurent\_expansion } F$  and is_pole f z
  shows fls_subdegree F < 0
proof -
  have is_pole  $(\lambda x. f (z + x))$  0
  using assms(2) is_pole_shift_0 by blast

```



```

    then show ?thesis
      using  $F$  is_pole_0_imp_neg_fls_subdegree by blast
qed

lemma is_pole_fls_subdegree_iff:
  assumes  $(\lambda x. f(z + x))$  has_laurent_expansion  $F$ 
  shows  $\text{is\_pole } f \ z \longleftrightarrow \text{fls\_subdegree } F < 0$ 
  using assms is_pole_imp_neg_fls_subdegree has_laurent_expansion_imp_is_pole
  by auto

lemma
  assumes  $f$  has_laurent_expansion  $F$ 
  shows has_laurent_expansion_isolated_0: isolated_singularity_at  $f$  0
    and has_laurent_expansion_not_essential_0: not_essential  $f$  0
  proof -
    from assms have eventually  $(\lambda z. \text{eval\_fls } F \ z = f \ z)$  (at 0)
      by (auto simp: has_laurent_expansion_def)
    then obtain  $r$  where  $r: r > 0 \wedge z. z \in \text{ball } 0 \ r - \{0\} \implies \text{eval\_fls } F \ z = f \ z$ 
      by (auto simp: eventually_at_filter ball_def eventually_nhds_metric)

    have fls_conv_radius  $F > 0$ 
      using assms by (auto simp: has_laurent_expansion_def)
    then obtain  $R :: \text{real}$  where  $R: R > 0 \wedge R \leq \min r (\text{fls\_conv\_radius } F)$ 
      using  $\langle r > 0 \rangle$  by (metis dual_order.strict_implies_order ereal_dense2 ereal_less2)
    min_def

    have  $\text{eval\_fls } F$  holomorphic_on  $\text{ball } 0 \ R - \{0\}$ 
      using  $r \ R$  by (intro holomorphic_intros ball_eball_mono Diff_mono) (auto
    simp: ereal_le_less)
    also have ?this  $\longleftrightarrow f$  holomorphic_on  $\text{ball } 0 \ R - \{0\}$ 
      using  $r \ R$  by (intro holomorphic_cong) auto
    also have  $\dots \longleftrightarrow f$  analytic_on  $\text{ball } 0 \ R - \{0\}$ 
      by (subst analytic_on_open) auto
    finally show isolated_singularity_at  $f$  0
      unfolding isolated_singularity_at_def using  $\langle R > 0 \rangle$  by blast

  show not_essential  $f$  0
  proof (cases fls_subdegree  $F \geq 0$ )
    case True
      hence  $f - 0 \rightarrow \text{fls\_nth } F \ 0$ 
        by (intro has_laurent_expansion_imp_tendsto_0[OF assms])
      thus ?thesis
        by (auto simp: not_essential_def)
    next
      case False
      hence is_pole  $f$  0
        by (intro has_laurent_expansion_imp_is_pole_0[OF assms]) auto
      thus ?thesis
        by (auto simp: not_essential_def)
  end

```

qed
qed

lemma

assumes $(\lambda w. f (z + w))$ *has_laurent_expansion* F
shows *has_laurent_expansion_isolated*: *isolated_singularity_at* $f z$
and *has_laurent_expansion_not_essential*: *not_essential* $f z$
using *has_laurent_expansion_isolated_0* [*OF assms*] *has_laurent_expansion_not_essential_0* [*OF assms*]
by (*simp_all add: isolated_singularity_at_shift_0 not_essential_shift_0*)

lemma *has_laurent_expansion_fps*:

assumes f *has_fps_expansion* F
shows f *has_laurent_expansion_fps_to_fls* F
proof –
from *assms* **have** *radius*: $0 < \text{fps_conv_radius } F$ **and** *eval*: $\forall_F z \text{ in nhds } 0. \text{eval_fps } F z = f z$
by (*auto simp: has_fps_expansion_def*)
from *eval* **have** *eval'*: $\forall_F z \text{ in at } 0. \text{eval_fps } F z = f z$
using *eventually_at_filter eventually_mono* **by** *fastforce*
moreover **have** *eventually* $(\lambda z. z \in \text{eball } 0 (\text{fps_conv_radius } F) - \{0\})$ (at 0)
using *radius* **by** (*intro eventually_at_in_open*) (*auto simp: zero_ereal_def*)
ultimately **have** *eventually* $(\lambda z. \text{eval_fls } (\text{fps_to_fls } F) z = f z)$ (at 0)
by *eventually_elim* (*auto simp: eval_fps_to_fls*)
thus *?thesis* **using** *radius*
by (*auto simp: has_laurent_expansion_def*)
qed

lemma *has_laurent_expansion_const* [*simp, intro, laurent_expansion_intros*]:

$(\lambda_. c)$ *has_laurent_expansion_fls_const* c
by (*auto simp: has_laurent_expansion_def*)

lemma *has_laurent_expansion_0* [*simp, intro, laurent_expansion_intros*]:

$(\lambda_. 0)$ *has_laurent_expansion* 0
by (*auto simp: has_laurent_expansion_def*)

lemma *has_fps_expansion_0_iff*: f *has_fps_expansion* 0 \longleftrightarrow *eventually* $(\lambda z. f z = 0)$ (nhds 0)

by (*auto simp: has_fps_expansion_def*)

lemma *has_laurent_expansion_1* [*simp, intro, laurent_expansion_intros*]:

$(\lambda_. 1)$ *has_laurent_expansion* 1
by (*auto simp: has_laurent_expansion_def*)

lemma *has_laurent_expansion_numeral* [*simp, intro, laurent_expansion_intros*]:

$(\lambda_. \text{numeral } n)$ *has_laurent_expansion* *numeral* n
by (*auto simp: has_laurent_expansion_def*)

lemma *has_laurent_expansion_fps_X_power* [*laurent_expansion_intros*]:

```

  ( $\lambda x. x \wedge n$ ) has_laurent_expansion (fls_X_intpow n)
  by (auto simp: has_laurent_expansion_def)

lemma has_laurent_expansion_fps_X_power_int [laurent_expansion_intros]:
  ( $\lambda x. x \text{ powi } n$ ) has_laurent_expansion (fls_X_intpow n)
  by (auto simp: has_laurent_expansion_def)

lemma has_laurent_expansion_fps_X [laurent_expansion_intros]:
  ( $\lambda x. x$ ) has_laurent_expansion fls_X
  by (auto simp: has_laurent_expansion_def)

lemma has_laurent_expansion_cmult_left [laurent_expansion_intros]:
  assumes f has_laurent_expansion F
  shows ( $\lambda x. c * f x$ ) has_laurent_expansion fls_const c * F
proof -
  from assms have eventually ( $\lambda z. z \in \text{eball } 0 (\text{fls\_conv\_radius } F) - \{0\}$ ) (at 0)
    by (intro eventually_at_in_open) (auto simp: has_laurent_expansion_def
    zero_ereal_def)
  moreover from assms have eventually ( $\lambda z. \text{eval\_fls } F z = f z$ ) (at 0)
    by (auto simp: has_laurent_expansion_def)
  ultimately have eventually ( $\lambda z. \text{eval\_fls } (\text{fls\_const } c * F) z = c * f z$ ) (at 0)
    by eventually_elim (simp_all add: eval_fls_mult)
  with assms show ?thesis
    by (auto simp: has_laurent_expansion_def intro!: less_le_trans[OF fls_conv_radius_mult])
qed

lemma has_laurent_expansion_cmult_right [laurent_expansion_intros]:
  assumes f has_laurent_expansion F
  shows ( $\lambda x. f x * c$ ) has_laurent_expansion F * fls_const c
proof -
  have  $F * \text{fls\_const } c = \text{fls\_const } c * F$ 
    by (intro fls_eqI) (auto simp: mult.commute)
  with has_laurent_expansion_cmult_left [OF assms] show ?thesis
    by (simp add: mult.commute)
qed

lemma has_fps_expansion_scaleR [fps_expansion_intros]:
  fixes  $F :: 'a :: \{\text{banach, real\_normed\_div\_algebra, comm\_ring\_1}\}$  fps
  shows  $f \text{ has\_fps\_expansion } F \implies (\lambda x. c *_{\mathbb{R}} f x) \text{ has\_fps\_expansion } \text{fps\_const}$ 
  ( $\text{of\_real } c$ ) * F
  unfolding scaleR_conv_of_real by (intro fps_expansion_intros)

lemma has_laurent_expansion_scaleR [laurent_expansion_intros]:
   $f \text{ has\_laurent\_expansion } F \implies (\lambda x. c *_{\mathbb{R}} f x) \text{ has\_laurent\_expansion } \text{fls\_const}$ 
  ( $\text{of\_real } c$ ) * F
  unfolding scaleR_conv_of_real by (intro laurent_expansion_intros)

lemma has_laurent_expansion_minus [laurent_expansion_intros]:
  assumes f has_laurent_expansion F

```

shows $(\lambda x. -f\ x) \text{ has_laurent_expansion } -F$
proof –
from *assms* **have** *eventually* $(\lambda x. x \in \text{eball } 0\ (\text{fls_conv_radius } F) - \{0\})\ (\text{at } 0)$
by (*intro eventually_at_in_open*) (*auto simp: has_laurent_expansion_def zero_ereal_def*)
moreover from *assms* **have** *eventually* $(\lambda x. \text{eval_fls } F\ x = f\ x)\ (\text{at } 0)$
by (*auto simp: has_laurent_expansion_def*)
ultimately have *eventually* $(\lambda x. \text{eval_fls } (-F)\ x = -f\ x)\ (\text{at } 0)$
by *eventually_elim* (*auto simp: eval_fls_minus*)
thus ?thesis using *assms* **by** (*auto simp: has_laurent_expansion_def*)
qed

lemma *has_laurent_expansion_add* [*laurent_expansion_intros*]:
assumes *f* *has_laurent_expansion* *F* *g* *has_laurent_expansion* *G*
shows $(\lambda x. f\ x + g\ x) \text{ has_laurent_expansion } F + G$
proof –
from *assms* **have** $0 < \min (\text{fls_conv_radius } F)\ (\text{fls_conv_radius } G)$
by (*auto simp: has_laurent_expansion_def*)
also have $\dots \leq \text{fls_conv_radius } (F + G)$
by (*rule fls_conv_radius_add*)
finally have *radius: ... > 0* .

from *assms* **have** *eventually* $(\lambda x. x \in \text{eball } 0\ (\text{fls_conv_radius } F) - \{0\})\ (\text{at } 0)$
by (*intro eventually_at_in_open; force simp: has_laurent_expansion_def zero_ereal_def*) +
moreover have *eventually* $(\lambda x. \text{eval_fls } F\ x = f\ x)\ (\text{at } 0)$
and *eventually* $(\lambda x. \text{eval_fls } G\ x = g\ x)\ (\text{at } 0)$
using *assms* **by** (*auto simp: has_laurent_expansion_def*)
ultimately have *eventually* $(\lambda x. \text{eval_fls } (F + G)\ x = f\ x + g\ x)\ (\text{at } 0)$
by *eventually_elim* (*auto simp: eval_fls_add*)
with *radius* **show** ?thesis **by** (*auto simp: has_laurent_expansion_def*)
qed

lemma *has_laurent_expansion_diff* [*laurent_expansion_intros*]:
assumes *f* *has_laurent_expansion* *F* *g* *has_laurent_expansion* *G*
shows $(\lambda x. f\ x - g\ x) \text{ has_laurent_expansion } F - G$
using *has_laurent_expansion_add* [*of f F* $\lambda x. -g\ x - G$] *assms*
by (*simp add: has_laurent_expansion_minus*)

lemma *has_laurent_expansion_mult* [*laurent_expansion_intros*]:
assumes *f* *has_laurent_expansion* *F* *g* *has_laurent_expansion* *G*
shows $(\lambda x. f\ x * g\ x) \text{ has_laurent_expansion } F * G$
proof –
from *assms* **have** $0 < \min (\text{fls_conv_radius } F)\ (\text{fls_conv_radius } G)$
by (*auto simp: has_laurent_expansion_def*)
also have $\dots \leq \text{fls_conv_radius } (F * G)$
by (*rule fls_conv_radius_mult*)

finally have $\text{radius} : \dots > 0$.

from *assms* **have** *eventually* $(\lambda x. x \in \text{eball } 0 (\text{fls_conv_radius } F) - \{0\})$ (at 0)
 eventually $(\lambda x. x \in \text{eball } 0 (\text{fls_conv_radius } G) - \{0\})$ (at 0)
by (intro *eventually_at_in_open*; force *simp*: *has_laurent_expansion_def* *zero_ereal_def*) +
moreover have *eventually* $(\lambda x. \text{eval_fls } F x = f x)$ (at 0)
 and *eventually* $(\lambda x. \text{eval_fls } G x = g x)$ (at 0)
using *assms* **by** (auto *simp*: *has_laurent_expansion_def*)
ultimately have *eventually* $(\lambda x. \text{eval_fls } (F * G) x = f x * g x)$ (at 0)
 by *eventually_elim* (auto *simp*: *eval_fls_mult*)
with *radius* **show** *?thesis* **by** (auto *simp*: *has_laurent_expansion_def*)
qed

lemma *has_fps_expansion_power* [*fps_expansion_intros*]:
 fixes $F :: 'a :: \{\text{banach}, \text{real_normed_div_algebra}, \text{comm_ring_1}\}$ *fps*
 shows $f \text{ has_fps_expansion } F \implies (\lambda x. f x \wedge^m) \text{ has_fps_expansion } F \wedge^m$
 by (induction *m*) (auto intro!: *fps_expansion_intros*)

lemma *has_laurent_expansion_power* [*laurent_expansion_intros*]:
 assumes $f \text{ has_laurent_expansion } F$
 shows $(\lambda x. f x \wedge^n) \text{ has_laurent_expansion } F \wedge^n$
 by (induction *n*) (auto intro!: *laurent_expansion_intros* *assms*)

lemma *has_laurent_expansion_sum* [*laurent_expansion_intros*]:
 assumes $\bigwedge x. x \in I \implies f x \text{ has_laurent_expansion } F x$
 shows $(\lambda y. \sum_{x \in I}. f x y) \text{ has_laurent_expansion } (\sum_{x \in I}. F x)$
 using *assms* **by** (induction *I* rule: *infinite_finite_induct*) (auto intro!: *laurent_expansion_intros*)

lemma *has_laurent_expansion_prod* [*laurent_expansion_intros*]:
 assumes $\bigwedge x. x \in I \implies f x \text{ has_laurent_expansion } F x$
 shows $(\lambda y. \prod_{x \in I}. f x y) \text{ has_laurent_expansion } (\prod_{x \in I}. F x)$
 using *assms* **by** (induction *I* rule: *infinite_finite_induct*) (auto intro!: *laurent_expansion_intros*)

lemma *has_laurent_expansion_deriv* [*laurent_expansion_intros*]:
 assumes $f \text{ has_laurent_expansion } F$
 shows $\text{deriv } f \text{ has_laurent_expansion_fls_deriv } F$

proof –

have *eventually* $(\lambda z. z \in \text{eball } 0 (\text{fls_conv_radius } F) - \{0\})$ (at 0)
 using *assms* **by** (intro *eventually_at_in_open*)
 (auto *simp*: *has_laurent_expansion_def* *zero_ereal_def*)
moreover from *assms* **have** *eventually* $(\lambda z. \text{eval_fls } F z = f z)$ (at 0)
 by (auto *simp*: *has_laurent_expansion_def*)
then obtain *s* **where** *open* s $0 \in s$ **and** $s : \bigwedge w. w \in s - \{0\} \implies \text{eval_fls } F w = f w$
 by (auto *simp*: *eventually_nhds* *eventually_at_filter*)
hence *eventually* $(\lambda w. w \in s - \{0\})$ (at 0)
 by (intro *eventually_at_in_open*) auto
ultimately have *eventually* $(\lambda z. \text{eval_fls } (\text{fls_deriv } F) z = \text{deriv } f z)$ (at 0)

```

proof eventually_elim
  case (elim z)
  hence eval_fls (fls_deriv F) z = deriv (eval_fls F) z
    by (simp add: eval_fls_deriv)
  also have eventually ( $\lambda w. w \in s - \{0\}$ ) (nhds z)
    using elim and ‹open s› by (intro eventually_nhds_in_open) auto
  hence eventually ( $\lambda w. \text{eval\_fls } F \ w = f \ w$ ) (nhds z)
    by eventually_elim (use s in auto)
  hence deriv (eval_fls F) z = deriv f z
    by (intro deriv_cong_ev refl)
  finally show ?case .
qed
with assms show ?thesis
  by (auto simp: has_laurent_expansion_def intro!: less_le_trans[OF fls_conv_radius_deriv])
qed

```

```

lemma has_laurent_expansion_shift [laurent_expansion_intros]:
  assumes f has_laurent_expansion F
  shows ( $\lambda x. f \ x * x^{\text{powi } n}$ ) has_laurent_expansion (fls_shift (-n) F)
proof -
  have eventually ( $\lambda x. x \in \text{eball } 0 \ (\text{fls\_conv\_radius } F) - \{0\}$ ) (at 0)
    using assms by (intro eventually_at_in_open) (auto simp: has_laurent_expansion_def
    zero_ereal_def)
  moreover have eventually ( $\lambda x. \text{eval\_fls } F \ x = f \ x$ ) (at 0)
    using assms by (auto simp: has_laurent_expansion_def)
  ultimately have eventually ( $\lambda x. \text{eval\_fls } (\text{fls\_shift } (-n) \ F) \ x = f \ x * x^{\text{powi } n}$ ) (at 0)
    by eventually_elim (auto simp: eval_fls_shift assms)
  with assms show ?thesis by (auto simp: has_laurent_expansion_def)
qed

```

```

lemma has_laurent_expansion_shift' [laurent_expansion_intros]:
  assumes f has_laurent_expansion F
  shows ( $\lambda x. f \ x * x^{\text{powi } (-n)}$ ) has_laurent_expansion (fls_shift n F)
  using has_laurent_expansion_shift[OF assms, of -n] by simp

```

```

lemma has_laurent_expansion_deriv':
  assumes f has_laurent_expansion F
  assumes open A 0  $\in A \wedge x. x \in A - \{0\} \implies (f \text{ has\_field\_derivative } f' \ x)$  (at x)
  shows f' has_laurent_expansion fls_deriv F
proof -
  have deriv f has_laurent_expansion fls_deriv F
    by (intro laurent_expansion_intros assms)
  also have ?this  $\longleftrightarrow$  ?thesis
  proof (intro has_laurent_expansion_cong refl)
    have eventually ( $\lambda z. z \in A - \{0\}$ ) (at 0)
      by (intro eventually_at_in_open assms)

```

```

thus eventually ( $\lambda z. \text{deriv } f \, z = f' \, z$ ) (at 0)
by eventually_elim (auto intro!: DERIV_imp_deriv assms)
qed
finally show ?thesis .
qed

```

definition laurent_expansion :: (complex \Rightarrow complex) \Rightarrow complex \Rightarrow complex fls
where

```

  laurent_expansion f z =
    (if eventually ( $\lambda z. f \, z = 0$ ) (at z) then 0
     else fls_shift (-zorder f z) (fps_to_fls (fps_expansion (zor_poly f z) z)))

```

lemma laurent_expansion_cong:

```

assumes eventually ( $\lambda w. f \, w = g \, w$ ) (at z) z = z'
shows laurent_expansion f z = laurent_expansion g z'
unfolding laurent_expansion_def
using zor_poly_cong[OF assms(1,2)] zorder_cong[OF assms] assms
by (intro if_cong refl) (auto elim: eventually_elim2)

```

theorem not_essential_has_laurent_expansion_0:

```

assumes isolated_singularity_at f 0 not_essential f 0
shows f has_laurent_expansion laurent_expansion f 0
proof (cases  $\exists_F w$  in at 0.  $f \, w \neq 0$ )
  case False
    have ( $\lambda_. 0$ ) has_laurent_expansion 0
    by simp
    also have ?this  $\longleftrightarrow$  f has_laurent_expansion 0
    using False by (intro has_laurent_expansion_cong) (auto simp: frequently_def)
    finally show ?thesis
    using False by (simp add: laurent_expansion_def frequently_def)
  next
    case True
      define n where n = zorder f 0
      obtain r where r: zor_poly f 0 0  $\neq 0$  zor_poly f 0 holomorphic_on cball 0 r r
      > 0

```

$$\forall w \in \text{cball } 0 \, r - \{0\}. f \, w = \text{zor_poly } f \, 0 \, w * w^{\text{powi } n} \wedge \text{zor_poly } f \, 0 \, w \neq 0$$

```

    using zorder_exist[OF assms True] unfolding n_def by auto
    have holo: zor_poly f 0 holomorphic_on ball 0 r
    by (rule holomorphic_on_subset[OF r(2)]) auto

    define F where F = fps_expansion (zor_poly f 0) 0
    have F: zor_poly f 0 has_fps_expansion F
    unfolding F_def by (rule has_fps_expansion_fps_expansion[OF _ _ holo])
    (use <r > 0> in auto)
    have ( $\lambda z. \text{zor\_poly } f \, 0 \, z * z^{\text{powi } n}$ ) has_laurent_expansion fls_shift (-n)
    (fps_to_fls F)
    by (intro laurent_expansion_intros has_laurent_expansion_fps[OF F])
    also have ?this  $\longleftrightarrow$  f has_laurent_expansion fls_shift (-n) (fps_to_fls F)

```

```

  by (intro has_laurent_expansion_cong refl eventually_mono[OF eventually_at_in_open[of
ball 0 r]])
    (use r in ‹auto simp: complex_powr_of_int›)
  finally show ?thesis using True
    by (simp add: laurent_expansion_def F_def n_def frequently_def)
qed

```

lemma *not_essential_has_laurent_expansion*:

```

  assumes isolated_singularity_at f z not_essential f z
  shows (λx. f (z + x)) has_laurent_expansion laurent_expansion f z
proof -
  from assms(1) have iso:isolated_singularity_at (λx. f (z + x)) 0
    by (simp add: isolated_singularity_at_shift_0)
  moreover from assms(2) have ness:not_essential (λx. f (z + x)) 0
    by (simp add: not_essential_shift_0)
  ultimately have (λx. f (z + x)) has_laurent_expansion laurent_expansion (λx.
f (z + x)) 0
    by (rule not_essential_has_laurent_expansion_0)

```

also have $\dots = \text{laurent_expansion } f \ z$

proof (cases $\exists_F w \text{ in at } z. f \ w \neq 0$)

case *False*

then have $\forall_F w \text{ in at } z. f \ w = 0$ **using** *not_frequently by force*

then have $\text{laurent_expansion } (\lambda x. f (z + x)) \ 0 = 0$

by (smt (verit, best) add.commute eventually_at_to_0 eventually_mono
laurent_expansion_def)

moreover have $\text{laurent_expansion } f \ z = 0$

using $\langle \forall_F w \text{ in at } z. f \ w = 0 \rangle$ **unfolding** *laurent_expansion_def* **by** *auto*

ultimately show *?thesis* **by** *auto*

next

case *True*

define *df* **where** $df = \text{zor_poly } (\lambda x. f (z + x)) \ 0$

define *g* **where** $g = (\lambda u. u - z)$

have $\text{fps_expansion } df \ 0$

$= \text{fps_expansion } (df \circ g) \ z$

proof -

have $\exists_F w \text{ in at } 0. f (z + w) \neq 0$ **using** *True*

by (smt (verit, best) add.commute eventually_at_to_0
eventually_mono not_frequently)

from *zorder_exist[OF iso ness this,folded df_def]*

obtain *r* **where** $r > 0$ **and** *df_holo*: $df \text{ holomorphic_on } cball \ 0 \ r$ **and** $df \ 0 \neq$

0

$\forall w \in cball \ 0 \ r - \{0\}.$

$f (z + w) = df \ w * w \text{ powi } (zorder (\lambda w. f (z + w)) \ 0) \wedge$

$df \ w \neq 0$

by *auto*

then have $df_nz: \forall w \in ball \ 0 \ r. df \ w \neq 0$ **by** *auto*


```

    have (deriv  $\sim$  n) df 0 = (deriv  $\sim$  n) (df  $\circ$  g) z for n
      unfolding comp_def g_def
    proof (subst higher_deriv_compose_linear'[where u=1 and c=-z,simplified])
      show df holomorphic_on ball 0 r
        using df_holo by auto
      show open (ball z r) open (ball 0 r) z  $\in$  ball z r
        using <r>0> by auto
      show  $\bigwedge w. w \in \text{ball } z \ r \implies w - z \in \text{ball } 0 \ r$ 
        by (simp add: dist_norm)
    qed auto
    then show ?thesis
      unfolding fps_expansion_def by auto
  qed
  also have ... = fps_expansion (zor_poly f z) z
  proof (rule fps_expansion_cong)
    have  $\forall_F w \text{ in nhds } z. \text{zor\_poly } f \ z \ w$ 
      = zor_poly ( $\lambda u. f \ (z + u)$ ) 0 (w - z)
    apply (rule zor_poly_shift)
    using True assms by auto
    then show  $\forall_F w \text{ in nhds } z. (df \circ g) \ w = \text{zor\_poly } f \ z \ w$ 
      unfolding df_def g_def comp_def
      by (auto elim:eventually_mono)
  qed
  finally show ?thesis unfolding df_def
    by (auto simp: laurent_expansion_def at_to_0[of z]
      eventually_filtermap add_ac zorder_shift')
  qed
  finally show ?thesis .
  qed

lemma has_fps_expansion_to_laurent:
  f has_fps_expansion F  $\longleftrightarrow$  f has_laurent_expansion fps_to_fls F  $\wedge$  f 0 =
  fps_nth F 0
  proof safe
    assume *: f has_laurent_expansion fps_to_fls F f 0 = fps_nth F 0
    have eventually ( $\lambda z. z \in \text{eball } 0 \ (\text{fps\_conv\_radius } F)$ ) (nhds 0)
      using * by (intro eventually_nhds_in_open) (auto simp: has_laurent_expansion_def
        zero_ereal_def)
    moreover have eventually ( $\lambda z. z \neq 0 \longrightarrow \text{eval\_fls } (\text{fps\_to\_fls } F) \ z = f \ z$ ) (nhds
      0)
      using * by (auto simp: has_laurent_expansion_def eventually_at_filter)
    ultimately have eventually ( $\lambda z. f \ z = \text{eval\_fps } F \ z$ ) (nhds 0)
      by eventually_elim
      (auto simp: has_laurent_expansion_def eventually_at_filter eval_fps_at_0
        eval_fps_to_fls *(2))
    thus f has_fps_expansion F
      using * by (auto simp: has_fps_expansion_def has_laurent_expansion_def
        eq_commute)
  next

```

```

  assume f has_fps_expansion F
  thus f 0 = fps_nth F 0
    by (metis eval_fps_at_0 has_fps_expansion_imp_holomorphic)
qed (auto intro: has_laurent_expansion_fps)

```

```

lemma eval_fps_fls_base_factor [simp]:
  assumes  $z \neq 0$ 
  shows  $\text{eval\_fps} (\text{fls\_base\_factor\_to\_fps } F) z = \text{eval\_fls } F z * z^{\text{powi } -\text{fls\_subdegree } F}$ 
  using assms unfolding eval_fls_def by (simp add: power_int_minus field_simps)

```

```

lemma has_fps_expansion_imp_analytic_0:
  assumes f has_fps_expansion F
  shows f analytic_on {0}
  by (meson analytic_at_two assms has_fps_expansion_imp_holomorphic)

```

```

lemma has_fps_expansion_imp_analytic:
  assumes  $(\lambda x. f (z + x))$  has_fps_expansion F
  shows f analytic_on {z}
proof -
  have  $(\lambda x. f (z + x))$  analytic_on {0}
    by (rule has_fps_expansion_imp_analytic_0) fact
  hence  $(\lambda x. f (z + x)) \circ (\lambda x. x - z)$  analytic_on {z}
    by (intro analytic_on_compose_gen analytic_intros) auto
  thus ?thesis
    by (simp add: o_def)
qed

```

```

lemma is_pole_cong_asymp_equiv:
  assumes  $f \sim[at\ z] g$   $z = z'$ 
  shows is_pole f z = is_pole g z'
  using asymp_equiv_at_infinity_transfer[OF assms(1)]
    asymp_equiv_at_infinity_transfer[OF asymp_equiv_symI[OF assms(1)]]
  by (simp add: assms(2) unfolding is_pole_def by auto)

```

```

lemma not_is_pole_const [simp]:  $\neg \text{is\_pole } (\lambda \_::'a::\text{perfect\_space}. c :: \text{complex})$ 
   $z$ 
  using not_tendsto_and_filterlim_at_infinity[of at z  $\lambda \_::'a. c$ ] by (auto simp:
  is_pole_def)

```

```

lemma has_laurent_expansion_imp_is_pole_iff:
  assumes F:  $(\lambda x. f (z + x))$  has_laurent_expansion F
  shows  $\text{is\_pole } f z \longleftrightarrow \text{fls\_subdegree } F < 0$ 
proof
  assume pole: is_pole f z
  have [simp]:  $F \neq 0$ 
  proof
    assume  $F = 0$ 

```

```

    hence is_pole f z  $\longleftrightarrow$  is_pole ( $\lambda \_. 0 :: \text{complex}$ ) z using assms
    by (intro is_pole_cong)
    (auto simp: has_laurent_expansion_def at_to_0[of z] eventually_filtermap
add_ac)
    with pole show False
    by simp
  qed

  note pole
  also have is_pole f z  $\longleftrightarrow$ 
    is_pole ( $\lambda w. \text{fls\_nth } F (\text{fls\_subdegree } F) * (w - z) \text{ powi fls\_subdegree } F$ ) z
  using has_laurent_expansion_imp_asymp_equiv[OF F] by (intro is_pole_cong_asymp_equiv
refl)
  also have  $\dots \longleftrightarrow$  is_pole ( $\lambda w. (w - z) \text{ powi fls\_subdegree } F$ ) z
    by simp
  finally have pole':  $\dots$  .

  have False if fls_subdegree F  $\geq 0$ 
  proof -
    have ( $\lambda w. (w - z) \text{ powi fls\_subdegree } F$ ) holomorphic_on UNIV
    using that by (intro holomorphic_intros) auto
    hence  $\neg \text{is\_pole } (\lambda w. (w - z) \text{ powi fls\_subdegree } F) z$ 
    by (meson UNIV_I not_is_pole_holomorphic_open_UNIV)
    with pole' show False
    by simp
  qed
  thus fls_subdegree F < 0
  by force
qed (use has_laurent_expansion_imp_is_pole[OF assms] in auto)

lemma analytic_at_imp_has_fps_expansion_0:
  assumes f analytic_on {0}
  shows f has_fps_expansion fps_expansion f 0
  using assms has_fps_expansion_fps_expansion analytic_at by fast

lemma analytic_at_imp_has_fps_expansion:
  assumes f analytic_on {z}
  shows ( $\lambda x. f (z + x)$ ) has_fps_expansion fps_expansion f z
  proof -
    have f  $\circ (\lambda x. z + x)$  analytic_on {0}
    by (intro analytic_on_compose_gen[OF _ assms] analytic_intros) auto
    hence (f  $\circ (\lambda x. z + x)$ ) has_fps_expansion fps_expansion (f  $\circ (\lambda x. z + x)$ ) 0
    unfolding o_def by (intro analytic_at_imp_has_fps_expansion_0) auto
    also have  $\dots = \text{fps\_expansion } f z$ 
    by (simp add: fps_expansion_def higher_deriv_shift_0')
    finally show ?thesis by (simp add: add_ac)
  qed

```

```

lemma has_laurent_expansion_zorder_0:
  assumes f has_laurent_expansion F F  $\neq$  0
  shows zorder f 0 = fls_subdegree F
proof -
  define G where G = fls_base_factor_to_fps F
  from assms obtain A where A: 0  $\in$  A open A  $\wedge$  x. x  $\in$  A - {0}  $\implies$  eval_fls
    F x = f x
    unfolding has_laurent_expansion_def eventually_at_filter eventually_nhds
    by blast

  show ?thesis
  proof (rule zorder_eqI)
    show open (A  $\cap$  eball 0 (fls_conv_radius F)) 0  $\in$  A  $\cap$  eball 0 (fls_conv_radius
    F)
      using assms A by (auto simp: has_laurent_expansion_def zero_ereal_def)
    show eval_fps G holomorphic_on A  $\cap$  eball 0 (fls_conv_radius F)
      by (intro holomorphic_intros) (auto simp: fls_conv_radius_altdef G_def)
    show eval_fps G 0  $\neq$  0 using  $\langle F \neq 0 \rangle$ 
      by (auto simp: eval_fps_at_0 G_def)
    next
      fix w :: complex assume w  $\in$  A  $\cap$  eball 0 (fls_conv_radius F) w  $\neq$  0
      thus f w = eval_fps G w * (w - 0) powi (fls_subdegree F)
        using A unfolding G_def
        by (subst eval_fps_fls_base_factor)
          (auto simp: complex_powr_of_int power_int_minus field_simps)
    qed
  qed

lemma has_laurent_expansion_zorder:
  assumes ( $\lambda w. f (z + w)$ ) has_laurent_expansion F F  $\neq$  0
  shows zorder f z = fls_subdegree F
  using has_laurent_expansion_zorder_0[OF assms] by (simp add: zorder_shift'
  add_ac)

lemma has_fps_expansion_zorder_0:
  assumes f has_fps_expansion F F  $\neq$  0
  shows zorder f 0 = int (subdegree F)
  using assms has_laurent_expansion_zorder_0[of f fps_to_fls F]
  by (auto simp: has_fps_expansion_to_laurent fls_subdegree_fls_to_fps)

lemma has_fps_expansion_zorder:
  assumes ( $\lambda w. f (z + w)$ ) has_fps_expansion F F  $\neq$  0
  shows zorder f z = int (subdegree F)
  using has_fps_expansion_zorder_0[OF assms]
  by (simp add: zorder_shift' add_ac)

lemma has_fps_expansion_fls_base_factor_to_fps:
  assumes f has_laurent_expansion F
  defines n  $\equiv$  fls_subdegree F

```

```

defines  $c \equiv \text{fps\_nth } (\text{fls\_base\_factor\_to\_fps } F) \ 0$ 
shows  $(\lambda z. \text{if } z = 0 \text{ then } c \text{ else } f z * z^{\text{powi } -n}) \text{ has\_fps\_expansion } \text{fls\_base\_factor\_to\_fps } F$ 
proof –
  have  $(\lambda z. f z * z^{\text{powi } -n}) \text{ has\_laurent\_expansion } \text{fls\_shift } (-(-n)) \ F$ 
    by (intro laurent_expansion_intros assms)
  also have  $\text{fls\_shift } (-(-n)) \ F = \text{fps\_to\_fls } (\text{fls\_base\_factor\_to\_fps } F)$ 
    by (simp add: n_def fls_shift_nonneg_subdegree)
  also have  $(\lambda z. f z * z^{\text{powi } -n}) \text{ has\_laurent\_expansion } \text{fps\_to\_fls } (\text{fls\_base\_factor\_to\_fps } F) \longleftrightarrow$ 
 $(\lambda z. \text{if } z = 0 \text{ then } c \text{ else } f z * z^{\text{powi } -n}) \text{ has\_laurent\_expansion } \text{fps\_to\_fls } (\text{fls\_base\_factor\_to\_fps } F)$ 
    by (intro has_laurent_expansion_cong) (auto simp: eventually_at_filter)
  also have  $\dots \longleftrightarrow (\lambda z. \text{if } z = 0 \text{ then } c \text{ else } f z * z^{\text{powi } -n}) \text{ has\_fps\_expansion } \text{fls\_base\_factor\_to\_fps } F$ 
    by (subst has_fps_expansion_to_laurent) (auto simp: c_def)
  finally show ?thesis .
qed

```

```

lemma zero_has_laurent_expansion_imp_eq_0:
  assumes  $(\lambda_. 0) \text{ has\_laurent\_expansion } F$ 
  shows  $F = 0$ 
proof –
  have  $at \ (0 :: \text{complex}) \neq \text{bot}$ 
    by auto
  moreover have  $(\lambda z. \text{if } z = 0 \text{ then } \text{fls\_nth } F \ (\text{fls\_subdegree } F) \text{ else } 0) \text{ has\_fps\_expansion } \text{fls\_base\_factor\_to\_fps } F$ 
    (is ?f has\_fps\_expansion _)
    using has_fps_expansion_fls_base_factor_to_fps[OF assms] by (simp cong: if_cong)
  hence  $\text{isCont } ?f \ 0$ 
    using has_fps_expansion_imp_continuous by blast
  hence  $?f \ -0 \rightarrow \text{fls\_nth } F \ (\text{fls\_subdegree } F)$ 
    by (auto simp: isCont_def)
  moreover have  $?f \ -0 \rightarrow 0 \longleftrightarrow (\lambda_::\text{complex}. 0 :: \text{complex}) \ -0 \rightarrow 0$ 
    by (intro filterlim_cong) (auto simp: eventually_at_filter)
  hence  $?f \ -0 \rightarrow 0$ 
    by simp
  ultimately have  $\text{fls\_nth } F \ (\text{fls\_subdegree } F) = 0$ 
    by (rule tendsto_unique)
  thus ?thesis
    by (meson nth_fls_subdegree_nonzero)
qed

```

```

lemma has_laurent_expansion_unique:
  assumes  $f \text{ has\_laurent\_expansion } F$   $f \text{ has\_laurent\_expansion } G$ 
  shows  $F = G$ 
proof –
  from assms have  $(\lambda x. f \ x - f \ x) \text{ has\_laurent\_expansion } F - G$ 
    by (intro laurent_expansion_intros)

```

```

  hence  $(\lambda_. 0)$  has_laurent_expansion  $F - G$ 
    by simp
  hence  $F - G = 0$ 
    by (rule zero_has_laurent_expansion_imp_eq_0)
  thus ?thesis
    by simp
qed

```

```

lemma laurent_expansion_eqI:
  assumes  $(\lambda x. f (z + x))$  has_laurent_expansion  $F$ 
  shows laurent_expansion  $f z = F$ 
  using assms has_laurent_expansion_isolated has_laurent_expansion_not_essential
    has_laurent_expansion_unique not_essential_has_laurent_expansion by
blast

```

```

lemma laurent_expansion_0_eqI:
  assumes  $f$  has_laurent_expansion  $F$ 
  shows laurent_expansion  $f 0 = F$ 
  using assms laurent_expansion_eqI[of  $f 0$ ] by simp

```

```

lemma has_laurent_expansion_nonzero_imp_eventually_nonzero:
  assumes  $f$  has_laurent_expansion  $F$   $F \neq 0$ 
  shows eventually  $(\lambda x. f x \neq 0)$  (at 0)
proof (rule ccontr)
  assume  $\neg$ eventually  $(\lambda x. f x \neq 0)$  (at 0)
  with assms have eventually  $(\lambda x. f x = 0)$  (at 0)
  by (intro not_essential_frequently_0_imp_eventually_0 has_laurent_expansion_isolated
    has_laurent_expansion_not_essential)
    (auto simp: frequently_def)
  hence  $(f$  has_laurent_expansion 0)  $\longleftrightarrow ((\lambda_. 0)$  has_laurent_expansion 0)
  by (intro has_laurent_expansion_cong) auto
  hence  $f$  has_laurent_expansion 0
  by simp
  with assms(1) have  $F = 0$ 
  using has_laurent_expansion_unique by blast
  with  $\langle F \neq 0 \rangle$  show False
  by contradiction
qed

```

```

lemma has_laurent_expansion_eventually_nonzero_iff':
  assumes  $f$  has_laurent_expansion  $F$ 
  shows eventually  $(\lambda x. f x \neq 0)$  (at 0)  $\longleftrightarrow F \neq 0$ 
proof
  assume  $\forall_F x$  in at 0.  $f x \neq 0$ 
  moreover have  $\neg (\forall_F x$  in at 0.  $f x \neq 0)$  if  $F=0$ 
  proof -
    have  $\forall_F x$  in at 0.  $f x = 0$ 
    using assms that unfolding has_laurent_expansion_def by simp
    then show ?thesis unfolding not_eventually

```

```

    by (auto elim: eventually_frequentlyE)
  qed
  ultimately show  $F \neq 0$  by auto
qed (simp add: has_laurent_expansion_nonzero_imp_eventually_nonzero[OF assms])

lemma has_laurent_expansion_eventually_nonzero_iff:
  assumes  $(\lambda w. f (z+w)) \text{ has\_laurent\_expansion } F$ 
  shows  $\text{eventually } (\lambda x. f x \neq 0) \text{ (at } z) \longleftrightarrow F \neq 0$ 
  apply (subst eventually_at_to_0)
  apply (rule has_laurent_expansion_eventually_nonzero_iff')
  using assms by (simp add: add.commute)

lemma has_laurent_expansion_inverse [laurent_expansion_intros]:
  assumes  $f \text{ has\_laurent\_expansion } F$ 
  shows  $(\lambda x. \text{inverse } (f x)) \text{ has\_laurent\_expansion inverse } F$ 
proof (cases  $F = 0$ )
  case True
  thus ?thesis using assms
    by (auto simp: has_laurent_expansion_def)
next
  case False
  define G where  $G = \text{laurent\_expansion } (\lambda x. \text{inverse } (f x)) 0$ 
  from False have ev:  $\text{eventually } (\lambda z. f z \neq 0) \text{ (at } 0)$ 
  by (intro has_laurent_expansion_nonzero_imp_eventually_nonzero[OF assms])

  have *:  $(\lambda x. \text{inverse } (f x)) \text{ has\_laurent\_expansion } G \text{ unfolding } G\_def$ 
  by (intro not_essential_has_laurent_expansion_0 isolated_singularity_at_inverse
    not_essential_inverse
    has_laurent_expansion_isolated_0[OF assms] has_laurent_expansion_not_essential_0[OF
    assms])
  have  $(\lambda x. f x * \text{inverse } (f x)) \text{ has\_laurent\_expansion } F * G$ 
  by (intro laurent_expansion_intros assms *)
  also have  $?this \longleftrightarrow (\lambda x. 1) \text{ has\_laurent\_expansion } F * G$ 
  by (intro has_laurent_expansion_cong_refl eventually_mono[OF ev]) auto
  finally have  $(\lambda x. 1) \text{ has\_laurent\_expansion } F * G$  .
  moreover have  $(\lambda x. 1) \text{ has\_laurent\_expansion } 1$ 
  by simp
  ultimately have  $F * G = 1$ 
  using has_laurent_expansion_unique by blast
  hence  $G = \text{inverse } F$ 
  using inverse_unique by blast
  with * show ?thesis
  by simp
qed

lemma has_laurent_expansion_power_int [laurent_expansion_intros]:
   $f \text{ has\_laurent\_expansion } F \implies (\lambda x. f x \text{ powi } n) \text{ has\_laurent\_expansion } (F \text{ powi } n)$ 
  by (auto simp: power_int_def intro!: laurent_expansion_intros)

```

```

lemma has_fps_expansion_0_analytic_continuation:
  assumes f has_fps_expansion 0 f holomorphic_on A
  assumes open A connected A 0 ∈ A x ∈ A
  shows  $f x = 0$ 
proof -
  have eventually ( $\lambda z. z \in A \wedge f z = 0$ ) (nhds 0) using assms
  by (intro eventually_conj eventually_nhds_in_open) (auto simp: has_fps_expansion_def)
  then obtain B where B: open B 0 ∈ B  $\forall z \in B. z \in A \wedge f z = 0$ 
  unfolding eventually_nhds by blast
  show ?thesis
proof (rule analytic_continuation_open[where f = f and g =  $\lambda_. 0$ ])
  show  $B \neq \{0\}$ 
  using  $\langle \text{open } B \rangle B$  by auto
  show connected A
  using assms by auto
  qed (use assms B in auto)
qed

```

```

lemma has_laurent_expansion_0_analytic_continuation:
  assumes f has_laurent_expansion 0 f holomorphic_on A - {0}
  assumes open A connected A 0 ∈ A x ∈ A - {0}
  shows  $f x = 0$ 
proof -
  have eventually ( $\lambda z. z \in A - \{0\} \wedge f z = 0$ ) (at 0) using assms
  by (intro eventually_conj eventually_at_in_open) (auto simp: has_laurent_expansion_def)
  then obtain B where B: open B 0 ∈ B  $\forall z \in B - \{0\}. z \in A - \{0\} \wedge f z = 0$ 
  unfolding eventually_at_filter eventually_nhds by blast
  show ?thesis
proof (rule analytic_continuation_open[where f = f and g =  $\lambda_. 0$ ])
  show  $B - \{0\} \neq \{0\}$ 
  using  $\langle \text{open } B \rangle \langle 0 \in B \rangle$  by (metis insert_Diff not_open_singleton)
  show connected (A - {0})
  using assms by (intro connected_open_delete) auto
  qed (use assms B in auto)
qed

```

```

lemma has_fps_expansion_cong:
  assumes eventually ( $\lambda x. f x = g x$ ) (nhds 0)  $F = G$ 
  shows  $f \text{ has\_fps\_expansion } F \longleftrightarrow g \text{ has\_fps\_expansion } G$ 
  using assms(2) by (auto simp: has_fps_expansion_def elim!: eventually_elim2[OF assms(1)])

```

```

lemma zor_poly_has_fps_expansion:
  assumes  $f \text{ has\_laurent\_expansion } F F \neq 0$ 
  shows  $\text{zor\_poly } f 0 \text{ has\_fps\_expansion } \text{fls\_base\_factor\_to\_fps } F$ 
proof -
  note [simp] =  $\langle F \neq 0 \rangle$ 

```



```

have eventually (λz. f z ≠ 0) (at 0)
by (rule has_laurent_expansion_nonzero_imp_eventually_nonzero[OF assms])
hence freq: frequently (λz. f z ≠ 0) (at 0)
by (rule eventually_frequently[rotated]) auto

have *: isolated_singularity_at f 0 not_essential f 0
using has_laurent_expansion_isolated_0[OF assms(1)] has_laurent_expansion_not_essential_0[OF
assms(1)]
by auto

define G where G = fls_base_factor_to_fps F
define n where n = zorder f 0
have n_altdef: n = fls_subdegree F
using has_laurent_expansion_zorder_0 [OF assms(1)] by (simp add: n_def)
obtain r where r: zor_poly f 0 0 ≠ 0 zor_poly f 0 holomorphic_on cball 0 r r
> 0
    ∀ w ∈ cball 0 r - {0}. f w = zor_poly f 0 w * w powi n ∧
    zor_poly f 0 w ≠ 0
using zorder_exist[OF * freq] unfolding n_def by auto
obtain r' where r': r' > 0 ∀ x ∈ ball 0 r' - {0}. eval_fls F x = f x
using assms(1) unfolding has_laurent_expansion_def eventually_at_filter
eventually_nhds_metric ball_def
by (auto simp: dist_commute)
have holo: zor_poly f 0 holomorphic_on ball 0 r
by (rule holomorphic_on_subset[OF r(2)]) auto

have (λz. if z = 0 then fps_nth G 0 else f z * z powi -n) has_fps_expansion G
unfolding G_def n_altdef by (intro has_fps_expansion_fls_base_factor_to_fps
assms)
also have ?this ⟷ zor_poly f 0 has_fps_expansion G
proof (intro has_fps_expansion_cong)
have eventually (λz. z ∈ ball 0 (min r r')) (nhds 0)
using ⟨r > 0⟩ ⟨r' > 0⟩ by (intro eventually_nhds_in_open) auto
thus ∀ f x in nhds 0. (if x = 0 then G $ 0 else f x * x powi -n) = zor_poly f
0 x
proof eventually_elim
case (elim w)
have w: w ∈ ball 0 r w ∈ ball 0 r'
using elim by auto
show ?case
proof (cases w = 0)
case False
hence f w = zor_poly f 0 w * w powi n
using r w by auto
thus ?thesis using False
by (simp add: powr_minus complex_powr_of_int power_int_minus)
next
case [simp]: True
obtain R where R: R > 0 R ≤ r R ≤ r' R ≤ fls_conv_radius F

```

```

      using ⟨r > 0⟩ ⟨r' > 0⟩ assms(1) unfolding has_laurent_expansion_def
      by (smt (verit, ccfv_SIG) ereal_dense2 ereal_less(2) less_ereal.simps(1)
order.strict_implies_order order_trans)
    have eval_fps G 0 = zor_poly f 0 0
    proof (rule analytic_continuation_open[where f = eval_fps G and g =
zor_poly f 0])
      show connected (ball 0 R :: complex set)
      by auto
      have of_real R / 2 ∈ ball 0 R - {0 :: complex}
      using R by auto
      thus ball 0 R - {0 :: complex} ≠ {}
      by blast
      show eval_fps G holomorphic_on ball 0 R
      using R less_le_trans[OF _ R(4)] unfolding G_def
      by (intro holomorphic_intros) (auto simp: fls_conv_radius_altdef)
      show zor_poly f 0 holomorphic_on ball 0 R
      by (rule holomorphic_on_subset[OF holo]) (use R in auto)
      show eval_fps G z = zor_poly f 0 z if z ∈ ball 0 R - {0} for z
      using that r r' R n_altdef unfolding G_def
      by (subst eval_fps_fls_base_factor)
      (auto simp: complex_pow_of_int field_simps power_int_minus
n_def)
    qed (use R in auto)
    hence zor_poly f 0 0 = fps_nth G 0
    by (simp add: eval_fps_at_0)
    thus ?thesis by simp
  qed
qed
qed (use r' in auto)
finally show ?thesis
  by (simp add: G_def)
qed

lemma zorder_geI_0:
  assumes f_analytic_on {0} f_holomorphic_on A open A connected A 0 ∈ A z ∈
A f z ≠ 0
  assumes ∧k. k < n ⟹ (deriv ~ k) f 0 = 0
  shows zorder f 0 ≥ n
proof -
  define F where F = fps_expansion f 0
  from assms have f_has_fps_expansion F
  unfolding F_def using analytic_at_imp_has_fps_expansion_0 by blast
  hence laurent: f_has_laurent_expansion fps_to_fls F and [simp]: f 0 = fps_nth
F 0
  by (simp_all add: has_fps_expansion_to_laurent)

  have [simp]: F ≠ 0
  proof
    assume [simp]: F = 0

```

```

hence  $f z = 0$ 
proof (cases  $z = 0$ )
  case False
  have  $f \text{ has\_laurent\_expansion } 0$ 
    using laurent by simp
  thus ?thesis
  proof (rule has_laurent_expansion_0_analytic_continuation)
    show  $f \text{ holomorphic\_on } A - \{0\}$ 
      using assms(2) by (rule holomorphic_on_subset) auto
    qed (use assms False in auto)
  qed auto
  with  $\langle f z \neq 0 \rangle$  show False by contradiction
qed

have  $\text{zorder } f \ 0 = \text{int } (\text{subdegree } F)$ 
  using has_laurent_expansion_zorder_0[OF laurent] by (simp add: fls_subdegree_fls_to_fps)
also have  $\text{subdegree } F \geq n$ 
  using assms by (intro subdegree_geI  $\langle F \neq 0 \rangle$ ) (auto simp: F_def fps_expansion_def)
hence  $\text{int } (\text{subdegree } F) \geq \text{int } n$ 
  by simp
finally show ?thesis .
qed

lemma zorder_geI:
  assumes  $f \text{ analytic\_on } \{x\}$   $f \text{ holomorphic\_on } A$   $A \text{ open}$   $A \text{ connected}$   $A x \in A$   $z \in A$   $f z \neq 0$ 
  assumes  $\bigwedge k. k < n \implies (\text{deriv } \smallfrown k) f x = 0$ 
  shows  $\text{zorder } f x \geq n$ 
proof -
  have  $\text{zorder } f x = \text{zorder } (f \circ (\lambda u. u + x)) \ 0$ 
    by (subst zorder_shift) (auto simp: o_def)
  also have  $\dots \geq n$ 
  proof (rule zorder_geI_0)
    show  $(f \circ (\lambda u. u + x)) \text{ analytic\_on } \{0\}$ 
      by (intro analytic_on_compose_gen[OF assms(1)] analytic_intros) auto
    show  $f \circ (\lambda u. u + x) \text{ holomorphic\_on } ((+) (-x)) \text{ ' } A$ 
      by (intro holomorphic_on_compose_gen[OF assms(2)] holomorphic_intros)
  auto
  show  $\text{connected } ((+) (-x)) \text{ ' } A$ 
    by (intro connected_continuous_image continuous_intros assms)
  show  $\text{open } ((+) (-x)) \text{ ' } A$ 
    by (intro open_translation assms)
  show  $z - x \in (+) (-x) \text{ ' } A$ 
    using  $\langle z \in A \rangle$  by auto
  show  $0 \in (+) (-x) \text{ ' } A$ 
    using  $\langle x \in A \rangle$  by auto
  show  $(f \circ (\lambda u. u + x)) (z - x) \neq 0$ 
    using  $\langle f z \neq 0 \rangle$  by auto
next

```

```

fix k :: nat assume k < n
hence (deriv  $\sim$  k) f x = 0
  using assms by simp
also have (deriv  $\sim$  k) f x = (deriv  $\sim$  k) (f  $\circ$  (+) x) 0
  by (subst higher_deriv_shift_0) auto
finally show (deriv  $\sim$  k) (f  $\circ$  ( $\lambda$ u. u + x)) 0 = 0
  by (subst add commute) auto
qed
finally show ?thesis .
qed

```

```

lemma has_laurent_expansion_divide [laurent_expansion_intros]:
  assumes f has_laurent_expansion F and g has_laurent_expansion G
  shows ( $\lambda$ x. f x / g x) has_laurent_expansion (F / G)
proof -
  have ( $\lambda$ x. f x * inverse (g x)) has_laurent_expansion (F * inverse G)
    by (intro laurent_expansion_intros assms)
  thus ?thesis
    by (simp add: field_simps)
qed

```

```

lemma has_laurent_expansion_residue_0:
  assumes f has_laurent_expansion F
  shows residue f 0 = fls_residue F
proof (cases fls_subdegree F  $\geq$  0)
case True
  have residue f 0 = residue (eval_fls F) 0
    using assms by (intro residue_cong) (auto simp: has_laurent_expansion_def
eq_commute)
  also have ... = 0
    by (rule residue_holo[OF _ holomorphic_on_eval_fls[OF order_refl]])
      (use True assms in  $\langle$ auto simp: has_laurent_expansion_def zero_ereal_def $\rangle$ )
  also have ... = fls_residue F
    using True by simp
  finally show ?thesis .
next
case False
  hence F  $\neq$  0
    by auto
  have *: zor_poly f 0 has_fps_expansion fls_base_factor_to_fps F
    by (intro zor_poly_has_fps_expansion False assms  $\langle$ F  $\neq$  0 $\rangle$ )

  have residue f 0 = (deriv  $\sim$  (nat (-zorder f 0) - 1)) (zor_poly f 0) 0 / fact
(nat (-zorder f 0) - 1)
    by (intro residue_pole_order has_laurent_expansion_isolated_0[OF assms]
has_laurent_expansion_imp_is_pole_0[OF assms]) (use False in auto)
  also have ... = fls_residue F
    using has_laurent_expansion_zorder_0[OF assms  $\langle$ F  $\neq$  0 $\rangle$ ] False
    by (subst fps_nth_fps_expansion [OF *, symmetric]) (auto simp: of_nat_diff)

```

finally show ?thesis .
qed

lemma has_laurent_expansion_residue:
assumes $(\lambda x. f (z + x))$ has_laurent_expansion F
shows residue $f z = fls_residue F$
using has_laurent_expansion_residue_0[OF assms] by (simp add: residue_shift_0')

lemma eval_fls_has_laurent_expansion [laurent_expansion_intros]:
assumes fls_conv_radius $F > 0$
shows eval_fls F has_laurent_expansion F
using assms by (auto simp: has_laurent_expansion_def)

lemma fps_expansion_unique_complex:
fixes $F G :: \text{complex fps}$
assumes f has_fps_expansion F f has_fps_expansion G
shows $F = G$
using assms unfolding fps_eq_iff by (auto simp: fps_eq_iff fps_nth_fps_expansion)

lemma fps_expansion_eqI:
assumes f has_fps_expansion F
shows fps_expansion $f 0 = F$
using assms unfolding fps_eq_iff
by (auto simp: fps_eq_iff fps_nth_fps_expansion fps_expansion_def)

lemma holomorphic_on_imp_fps_conv_radius_ge:
assumes f has_fps_expansion F f holomorphic_on eball $0 r$
shows fps_conv_radius $F \geq r$
proof -
define n where $n = \text{subdegree } F$
have fps_conv_radius (fps_expansion $f 0$) $\geq r$
by (intro conv_radius_fps_expansion assms)
also have fps_expansion $f 0 = F$
using assms by (intro fps_expansion_eqI)
finally show ?thesis
by simp
qed

lemma has_fps_expansion_imp_eval_fps_eq:
assumes f has_fps_expansion F norm $z < r$
assumes f holomorphic_on ball $0 r$
shows eval_fps $F z = f z$
proof -
have [simp]: fps_expansion $f 0 = F$
by (rule fps_expansion_eqI) fact
have *: f holomorphic_on eball 0 (ereal r)
using assms by simp
from conv_radius_fps_expansion[OF *] have fps_conv_radius $F \geq \text{ereal } r$
by simp

```

have eval_fps (fps_expansion f 0) z = f (0 + z)
  by (rule eval_fps_expansion'[OF *]) (use assms in auto)
thus ?thesis
  by simp
qed

```

```

lemma has_fps_expansion_imp_sums_complex:
  fixes F :: complex fps
  assumes f has_fps_expansion F f holomorphic_on eball 0 r ereal (norm z) < r
  shows (λn. fps_nth F n * z ^ n) sums f z
proof -
  have r: fps_conv_radius F ≥ r
    using assms(1,2) by (rule holomorphic_on_imp_fps_conv_radius_ge)
  from assms obtain R where R: norm z < R ereal R < r
    using ereal_dense2 less_ereal.simps(1) by blast
  have z: norm z < fps_conv_radius F
    using r R assms(3) by order

  have summable (λn. fps_nth F n * z ^ n)
    by (rule summable_fps) (use z in auto)
  moreover have eval_fps F z = f z
  proof (rule has_fps_expansion_imp_eval_fps_eq[where r = R])
    have *: ereal (norm z) < r if norm z < R for z :: complex
      using that R ereal_le_less less_imp_le by blast
    show f holomorphic_on ball 0 R
      using assms(2) by (rule holomorphic_on_subset) (use * in auto)
  qed (use R assms(1) in auto)
  ultimately show ?thesis
    unfolding eval_fps_def sums_iff by simp
qed

```

```

lemma fls_conv_radius_ge:
  assumes f has_laurent_expansion F
  assumes f holomorphic_on eball 0 r - {0}
  shows fls_conv_radius F ≥ r
proof -
  define n where n = fls_subdegree F
  define G where G = fls_base_factor_to_fps F
  define g where g = (λz. if z = 0 then fps_nth G 0 else f z * z powi -n)
  have G: g has_fps_expansion G
    unfolding G_def g_def n_def
    by (intro has_fps_expansion_fls_base_factor_to_fps assms)
  have (λz. f z * z powi -n) holomorphic_on eball 0 r - {0}
    by (intro holomorphic_intros assms) auto
  also have ?this ⟷ g holomorphic_on eball 0 r - {0}
    by (intro holomorphic_cong) (auto simp: g_def)
  finally have g analytic_on eball 0 r - {0}
    by (subst analytic_on_open) auto
  moreover have g analytic_on {0}

```

```

    using G has_fps_expansion_imp_analytic_0 by auto
  ultimately have g analytic_on (eball 0 r - {0}  $\cup$  {0})
    by (subst analytic_on_Un) auto
  hence g analytic_on eball 0 r
    by (rule analytic_on_subset) auto
  hence g holomorphic_on eball 0 r
    by (subst (asm) analytic_on_open) auto
  hence fps_conv_radius (fps_expansion g 0)  $\geq$  r
    by (intro conv_radius_fps_expansion)
  also have fps_expansion g 0 = G
    using G by (intro fps_expansion_eqI)
  finally show ?thesis
    by (simp add: fls_conv_radius_altdef G_def)
qed

```

lemma eval_fls_eqI:

```

  assumes f has_laurent_expansion F f holomorphic_on eball 0 r - {0}
  assumes z  $\in$  eball 0 r - {0}
  shows eval_fls F z = f z
proof -
  have conv: fls_conv_radius F  $\geq$  r
    by (intro fls_conv_radius_ge[OF assms(1,2)])
  have ( $\lambda z$ . eval_fls F z - f z) has_laurent_expansion F - F
    using assms by (intro laurent_expansion_intros assms) (auto simp: has_laurent_expansion_def)
  hence ( $\lambda z$ . eval_fls F z - f z) has_laurent_expansion 0
    by simp
  hence eval_fls F z - f z = 0
  proof (rule has_laurent_expansion_0_analytic_continuation)
    have ereal 0  $\leq$  ereal (norm z)
      by simp
    also have norm z < r
      using assms by auto
    finally have r > 0
      by (simp add: zero_ereal_def)
    thus open (eball 0 r :: complex set) connected (eball 0 r :: complex set)
      0  $\in$  eball 0 r z  $\in$  eball 0 r - {0}
      using assms by (auto simp: zero_ereal_def)
    qed (auto intro!: holomorphic_intros assms less_le_trans[OF _ conv] split:
if_splits)
    thus ?thesis by simp
  qed
qed

```

lemma fls_nth_as_contour_integral:

```

  assumes F: f has_laurent_expansion F
  assumes holo: f holomorphic_on ball 0 r - {0}
  assumes R: 0 < R R < r
  shows (( $\lambda z$ . f z * z powi (-(n+1))) has_contour_integral
    complex_of_real (2 * pi) * i * fls_nth F n) (circlepath 0 R)
proof -

```

```

define  $I$  where  $I = (\lambda z. f\ z * z\ \text{powi}\ (-(n+1)))$ 
  have ( $I$  has_contour_integral_complex_of_real ( $2 * \pi$ ) *  $i$  * residue  $I$   $0$ )
(circlepath  $0$   $R$ )
proof (rule base_residue)
  show open (ball ( $0::\text{complex}$ )  $r$ )  $0 \in \text{ball}\ (0::\text{complex})\ r$ 
  using  $R$   $F$  by (auto simp: has_laurent_expansion_def zero_ereal_def)
qed (use  $R$  in  $\langle \text{auto intro!; holomorphic_intros holomorphic_on_subset}[OF\ \text{holo}]$ 
      simp: I_def split: if_splits)
also have residue  $I$   $0 = \text{fls\_residue}\ (\text{fls\_shift}\ (n + 1)\ F)$ 
unfolding  $I\_def$  by (intro has_laurent_expansion_residue_0 laurent_expansion_intros
 $F$ )
also have  $\dots = \text{fls\_nth}\ F\ n$ 
by simp
finally show ?thesis
by (simp add: I_def)
qed

lemma tendsto_0_subdegree_iff_0:
assumes  $F: f$  has_laurent_expansion  $F$  and  $F \neq 0$ 
shows ( $f \rightarrow 0 \rightarrow 0$ )  $\longleftrightarrow \text{fls\_subdegree}\ F > 0$ 
proof -
  have ?thesis if is_pole  $f$   $0$ 
proof -
    have  $\text{fls\_subdegree}\ F < 0$ 
    using is_pole_0_imp_neg_fls_subdegree [ $OF\ F\ that$ ] .
    moreover then have  $\neg f \rightarrow 0 \rightarrow 0$ 
    using  $\langle \text{is\_pole}\ f\ 0 \rangle\ F$  at_neq_bot
      has_laurent_expansion_imp_filterlim_infinity_0
      not_tendsto_and_filterlim_at_infinity that
    by blast
    ultimately show ?thesis by auto
  qed
moreover have ?thesis if not_is_pole  $f$   $0 \exists x. f \rightarrow 0 \rightarrow x$ 
proof -
  have  $\text{fls\_subdegree}\ F \geq 0$ 
  using has_laurent_expansion_imp_is_pole_0 [ $OF\ F$ ] that(1)
  by linarith
  have  $f \rightarrow 0 \rightarrow 0$  if  $\text{fls\_subdegree}\ F > 0$ 
  using fls_eq0_below_subdegree [ $OF\ that$ ]
  by (metis  $F \langle 0 \leq \text{fls\_subdegree}\ F \rangle$  has_laurent_expansion_imp_tendsto_0)
  moreover have  $\text{fls\_subdegree}\ F > 0$  if  $f \rightarrow 0 \rightarrow 0$ 
proof -
    have False if  $\text{fls\_subdegree}\ F = 0$ 
    proof -
      have  $f \rightarrow 0 \rightarrow \text{fls\_nth}\ F\ 0$ 
      using has_laurent_expansion_imp_tendsto_0
        [ $OF\ F\ \langle \text{fls\_subdegree}\ F \geq 0 \rangle$ ] .
      then have  $\text{fls\_nth}\ F\ 0 = 0$  using  $\langle f \rightarrow 0 \rightarrow 0 \rangle$ 
      using LIM_unique by blast
    
```



```

    then have  $F = 0$ 
      using nth_fls_subdegree_zero_iff ‹fls_subdegree  $F = 0$ ›
      by metis
    with ‹ $F \neq 0$ › show False by auto
  qed
  with ‹fls_subdegree  $F \geq 0$ ›
  show ?thesis by fastforce
  qed
  ultimately show ?thesis by auto
  qed
  moreover have is_pole  $f\ 0 \vee (\exists x. f - 0 \rightarrow x)$ 
  proof -
    have not_essential  $f\ 0$ 
      using  $F$  has_laurent_expansion_not_essential_0 by auto
    then show ?thesis unfolding not_essential_def
      by auto
  qed
  ultimately show ?thesis by auto
  qed

lemma tendsto_0_subdegree_iff:
  assumes  $F: (\lambda w. f\ (z+w))$  has_laurent_expansion  $F$  and  $F \neq 0$ 
  shows  $(f - z \rightarrow 0) \longleftrightarrow \text{fls\_subdegree } F > 0$ 
  apply (subst Lim_at_zero)
  apply (rule tendsto_0_subdegree_iff_0)
  using assms by auto

lemma is_pole_0_deriv_divide_iff:
  assumes  $F: f$  has_laurent_expansion  $F$  and  $F \neq 0$ 
  shows is_pole  $(\lambda x. \text{deriv } f\ x / f\ x)\ 0 \longleftrightarrow \text{is\_pole } f\ 0 \vee (f - 0 \rightarrow 0)$ 
  proof -
    have  $(\lambda x. \text{deriv } f\ x / f\ x)$  has_laurent_expansion fls_deriv  $F / F$ 
      using  $F$  by (auto intro:laurent_expansion_intros)

    have is_pole  $(\lambda x. \text{deriv } f\ x / f\ x)\ 0 \longleftrightarrow$ 
      fls_subdegree (fls_deriv  $F / F$ )  $< 0$ 
      apply (rule is_pole_fls_subdegree_iff)
      using  $F$  by (auto intro:laurent_expansion_intros)
    also have ...  $\longleftrightarrow \text{is\_pole } f\ 0 \vee (f - 0 \rightarrow 0)$ 
  proof (cases fls_subdegree  $F = 0$ )
    case True
      then have fls_subdegree (fls_deriv  $F / F$ )  $\geq 0$ 
        by (metis diff_zero div_0 ‹ $F \neq 0$ › fls_deriv_subdegree0
          fls_divide_subdegree)
      moreover then have  $\neg \text{is\_pole } f\ 0$ 
        by (metis  $F$  True is_pole_0_imp_neg_fls_subdegree less_le)
      moreover have  $\neg (f - 0 \rightarrow 0)$ 
        using tendsto_0_subdegree_iff_0[OF  $F$  ‹ $F \neq 0$ ›] True by auto
      ultimately show ?thesis by auto
  end
  end

```

```

next
  case False
  then have fls_deriv F  $\neq$  0
    by (metis fls_const_subdegree fls_deriv_eq_0_iff)
  then have fls_subdegree (fls_deriv F / F) =
    fls_subdegree (fls_deriv F) - fls_subdegree F
    by (rule fls_divide_subdegree[OF  $\langle F \neq 0 \rangle$ ])
  moreover have fls_subdegree (fls_deriv F) = fls_subdegree F - 1
    using fls_subdegree_deriv[OF False] .
  ultimately have fls_subdegree (fls_deriv F / F) < 0 by auto
  moreover have  $f - 0 \rightarrow 0 = (0 < \text{fls\_subdegree } F)$ 
    using tendsto_0_subdegree_iff_0[OF F  $\langle F \neq 0 \rangle$ ] .
  moreover have is_pole f 0 = (fls_subdegree F < 0)
    using is_pole_fls_subdegree_iff F by auto
  ultimately show ?thesis using False by auto
qed
finally show ?thesis .
qed

lemma is_pole_deriv_divide_iff:
  assumes F:  $(\lambda w. f (z+w))$  has_laurent_expansion F and  $F \neq 0$ 
  shows is_pole  $(\lambda x. \text{deriv } f x / f x) z \longleftrightarrow \text{is\_pole } f z \vee (f - z \rightarrow 0)$ 
proof -
  define ff df where ff =  $(\lambda w. f (z+w))$  and df =  $(\lambda w. \text{deriv } f (z + w))$ 
  have is_pole  $(\lambda x. \text{deriv } f x / f x) z$ 
     $\longleftrightarrow \text{is\_pole } (\lambda x. \text{deriv } ff x / ff x) 0$ 
    unfolding ff_def df_def
    by (simp add: deriv_shift_0' is_pole_shift_0' comp_def algebra_simps)
  moreover have is_pole f z  $\longleftrightarrow \text{is\_pole } ff 0$ 
    unfolding ff_def by (auto simp: is_pole_shift_0')
  moreover have  $(f - z \rightarrow 0) \longleftrightarrow (ff - 0 \rightarrow 0)$ 
    unfolding ff_def by (simp add: LIM_offset_zero_iff)
  moreover have is_pole  $(\lambda x. \text{deriv } ff x / ff x) 0 = (\text{is\_pole } ff 0 \vee ff - 0 \rightarrow 0)$ 
    apply (rule is_pole_0_deriv_divide_iff)
    using F ff_def  $\langle F \neq 0 \rangle$  by blast+
  ultimately show ?thesis by auto
qed

lemma subdegree_imp_eventually_deriv_nzero_0:
  assumes F: f has_laurent_expansion F and fls_subdegree F  $\neq$  0
  shows eventually  $(\lambda z. \text{deriv } f z \neq 0)$  (at 0)
proof -
  have deriv f has_laurent_expansion fls_deriv F
    using has_laurent_expansion_deriv[OF F] .
  moreover have fls_deriv F  $\neq$  0
    using  $\langle \text{fls\_subdegree } F \neq 0 \rangle$ 
    by (metis fls_const_subdegree fls_deriv_eq_0_iff)
  ultimately show ?thesis
    using has_laurent_expansion_eventually_nonzero_iff' by blast

```

qed

lemma *subdegree_imp_eventually_deriv_nzero*:
 assumes $F: (\lambda w. f (z+w)) \text{ has_laurent_expansion } F$
 and $\text{fls_subdegree } F \neq 0$
 shows *eventually* $(\lambda w. \text{deriv } f w \neq 0) \text{ (at } z)$
proof –
 have $\forall_F x \text{ in at } 0. \text{deriv } (\lambda w. f (z+w)) x \neq 0$
 using *subdegree_imp_eventually_deriv_nzero_0* **assms** **by** *auto*
 then **show** *?thesis*
 apply (*subst eventually_at_to_0*)
 by (*simp add: deriv_shift_0' comp_def algebra_simps*)

qed

lemma *has_fps_expansion_imp_asymp_equiv_0*:
 fixes $f :: \text{complex} \Rightarrow \text{complex}$
 assumes $F: f \text{ has_fps_expansion } F$
 defines $n \equiv \text{subdegree } F$
 shows $f \sim[\text{nhds } 0] (\lambda z. \text{fps_nth } F n * z^n)$
proof –
 have $F': f \text{ has_laurent_expansion fps_to_fls } F$
 using $F \text{ has_laurent_expansion_fps}$ **by** *blast*

 have $f \sim[\text{at } 0] (\lambda z. \text{fps_nth } F n * z^n)$
 using *has_laurent_expansion_imp_asymp_equiv_0[OF F']*
 by (*simp add: fls_subdegree_fls_to_fps_n_def*)
 moreover have $f 0 = \text{fps_nth } F n * 0^n$
 using F **by** (*auto simp: n_def has_fps_expansion_to_laurent_power_0_left*)
 ultimately **show** *?thesis*
 by (*auto simp: asymp_equiv_nhds_iff*)

qed

lemma *has_fps_expansion_imp_tendsto_0*:
 fixes $f :: \text{complex} \Rightarrow \text{complex}$
 assumes $f \text{ has_fps_expansion } F$
 shows $(f \longrightarrow \text{fps_nth } F 0) \text{ (nhds } 0)$
proof (*rule asymp_equiv_tendsto_transfer*)
 show $(\lambda z. \text{fps_nth } F (\text{subdegree } F) * z^{\text{subdegree } F}) \sim[\text{nhds } 0] f$
 by (*rule asymp_equiv_symI, rule has_fps_expansion_imp_asymp_equiv_0*)
fact
 have $((\lambda z. F \$ \text{subdegree } F * z^{\text{subdegree } F}) \longrightarrow F \$ 0) \text{ (at } 0)$
 by (*rule tendsto_eq_intros refl | simp*) + (*auto simp: power_0_left*)
 thus $((\lambda z. F \$ \text{subdegree } F * z^{\text{subdegree } F}) \longrightarrow F \$ 0) \text{ (nhds } 0)$
 by (*auto simp: tendsto_nhds_iff power_0_left*)

qed

lemma *has_fps_expansion_imp_0_eq_fps_nth_0*:
 assumes $f \text{ has_fps_expansion } F$
 shows $f 0 = \text{fps_nth } F 0$

proof –

```

  have eventually ( $\lambda x. f\ x = \text{eval\_fps}\ F\ x$ ) (nhds 0)
    using assms by (auto simp: has_fps_expansion_def eq_commute)
  then obtain A where open A 0  $\in A$   $\forall x \in A. f\ x = \text{eval\_fps}\ F\ x$ 
    unfolding eventually_nhds by blast
  hence  $f\ 0 = \text{eval\_fps}\ F\ 0$ 
    by blast
  thus ?thesis
    by (simp add: eval_fps_at_0)

```

qed

lemma *fls_nth_compose_aux*:

```

  assumes f_has_fps_expansion F
  assumes G:  $g$  has_fps_expansion  $G$   $\text{fps\_nth}\ G\ 0 = 0$   $\text{fps\_deriv}\ G \neq 0$ 
  assumes (f  $\circ$  g) has_laurent_expansion H
  shows fls_nth H (int n) =  $\text{fps\_nth}\ (\text{fps\_compose}\ F\ G)\ n$ 
    using assms(1,5)
proof (induction n arbitrary: f F H rule: less_induct)
  case (less n f F H)
  have [simp]:  $g\ 0 = 0$ 
    using has_fps_expansion_imp_0_eq_fps_nth_0[OF G(1)] G(2) by simp
  have ana_f:  $f$  analytic_on {0}
    using less.prem1 by (meson has_fps_expansion_imp_analytic_0)
  have ana_g:  $g$  analytic_on {0}
    using G by (meson has_fps_expansion_imp_analytic_0)
  have (f  $\circ$  g) has_laurent_expansion  $\text{fps\_to\_fls}\ (\text{fps\_expansion}\ (f \circ g)\ 0)$ 
    by (rule analytic_at_imp_has_fps_expansion_0 analytic_intros has_laurent_expansion_fps
        analytic_on_compose_gen ana_f ana_g) + auto
  with less.prem2 have  $H = \text{fps\_to\_fls}\ (\text{fps\_expansion}\ (f \circ g)\ 0)$ 
    using has_laurent_expansion_unique by blast
  also have fls_subdegree ...  $\geq 0$ 
    by (simp add: fls_subdegree_fps_to_fps)
  finally have subdeg: fls_subdegree H  $\geq 0$  .

```

show ?case

proof (cases n = 0)

case [simp]: True

have $\lim_g: g \rightarrow 0 \rightarrow 0$

using has_laurent_expansion_imp_tendsto_0[of g $\text{fps_to_fls}\ G$] G

by (auto simp: fls_subdegree_fps_to_fps_gt0 has_fps_expansion_to_laurent)

have $\lim_f: (f \longrightarrow \text{fps_nth}\ F\ 0)$ (nhds 0)

by (intro has_fps_expansion_imp_tendsto_0 less.prem3)

have $(\lambda x. f\ (g\ x)) \rightarrow 0 \rightarrow \text{fps_nth}\ F\ 0$

by (rule filterlim_compose[OF $\lim_f\ \lim_g$])

moreover have $(f \circ g) \rightarrow 0 \rightarrow \text{fps_nth}\ H\ 0$

by (intro has_laurent_expansion_imp_tendsto_0 less.prem2 subdeg)

ultimately have $\text{fps_nth}\ F\ 0 = \text{fps_nth}\ H\ 0$

using tendsto_unique by (force simp: o_def)

thus ?thesis

```

    by simp
  next
    case n: False
    define GH where GH = (fls_deriv H / fls_deriv (fps_to_fls G))
    define GH' where GH' = fls_regpart GH

    have ( $\lambda x. \text{deriv } (f \circ g) x / \text{deriv } g x$ ) has_laurent_expansion
      fls_deriv H / fls_deriv (fps_to_fls G)
    by (intro laurent_expansion_intros less.prem has_laurent_expansion_fps[of
      _ G] G)
    also have ?this  $\longleftrightarrow$  ( $\text{deriv } f \circ g$ ) has_laurent_expansion fls_deriv H /
      fls_deriv (fps_to_fls G)
    proof (rule has_laurent_expansion_cong)
      from ana_f obtain r1 where r1:  $r1 > 0$  f holomorphic_on ball 0 r1
      unfolding analytic_on_def by blast
      from ana_g obtain r2 where r2:  $r2 > 0$  g holomorphic_on ball 0 r2
      unfolding analytic_on_def by blast
      have lim_g:  $g \rightarrow 0 \rightarrow 0$ 
      using has_laurent_expansion_imp_tendsto_0[of g fps_to_fls G] G
      by (auto simp: fls_subdegree_fls_to_fps_gt0 has_fps_expansion_to_laurent)
      moreover have open (ball 0 r1)  $0 \in \text{ball } 0 r1$ 
      using r1 by auto
      ultimately have eventually ( $\lambda x. g x \in \text{ball } 0 r1$ ) (at 0)
      unfolding tendsto_def by blast
      moreover have eventually ( $\lambda x. \text{deriv } g x \neq 0$ ) (at 0)
      using G fps_to_fls_eq_0_iff has_fps_expansion_deriv has_fps_expansion_to_laurent
        has_laurent_expansion_nonzero_imp_eventually_nonzero by blast
      moreover have eventually ( $\lambda x. x \in \text{ball } 0 (\min r1 r2) - \{0\}$ ) (at 0)
      by (intro eventually_at_in_open) (use r1 r2 in auto)
      ultimately show eventually ( $\lambda x. \text{deriv } (f \circ g) x / \text{deriv } g x = (\text{deriv } f \circ g)$ 
x) (at 0)
      proof eventually_elim
        case (elim x)
        thus ?case using r1 r2
        by (subst deriv_chain)
          (auto simp: field_simps holomorphic_on_def at_within_open[of _ ball
      _])
      qed
    qed auto
    finally have GH: ( $\text{deriv } f \circ g$ ) has_laurent_expansion GH
      unfolding GH_def .

    have ( $\text{deriv } f \circ g$ ) has_laurent_expansion fps_to_fls (fps_expansion (deriv f
 $\circ g$ ) 0)
    by (rule analytic_at_imp_has_fps_expansion_0 analytic_intros has_laurent_expansion_fps
      analytic_on_compose_gen ana_f ana_g)+ auto
    with GH have GH = fps_to_fls (fps_expansion (deriv f  $\circ g$ ) 0)
    using has_laurent_expansion_unique by blast
    also have fls_subdegree ...  $\geq 0$ 

```

```

    by (simp add: fls_subdegree_fls_to_fps)
    finally have subdeg': fls_subdegree GH  $\geq$  0 .

    have deriv f has_fps_expansion fps_deriv F
    by (intro fps_expansion_intros less.premis)
    from this and GH have IH: fls_nth GH (int k) = fps_nth (fps_compose
(fps_deriv F) G) k
    if k < n for k
    by (intro less.IH that)

    have fps_nth (fps_compose (fps_deriv F) G) n = ( $\sum$  i=0..n. of_nat (Suc i)
* F $ Suc i * G ^ i $ n)
    by (simp add: fps_compose_nth)

    have fps_nth (fps_compose F G) n =
      fps_nth (fps_deriv (fps_compose F G)) (n - 1) / of_nat n
    using n by (cases n) (auto simp del: of_nat_Suc)
    also have fps_deriv (fps_compose F G) = fps_compose (fps_deriv F) G *
fps_deriv G
    using G by (subst fps_compose_deriv) auto
    also have fps_nth ... (n - 1) = ( $\sum$  i=0..n-1. (fps_deriv F oo G) $ i *
fps_deriv G $ (n - 1 - i))
    unfolding fps_mult_nth ..
    also have ... = ( $\sum$  i=0..n-1. fps_nth GH' i * of_nat (n - i) * G $ (n -
i))
    using n by (intro sum.cong) (auto simp: IH Suc_diff_Suc GH'_def)
    also have ... = ( $\sum$  i=0..n. fps_nth GH' i * of_nat (n - i) * G $ (n - i))
    by (intro sum.mono_neutral_left) auto
    also have ... = fps_nth (GH' * Abs_fps ( $\lambda$ i. of_nat i * fps_nth G i)) n
    by (simp add: fps_mult_nth mult_ac)
    also have Abs_fps ( $\lambda$ i. of_nat i * fps_nth G i) = fps_X * fps_deriv G
    by (simp add: fps_mult_fps_X_deriv_shift)
    also have fps_nth (GH' * (fps_X * fps_deriv G)) n =
      fls_nth (fps_to_fls (GH' * (fps_X * fps_deriv G))) (int n)
    by simp
    also have fps_to_fls (GH' * (fps_X * fps_deriv G)) =
      GH * fps_to_fls (fps_deriv G) * fls_X
    using subdeg' by (simp add: mult_ac fls_times_fps_to_fls GH'_def)
    also have GH * fps_to_fls (fps_deriv G) = fls_deriv H
    unfolding GH_def using G by (simp add: fls_deriv_fps_to_fls)
    also have fls_deriv H * fls_X = fls_shift (-1) (fls_deriv H)
    using fls_X_times_conv_shift(2) by blast
    finally show ?thesis
    using n by simp
qed
qed

lemma has_fps_expansion_compose [fps_expansion_intros]:
  fixes f g :: complex  $\Rightarrow$  complex

```

```

assumes  $F$ :  $f$  has_fps_expansion  $F$ 
assumes  $G$ :  $g$  has_fps_expansion  $G$  fps_nth  $G$  0 = 0
shows  $(f \circ g)$  has_fps_expansion fps_compose  $F$   $G$ 
proof (cases fps_deriv  $G$  = 0)
  case False
    have [simp]:  $g$  0 = 0
      using has_fps_expansion_imp_0_eq_fps_nth_0[OF  $G$ (1)]  $G$ (2) False by
simp
    have  $ana\_f$ :  $f$  analytic_on {0}
      using  $F$  by (meson has_fps_expansion_imp_analytic_0)
    have  $ana\_g$ :  $g$  analytic_on {0}
      using  $G$  by (meson has_fps_expansion_imp_analytic_0)
    have  $fg$ :  $(f \circ g)$  has_fps_expansion fps_expansion  $(f \circ g)$  0
      by (rule analytic_at_imp_has_fps_expansion_0 analytic_intros
        analytic_on_compose_gen  $ana\_f$   $ana\_g$ ) + auto

    have  $fls\_nth$  (fps_to_fls (fps_expansion  $(f \circ g)$  0)) (int  $n$ ) = fps_nth (fps_compose
 $F$   $G$ )  $n$  for  $n$ 
      by (rule fls_nth_compose_aux has_laurent_expansion_fps  $F$   $G$  False  $fg$ ) +
    hence fps_expansion  $(f \circ g)$  0 = fps_compose  $F$   $G$ 
      by (simp add: fps_eq_iff)
    thus ?thesis using  $fg$ 
      by simp
  next
    case True
    have [simp]:  $f$  0 = fps_nth  $F$  0
      using  $F$  by (auto dest: has_fps_expansion_imp_0_eq_fps_nth_0)
    from True have fps_nth  $G$   $n$  = 0 for  $n$ 
      using  $G$ (2) by (cases  $n$ ) (auto simp del: of_nat_Suc)
    hence [simp]:  $G$  = 0
      by (auto simp: fps_eq_iff)
    have  $(\lambda \_. f$  0) has_fps_expansion fps_const  $(f$  0)
      by (intro fps_expansion_intros)
    also have eventually  $(\lambda x. g$   $x$  = 0) (nhds 0)
      using  $G$  by (auto simp: has_fps_expansion_def)
    hence  $(\lambda \_. f$  0) has_fps_expansion fps_const  $(f$  0)  $\longleftrightarrow$   $(f \circ g)$  has_fps_expansion
fps_const  $(f$  0)
      by (intro has_fps_expansion_cong) (auto elim!: eventually_mono)
    thus ?thesis
      by simp
qed

lemma has_fps_expansion_fps_to_fls:
  assumes  $f$  has_laurent_expansion fps_to_fls  $F$ 
  shows  $(\lambda z. \text{if } z = 0 \text{ then fps_nth } F$  0 else  $f$   $z$ ) has_fps_expansion  $F$ 
  (is ? $f'$  has_fps_expansion  $\_$ )
proof –
  have  $f$  has_laurent_expansion fps_to_fls  $F$   $\longleftrightarrow$  ? $f'$  has_laurent_expansion

```

```

fps_to_fls F
  by (intro has_laurent_expansion_cong) (auto simp: eventually_at_filter)
  with assms show ?thesis
  by (auto simp: has_fps_expansion_to_laurent)
qed

lemma has_laurent_expansion_compose [laurent_expansion_intros]:
  fixes f g :: complex  $\Rightarrow$  complex
  assumes F: f has_laurent_expansion F
  assumes G: g has_laurent_expansion fps_to_fls G fps_nth G 0 = 0 G  $\neq$  0
  shows (f  $\circ$  g) has_laurent_expansion fls_compose_fps F G
proof -
  from assms have lim_g: g  $\rightarrow$  0
  by (subst tendsto_0_subdegree_iff_0[OF G(1)])
  (auto simp: fls_subdegree_fls_to_fps_subdegree_pos_iff)
  have ev1: eventually ( $\lambda z. g z \neq 0$ ) (at 0)
  using  $\langle G \neq 0 \rangle$  G(1) fps_to_fls_eq_0_iff has_laurent_expansion_fps
  has_laurent_expansion_nonzero_imp_eventually_nonzero by blast
  moreover have eventually ( $\lambda z. z \neq 0$ ) (at (0 :: complex))
  by (auto simp: eventually_at_filter)
  ultimately have ev: eventually ( $\lambda z. z \neq 0 \wedge g z \neq 0$ ) (at 0)
  by eventually_elim blast
  from ev1 and lim_g have lim_g': filterlim g (at 0) (at 0)
  by (auto simp: filterlim_at)
  define g' where g' = ( $\lambda z. \text{if } z = 0 \text{ then } \text{fps\_nth } G \ 0 \text{ else } g z$ )

  show ?thesis
  proof (cases F = 0)
    assume [simp]: F = 0
    have eventually ( $\lambda z. f z = 0$ ) (at 0)
    using F by (auto simp: has_laurent_expansion_def)
    hence eventually ( $\lambda z. f (g z) = 0$ ) (at 0)
    using lim_g' by (rule eventually_compose_filterlim)
    thus ?thesis
    by (auto simp: has_laurent_expansion_def)
  next
    assume [simp]: F  $\neq$  0
    define n where n = fls_subdegree F
    define f' where
      f' = ( $\lambda z. \text{if } z = 0 \text{ then } \text{fps\_nth } (\text{fls\_base\_factor\_to\_fps } F) \ 0 \text{ else } f z * z^{\text{powi } n}$ )
    have (( $\lambda z. (f' \circ g') z * g z^{\text{powi } n}$ )) has_laurent_expansion fls_compose_fps
    F G
    unfolding f'_def n_def fls_compose_fps_def g'_def
    by (intro fps_expansion_intros laurent_expansion_intros has_fps_expansion_fps_to_fls
      has_fps_expansion_fls_base_factor_to_fps assms has_laurent_expansion_fps)
    also have ?this  $\longleftrightarrow$  ?thesis
    by (intro has_laurent_expansion_cong eventually_mono[OF ev])
    (auto simp: f'_def power_int_minus g'_def)
  end
end

```


finally show ?thesis .
 qed
 qed

lemma has_laurent_expansion_fls_X_inv [laurent_expansion_intros]:
 inverse has_laurent_expansion_fls_X_inv
 using has_laurent_expansion_inverse[OF has_laurent_expansion_fps_X]
 by (simp add: fls_inverse_X)

lemma zorder_times_analytic:
 assumes f analytic_on $\{z\}$ g analytic_on $\{z\}$
 assumes eventually $(\lambda z. f z * g z \neq 0)$ (at z)
 shows $\text{zorder } (\lambda z. f z * g z) z = \text{zorder } f z + \text{zorder } g z$
proof –
 have *: $(\lambda w. f (z + w)) \text{ has_fps_expansion } \text{fps_expansion } f z$
 $(\lambda w. g (z + w)) \text{ has_fps_expansion } \text{fps_expansion } g z$
 $(\lambda w. f (z + w) * g (z + w)) \text{ has_fps_expansion } \text{fps_expansion } f z *$
 $\text{fps_expansion } g z$
 by (intro fps_expansion_intros analytic_at_imp_has_fps_expansion assms)+
 have [simp]: $\text{fps_expansion } f z \neq 0$
proof
 assume $\text{fps_expansion } f z = 0$
 hence eventually $(\lambda z. f z * g z = 0)$ (at z) using *(1)
 by (auto simp: has_fps_expansion_0_iff nhds_to_0' eventually_filtermap
 eventually_at_filter
 elim: eventually_mono)
 with assms(3) have eventually $(\lambda z. \text{False})$ (at z)
 by eventually_elim auto
 thus False by simp
qed
 have [simp]: $\text{fps_expansion } g z \neq 0$
proof
 assume $\text{fps_expansion } g z = 0$
 hence eventually $(\lambda z. f z * g z = 0)$ (at z) using *(2)
 by (auto simp: has_fps_expansion_0_iff nhds_to_0' eventually_filtermap
 eventually_at_filter
 elim: eventually_mono)
 with assms(3) have eventually $(\lambda z. \text{False})$ (at z)
 by eventually_elim auto
 thus False by simp
qed
 from *[THEN has_fps_expansion_zorder] show ?thesis
 by auto
qed

lemma zorder_const [simp]: $c \neq 0 \implies \text{zorder } (\lambda_. c) z = 0$
 by (intro zorder_eqI[where $S = \text{UNIV}$]) auto

lemma zorder_prod_analytic:

```

assumes  $\bigwedge x. x \in A \implies f\ x \text{ analytic\_on } \{z\}$ 
assumes eventually  $(\lambda z. (\prod x \in A. f\ x\ z) \neq 0) \text{ (at } z)$ 
shows  $\text{zorder } (\lambda z. \prod x \in A. f\ x\ z)\ z = (\sum x \in A. \text{zorder } (f\ x)\ z)$ 
using assms
proof (induction A rule: infinite_finite_induct)
  case (insert x A)
    have  $\text{zorder } (\lambda z. f\ x\ z * (\prod x \in A. f\ x\ z))\ z = \text{zorder } (f\ x)\ z + \text{zorder } (\lambda z. \prod x \in A.$ 
 $f\ x\ z)\ z$ 
    using insert.premis insert.hyps by (intro zorder_times_analytic analytic_intros)
  auto
  also have  $\text{zorder } (\lambda z. \prod x \in A. f\ x\ z)\ z = (\sum x \in A. \text{zorder } (f\ x)\ z)$ 
  using insert.premis insert.hyps by (intro insert.IH) (auto elim!: eventually_mono)
  finally show ?case using insert
    by simp
qed auto

```

```

lemma zorder_eq_0I:
  assumes  $g \text{ analytic\_on } \{z\}$   $g\ z \neq 0$ 
  shows  $\text{zorder } g\ z = 0$ 
  using analytic_at assms zorder_eqI by fastforce

```

```

lemma zorder_pos_iff:
  assumes  $f \text{ holomorphic\_on } A$  open A  $z \in A$  frequently  $(\lambda z. f\ z \neq 0) \text{ (at } z)$ 
  shows  $\text{zorder } f\ z > 0 \iff f\ z = 0$ 
proof -
  have  $f \text{ analytic\_on } \{z\}$ 
  using assms analytic_at by blast
  hence  $*(\lambda w. f\ (z + w)) \text{ has\_fps\_expansion } \text{fps\_expansion } f\ z$ 
  using analytic_at_imp_has_fps_expansion by blast
  have  $\text{nz: fps\_expansion } f\ z \neq 0$ 
  proof
    assume  $\text{fps\_expansion } f\ z = 0$ 
    hence eventually  $(\lambda z. f\ z = 0) \text{ (nhds } z)$ 
    using  $*$  by (auto simp: has_fps_expansion_def nhds_to_0' eventually_filtermap
add_ac)
    hence eventually  $(\lambda z. f\ z = 0) \text{ (at } z)$ 
    by (auto simp: eventually_at_filter elim: eventually_mono)
    with assms show False
    by (auto simp: frequently_def)
  qed
  from has_fps_expansion_zorder[OF * this] have  $\text{eq: zorder } f\ z = \text{int } (\text{subdegree } (\text{fps\_expansion } f\ z))$ 
  by auto
  moreover have  $\text{subdegree } (\text{fps\_expansion } f\ z) = 0 \iff \text{fps\_nth } (\text{fps\_expansion } f\ z)\ 0 \neq 0$ 
  using nz by (auto simp: subdegree_eq_0_iff)
  moreover have  $\text{fps\_nth } (\text{fps\_expansion } f\ z)\ 0 = f\ z$ 
  by (auto simp: fps_expansion_def)
  ultimately show ?thesis

```

by auto
qed

lemma *zorder_pos_iff'*:
assumes *f analytic_on {z} frequently* ($\lambda z. f z \neq 0$) (*at z*)
shows $\text{zorder } f z > 0 \longleftrightarrow f z = 0$
using *analytic_at assms zorder_pos_iff* by blast

lemma *zorder_ge_0*:
assumes *f analytic_on {z} frequently* ($\lambda z. f z \neq 0$) (*at z*)
shows $\text{zorder } f z \geq 0$
proof –
have *: ($\lambda w. f (z + w)$) *has_laurent_expansion fps_to_fls* (*fps_expansion f z*)
using *assms* by (*simp add: analytic_at_imp_has_fps_expansion has_laurent_expansion_fps*)
from * *assms*(2) have *fps_to_fls* (*fps_expansion f z*) $\neq 0$
by (*auto simp: has_laurent_expansion_def frequently_def at_to_0' eventually_filtermap add_ac*)
with *has_laurent_expansion_zorder*[*OF* *] **show** ?thesis
by (*simp add: fls_subdegree_fls_to_fps*)
qed

lemma *zorder_eq_0_iff*:
assumes *f analytic_on {z} frequently* ($\lambda w. f w \neq 0$) (*at z*)
shows $\text{zorder } f z = 0 \longleftrightarrow f z \neq 0$
using *assms zorder_eq_0I zorder_pos_iff'* by fastforce

lemma *zorder_scale*:
assumes *f analytic_on {a * z} eventually* ($\lambda w. f w \neq 0$) (*at (a * z)*) *a* $\neq 0$
shows $\text{zorder } (\lambda w. f (a * w)) z = \text{zorder } f (a * z)$
proof –
from *assms*(1) **obtain** *r* **where** *r*: $r > 0$ *f holomorphic_on ball (a * z) r*
by (*auto simp: analytic_on_def*)
have *: *open* (*ball (a * z) r*) *connected* (*ball (a * z) r*) *a * z* \in *ball (a * z) r*
using *r* $\langle a \neq 0 \rangle$ by (*auto simp: dist_norm*)
from *assms*(2) have *eventually* ($\lambda w. f w \neq 0 \wedge w \in \text{ball } (a * z) r - \{a * z\}$)
(*at (a * z)*)
using $\langle r > 0 \rangle$ by (*intro eventually_conj eventually_at_in_open*) *auto*
then obtain *z0* **where** $f z0 \neq 0 \wedge z0 \in \text{ball } (a * z) r - \{a * z\}$
using *eventually_happens*[*of _ at (a * z)*] by force
hence **: $\exists w \in \text{ball } (a * z) r. f w \neq 0$
by blast

define *n* **where** *n* = *nat* (*zorder f (a * z)*)
obtain *r'* **where** *r'*:
(*if* *f (a * z)* = 0 *then* 0 < *zorder f (a * z)* *else* *zorder f (a * z)* = 0)
r' > 0 *cball* (*a * z*) *r'* \subseteq *ball* (*a * z*) *r* *zor_poly f (a * z) holomorphic_on*
cball (*a * z*) *r'*
 $\bigwedge w. w \in \text{cball } (a * z) r' \implies$
 $f w = \text{zor_poly } f (a * z) w * (w - a * z) ^ n \wedge \text{zor_poly } f (a * z) w \neq 0$

```

    unfolding n_def using zorder_exist_zero[OF r(2) * **] by blast

show ?thesis
proof (rule zorder_eqI)
  show open (ball z (r' / norm a)) z ∈ ball z (r' / norm a)
    using r ⟨r' > 0⟩ ⟨a ≠ 0⟩ by auto
  have (*) a ' ball z (r' / cmod a) ⊆ cball (a * z) r'
  proof safe
    fix w assume w ∈ ball z (r' / cmod a)
    thus a * w ∈ cball (a * z) r'
    using dist_mult_left[of a z w] ⟨a ≠ 0⟩ by (auto simp: divide_simps mult_ac)
  qed
  thus (λw. a ^ n * (zor_poly f (a * z) ∘ (λw. a * w)) w) holomorphic_on ball
    z (r' / norm a)
    using ⟨a ≠ 0⟩ by (intro holomorphic_on_compose_gen[OF _ r'(4)] holo-
      morphic_intros) auto
  show a ^ n * (zor_poly f (a * z) ∘ (λw. a * w)) z ≠ 0
    using r' ⟨a ≠ 0⟩ by auto
  show f (a * w) = a ^ n * (zor_poly f (a * z) ∘ (*) a) w * (w - z) powi (zorder
    f (a * z))
    if w ∈ ball z (r' / norm a) w ≠ z for w
  proof -
    have f (a * w) = zor_poly f (a * z) (a * w) * (a * (w - z)) ^ n
      using that r'(5)[of a * w] dist_mult_left[of a z w] ⟨a ≠ 0⟩ unfolding
        ring_distrib
      by (auto simp: divide_simps mult_ac)
    also have ... = a ^ n * zor_poly f (a * z) (a * w) * (w - z) ^ n
      by (subst power_mult_distrib) (auto simp: mult_ac)
    also have (w - z) ^ n = (w - z) powi of_nat n
      by simp
    also have of_nat n = zorder f (a * z)
      using r'(1) by (auto simp: n_def split: if_splits)
    finally show ?thesis
      unfolding o_def n_def .
  qed
qed
qed
qed

lemma zorder_compose_aux:
  assumes isolated_singularity_at f 0 not_essential f 0
  assumes G: g has_fps_expansion G G ≠ 0 g 0 = 0
  assumes eventually (λw. f w ≠ 0) (at 0)
  shows zorder (f ∘ g) 0 = zorder f 0 * subdegree G
proof -
  obtain F where F: f has_laurent_expansion F
    using not_essential_has_laurent_expansion_0[OF assms(1,2)] by blast
  have [simp]: fps_nth G 0 = 0
    using G ⟨g 0 = 0⟩ by (simp add: has_fps_expansion_imp_0_eq_fps_nth_0)
  note [simp] = ⟨G ≠ 0⟩ ⟨g 0 = 0⟩

```

```

have [simp]:  $F \neq 0$ 
using has_laurent_expansion_eventually_nonzero_iff[of  $f$  0  $F$ ]  $F$  assms by
simp
have  $FG: (f \circ g) \text{ has\_laurent\_expansion fls\_compose\_fps } F \ G$ 
by (intro has_laurent_expansion_compose has_laurent_expansion_fps  $F \ G$ )
auto

have  $\text{zorder } (f \circ g) \ 0 = \text{fls\_subdegree } (\text{fls\_compose\_fps } F \ G)$ 
using has_laurent_expansion_zorder_0 [OF  $FG$ ] by (auto simp: fls_compose_fps_eq_0_iff)
also have  $\dots = \text{fls\_subdegree } F * \text{int } (\text{subdegree } G)$ 
by simp
also have  $\text{fls\_subdegree } F = \text{zorder } f \ 0$ 
using has_laurent_expansion_zorder_0 [OF  $F$ ] by auto
finally show ?thesis .
qed

lemma zorder_compose:
assumes isolated_singularity_at  $f \ (g \ z)$  not_essential  $f \ (g \ z)$ 
assumes  $G: (\lambda x. g \ (z + x) - g \ z) \text{ has\_fps\_expansion } G \ G \neq 0$ 
assumes eventually  $(\lambda w. f \ w \neq 0)$  (at  $(g \ z)$ )
shows  $\text{zorder } (f \circ g) \ z = \text{zorder } f \ (g \ z) * \text{subdegree } G$ 
proof -
define  $f'$  where  $f' = (\lambda w. f \ (g \ z + w))$ 
define  $g'$  where  $g' = (\lambda w. g \ (z + w) - g \ z)$ 
have  $\text{zorder } f \ (g \ z) = \text{zorder } f' \ 0$ 
by (simp add: f'_def zorder_shift' add_ac)
have  $\text{zorder } (\lambda x. g \ x - g \ z) \ z = \text{zorder } g' \ 0$ 
by (simp add: g'_def zorder_shift' add_ac)
have  $\text{zorder } (f \circ g) \ z = \text{zorder } (f' \circ g') \ 0$ 
by (simp add: zorder_shift' f'_def g'_def add_ac o_def)
also have  $\dots = \text{zorder } f' \ 0 * \text{int } (\text{subdegree } G)$ 
proof (rule zorder_compose_aux)
show isolated_singularity_at  $f' \ 0$  unfolding f'_def
using assms has_laurent_expansion_isolated_0 not_essential_has_laurent_expansion
by blast
show not_essential  $f' \ 0$  unfolding f'_def
using assms has_laurent_expansion_not_essential_0 not_essential_has_laurent_expansion
by blast
qed (use assms in ⟨auto simp: f'_def g'_def at_to_0' eventually_filtermap
add_ac⟩)
also have  $\text{zorder } f' \ 0 = \text{zorder } f \ (g \ z)$ 
by (simp add: f'_def zorder_shift' add_ac)
finally show ?thesis .
qed

lemma fps_to_fls_eq_fls_const_iff [simp]:  $\text{fps\_to\_fls } F = \text{fls\_const } c \longleftrightarrow F = \text{fps\_const } c$ 
using fps_to_fls_eq_iff by fastforce

```

```

lemma zorder_compose':
  assumes isolated_singularity_at f (g z) not_essential f (g z)
  assumes g_analytic_on {z}
  assumes eventually ( $\lambda w. f w \neq 0$ ) (at (g z))
  assumes eventually ( $\lambda w. g w \neq g z$ ) (at z)
  shows zorder (f  $\circ$  g) z = zorder f (g z) * zorder ( $\lambda x. g x - g z$ ) z
proof -
  obtain G where G [fps_expansion_intros]: ( $\lambda x. g (z + x)$ ) has_fps_expansion G
  using assms analytic_at_imp_has_fps_expansion by blast
  have G': ( $\lambda x. g (z + x) - g z$ ) has_fps_expansion G - fps_const (g z)
  by (intro fps_expansion_intros)
  hence G'': ( $\lambda x. g (z + x) - g z$ ) has_laurent_expansion fps_to_fls (G - fps_const (g z))
  using has_laurent_expansion_fps by blast
  have nz: G - fps_const (g z)  $\neq 0$ 
  using has_laurent_expansion_eventually_nonzero_iff[OF G''] assms by auto
  have zorder (f  $\circ$  g) z = zorder f (g z) * subdegree (G - fps_const (g z))
  proof (rule zorder_compose)
    show ( $\lambda x. g (z + x) - g z$ ) has_fps_expansion G - fps_const (g z)
    by (intro fps_expansion_intros)
  qed (use assms nz in auto)
  also have int (subdegree (G - fps_const (g z))) = fls_subdegree (fps_to_fls G - fls_const (g z))
  by (simp flip: fls_subdegree_fls_to_fps)
  also have ... = zorder ( $\lambda x. g x - g z$ ) z
  using has_laurent_expansion_zorder [OF G''] nz by auto
  finally show ?thesis .
qed

lemma analytic_at_cong:
  assumes eventually ( $\lambda x. f x = g x$ ) (nhds x) x = y
  shows f_analytic_on {x}  $\longleftrightarrow$  g_analytic_on {y}
proof -
  have g_analytic_on {x} if f_analytic_on {x} eventually ( $\lambda x. f x = g x$ ) (nhds x)
for f g
  proof -
    have ( $\lambda y. f (x + y)$ ) has_fps_expansion fps_expansion f x
    by (rule analytic_at_imp_has_fps_expansion) fact
    also have ?this  $\longleftrightarrow$  ( $\lambda y. g (x + y)$ ) has_fps_expansion fps_expansion f x
    using that by (intro has_fps_expansion_cong refl) (auto simp: nhds_to_0' eventually_filtermap)
    finally show ?thesis
    by (rule has_fps_expansion_imp_analytic)
  qed
from this[of f g] this[of g f] show ?thesis using assms
  by (auto simp: eq_commute)
qed

```

lemma *has_laurent_expansion_sin'* [laurent_expansion_intros]:
 \sin has_laurent_expansion fps_to_fls (fps_sin 1)
 using has_fps_expansion_sin' has_fps_expansion_to_laurent by blast

lemma *has_laurent_expansion_cos'* [laurent_expansion_intros]:
 \cos has_laurent_expansion fps_to_fls (fps_cos 1)
 using has_fps_expansion_cos' has_fps_expansion_to_laurent by blast

lemma *has_laurent_expansion_sin* [laurent_expansion_intros]:
 $(\lambda z. \sin (c * z))$ has_laurent_expansion fps_to_fls (fps_sin c)
 by (intro has_laurent_expansion_fps has_fps_expansion_sin)

lemma *has_laurent_expansion_cos* [laurent_expansion_intros]:
 $(\lambda z. \cos (c * z))$ has_laurent_expansion fps_to_fls (fps_cos c)
 by (intro has_laurent_expansion_fps has_fps_expansion_cos)

lemma *has_laurent_expansion_tan'* [laurent_expansion_intros]:
 \tan has_laurent_expansion fps_to_fls (fps_tan 1)
 using has_fps_expansion_tan' has_fps_expansion_to_laurent by blast

lemma *has_laurent_expansion_tan* [laurent_expansion_intros]:
 $(\lambda z. \tan (c * z))$ has_laurent_expansion fps_to_fls (fps_tan c)
 by (intro has_laurent_expansion_fps has_fps_expansion_tan)

8.5 More Laurent expansions

lemma *has_laurent_expansion_frequently_zero_iff*:
 assumes $(\lambda w. f (z + w))$ has_laurent_expansion F
 shows frequently $(\lambda z. f z = 0)$ (at z) $\longleftrightarrow F = 0$
 using assms by (simp add: frequently_def has_laurent_expansion_eventually_nonzero_iff)

lemma *has_laurent_expansion_eventually_zero_iff*:
 assumes $(\lambda w. f (z + w))$ has_laurent_expansion F
 shows eventually $(\lambda z. f z = 0)$ (at z) $\longleftrightarrow F = 0$
 using assms
 by (metis has_laurent_expansion_frequently_zero_iff has_laurent_expansion_isolated)

$\text{has_laurent_expansion_not_essential laurent_expansion_def}$
 $\text{not_essential_frequently_0_imp_eventually_0 not_essential_has_laurent_expansion})$

lemma *has_laurent_expansion_frequently_nonzero_iff*:
 assumes $(\lambda w. f (z + w))$ has_laurent_expansion F
 shows frequently $(\lambda z. f z \neq 0)$ (at z) $\longleftrightarrow F \neq 0$
 using assms by (metis has_laurent_expansion_eventually_zero_iff not_eventually)

lemma *has_laurent_expansion_sum_list* [laurent_expansion_intros]:
 assumes $\bigwedge x. x \in \text{set } xs \implies f x$ has_laurent_expansion $F x$
 shows $(\lambda y. \sum x \leftarrow xs. f x y)$ has_laurent_expansion $(\sum x \leftarrow xs. F x)$
 using assms by (induction xs) (auto intro!: laurent_expansion_intros)

```

lemma has_laurent_expansion_prod_list [laurent_expansion_intros]:
  assumes  $\bigwedge x. x \in \text{set } xs \implies f\ x \text{ has\_laurent\_expansion } F\ x$ 
  shows  $(\lambda y. \prod x \leftarrow xs. f\ x\ y) \text{ has\_laurent\_expansion } (\prod x \leftarrow xs. F\ x)$ 
  using assms by (induction xs) (auto intro!: laurent_expansion_intros)

```

```

lemma has_laurent_expansion_sum_mset [laurent_expansion_intros]:
  assumes  $\bigwedge x. x \in \# I \implies f\ x \text{ has\_laurent\_expansion } F\ x$ 
  shows  $(\lambda y. \sum x \in \# I. f\ x\ y) \text{ has\_laurent\_expansion } (\sum x \in \# I. F\ x)$ 
  using assms by (induction I) (auto intro!: laurent_expansion_intros)

```

```

lemma has_laurent_expansion_prod_mset [laurent_expansion_intros]:
  assumes  $\bigwedge x. x \in \# I \implies f\ x \text{ has\_laurent\_expansion } F\ x$ 
  shows  $(\lambda y. \prod x \in \# I. f\ x\ y) \text{ has\_laurent\_expansion } (\prod x \in \# I. F\ x)$ 
  using assms by (induction I) (auto intro!: laurent_expansion_intros)

```

8.6 Formal convergence versus analytic convergence

The convergence of a sequence of formal power series and the convergence of the functions in the complex plane do not imply each other:

- If we have the sequence of constant power series $(1/n)_{n \geq 0}$, this clearly converges to the zero function analytically, but as a series of formal power series it is divergent (since the 0-th coefficient never stabilises).
- Conversely, the sequence of series $(n!x^n)_{n \geq 0}$ converges formally to 0, but the corresponding sequence of functions diverges for every $x \neq 0$.

However, if the sequence of series converges to some limit series h and the corresponding series of functions converges uniformly to some limit function $g(x)$, then h is also a series expansion of $g(x)$, i.e. in that case, formal and analytic convergence agree.

```

proposition uniform_limit_imp_fps_expansion_eq:
  fixes  $f :: 'a \Rightarrow \text{complex fps}$ 
  assumes lim1:  $(f \longrightarrow h)\ F$ 
  assumes lim2: uniform_limit A  $(\lambda x\ z. f'\ x\ z)\ g'\ F$ 
  assumes expansions: eventually  $(\lambda x. f'\ x \text{ has\_fps\_expansion } f\ x)\ F\ g' \text{ has\_fps\_expansion } g$ 
  assumes holo: eventually  $(\lambda x. f'\ x \text{ holomorphic\_on } A)\ F$ 
  assumes A: open A  $0 \in A$ 
  assumes nontriv [simp]:  $F \neq \text{bot}$ 
  shows  $g = h$ 
proof (rule fps_ext)
  fix  $n :: \text{nat}$ 
  have eventually  $(\lambda x. \text{fps\_nth } (f\ x)\ n = \text{fps\_nth } h\ n)\ F$ 
    using lim1 unfolding tendsto_fps_iff by blast
  hence eventually  $(\lambda x. (\text{deriv } \sim^n)\ (f'\ x)\ 0 / \text{fact } n = \text{fps\_nth } h\ n)\ F$ 

```



```

    using expansions(1)
  proof eventually_elim
    case (elim x)
    have fps_nth (f x) n = (deriv  $\sim$  n) (f' x) 0 / fact n
      by (rule fps_nth_fps_expansion) (use elim in auto)
    with elim show ?case
      by simp
  qed
  hence (( $\lambda$ x. (deriv  $\sim$  n) (f' x) 0 / fact n)  $\longrightarrow$  fps_nth h n) F
    by (simp add: tendsto_eventually)

  moreover have (( $\lambda$ x. (deriv  $\sim$  n) (f' x) 0)  $\longrightarrow$  (deriv  $\sim$  n) g' 0) F
    using lim2
  proof (rule higher_deriv_complex_uniform_limit)
    show eventually ( $\lambda$ x. f' x holomorphic_on A) F
      using holo by eventually_elim auto
  qed (use A in auto)
  hence (( $\lambda$ x. (deriv  $\sim$  n) (f' x) 0 / fact n)  $\longrightarrow$  (deriv  $\sim$  n) g' 0 / fact n) F
    by (intro tendsto_divide) auto

  ultimately have fps_nth h n = (deriv  $\sim$  n) g' 0 / fact n
    using tendsto_unique[OF nontriv] by blast
  also have ... = fps_nth g n
    by (rule fps_nth_fps_expansion [symmetric]) fact
  finally show fps_nth g n = fps_nth h n ..
qed

end

theory Meromorphic imports
  Laurent_Convergence
  Cauchy_Integral_Formula
  HOL-Analysis.Sparse_In
begin

```

8.7 Remove singular points

This function takes a complex function and returns a version of it where all removable singularities have been removed and all other singularities (to be more precise, unremovable discontinuities) are set to 0.

This is very useful since it is sometimes difficult to avoid introducing removable singularities. For example, consider the meromorphic functions $f(z) = z$ and $g(z) = 1/z$. Then a mathematician would write $f(z)g(z) = 1$, but in Isabelle of course this is not so.

Using the `remove_sings` function, we indeed have $\text{remove_sings } (\lambda z. f z * g z) = (\lambda _. 1)$.

definition `remove_sings` :: $(\text{complex} \Rightarrow \text{complex}) \Rightarrow \text{complex} \Rightarrow \text{complex}$ **where**

$\text{remove_sings } f z = (\text{if } \exists c. f - z \rightarrow c \text{ then } \text{Lim } (\text{at } z) f \text{ else } 0)$

lemma *remove_sings_eqI* [intro]:

assumes $f - z \rightarrow c$

shows $\text{remove_sings } f z = c$

using *assms* **unfolding** *remove_sings_def* **by** (auto simp: *tendsto_Lim*)

lemma *remove_sings_at_analytic* [simp]:

assumes $f \text{ analytic_on } \{z\}$

shows $\text{remove_sings } f z = f z$

using *assms* **by** (intro *remove_sings_eqI*) (simp add: *analytic_at_imp_isCont isContD*)

lemma *remove_sings_at_pole* [simp]:

assumes $\text{is_pole } f z$

shows $\text{remove_sings } f z = 0$

using *assms* **unfolding** *remove_sings_def is_pole_def*

by (*meson at_neq_bot not_tendsto_and_filterlim_at_infinity*)

lemma *eventually_remove_sings_eq_at*:

assumes $\text{isolated_singularity_at } f z$

shows $\text{eventually } (\lambda w. \text{remove_sings } f w = f w) (\text{at } z)$

proof –

from *assms* **obtain** r **where** $r: r > 0$ $f \text{ analytic_on ball } z r - \{z\}$

by (auto simp: *isolated_singularity_at_def*)

hence $*$: $f \text{ analytic_on } \{w\}$ **if** $w \in \text{ball } z r - \{z\}$ **for** w

using r **that** **by** (auto intro: *analytic_on_subset*)

have $\text{eventually } (\lambda w. w \in \text{ball } z r - \{z\}) (\text{at } z)$

using r **by** (intro *eventually_at_in_open*) auto

thus *?thesis*

by *eventually_elim* (auto simp: *remove_sings_at_analytic **)

qed

lemma *eventually_remove_sings_eq_nhds*:

assumes $f \text{ analytic_on } \{z\}$

shows $\text{eventually } (\lambda w. \text{remove_sings } f w = f w) (\text{nhds } z)$

proof –

from *assms* **obtain** A **where** $A: \text{open } A \ z \in A \ f \text{ holomorphic_on } A$

by (auto simp: *analytic_at*)

have $\text{eventually } (\lambda z. z \in A) (\text{nhds } z)$

by (intro *eventually_nhds_in_open A*)

thus *?thesis*

proof *eventually_elim*

case (*elim w*)

from *elim* **have** $f \text{ analytic_on } \{w\}$

using $A \text{ analytic_at}$ **by** *blast*

thus *?case* **by** *auto*

qed

qed

```

lemma remove_sings_compose:
  assumes filtermap g (at z) = at z'
  shows remove_sings (f ∘ g) z = remove_sings f z'
proof (cases ∃ c. f -z'→ c)
  case True
  then obtain c where c: f -z'→ c
  by auto
  from c have remove_sings f z' = c
  by blast
  moreover from c have remove_sings (f ∘ g) z = c
  using c by (intro remove_sings_eqI) (auto simp: filterlim_def filtermap_compose
assms)
  ultimately show ?thesis
  by simp
next
  case False
  hence ¬(∃ c. (f ∘ g) -z→ c)
  by (auto simp: filterlim_def filtermap_compose assms)
  with False show ?thesis
  by (auto simp: remove_sings_def)
qed

```

```

lemma remove_sings_cong:
  assumes eventually (λx. f x = g x) (at z) z = z'
  shows remove_sings f z = remove_sings g z'
proof (cases ∃ c. f -z→ c)
  case True
  then obtain c where c: f -z→ c by blast
  hence remove_sings f z = c
  by blast
  moreover have f -z→ c ↔ g -z'→ c
  using assms by (intro filterlim_cong refl) auto
  with c have remove_sings g z' = c
  by (intro remove_sings_eqI) auto
  ultimately show ?thesis
  by simp
next
  case False
  have f -z→ c ↔ g -z'→ c for c
  using assms by (intro filterlim_cong) auto
  with False show ?thesis
  by (auto simp: remove_sings_def)
qed

```

```

lemma deriv_remove_sings_at_analytic [simp]:
  assumes f analytic_on {z}
  shows deriv (remove_sings f) z = deriv f z

```

```

apply (rule deriv_cong_ev)
apply (rule eventually_remove_sings_eq_nhds)
using assms by auto

lemma isolated_singularity_at_remove_sings [simp, intro]:
  assumes isolated_singularity_at f z
  shows isolated_singularity_at (remove_sings f) z
  using isolated_singularity_at_cong[OF eventually_remove_sings_eq_at[OF assms]
refl] assms
  by simp

lemma not_essential_remove_sings_iff [simp]:
  assumes isolated_singularity_at f z
  shows not_essential (remove_sings f) z  $\longleftrightarrow$  not_essential f z
  using not_essential_cong[OF eventually_remove_sings_eq_at[OF assms(1)]
refl]
  by simp

lemma not_essential_remove_sings [intro]:
  assumes isolated_singularity_at f z not_essential f z
  shows not_essential (remove_sings f) z
  by (subst not_essential_remove_sings_iff) (use assms in auto)

lemma
  assumes isolated_singularity_at f z
  shows is_pole_remove_sings_iff [simp]:
    is_pole (remove_sings f) z  $\longleftrightarrow$  is_pole f z
  and zorder_remove_sings [simp]:
    zorder (remove_sings f) z = zorder f z
  and zor_poly_remove_sings [simp]:
    zor_poly (remove_sings f) z = zor_poly f z
  and has_laurent_expansion_remove_sings_iff [simp]:
    ( $\lambda w. \text{remove\_sings } f \ (z + w)$ ) has_laurent_expansion F  $\longleftrightarrow$ 
    ( $\lambda w. f \ (z + w)$ ) has_laurent_expansion F
  and tendsto_remove_sings_iff [simp]:
    remove_sings f  $-z \rightarrow c \longleftrightarrow f -z \rightarrow c$ 
  by (intro is_pole_cong eventually_remove_sings_eq_at refl zorder_cong
    zor_poly_cong has_laurent_expansion_cong' tendsto_cong assms)+

lemma remove_sings_has_laurent_expansion [laurent_expansion_intros]:
  assumes f has_laurent_expansion F
  shows remove_sings f has_laurent_expansion F
proof -
  have remove_sings f has_laurent_expansion F  $\longleftrightarrow$  f has_laurent_expansion F
  proof (rule has_laurent_expansion_cong)
    have isolated_singularity_at f 0
    using assms by (metis has_laurent_expansion_isolated_0)
    thus eventually ( $\lambda x. \text{remove\_sings } f \ x = f \ x$ ) (at 0)
    by (rule eventually_remove_sings_eq_at)
  qed

```

```

qed auto
with assms show ?thesis
  by simp
qed

lemma get_all_poles_from_remove_sings:
  fixes f:: complex  $\Rightarrow$  complex
  defines ff $\equiv$ remove_sings f
  assumes f_holo:f holomorphic_on s - pts and finite pts
    pts $\subseteq$ s open s and not_ess: $\forall x \in \text{pts}. \text{not\_essential } f x$ 
  obtains pts' where
    pts'  $\subseteq$  pts finite pts' ff holomorphic_on s - pts'  $\forall x \in \text{pts}'. \text{is\_pole } ff x$ 
proof -
  define pts' where pts' = {x $\in$ pts. is_pole f x}

  have pts'  $\subseteq$  pts unfolding pts'_def by auto
  then have finite_pts' using ⟨finite pts⟩
    using rev_finite_subset by blast
  then have open (s - pts') using ⟨open s⟩
    by (simp add: finite_imp_closed open_Diff)

  have isolated:isolated_singularity_at f z if z $\in$ pts for z
proof (rule isolated_singularity_at_holomorphic)
  show f holomorphic_on (s-(pts-{z})) - {z}
    by (metis Diff_insert f_holo insert_Diff that)
  show open (s - (pts - {z}))
    by (meson assms(3) assms(5) finite_Diff finite_imp_closed open_Diff)
  show z  $\in$  s - (pts - {z})
    using assms(4) that by auto
qed

have ff holomorphic_on s - pts'
proof (rule no_isolated_singularity')
  show (ff  $\longrightarrow$  ff z) (at z within s - pts') if z  $\in$  pts-pts' for z
proof -
  have at z within s - pts' = at z
    apply (rule at_within_open)
    using ⟨open (s - pts')⟩ that ⟨pts $\subseteq$ s⟩ by auto
  moreover have ff -z $\rightarrow$  ff z
    unfolding ff_def
  proof (subst tendsto_remove_sings_iff)
    show isolated_singularity_at f z
      apply (rule isolated)
      using that by auto
    have not_essential f z
      using not_ess that by auto
    moreover have  $\neg \text{is\_pole } f z$ 
      using that unfolding pts'_def by auto
    ultimately have  $\exists c. f -z \rightarrow c$ 

```

```

      unfolding not_essential_def by auto
    then show  $f - z \rightarrow \text{remove\_sings } f \ z$ 
      using remove_sings_eqI by blast
  qed
  ultimately show ?thesis by auto
qed
have ff holomorphic_on s - pts
  using f_holo
proof (elim holomorphic_transform)
  fix x assume  $x \in s - pts$ 
  then have f analytic_on {x}
    using assms(3) assms(5) f_holo
  by (meson finite_imp_closed
      holomorphic_on_imp_analytic_at open_Diff)
  from remove_sings_at_analytic[OF this]
  show  $f \ x = \text{ff } x$  unfolding ff_def by auto
qed
then show ff holomorphic_on s - pts' - (pts - pts')
  apply (elim holomorphic_on_subset)
  by blast
show open (s - pts')
  by (simp add: open (s - pts'))
show finite (pts - pts')
  by (simp add: assms(3))
qed
moreover have  $\forall x \in \text{pts}'. \text{is\_pole } \text{ff } x$ 
  unfolding pts'_def
  using ff_def is_pole_remove_sings_iff isolated by blast
moreover note  $\langle \text{pts}' \subseteq \text{pts} \rangle \langle \text{finite } \text{pts}' \rangle$ 
ultimately show ?thesis using that by auto
qed

lemma remove_sings_eq_0_iff:
  assumes not_essential f w
  shows  $\text{remove\_sings } f \ w = 0 \iff \text{is\_pole } f \ w \vee f - w \rightarrow 0$ 
proof (cases is_pole f w)
  case False
  then obtain c where  $c: f - w \rightarrow c$ 
    using not_essential f w unfolding not_essential_def by auto
  then show ?thesis
    using False remove_sings_eqI by auto
qed simp

lemma remove_sings_analytic_at:
  assumes isolated_singularity_at f z  $f - z \rightarrow c$ 
  shows  $\text{remove\_sings } f \text{ analytic\_on } \{z\}$ 
proof -
  from assms(1) obtain A where A: open A  $z \in A$   $f \text{ holomorphic\_on } (A - \{z\})$ 
  using analytic_imp_holomorphic isolated_singularity_at_iff_analytic_nhd by

```

```

auto
  have ana:  $f$  analytic_on  $(A - \{z\})$ 
    by (subst analytic_on_open) (use A in auto)

  have remove_sings  $f$  holomorphic_on A
  proof (rule no_isolated_singularity)
    have  $f$  holomorphic_on  $(A - \{z\})$ 
    by fact
    moreover have remove_sings  $f$  holomorphic_on  $(A - \{z\}) \longleftrightarrow f$  holomor-
    phic_on  $(A - \{z\})$ 
    by (intro holomorphic_cong remove_sings_at_analytic) (auto intro!: ana-
    lytic_on_subset[OF ana])
    ultimately show remove_sings  $f$  holomorphic_on  $(A - \{z\})$ 
    by blast
    hence continuous_on  $(A - \{z\})$  (remove_sings  $f$ )
    by (intro holomorphic_on_imp_continuous_on)
    moreover have isCont (remove_sings  $f$ )  $z$ 
    using assms isCont_def remove_sings_eqI tendsto_remove_sings_iff by
    blast
    ultimately show continuous_on A (remove_sings  $f$ )
    by (metis A(1) DiffI closed_singleton continuous_on_eq_continuous_at
    open_Diff singletonD)
  qed (use A(1) in auto)
  thus ?thesis
    using A(1,2) analytic_at by blast
qed

lemma remove_sings_analytic_on:
  assumes  $f$  analytic_on A
  shows remove_sings  $f$  analytic_on A
proof -
  from assms obtain B where B: open B  $A \subseteq B$   $f$  holomorphic_on B
  by (metis analytic_on_holomorphic)
  have remove_sings  $f$  holomorphic_on B  $\longleftrightarrow f$  holomorphic_on B
  proof (rule holomorphic_cong)
    fix  $z$  assume  $z \in B$ 
    have  $f$  analytic_on  $\{z\}$ 
    using  $\langle z \in B \rangle$  B holomorphic_on_imp_analytic_at by blast
    thus remove_sings  $f$   $z = f$   $z$ 
    by (rule remove_sings_at_analytic)
  qed auto
  thus ?thesis
    using B analytic_on_holomorphic by blast
qed

lemma residue_remove_sings [simp]:
  assumes isolated_singularity_at  $f$   $z$ 
  shows residue (remove_sings  $f$ )  $z =$  residue  $f$   $z$ 
proof -

```

from *assms* **have** *eventually* $(\lambda w. \text{remove_sings } f \ w = f \ w)$ *(at z)*
using *eventually_remove_sings_eq_at* **by** *blast*
then obtain *A* **where** *A*: *open* *A* $z \in A \wedge w. w \in A - \{z\} \implies \text{remove_sings } f$
 $w = f \ w$
by *(subst (asm) eventually_at_topological)* *blast*
from *A(1,2)* **obtain** ε **where** $\varepsilon: \varepsilon > 0 \text{ cball } z \ \varepsilon \subseteq A$
using *open_contains_cball_eq* **by** *blast*
hence *eq*: $\text{remove_sings } f \ w = f \ w$ **if** $w \in \text{cball } z \ \varepsilon - \{z\}$ **for** *w*
using *that A ε by blast*

define *P* **where** $P = (\lambda f \ c \ \varepsilon. (f \ \text{has_contour_integral_of_real } (2 * \pi) * i * c))$
(circlepath z ε)
have $P \ (\text{remove_sings } f) \ c \ \delta \longleftrightarrow P \ f \ c \ \delta$ **if** $0 < \delta < \varepsilon$ **for** $c \ \delta$
unfolding *P_def* **using** $\langle \varepsilon > 0 \rangle$ **that by** *(intro has_contour_integral_cong)*
(auto simp: eq)
hence $*$: $(\forall \varepsilon > 0. \varepsilon < e \longrightarrow P \ (\text{remove_sings } f) \ c \ \varepsilon) \longleftrightarrow (\forall \varepsilon > 0. \varepsilon < e \longrightarrow P$
 $f \ c \ \varepsilon)$ **if** $e \leq \varepsilon$ **for** $c \ e$
using *that by force*
have $**$: $(\exists e > 0. \forall \varepsilon > 0. \varepsilon < e \longrightarrow P \ (\text{remove_sings } f) \ c \ \varepsilon) \longleftrightarrow (\exists e > 0. \forall \varepsilon > 0.$
 $\varepsilon < e \longrightarrow P \ f \ c \ \varepsilon)$ **for** *c*
proof
assume $(\exists e > 0. \forall \varepsilon > 0. \varepsilon < e \longrightarrow P \ (\text{remove_sings } f) \ c \ \varepsilon)$
then obtain *e* **where** $e > 0 \ \forall \varepsilon > 0. \varepsilon < e \longrightarrow P \ (\text{remove_sings } f) \ c \ \varepsilon$
by *blast*
thus $(\exists e > 0. \forall \varepsilon > 0. \varepsilon < e \longrightarrow P \ f \ c \ \varepsilon)$
by *(intro exI[of _ min e ε]) (use *[of min e ε c] ε(1) in auto)*
next
assume $(\exists e > 0. \forall \varepsilon > 0. \varepsilon < e \longrightarrow P \ f \ c \ \varepsilon)$
then obtain *e* **where** $e > 0 \ \forall \varepsilon > 0. \varepsilon < e \longrightarrow P \ f \ c \ \varepsilon$
by *blast*
thus $(\exists e > 0. \forall \varepsilon > 0. \varepsilon < e \longrightarrow P \ (\text{remove_sings } f) \ c \ \varepsilon)$
by *(intro exI[of _ min e ε]) (use *[of min e ε c] ε(1) in auto)*
qed
show *?thesis*
unfolding *residue_def* **by** *(intro arg_cong[of _ _ Eps] ext **[unfolded P_def])*
qed

lemma *remove_sings_shift_0*:
 $\text{remove_sings } f \ z = \text{remove_sings } (\lambda w. f \ (z + w)) \ 0$
proof *(cases $\exists c. f \ -z \rightarrow c$)*
case *True*
then obtain *c* **where** $c: f \ -z \rightarrow c$
by *blast*
from *c* **have** $\text{remove_sings } f \ z = c$
by *(rule remove_sings_eqI)*
moreover have $\text{remove_sings } (\lambda w. f \ (z + w)) \ 0 = c$
by *(rule remove_sings_eqI) (use c in <simp_all add: at_to_0' filterlim_filtermap add_ac>)*
ultimately show *?thesis*


```

    by simp
next
case False
hence  $\neg(\exists c. (\lambda w. f (z + w)) - 0 \rightarrow c)$ 
  by (simp add: at_to_0' filterlim_filtermap add_ac)
with False show ?thesis
  by (simp add: remove_sings_def)
qed

```

```

lemma remove_sings_shift_0':
  NO_MATCH 0 z  $\implies$  remove_sings f z = remove_sings ( $\lambda w. f (z + w)$ ) 0
  by (rule remove_sings_shift_0)

```

8.8 Meromorphicity

We define meromorphicity in terms of Laurent series expansions. This has the advantage of giving us a particularly simple definition that makes many of the lemmas below trivial because they reduce to similar statements about Laurent series that are already in the library.

On open domains, this definition coincides with the usual one from the literature, namely that the function be holomorphic on its domain except for a set of non-essential singularities that is *sparse*, i.e. that has no limit point inside the domain.

However, unlike the definitions found in the literature, our definition also makes sense for non-open domains: similarly to (*analytic_on*), we consider a function meromorphic on a non-open domain if it is meromorphic on some open superset of that domain.

We will prove all of this below.

```

definition meromorphic_on :: (complex  $\Rightarrow$  complex)  $\Rightarrow$  complex set  $\Rightarrow$  bool
  (infixl  $\langle$ (meromorphic'_on) $\rangle$  50) where
    f meromorphic_on A  $\longleftrightarrow$  ( $\forall z \in A. \exists F. (\lambda w. f (z + w))$  has_laurent_expansion F)

```

```

lemma meromorphic_at_iff: f meromorphic_on {z}  $\longleftrightarrow$  isolated_singularity_at f z  $\wedge$  not_essential f z

```

```

  unfolding meromorphic_on_def
  by (metis has_laurent_expansion_isolated has_laurent_expansion_not_essential
    insertI1 singletonD not_essential_has_laurent_expansion)

```

```

named_theorems meromorphic_intros

```

```

lemma meromorphic_on_empty [simp, intro]: f meromorphic_on {}
  by (auto simp: meromorphic_on_def)

```

```

lemma meromorphic_on_def':
  f meromorphic_on A  $\longleftrightarrow$  ( $\forall z \in A. (\lambda w. f (z + w))$  has_laurent_expansion laurent_expansion f z)

```

unfolding meromorphic_on_def **using** laurent_expansion_eqI **by** blast

lemma meromorphic_on_meromorphic_at: $f \text{ meromorphic_on } A \longleftrightarrow (\forall x \in A. f \text{ meromorphic_on } \{x\})$
by (auto simp: meromorphic_on_def)

lemma meromorphic_on_altdef:
 $f \text{ meromorphic_on } A \longleftrightarrow (\forall z \in A. \text{isolated_singularity_at } f z \wedge \text{not_essential } f z)$
by (subst meromorphic_on_meromorphic_at) (auto simp: meromorphic_at_iff)

lemma meromorphic_on_cong:
assumes $\bigwedge z. z \in A \implies \text{eventually } (\lambda w. f w = g w) \text{ (at } z) A = B$
shows $f \text{ meromorphic_on } A \longleftrightarrow g \text{ meromorphic_on } B$
unfolding meromorphic_on_def **using** assms
by (intro ball_cong refl arg_cong[of _ _ Ex] has_laurent_expansion_cong ext) (simp_all add: at_to_0' eventually_filtermap_add_ac)

lemma meromorphic_on_subset: $f \text{ meromorphic_on } A \implies B \subseteq A \implies f \text{ meromorphic_on } B$
by (auto simp: meromorphic_on_def)

lemma meromorphic_on_Un:
assumes $f \text{ meromorphic_on } A \wedge f \text{ meromorphic_on } B$
shows $f \text{ meromorphic_on } (A \cup B)$
using assms **unfolding** meromorphic_on_def **by** blast

lemma meromorphic_on_Union:
assumes $\bigwedge A. A \in B \implies f \text{ meromorphic_on } A$
shows $f \text{ meromorphic_on } (\bigcup B)$
using assms **unfolding** meromorphic_on_def **by** blast

lemma meromorphic_on_UN:
assumes $\bigwedge x. x \in X \implies f \text{ meromorphic_on } (A x)$
shows $f \text{ meromorphic_on } (\bigcup x \in X. A x)$
using assms **unfolding** meromorphic_on_def **by** blast

lemma meromorphic_on_imp_has_laurent_expansion:
assumes $f \text{ meromorphic_on } A \wedge z \in A$
shows $(\lambda w. f (z + w)) \text{ has_laurent_expansion } \text{laurent_expansion } f z$
using assms laurent_expansion_eqI **unfolding** meromorphic_on_def **by** blast

lemma meromorphic_on_open_nhd:
assumes $f \text{ meromorphic_on } A$
obtains B **where** $\text{open } B \wedge A \subseteq B \wedge f \text{ meromorphic_on } B$
proof –
obtain F **where** $F: \bigwedge z. z \in A \implies (\lambda w. f (z + w)) \text{ has_laurent_expansion } F z$
using assms **unfolding** meromorphic_on_def **by** metis
have $\exists Z. \text{open } Z \wedge z \in Z \wedge (\forall w \in Z - \{z\}. \text{eval_fls } (F z) (w - z) = f w)$ **if** $z: z$

```

∈ A for z
proof -
  obtain Z where Z: open Z 0 ∈ Z ∧ w. w ∈ Z - {0} ⇒ eval_fls (F z) w =
f (z + w)
  using F[OF z] unfolding has_laurent_expansion_def eventually_at_topological
by blast
  hence open ((+) z ` Z) and z ∈ (+) z ` Z
  using open_translation by auto
  moreover have eval_fls (F z) (w - z) = f w if w ∈ (+) z ` Z - {z} for w
  using Z(3)[of w - z] that by auto
  ultimately show ?thesis by blast
qed
then obtain Z where Z:
  ∧ z. z ∈ A ⇒ open (Z z) ∧ z ∈ Z z ∧ (∀ w ∈ Z z - {z}. eval_fls (F z) (w - z)
= f w)
  by metis

define B where B = (⋃ z ∈ A. Z z ∩ eball z (fls_conv_radius (F z)))
show ?thesis
proof (rule that[of B])
  show open B
  using Z unfolding B_def by auto
  show A ⊆ B
  unfolding B_def using F Z by (auto simp: has_laurent_expansion_def
zero_ereal_def)
  show f meromorphic_on B
  unfolding meromorphic_on_def
proof
  fix z assume z: z ∈ B
  show ∃ F. (λ w. f (z + w)) has_laurent_expansion F
  proof (cases z ∈ A)
    case True
    thus ?thesis using F by blast
  next
    case False
    then obtain z0 where z0: z0 ∈ A z ∈ Z z0 - {z0} dist z0 z <
fls_conv_radius (F z0)
    using z False Z unfolding B_def by auto
    hence (λ w. eval_fls (F z0) (w - z0)) analytic_on {z}
    by (intro analytic_on_eval_fls' analytic_intros) (auto simp: dist_norm)
    also have ?this ⟷ f analytic_on {z}
    proof (intro analytic_at_cong refl)
      have eventually (λ w. w ∈ Z z0 - {z0}) (nhds z)
      using Z[of z0] z0 by (intro eventually_nhds_in_open) auto
      thus ∀ x in nhds z. eval_fls (F z0) (x - z0) = f x
      by eventually_elim (use Z[of z0] z0 in auto)
    qed
  finally show ?thesis
  using analytic_at_imp_has_fps_expansion has_fps_expansion_to_laurent

```

```

by blast
qed
qed
qed
qed

```

```

lemma meromorphic_on_not_essential:
  assumes f meromorphic_on {z}
  shows not_essential f z
  using assms has_laurent_expansion_not_essential unfolding meromorphic_on_def
  by blast

```

```

lemma meromorphic_on_isolated_singularity:
  assumes f meromorphic_on {z}
  shows isolated_singularity_at f z
  using assms has_laurent_expansion_isolated unfolding meromorphic_on_def
  by blast

```

```

lemma meromorphic_on_imp_not_islimpt_singularities:
  assumes f meromorphic_on A z ∈ A
  shows ¬z islimpt {w. ¬f analytic_on {w}}
proof -
  obtain B where B: open B A ⊆ B f meromorphic_on B
  using assms meromorphic_on_open_nhd by blast
  obtain F where F: (λw. f (z + w)) has_laurent_expansion F
  using B assms(2) unfolding meromorphic_on_def by blast
  from F have isolated_singularity_at f z
  using has_laurent_expansion_isolated assms(2) by blast
  then obtain r where r: r > 0 f analytic_on ball z r - {z}
  unfolding isolated_singularity_at_def by blast
  have f analytic_on {w} if w ∈ ball z r - {z} for w
  by (rule analytic_on_subset[OF r(2)]) (use that in auto)
  hence eventually (λw. f analytic_on {w}) (at z)
  using eventually_at_in_open[of ball z r z] ⟨r > 0⟩ by (auto elim!: eventu-
ally_mono)
  thus ¬z islimpt {w. ¬f analytic_on {w}}
  by (auto simp: islimpt_conv_frequently_at frequently_def)
qed

```

```

lemma meromorphic_on_imp_sparse_singularities:
  assumes f meromorphic_on A
  shows {w. ¬f analytic_on {w}} sparse_in A
  by (metis assms meromorphic_on_imp_not_islimpt_singularities
meromorphic_on_open_nhd sparse_in_open sparse_in_subset)

```

```

lemma meromorphic_on_imp_sparse_singularities':
  assumes f meromorphic_on A
  shows {w ∈ A. ¬f analytic_on {w}} sparse_in A
  using meromorphic_on_imp_sparse_singularities[OF assms]

```

```

by (rule sparse_in_subset2) auto

lemma meromorphic_onE:
  assumes f meromorphic_on A
  obtains pts where pts  $\subseteq$  A pts sparse_in A f analytic_on A - pts
     $\bigwedge z. z \in A \implies \text{not\_essential } f z \bigwedge z. z \in A \implies \text{isolated\_singularity\_at } f z$ 
proof (rule that)
  show  $\{z \in A. \neg f \text{ analytic\_on } \{z\}\}$  sparse_in A
    using assms by (rule meromorphic_on_imp_sparse_singularities')
  show f analytic_on A -  $\{z \in A. \neg f \text{ analytic\_on } \{z\}\}$ 
    by (subst analytic_on_analytic_at) auto
qed (use assms in <auto intro: meromorphic_on_isolated_singularity meromorphic_on_not_essential meromorphic_on_subset>)

lemma meromorphic_onI_weak:
  assumes f analytic_on A - pts  $\bigwedge z. z \in \text{pts} \implies \text{not\_essential } f z$  pts sparse_in A
  shows f meromorphic_on A
  unfolding meromorphic_on_def
proof
  fix z assume z:  $z \in A$ 
  show  $(\exists F. (\lambda w. f (z + w)) \text{ has\_laurent\_expansion } F)$ 
  proof (cases  $z \in \text{pts}$ )
    case False
    have f analytic_on  $\{z\}$ 
      using assms(1) by (rule analytic_on_subset) (use False z in auto)
    thus ?thesis
      using isolated_singularity_at_analytic not_essential_analytic
        not_essential_has_laurent_expansion by blast
  next
    case True
    show ?thesis
  proof (rule exI, rule not_essential_has_laurent_expansion)
    show not_essential f z
      using assms(2) True by blast
  next
    show isolated_singularity_at f z
  proof (rule isolated_singularity_at_holomorphic)
    show open (interior A - (pts -  $\{z\}$ ))
    proof (rule open_diff_sparse_pts)
      show pts -  $\{z\}$  sparse_in interior A
        using sparse_in_subset sparse_in_subset2 assms interior_subset
  Diff_subset by metis
    qed auto
  next
    show f analytic_on interior A - (pts -  $\{z\}$ ) -  $\{z\}$ 
      using assms(1) by (rule analytic_on_subset) (use interior_subset in
blast)

```

```

      thus  $f$  holomorphic_on interior  $A - (pts - \{z\}) - \{z\}$ 
      by (rule analytic_imp_holomorphic)
    next
      from assms(4) and True have  $z \in \text{interior } A$ 
      unfolding frontier_def using closure_subset  $z$  by blast
      thus  $z \in \text{interior } A - (pts - \{z\})$ 
      by blast
    qed
  qed
qed
qed

```

lemma meromorphic_onI_open:

```

  assumes open  $A$   $f$  analytic_on  $A - pts \wedge z. z \in pts \implies \text{not\_essential } f z$ 
  assumes  $\bigwedge z. z \in A \implies \neg z \text{ islimpt } pts \cap A$ 
  shows  $f$  meromorphic_on  $A$ 
proof (rule meromorphic_onI_weak)
  have *:  $A - pts \cap A = A - pts$ 
  by blast
  show  $f$  analytic_on  $A - pts \cap A$ 
  unfolding * by fact
  show  $pts \cap A$  sparse_in  $A$ 
  using assms(1,4) by (subst sparse_in_open) auto
  show not_essential  $f z$  if  $z \in pts \cap A$  for  $z$ 
  using assms(3) that by blast
  show  $pts \cap A \cap \text{frontier } A = \{\}$ 
  using <open  $A$ > frontier_disjoint_eq by blast
qed

```

lemma meromorphic_at_isCont_imp_analytic:

```

  assumes  $f$  meromorphic_on  $\{z\}$  isCont  $f z$ 
  shows  $f$  analytic_on  $\{z\}$ 
proof -
  have *:  $(\lambda w. f (z + w))$  has_laurent_expansion laurent_expansion  $f z$ 
  using assms by (auto intro: meromorphic_on_imp_has_laurent_expansion)
  from assms have  $\neg \text{is\_pole } f z$ 
  using is_pole_def not_tendsto_and_filterlim_at_infinity trivial_limit_at by
  (metis isContD)
  with * have fls_subdegree (laurent_expansion  $f z$ )  $\geq 0$ 
  using has_laurent_expansion_imp_is_pole linorder_not_le by blast
  hence **:  $(\lambda w. \text{eval\_fls } (\text{laurent\_expansion } f z) (w - z))$  analytic_on  $\{z\}$ 
  by (intro analytic_intros)+ (use * in <auto simp: has_laurent_expansion_def
  zero_ereal_def>)
  have  $(\lambda w. \text{eval\_fls } (\text{laurent\_expansion } f z) (w - z)) - z \rightarrow \text{eval\_fls } (\text{laurent\_expansion }
  f z) (z - z)$ 
  by (intro isContD analytic_at_imp_isCont **)
  also have  $?this \longleftrightarrow f - z \rightarrow \text{eval\_fls } (\text{laurent\_expansion } f z) (z - z)$ 
  by (intro filterlim_cong refl)
  (use * in <auto simp: has_laurent_expansion_def at_to_0' eventually_filtermap

```

```

add_ac)
  finally have  $f - z \rightarrow \text{eval\_fls } (\text{laurent\_expansion } f \ z) \ 0$ 
    by simp
  moreover from assms have  $f - z \rightarrow f \ z$ 
    by (auto intro: isContD)
  ultimately have ***:  $\text{eval\_fls } (\text{laurent\_expansion } f \ z) \ 0 = f \ z$ 
    by (rule LIM_unique)

  have eventually  $(\lambda w. f \ w = \text{eval\_fls } (\text{laurent\_expansion } f \ z) \ (w - z)) \ (\text{at } z)$ 
    using * by (simp add: has_laurent_expansion_def at_to_0' eventually_filtermap
add_ac eq_commute)
  hence eventually  $(\lambda w. f \ w = \text{eval\_fls } (\text{laurent\_expansion } f \ z) \ (w - z)) \ (\text{nhds } z)$ 
    unfolding eventually_at_filter by eventually_elim (use *** in auto)
  hence  $f \text{ analytic\_on } \{z\} \longleftrightarrow (\lambda w. \text{eval\_fls } (\text{laurent\_expansion } f \ z) \ (w - z))$ 
analytic_on  $\{z\}$ 
    by (intro analytic_at_cong refl)
  with ** show ?thesis
    by simp
qed

lemma analytic_on_imp_meromorphic_on:
  assumes  $f \text{ analytic\_on } A$ 
  shows  $f \text{ meromorphic\_on } A$ 
  by (rule meromorphic_onI_weak[of _ _  $\{z\}$ ]) (use assms in auto)

lemma meromorphic_on_compose:
  assumes  $g \text{ meromorphic\_on } A \ f \text{ analytic\_on } B \ f' \ B \subseteq A$ 
  shows  $(\lambda w. g \ (f \ w)) \text{ meromorphic\_on } B$ 
  unfolding meromorphic_on_def
proof safe
  fix  $z$  assume  $z: z \in B$ 
  have  $f \text{ analytic\_on } \{z\}$ 
    using assms(2) by (rule analytic_on_subset) (use assms(3)  $z$  in auto)
  hence  $(\lambda w. f \ w - f \ z) \text{ analytic\_on } \{z\}$ 
    by (intro analytic_intros)
  then obtain  $F$  where  $F: (\lambda w. f \ (z + w) - f \ z) \text{ has\_fps\_expansion } F$ 
    using analytic_at_imp_has_fps_expansion by blast

  from assms(3) and  $z$  have  $f \ z \in A$ 
    by auto
  with assms(1) obtain  $G$  where  $G: (\lambda w. g \ (f \ z + w)) \text{ has\_laurent\_expansion } G$ 
  G
    using  $z$  by (auto simp: meromorphic_on_def)

  have  $\exists H. ((\lambda w. g \ (f \ z + w)) \circ (\lambda w. f \ (z + w) - f \ z)) \text{ has\_laurent\_expansion } H$ 
  proof (cases  $F = 0$ )
  case False
  show ?thesis
  proof (rule exI, rule has_laurent_expansion_compose)

```

```

  show  $(\lambda w. f(z + w) - f z)$  has_laurent_expansion fps_to_fls F
    using F by (rule has_laurent_expansion_fps)
  show fps_nth F 0 = 0
    using has_fps_expansion_imp_0_eq_fps_nth_0[OF F] by simp
qed fact+
next
case True
have  $(\lambda w. g(f z))$  has_laurent_expansion fls_const (g (f z))
  by auto
also have ?this  $\longleftrightarrow (\lambda w. ((\lambda w. g(f z + w)) \circ (\lambda w. f(z + w) - f z)) w)$ 
  has_laurent_expansion fls_const (g (f z))
proof (rule has_laurent_expansion_cong, goal_cases)
case 1
from F and True have eventually  $(\lambda w. f(z + w) - f z = 0)$  (nhds 0)
  by (simp add: has_fps_expansion_0_iff)
hence eventually  $(\lambda w. f(z + w) - f z = 0)$  (at 0)
  by (simp add: eventually_nhds_conv_at)
thus ?case
  by eventually_elim auto
qed auto
finally show ?thesis
  by blast
qed
thus  $\exists H. (\lambda w. g(f(z + w)))$  has_laurent_expansion H
  by (simp add: o_def)
qed

lemma constant_on_imp_meromorphic_on:
  assumes f_constant_on A open A
  shows f_meromorphic_on A
  using assms analytic_on_imp_meromorphic_on
    constant_on_imp_analytic_on
  by blast

```

8.9 Nice meromorphicity

This is probably very non-standard, but we call a function “nicely meromorphic” if it is meromorphic and has no removable singularities. That means that the only singularities are poles. We furthermore require that the function return 0 at any pole, for convenience.

```

definition nicely_meromorphic_on :: (complex  $\Rightarrow$  complex)  $\Rightarrow$  complex set  $\Rightarrow$  bool
  (infixl 50 <('nicely'_meromorphic'_on)> 50)
  where f nicely_meromorphic_on A  $\longleftrightarrow$  f_meromorphic_on A
     $\wedge (\forall z \in A. (is\_pole f z \wedge f z = 0) \vee f -z \rightarrow f z)$ 

```

```

lemma nicely_meromorphic_on_subset:
  f nicely_meromorphic_on A  $\implies B \subseteq A \implies$  f nicely_meromorphic_on B
  using meromorphic_on_subset unfolding nicely_meromorphic_on_def by blast

```


lemma *constant_on_imp_nicely_meromorphic_on*:

assumes *f constant_on A open A*
shows *f nicely_meromorphic_on A*
by (*meson analytic_at_imp_isCont assms*
constant_on_imp_holomorphic_on
constant_on_imp_meromorphic_on
holomorphic_on_imp_analytic_at isCont_def
nicely_meromorphic_on_def)

lemma *nicely_meromorphic_on_imp_analytic_at*:

assumes *f nicely_meromorphic_on A z ∈ A ¬is_pole f z*
shows *f analytic_on {z}*
proof (*rule meromorphic_at_isCont_imp_analytic*)
show *f meromorphic_on {z}*
by (*rule meromorphic_on_subset[of _ A]*) (*use assms in <auto simp: nicely_meromorphic_on_def>*)
next
from *assms* **have** *f -z → f z*
by (*auto simp: nicely_meromorphic_on_def*)
thus *isCont f z*
by (*auto simp: isCont_def*)
qed

lemma *analytic_on_imp_nicely_meromorphic_on*:

f analytic_on A ⇒ f nicely_meromorphic_on A
by (*meson analytic_at_imp_isCont analytic_on_analytic_at*
analytic_on_imp_meromorphic_on isContD nicely_meromorphic_on_def)

lemma *remove_sings_meromorphic [meromorphic_intros]*:

assumes *f meromorphic_on A*
shows *remove_sings f meromorphic_on A*
unfolding *meromorphic_on_def*
proof *safe*
fix *z* **assume** *z: z ∈ A*
show $\exists F. (\lambda w. \text{remove_sings } f (z + w)) \text{ has_laurent_expansion } F$
using *assms meromorphic_on_isolated_singularity meromorphic_on_not_essential*
not_essential_has_laurent_expansion z meromorphic_on_subset **by** *blast*
qed

lemma *remove_sings_nicely_meromorphic*:

assumes *f meromorphic_on A*
shows *remove_sings f nicely_meromorphic_on A*
proof -
have *remove_sings f meromorphic_on A*
by (*simp add: assms remove_sings_meromorphic*)
moreover **have** *is_pole (remove_sings f) z*
 $\wedge \text{remove_sings } f z = 0 \vee$
 $\text{remove_sings } f -z \rightarrow \text{remove_sings } f z$
if *z ∈ A* **for** *z*

```

proof (cases  $\exists c. f - z \rightarrow c$ )
  case True
  then have remove_sings  $f - z \rightarrow$  remove_sings  $f z$ 
    by (metis remove_sings_eqI tendsto_remove_sings_iff
      assms meromorphic_onE that)
  then show ?thesis by simp
next
  case False
  then have is_pole (remove_sings  $f$ )  $z$ 
     $\wedge$  remove_sings  $f z = 0$ 
  by (meson is_pole_remove_sings_iff remove_sings_def
    remove_sings_eq_0_iff assms meromorphic_onE that)
  then show ?thesis by simp
qed
ultimately show ?thesis
  unfolding nicely_meromorphic_on_def by simp
qed

```

A nicely meromorphic function that frequently takes the same value in the neighbourhood of some point is constant.

lemma frequently_eq_meromorphic_imp_constant:

```

assumes frequently  $(\lambda z. f z = c)$  (at  $z$ )
assumes  $f$  nicely_meromorphic_on  $A$  open  $A$  connected  $A$   $z \in A$ 
shows  $\bigwedge w. w \in A \implies f w = c$ 
proof -
  from assms(2) have mero:  $f$  meromorphic_on  $A$ 
    by (auto simp: nicely_meromorphic_on_def)
  have sparse:  $\{z. \text{is\_pole } f z\}$  sparse_in  $A$ 
    using assms(2) mero
    by (meson assms(3) meromorphic_on_isolated_singularity meromorphic_on_meromorphic_at
      not_islimpt_poles_sparse_in_open)

  have eq:  $f w = c$  if  $w: w \in A \neg \text{is\_pole } f w$  for  $w$ 
  proof -
    have  $f w - c = 0$ 
  proof (rule analytic_continuation[of  $\lambda w. f w - c$ ])
    show  $(\lambda w. f w - c)$  holomorphic_on  $\{z \in A. \neg \text{is\_pole } f z\}$  using assms(2)
      by (intro holomorphic_intros)
    (metis (mono_tags, lifting) analytic_imp_holomorphic analytic_on_analytic_at

      mem_Collect_eq nicely_meromorphic_on_imp_analytic_at)
  next
    from sparse and assms(3) have open  $(A - \{z. \text{is\_pole } f z\})$ 
      by (simp add: open_diff_sparse_pts)
    also have  $A - \{z. \text{is\_pole } f z\} = \{z \in A. \neg \text{is\_pole } f z\}$ 
      by blast
    finally show open ... .
  next
    from sparse have connected  $(A - \{z. \text{is\_pole } f z\})$ 

```

```

    using assms(3,4) by (intro sparse_imp_connected) auto
  also have  $A - \{z. \text{is\_pole } f \ z\} = \{z \in A. \neg \text{is\_pole } f \ z\}$ 
    by blast
  finally show connected ... .
next
  have eventually  $(\lambda w. w \in A)$  (at  $z$ )
    using assms by (intro eventually_at_in_open') auto
  moreover have eventually  $(\lambda w. \neg \text{is\_pole } f \ w)$  (at  $z$ ) using mero
    by (metis assms(5) eventually_not_pole meromorphic_onE)
  ultimately have  $ev: \text{eventually } (\lambda w. w \in A \wedge \neg \text{is\_pole } f \ w)$  (at  $z$ )
    by eventually_elim auto
  show  $z \text{ islimpt } \{z \in A. \neg \text{is\_pole } f \ z \wedge f \ z = c\}$ 
    using frequently_eventually_frequently[OF assms(1) ev]
    unfolding islimpt_conv_frequently_at by (rule frequently_elim1) auto
next
  from assms(1) have  $\neg \text{is\_pole } f \ z$ 
    by (simp add: frequently_const_imp_not_is_pole)
  with  $\langle z \in A \rangle$  show  $z \in \{z \in A. \neg \text{is\_pole } f \ z\}$ 
    by auto
qed (use  $w$  in auto)
thus  $f \ w = c$ 
  by simp
qed

have not_pole:  $\neg \text{is\_pole } f \ w$  if  $w: w \in A$  for  $w$ 
proof -
  have eventually  $(\lambda w. \neg \text{is\_pole } f \ w)$  (at  $w$ )
    using mero by (metis eventually_not_pole meromorphic_onE that)
  moreover have eventually  $(\lambda w. w \in A)$  (at  $w$ )
    using  $w \in \text{open } A$  by (intro eventually_at_in_open')
  ultimately have eventually  $(\lambda w. f \ w = c)$  (at  $w$ )
    by eventually_elim (auto simp: eq)
  hence  $\text{is\_pole } f \ w \longleftrightarrow \text{is\_pole } (\lambda \_. c) \ w$ 
    by (intro is_pole_cong refl)
  thus ?thesis
    by simp
qed

show  $f \ w = c$  if  $w: w \in A$  for  $w$ 
  using eq[OF  $w$  not_pole[OF  $w$ ]] .
qed

lemma nicely_meromorphic_on_unop:
  assumes  $f \text{ nicely\_meromorphic\_on } A$ 
  assumes  $f \text{ meromorphic\_on } A \implies (\lambda z. h \ (f \ z)) \text{ meromorphic\_on } A$ 
  assumes  $\bigwedge z. z \in A \implies \text{is\_pole } f \ z \implies \text{is\_pole } (\lambda z. h \ (f \ z)) \ z$ 
  assumes  $\bigwedge z. z \in f^{-1} A \implies \text{isCont } h \ z$ 
  assumes  $h \ 0 = 0$ 
  shows  $(\lambda z. h \ (f \ z)) \text{ nicely\_meromorphic\_on } A$ 

```

```

unfolding nicely_meromorphic_on_def
proof (intro conjI ballI)
  show ( $\lambda z. h (f z)$ ) meromorphic_on A
    using assms(1,2) by (auto simp: nicely_meromorphic_on_def)
next
  fix z assume z:  $z \in A$ 
  hence is_pole f z  $\wedge f z = 0 \vee f -z \rightarrow f z$ 
    using assms by (auto simp: nicely_meromorphic_on_def)
  thus is_pole ( $\lambda z. h (f z)$ ) z  $\wedge h (f z) = 0 \vee (\lambda z. h (f z)) -z \rightarrow h (f z)$ 
proof (rule disj_forward)
  assume is_pole f z  $\wedge f z = 0$ 
  thus is_pole ( $\lambda z. h (f z)$ ) z  $\wedge h (f z) = 0$ 
    using assms z by auto
next
  assume *:  $f -z \rightarrow f z$ 
  from z assms have isCont h (f z)
    by auto
  with * show ( $\lambda z. h (f z)$ )  $-z \rightarrow h (f z)$ 
    using continuous_within continuous_within_compose3 by blast
qed
qed

```

8.10 Closure properties and proofs for individual functions

```

lemma meromorphic_on_const [intro, meromorphic_intros]: ( $\lambda \_. c$ ) meromorphic_on A
  by (rule analytic_on_imp_meromorphic_on) auto

```

```

lemma meromorphic_on_id [intro, meromorphic_intros]: ( $\lambda w. w$ ) meromorphic_on A
  by (auto simp: meromorphic_on_def intro!: exI laurent_expansion_intros)

```

```

lemma meromorphic_on_id' [intro, meromorphic_intros]: id meromorphic_on A
  by (auto simp: meromorphic_on_def intro!: exI laurent_expansion_intros)

```

```

lemma meromorphic_on_add [meromorphic_intros]:
  assumes f meromorphic_on A g meromorphic_on A
  shows ( $\lambda w. f w + g w$ ) meromorphic_on A
  unfolding meromorphic_on_def
  by (rule laurent_expansion_intros exI ballI
    assms[THEN meromorphic_on_imp_has_laurent_expansion] | assumption)+

```

```

lemma meromorphic_on_uminus [meromorphic_intros]:
  assumes f meromorphic_on A
  shows ( $\lambda w. -f w$ ) meromorphic_on A
  unfolding meromorphic_on_def
  by (rule laurent_expansion_intros exI ballI
    assms[THEN meromorphic_on_imp_has_laurent_expansion] | assumption)+

```

tion)+

lemma *meromorphic_on_diff* [*meromorphic_intros*]:
 assumes *f meromorphic_on A g meromorphic_on A*
 shows $(\lambda w. f w - g w)$ *meromorphic_on A*
 using *meromorphic_on_add*[*OF assms(1) meromorphic_on_uminus*[*OF assms(2)*]]
 by *simp*

lemma *meromorphic_on_mult* [*meromorphic_intros*]:
 assumes *f meromorphic_on A g meromorphic_on A*
 shows $(\lambda w. f w * g w)$ *meromorphic_on A*
 unfolding *meromorphic_on_def*
 by (rule *laurent_expansion_intros exI ballI*
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | *assump-*
tion)+

lemma *meromorphic_on_power* [*meromorphic_intros*]:
 assumes *f meromorphic_on A*
 shows $(\lambda w. f w ^ n)$ *meromorphic_on A*
 unfolding *meromorphic_on_def*
 by (rule *laurent_expansion_intros exI ballI*
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | *assump-*
tion)+

lemma *meromorphic_on_powi* [*meromorphic_intros*]:
 assumes *f meromorphic_on A*
 shows $(\lambda w. f w \text{ powi } n)$ *meromorphic_on A*
 unfolding *meromorphic_on_def*
 by (rule *laurent_expansion_intros exI ballI*
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | *assump-*
tion)+

lemma *meromorphic_on_scaleR* [*meromorphic_intros*]:
 assumes *f meromorphic_on A*
 shows $(\lambda w. \text{scaleR } x (f w))$ *meromorphic_on A*
 unfolding *meromorphic_on_def*
 by (rule *laurent_expansion_intros exI ballI*
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | *assump-*
tion)+

lemma *meromorphic_on_inverse* [*meromorphic_intros*]:
 assumes *f meromorphic_on A*
 shows $(\lambda w. \text{inverse } (f w))$ *meromorphic_on A*
 unfolding *meromorphic_on_def*
 by (rule *laurent_expansion_intros exI ballI*
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | *assump-*
tion)+

lemma *meromorphic_on_divide* [*meromorphic_intros*]:

assumes f meromorphic_on A g meromorphic_on A
shows $(\lambda w. f w / g w)$ meromorphic_on A
using meromorphic_on_mult[OF assms(1) meromorphic_on_inverse[OF assms(2)]]
by (simp add: field_simps)

lemma meromorphic_on_sum [meromorphic_intros]:
assumes $\bigwedge i. i \in I \implies f i$ meromorphic_on A
shows $(\lambda w. \sum_{i \in I}. f i w)$ meromorphic_on A
unfolding meromorphic_on_def
by (rule laurent_expansion_intros exI ballI
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | assumption)
 +

lemma meromorphic_on_sum_list [meromorphic_intros]:
assumes $\bigwedge i. i \in \text{set } fs \implies f i$ meromorphic_on A
shows $(\lambda w. \sum i \leftarrow fs. f i w)$ meromorphic_on A
unfolding meromorphic_on_def
by (rule laurent_expansion_intros exI ballI
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | assumption)
 +

lemma meromorphic_on_sum_mset [meromorphic_intros]:
assumes $\bigwedge i. i \in \# I \implies f i$ meromorphic_on A
shows $(\lambda w. \sum i \in \# I. f i w)$ meromorphic_on A
unfolding meromorphic_on_def
by (rule laurent_expansion_intros exI ballI
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | assumption)
 +

lemma meromorphic_on_prod [meromorphic_intros]:
assumes $\bigwedge i. i \in I \implies f i$ meromorphic_on A
shows $(\lambda w. \prod_{i \in I}. f i w)$ meromorphic_on A
unfolding meromorphic_on_def
by (rule laurent_expansion_intros exI ballI
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | assumption)
 +

lemma meromorphic_on_prod_list [meromorphic_intros]:
assumes $\bigwedge i. i \in \text{set } fs \implies f i$ meromorphic_on A
shows $(\lambda w. \prod i \leftarrow fs. f i w)$ meromorphic_on A
unfolding meromorphic_on_def
by (rule laurent_expansion_intros exI ballI
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | assumption)
 +

lemma meromorphic_on_prod_mset [meromorphic_intros]:
assumes $\bigwedge i. i \in \# I \implies f i$ meromorphic_on A
shows $(\lambda w. \prod i \in \# I. f i w)$ meromorphic_on A
unfolding meromorphic_on_def

by (rule laurent_expansion_intros exI ballI
 assms[THEN meromorphic_on_imp_has_laurent_expansion] | assumption)+

lemma nicely_meromorphic_on_const [intro]: $(\lambda_. c)$ nicely_meromorphic_on A
unfolding nicely_meromorphic_on_def **by** auto

lemma nicely_meromorphic_on_cmult_left [intro]:
 assumes f nicely_meromorphic_on A
 shows $(\lambda z. c * f z)$ nicely_meromorphic_on A
proof (cases $c = 0$)
 case [simp]: False
 show ?thesis
 using assms **by** (rule nicely_meromorphic_on_unop) (auto intro!: meromorphic_intros)
qed (auto intro!: nicely_meromorphic_on_const)

lemma nicely_meromorphic_on_cmult_right [intro]:
 assumes f nicely_meromorphic_on A
 shows $(\lambda z. f z * c)$ nicely_meromorphic_on A
 using nicely_meromorphic_on_cmult_left[OF assms, of c] **by** (simp add: mult.commute)

lemma nicely_meromorphic_on_scaleR [intro]:
 assumes f nicely_meromorphic_on A
 shows $(\lambda z. c *_R f z)$ nicely_meromorphic_on A
 using assms **by** (auto simp: scaleR_conv_of_real)

lemma nicely_meromorphic_on_uminus [intro]:
 assumes f nicely_meromorphic_on A
 shows $(\lambda z. -f z)$ nicely_meromorphic_on A
 using nicely_meromorphic_on_cmult_left[OF assms, of -1] **by** simp

lemma meromorphic_on_If [meromorphic_intros]:
 assumes f meromorphic_on A g meromorphic_on B
 assumes $\bigwedge z. z \in A \implies z \in B \implies f z = g z$ open A open B $C \subseteq A \cup B$
 shows $(\lambda z. \text{if } z \in A \text{ then } f z \text{ else } g z)$ meromorphic_on C
proof (rule meromorphic_on_subset)
 show $(\lambda z. \text{if } z \in A \text{ then } f z \text{ else } g z)$ meromorphic_on $(A \cup B)$
proof (rule meromorphic_on_Un)
 have $(\lambda z. \text{if } z \in A \text{ then } f z \text{ else } g z)$ meromorphic_on $A \longleftrightarrow f$ meromorphic_on A
 A
proof (rule meromorphic_on_cong)
 fix z **assume** $z \in A$
 hence eventually $(\lambda z. z \in A)$ (at z)
 using $\langle \text{open } A \rangle$ **by** (intro eventually_at_in_open') auto
 thus $\forall_F w \text{ in at } z. (\text{if } w \in A \text{ then } f w \text{ else } g w) = f w$
by eventually_elim auto
qed auto

```

with assms(1) show ( $\lambda z. \text{if } z \in A \text{ then } f\ z \text{ else } g\ z$ ) meromorphic_on A
  by blast
next
have ( $\lambda z. \text{if } z \in A \text{ then } f\ z \text{ else } g\ z$ ) meromorphic_on B  $\longleftrightarrow$  g meromorphic_on
B
proof (rule meromorphic_on_cong)
  fix z assume  $z \in B$ 
  hence eventually ( $\lambda z. z \in B$ ) (at z)
    using  $\langle \text{open } B \rangle$  by (intro eventually_at_in_open') auto
  thus  $\forall_F w \text{ in } \text{at } z. (\text{if } w \in A \text{ then } f\ w \text{ else } g\ w) = g\ w$ 
    by eventually_elim (use assms(3)) in auto
qed auto
with assms(2) show ( $\lambda z. \text{if } z \in A \text{ then } f\ z \text{ else } g\ z$ ) meromorphic_on B
  by blast
qed
qed fact

```

```

lemma meromorphic_on_deriv [meromorphic_intros]:
  f meromorphic_on A  $\implies$  deriv f meromorphic_on A
by (metis meromorphic_on_def isolated_singularity_at_deriv meromorphic_on_isolated_singularity

```

```

    meromorphic_on_meromorphic_at meromorphic_on_not_essential
not_essential_deriv
    not_essential_has_laurent_expansion)

```

```

lemma meromorphic_on_higher_deriv [meromorphic_intros]:
  f meromorphic_on A  $\implies$  (deriv  $\hat{\sim}$  n) f meromorphic_on A
by (induction n) (auto intro!: meromorphic_intros)

```

```

lemma analytic_on_eval_fps [analytic_intros]:
  assumes f analytic_on A
  assumes  $\bigwedge z. z \in A \implies \text{norm } (f\ z) < \text{fps\_conv\_radius } F$ 
  shows ( $\lambda w. \text{eval\_fps } F\ (f\ w)$ ) analytic_on A
  by (rule analytic_on_compose[OF assms(1) analytic_on_eval_fps, unfolded
o_def])
    (use assms(2)) in auto)

```

```

lemma meromorphic_on_eval_fps [meromorphic_intros]:
  assumes f analytic_on A
  assumes  $\bigwedge z. z \in A \implies \text{norm } (f\ z) < \text{fps\_conv\_radius } F$ 
  shows ( $\lambda w. \text{eval\_fps } F\ (f\ w)$ ) meromorphic_on A
by (rule analytic_on_imp_meromorphic_on analytic_intros analytic_on_eval_fps
assms)+

```

```

lemma meromorphic_on_eval_fls [meromorphic_intros]:
  assumes f analytic_on A
  assumes  $\bigwedge z. z \in A \implies \text{norm } (f\ z) < \text{fls\_conv\_radius } F$ 
  shows ( $\lambda w. \text{eval\_fls } F\ (f\ w)$ ) meromorphic_on A
proof (cases fls_conv_radius F > 0)

```



```

case False
with assms(2) have  $A = \{\}$ 
  by (metis all_not_in_conv ereal_less(2) norm_eq_zero order.strict_trans
      zero_ereal_def zero_less_norm_iff)
thus ?thesis
  by auto
next
case True
have  $F$ : eval_fls  $F$  has_laurent_expansion  $F$ 
  using True by (rule eval_fls_has_laurent_expansion)
show ?thesis
proof (rule meromorphic_on_compose[OF assms(1)])
  show eval_fls  $F$  meromorphic_on eball 0 (fls_conv_radius  $F$ )
  proof (rule meromorphic_onI_open)
    show eval_fls  $F$  analytic_on eball 0 (fls_conv_radius  $F$ ) - {0}
    by (rule analytic_on_eval_fls) auto
    show not_essential (eval_fls  $F$ )  $z$  if  $z \in \{0\}$  for  $z$ 
    using that  $F$  has_laurent_expansion_not_essential_0 by blast
    qed (auto simp: islimpt_finite)
  qed (use assms(2) in auto)
qed

lemma meromorphic_on_imp_analytic_cosparse:
  assumes  $f$  meromorphic_on  $A$ 
  shows eventually ( $\lambda z. f$  analytic_on { $z$ }) (cosparse  $A$ )
  unfolding eventually_cosparse using assms meromorphic_on_imp_sparse_singularities
by auto

lemma meromorphic_on_imp_not_pole_cosparse:
  assumes  $f$  meromorphic_on  $A$ 
  shows eventually ( $\lambda z. \neg$ is_pole  $f$   $z$ ) (cosparse  $A$ )
proof -
  have eventually ( $\lambda z. f$  analytic_on { $z$ }) (cosparse  $A$ )
  by (rule meromorphic_on_imp_analytic_cosparse) fact
  thus ?thesis
  by eventually_elim (blast dest: analytic_at_imp_no_pole)
qed

lemma eventually_remove_sings_eq:
  assumes  $f$  meromorphic_on  $A$ 
  shows eventually ( $\lambda z. \text{remove\_sings } f\ z = f\ z$ ) (cosparse  $A$ )
proof -
  have eventually ( $\lambda z. f$  analytic_on { $z$ }) (cosparse  $A$ )
  using assms by (rule meromorphic_on_imp_analytic_cosparse)
  thus ?thesis
  by eventually_elim auto
qed

lemma remove_sings_constant_on_open_iff:

```

```

assumes  $f$  meromorphic_on  $A$  open  $A$ 
shows  $\text{remove\_sings } f \text{ constant\_on } A \longleftrightarrow (\exists c. \forall_{\approx} x \in A. f x = c)$ 
proof
  assume  $\text{remove\_sings } f \text{ constant\_on } A$ 
  then obtain  $c$  where  $c: \text{remove\_sings } f z = c$  if  $z \in A$  for  $z$ 
    using that by (auto simp: constant_on_def)
  have  $\forall_{\approx} x \in A. x \in A$ 
    using  $\langle \text{open } A \rangle$  by (simp add: eventually_in_cosparse)
  hence  $\forall_{\approx} x \in A. f x = c$ 
    using  $\text{eventually\_remove\_sings\_eq}[OF \text{ assms}(1)]$  by eventually_elim (use c in auto)
  thus  $\exists c. \forall_{\approx} x \in A. f x = c$ 
    by blast
next
  assume  $\exists c. \forall_{\approx} x \in A. f x = c$ 
  then obtain  $c$  where  $c: \forall_{\approx} x \in A. f x = c$ 
    by blast
  have  $\forall_{\approx} x \in A. \text{remove\_sings } f x = c$ 
    using  $\text{eventually\_remove\_sings\_eq}[OF \text{ assms}(1)]$   $c$  by eventually_elim auto
  hence  $\text{remove\_sings } f z = c$  if  $z \in A$  for  $z$  using that
    by (meson assms(2) c eventually_cosparse_open_eq remove_sings_eqI tendsto_eventually)
  thus  $\text{remove\_sings } f \text{ constant\_on } A$ 
    unfolding constant_on_def by blast
qed

```

A meromorphic function on a connected domain takes any given value either almost everywhere or almost nowhere.

lemma *meromorphic_imp_constant_or_avoid*:

```

assumes mero:  $f$  meromorphic_on  $A$  and  $A$ : open  $A$  connected  $A$ 
shows  $\text{eventually } (\lambda z. f z = c) (\text{cosparse } A) \vee \text{eventually } (\lambda z. f z \neq c) (\text{cosparse } A)$ 
proof –
  have  $\text{eventually } (\lambda z. f z = c) (\text{cosparse } A)$  if freq:  $\text{frequently } (\lambda z. f z = c) (\text{cosparse } A)$ 
  proof –
    let  $?f = \text{remove\_sings } f$ 
    have  $ev: \text{eventually } (\lambda z. ?f z = f z) (\text{cosparse } A)$ 
      by (rule eventually_remove_sings_eq fact)
    have  $\text{frequently } (\lambda z. ?f z = c) (\text{cosparse } A)$ 
      using  $\text{frequently\_eventually\_frequently}[OF \text{ freq } ev]$  by (rule frequently_elim1 auto)
    then obtain  $z0$  where  $z0: z0 \in A$   $\text{frequently } (\lambda z. ?f z = c) (\text{at } z0)$ 
      using  $A$  by (auto simp: eventually_cosparse_open_eq frequently_def)
    have  $\text{mero}'$ :  $?f$  nicely_meromorphic_on  $A$ 
      using mero  $\text{remove\_sings\_nicely\_meromorphic}$  by blast
    have  $eq: ?f w = c$  if  $w: w \in A$  for  $w$ 
      using  $\text{frequently\_eq\_meromorphic\_imp\_constant}[OF \text{ z0}(2) \text{ mero}']$   $A \text{ z0}(1)$ 
  w by blast

```

```

    have eventually ( $\lambda z. z \in A$ ) (cosparse A)
    by (rule eventually_in_cosparse) (use A in auto)
    thus eventually ( $\lambda z. f z = c$ ) (cosparse A)
    using ev by eventually_elim (use eq in auto)
  qed
  thus ?thesis
  by (auto simp: frequently_def)
qed

lemma nicely_meromorphic_imp_constant_or_avoid:
  assumes f_nicely_meromorphic_on A open A connected A
  shows ( $\forall x \in A. f x = c$ )  $\vee$  ( $\forall \approx x \in A. f x \neq c$ )
proof -
  have ( $\forall \approx x \in A. f x = c$ )  $\vee$  ( $\forall \approx x \in A. f x \neq c$ )
  by (intro meromorphic_imp_constant_or_avoid)
    (use assms in ‹auto simp: nicely_meromorphic_on_def›)
  thus ?thesis
  proof
    assume ev:  $\forall \approx x \in A. f x = c$ 
    have  $\forall x \in A. f x = c$ 
    proof
      fix x assume x:  $x \in A$ 
      have not_essential f x
      using assms x unfolding nicely_meromorphic_on_def by blast
      moreover have is_pole f x  $\longleftrightarrow$  is_pole ( $\lambda \_. c$ ) x
      by (intro is_pole_cong) (use ev x in ‹auto simp: eventually_cosparse_open_eq
        assms›)
      hence  $\neg$ is_pole f x
      by auto
      ultimately have f_analytic_on {x}
      using assms(1) nicely_meromorphic_on_imp_analytic_at x by blast
      hence f  $\rightarrow x \rightarrow f x$ 
      by (intro isContD analytic_at_imp_isCont)
      also have ?this  $\longleftrightarrow$  ( $\lambda \_. c$ )  $\rightarrow x \rightarrow f x$ 
      by (intro tendsto_cong) (use ev x in ‹auto simp: eventually_cosparse_open_eq
        assms›)
      finally have ( $\lambda \_. c$ )  $\rightarrow x \rightarrow f x$  .
      moreover have ( $\lambda \_. c$ )  $\rightarrow x \rightarrow c$ 
      by simp
      ultimately show f x = c
      using LIM_unique by blast
    qed
  qed
  thus ?thesis
  by blast
qed blast
qed

```

```

lemma nicely_meromorphic_onE:
  assumes f_nicely_meromorphic_on A

```

```

obtains pts where  $pts \subseteq A$  pts sparse_in A
  f analytic_on A - pts
   $\bigwedge z. z \in pts \implies is\_pole\ f\ z \wedge f\ z = 0$ 
proof -
  define pts where  $pts = \{z \in A. \neg f\ analytic\_on\ \{z\}\}$ 
  have  $pts \subseteq A$  pts sparse_in A
    using assms unfolding pts_def nicely_meromorphic_on_def
    by (auto intro: meromorphic_on_imp_sparse_singularities')
  moreover have f analytic_on A - pts unfolding pts_def
    by (subst analytic_on_analytic_at) auto
  moreover have  $\bigwedge z. z \in pts \implies is\_pole\ f\ z \wedge f\ z = 0$ 
    by (metis (no_types, lifting) remove_sings_eqI
      remove_sings_eq_0_iff assms is_pole_imp_not_essential
      mem_Collect_eq nicely_meromorphic_on_def
      nicely_meromorphic_on_imp_analytic_at pts_def)
  ultimately show ?thesis using that by auto
qed

lemma nicely_meromorphic_onI_open:
  assumes open A and
    analytic: f analytic_on A - pts and
    pole:  $\bigwedge x. x \in pts \implies is\_pole\ f\ x \wedge f\ x = 0$  and
    isolated:  $\bigwedge x. x \in A \implies isolated\_singularity\_at\ f\ x$ 
  shows f nicely_meromorphic_on A
proof -
  have f meromorphic_on A
  proof (rule meromorphic_onI_open)
    show  $\bigwedge z. z \in pts \implies not\_essential\ f\ z$ 
      using pole unfolding not_essential_def by auto
    show  $\bigwedge z. z \in A \implies \neg z\ islimpt\ pts \cap A$ 
      by (metis assms(3) assms(4) inf_commute inf_le2
        islimpt_subset mem_Collect_eq not_islimpt_poles subsetI)
  qed fact+
  moreover have  $(\bigvee z \in A. (is\_pole\ f\ z \wedge f\ z = 0) \vee f\ -z \rightarrow f\ z)$ 
    by (meson DiffI analytic_analytic_at_imp_isCont
      analytic_on_analytic_at assms(3) isContD)
  ultimately show ?thesis unfolding nicely_meromorphic_on_def
    by auto
qed

lemma nicely_meromorphic_without_singularities:
  assumes f nicely_meromorphic_on A  $\bigvee z \in A. \neg is\_pole\ f\ z$ 
  shows f analytic_on A
  by (meson analytic_on_analytic_at assms
    nicely_meromorphic_on_imp_analytic_at)

lemma meromorphic_on_cong':
  assumes eventually  $(\lambda z. f\ z = g\ z)$   $(cospars\ A)\ A = B$ 
  shows f meromorphic_on A  $\longleftrightarrow$  g meromorphic_on B

```

```

unfolding assms(2)[symmetric]
by (rule meromorphic_on_cong eventually_cospars_imp_eventually_at assms)+
auto

```

8.11 Meromorphic functions and zorder

```

lemma zorder_power_int:
  assumes f meromorphic_on {z} frequently ( $\lambda z. f\ z \neq 0$ ) (at z)
  shows zorder ( $\lambda z. f\ z\ \text{powi}\ n$ ) z = n * zorder f z
proof -
  from assms(1) obtain L where L: ( $\lambda w. f\ (z + w)$ ) has_laurent_expansion L
  by (auto simp: meromorphic_on_def)
  from assms(2) and L have [simp]: L  $\neq 0$ 
  by (metis assms(1) has_laurent_expansion_eventually_nonzero_iff meromor-
    phic_at_iff
      not_essential_frequently_0_imp_eventually_0 not_eventually_not_frequently)
  from L have L': ( $\lambda w. f\ (z + w)\ \text{powi}\ n$ ) has_laurent_expansion L powi n
  by (intro laurent_expansion_intros)
  have zorder f z = fls_subdegree L
  using L assms(2)  $\langle L \neq 0 \rangle$  by (simp add: has_laurent_expansion_zorder)
  moreover have zorder ( $\lambda z. f\ z\ \text{powi}\ n$ ) z = fls_subdegree (L powi n)
  using L' assms(2)  $\langle L \neq 0 \rangle$  by (simp add: has_laurent_expansion_zorder)
  moreover have fls_subdegree (L powi n) = n * fls_subdegree L
  by simp
  ultimately show ?thesis
  by simp
qed

```

```

lemma zorder_power:
  assumes f meromorphic_on {z} frequently ( $\lambda z. f\ z \neq 0$ ) (at z)
  shows zorder ( $\lambda z. f\ z^{\wedge} n$ ) z = n * zorder f z
  using zorder_power_int[OF assms, of int n] by simp

```

```

lemma zorder_add1:
  assumes f meromorphic_on {z} frequently ( $\lambda z. f\ z \neq 0$ ) (at z)
  assumes g meromorphic_on {z} frequently ( $\lambda z. g\ z \neq 0$ ) (at z)
  assumes zorder f z < zorder g z
  shows zorder ( $\lambda z. f\ z + g\ z$ ) z = zorder f z
proof -
  from assms(1) obtain F where F: ( $\lambda w. f\ (z + w)$ ) has_laurent_expansion F
  by (auto simp: meromorphic_on_def)
  from assms(3) obtain G where G: ( $\lambda w. g\ (z + w)$ ) has_laurent_expansion G
  by (auto simp: meromorphic_on_def)
  have [simp]: F  $\neq 0$  G  $\neq 0$ 
  by (metis assms has_laurent_expansion_eventually_nonzero_iff meromor-
    phic_at_iff
      not_essential_frequently_0_imp_eventually_0 not_eventually_not_frequently
      F G)+
  have *: zorder f z = fls_subdegree F zorder g z = fls_subdegree G

```

```

    using F G assms by (simp_all add: has_laurent_expansion_zorder)
  from assms * have F ≠ -G
  by auto
  hence [simp]: F + G ≠ 0
  by (simp add: add_eq_0_iff2)
  moreover have zorder (λz. f z + g z) z = fls_subdegree (F + G)
  using has_laurent_expansion_zorder[OF has_laurent_expansion_add[OF F
G]] ⟨F ≠ -G⟩ by simp
  moreover have fls_subdegree (F + G) = fls_subdegree F
  using assms by (simp add: * fls_subdegree_add_eq1)
  ultimately show ?thesis
  by (simp add: *)
qed

```

lemma zorder_add2:

```

  assumes f meromorphic_on {z} frequently (λz. f z ≠ 0) (at z)
  assumes g meromorphic_on {z} frequently (λz. g z ≠ 0) (at z)
  assumes zorder f z > zorder g z
  shows zorder (λz. f z + g z) z = zorder g z
  using zorder_add1[OF assms(3,4) assms(1,2)] assms(5-) by (simp add: add.commute)

```

lemma zorder_add_ge:

```

  fixes f g :: complex ⇒ complex
  assumes f meromorphic_on {z} frequently (λz. f z ≠ 0) (at z)
  assumes g meromorphic_on {z} frequently (λz. g z ≠ 0) (at z)
  assumes frequently (λz. f z + g z ≠ 0) (at z) zorder f z ≥ c zorder g z ≥ c
  shows zorder (λz. f z + g z) z ≥ c
proof -
  from assms(1) obtain F where F: (λw. f (z + w)) has_laurent_expansion F
  by (auto simp: meromorphic_on_def)
  from assms(3) obtain G where G: (λw. g (z + w)) has_laurent_expansion G
  by (auto simp: meromorphic_on_def)
  have [simp]: F ≠ 0 G ≠ 0
  using assms F G has_laurent_expansion_frequently_nonzero_iff by blast+
  have FG: (λw. f (z + w) + g (z + w)) has_laurent_expansion F + G
  by (intro laurent_expansion_intros F G)
  have [simp]: F + G ≠ 0
  using assms(5) has_laurent_expansion_frequently_nonzero_iff[OF FG] by
blast

```

```

  have *: zorder f z = fls_subdegree F zorder g z = fls_subdegree G
  zorder (λz. f z + g z) z = fls_subdegree (F + G)
  using F G FG has_laurent_expansion_zorder by simp_all
  moreover have zorder (λz. f z + g z) z = fls_subdegree (F + G)
  using has_laurent_expansion_zorder[OF has_laurent_expansion_add[OF F
G]] by simp
  moreover have fls_subdegree (F + G) ≥ min (fls_subdegree F) (fls_subdegree
G)

```

```

    by (intro fls_plus_subdegree) simp
  ultimately show ?thesis
    using assms(6,7) unfolding * by linarith
qed

```

```

lemma zorder_diff_ge:
  fixes f g :: complex  $\Rightarrow$  complex
  assumes f meromorphic_on {z} frequently ( $\lambda z. f z \neq 0$ ) (at z)
  assumes g meromorphic_on {z} frequently ( $\lambda z. g z \neq 0$ ) (at z)
  assumes frequently ( $\lambda z. f z \neq g z$ ) (at z) zorder f z  $\geq$  c zorder g z  $\geq$  c
  shows zorder ( $\lambda z. f z - g z$ ) z  $\geq$  c
proof -
  have ( $\lambda z. -g z$ ) meromorphic_on {z}
    by (auto intro: meromorphic_intros assms)
  thus ?thesis
    using zorder_add_ge[of f z  $\lambda z. -g z$  c] assms by simp
qed

```

```

lemma zorder_diff1:
  assumes f meromorphic_on {z} frequently ( $\lambda z. f z \neq 0$ ) (at z)
  assumes g meromorphic_on {z} frequently ( $\lambda z. g z \neq 0$ ) (at z)
  assumes zorder f z < zorder g z
  shows zorder ( $\lambda z. f z - g z$ ) z = zorder f z
proof -
  have zorder ( $\lambda z. f z + (-g z)$ ) z = zorder f z
    by (intro zorder_add1 meromorphic_intros assms) (use assms in auto)
  thus ?thesis
    by simp
qed

```

```

lemma zorder_diff2:
  assumes f meromorphic_on {z} frequently ( $\lambda z. f z \neq 0$ ) (at z)
  assumes g meromorphic_on {z} frequently ( $\lambda z. g z \neq 0$ ) (at z)
  assumes zorder f z > zorder g z
  shows zorder ( $\lambda z. f z - g z$ ) z = zorder g z
proof -
  have zorder ( $\lambda z. f z + (-g z)$ ) z = zorder ( $\lambda z. -g z$ ) z
    by (intro zorder_add2 meromorphic_intros assms) (use assms in auto)
  thus ?thesis
    by simp
qed

```

```

lemma zorder_mult:
  assumes f meromorphic_on {z} frequently ( $\lambda z. f z \neq 0$ ) (at z)
  assumes g meromorphic_on {z} frequently ( $\lambda z. g z \neq 0$ ) (at z)
  shows zorder ( $\lambda z. f z * g z$ ) z = zorder f z + zorder g z
proof -
  from assms(1) obtain F where F: ( $\lambda w. f (z + w)$ ) has_laurent_expansion F
    by (auto simp: meromorphic_on_def)

```

```

from assms(3) obtain G where G: ( $\lambda w. g(z + w)$ ) has_laurent_expansion G
by (auto simp: meromorphic_on_def)
have [simp]:  $F \neq 0 \ G \neq 0$ 
by (metis assms has_laurent_expansion_eventually_nonzero_iff meromorphic_at_iff
    not_essential_frequently_0_imp_eventually_0 not_eventually_not_frequently
    F G) +
have *: zorder  $f z = \text{fls\_subdegree } F$  zorder  $g z = \text{fls\_subdegree } G$ 
using F G assms by (simp_all add: has_laurent_expansion_zorder)
moreover have zorder ( $\lambda z. f z * g z$ )  $z = \text{fls\_subdegree } (F * G)$ 
using has_laurent_expansion_zorder[OF has_laurent_expansion_mult][OF F G] by simp
moreover have  $\text{fls\_subdegree } (F * G) = \text{fls\_subdegree } F + \text{fls\_subdegree } G$ 
using assms by simp
ultimately show ?thesis
by (simp add: *)
qed

```

lemma *zorder_divide*:

```

assumes f meromorphic_on {z} frequently ( $\lambda z. f z \neq 0$ ) (at z)
assumes g meromorphic_on {z} frequently ( $\lambda z. g z \neq 0$ ) (at z)
shows zorder ( $\lambda z. f z / g z$ )  $z = \text{zorder } f z - \text{zorder } g z$ 
proof –
from assms(1) obtain F where F: ( $\lambda w. f(z + w)$ ) has_laurent_expansion F
by (auto simp: meromorphic_on_def)
from assms(3) obtain G where G: ( $\lambda w. g(z + w)$ ) has_laurent_expansion G
by (auto simp: meromorphic_on_def)
have [simp]:  $F \neq 0 \ G \neq 0$ 
by (metis assms has_laurent_expansion_eventually_nonzero_iff meromorphic_at_iff
    not_essential_frequently_0_imp_eventually_0 not_eventually_not_frequently
    F G) +
have *: zorder  $f z = \text{fls\_subdegree } F$  zorder  $g z = \text{fls\_subdegree } G$ 
using F G assms by (simp_all add: has_laurent_expansion_zorder)
moreover have zorder ( $\lambda z. f z / g z$ )  $z = \text{fls\_subdegree } (F / G)$ 
using has_laurent_expansion_zorder[OF has_laurent_expansion_divide][OF F G] by simp
moreover have  $\text{fls\_subdegree } (F / G) = \text{fls\_subdegree } F - \text{fls\_subdegree } G$ 
using assms by (simp add: fls_divide_subdegree)
ultimately show ?thesis
by (simp add: *)
qed

```

lemma *constant_on_extend_nicely_meromorphic_on*:

```

assumes f nicely_meromorphic_on B f constant_on A
assumes open A open B connected B  $A \neq \{\}$   $A \subseteq B$ 
shows f constant_on B
proof –
from assms obtain c where  $c: \bigwedge z. z \in A \implies f z = c$ 

```



```

    by (auto simp: constant_on_def)
  have eventually ( $\lambda z. z \in A$ ) (cosparses A)
    by (intro eventually_in_cosparses assms order.refl)
  hence eventually ( $\lambda z. f z = c$ ) (cosparses A)
    by eventually_elim (use c in auto)
  hence freq: frequently ( $\lambda z. f z = c$ ) (cosparses A)
    by (intro eventually_frequently) (use assms in auto)
  then obtain z0 where z0:  $z0 \in A$  frequently ( $\lambda z. f z = c$ ) (at z0)
    using assms by (auto simp: frequently_def eventually_cosparses_open_eq)

  have  $f z = c$  if  $z \in B$  for  $z$ 
  proof (rule frequently_eq_meromorphic_imp_constant[OF _ assms(1)])
    show  $z0 \in B$  frequently ( $\lambda z. f z = c$ ) (at z0)
      using z0 assms by auto
    qed (use assms that in auto)
  thus  $f$  constant_on B
    by (auto simp: constant_on_def)
qed

```

8.12 More on poles and zeros

lemma zorder_prod:

```

  assumes  $\bigwedge x. x \in A \implies f x$  meromorphic_on  $\{z\}$ 
  assumes eventually ( $\lambda z. (\prod_{x \in A} f x z) \neq 0$ ) (at z)
  shows zorder ( $\lambda z. \prod_{x \in A} f x z$ )  $z = (\sum_{x \in A} zorder (f x) z)$ 
    using assms
  proof (induction A rule: infinite_finite_induct)
    case (insert a A)
    have zorder ( $\lambda z. \prod_{x \in \text{insert } a A} f x z$ )  $z = zorder (\lambda z. f a z * (\prod_{x \in A} f x z)) z$ 
      using insert.hyps by simp
    also have  $\dots = zorder (f a) z + zorder (\lambda z. \prod_{x \in A} f x z) z$ 
    proof (subst zorder_mult)
      have  $\forall_F z$  in at  $z. f a z \neq 0$ 
        using insert.premis(2) by eventually_elim (use insert.hyps in auto)
      thus  $\exists_F z$  in at  $z. f a z \neq 0$ 
        using eventually_frequently_at_neq_bot by blast
    next
      have  $\forall_F z$  in at  $z. (\prod_{x \in A} f x z) \neq 0$ 
        using insert.premis(2) by eventually_elim (use insert.hyps in auto)
      thus  $\exists_F z$  in at  $z. (\prod_{x \in A} f x z) \neq 0$ 
        using eventually_frequently_at_neq_bot by blast
    qed (use insert.premis in <auto intro!: meromorphic_intros>)
    also have zorder ( $\lambda z. \prod_{x \in A} f x z$ )  $z = (\sum_{x \in A} zorder (f x) z)$ 
      by (intro insert.IH) (use insert.premis insert.hyps in <auto elim!: eventually_mono>)
    also have zorder (f a)  $z + \dots = (\sum_{x \in \text{insert } a A} zorder (f x) z)$ 
      using insert.hyps by simp
    finally show ?case .
  qed auto

```

lemma *zorder_scale*:

```

  assumes f meromorphic_on {a * z} a ≠ 0
  shows zorder ( $\lambda w. f (a * w)$ ) z = zorder f (a * z)
proof (cases eventually ( $\lambda z. f z = 0$ ) (at (a * z)))
  case True
  hence ev: eventually ( $\lambda z. f (a * z) = 0$ ) (at z)
  proof (rule eventually_compose_filterlim)
    show filterlim ((*) a) (at (a * z)) (at z)
    proof (rule filterlim_atI)
      show  $\forall_F x \text{ in } \textit{at } z. a * x \neq a * z$ 
      using eventually_neq_at_within[of z z] by eventually_elim (use  $\langle a \neq 0 \rangle$ )
    qed
  qed (auto intro!: tendsto_intros)
qed

```

```

  have zorder ( $\lambda w. f (a * w)$ ) z = zorder ( $\lambda \_. 0$ ) z
    by (rule zorder_cong) (use ev in auto)
  also have ... = zorder ( $\lambda \_. 0$ ) (a * z)
    by (simp add: zorder_shift')
  also have ... = zorder f (a * z)
    by (rule zorder_cong) (use True in auto)
  finally show ?thesis .

```

next

```

  case False
  define G where G = fps_const a * fps_X
  have zorder (f ∘ ( $\lambda z. a * z$ )) z = zorder f (a * z) * int (subdegree G)
  proof (rule zorder_compose)
    show isolated_singularity_at f (a * z) not_essential f (a * z)
      using assms(1) by (auto simp: meromorphic_on_altdef)
    next
    have ( $\lambda x. a * x$ ) has_fps_expansion G
      unfolding G_def by (intro fps_expansion_intros)
    thus ( $\lambda x. a * (z + x) - a * z$ ) has_fps_expansion G
      by (simp add: algebra_simps)
    next
    show  $\forall_F w \text{ in } \textit{at } (a * z). f w \neq 0$  using False
    by (metis assms(1) has_laurent_expansion_isolated has_laurent_expansion_not_essential
      meromorphic_on_def non_zero_neighbour not_eventually_singletonI)
    qed (use  $\langle a \neq 0 \rangle$  in  $\langle \textit{auto simp: G\_def} \rangle$ )
    also have subdegree G = 1
      using  $\langle a \neq 0 \rangle$  by (simp add: G_def)
    finally show ?thesis
      by (simp add: o_def)
    qed

```

lemma *zorder_uminus*:

```

  assumes f meromorphic_on {−z}
  shows zorder ( $\lambda w. f (-w)$ ) z = zorder f (−z)

```

```

using assms zorder_scale[of f -1 z] by simp

lemma is_pole_deriv_iff:
  assumes f_meromorphic_on {z}
  shows is_pole (deriv f) z  $\longleftrightarrow$  is_pole f z
proof -
  from assms obtain F where F: ( $\lambda w. f (z + w)$ ) has_laurent_expansion F
  by (auto simp: meromorphic_on_def)
  have deriv ( $\lambda w. f (z + w)$ ) has_laurent_expansion fls_deriv F
  using F by (rule has_laurent_expansion_deriv)
  also have deriv ( $\lambda w. f (z + w)$ ) = ( $\lambda w. \text{deriv } f (z + w)$ )
  by (simp add: deriv_shift_0' add_ac o_def fun_eq_iff)
  finally have F': ( $\lambda w. \text{deriv } f (z + w)$ ) has_laurent_expansion fls_deriv F .
  have is_pole (deriv f) z  $\longleftrightarrow$  fls_subdegree (fls_deriv F) < 0
  using is_pole_fls_subdegree_iff[OF F'] by simp
  also have ...  $\longleftrightarrow$  fls_subdegree F < 0
  using fls_deriv_subdegree0 fls_subdegree_deriv linorder_less_linear by fast-
  force
  also have ...  $\longleftrightarrow$  is_pole f z
  using F by (simp add: has_laurent_expansion_imp_is_pole_iff)
  finally show ?thesis .
qed

lemma isolated_zero_remove_sings_iff [simp]:
  assumes isolated_singularity_at f z
  shows isolated_zero (remove_sings f) z  $\longleftrightarrow$  isolated_zero f z
proof -
  have *: ( $\forall_F x \text{ in at } z. \text{remove\_sings } f \ x \neq 0$ )  $\longleftrightarrow$  ( $\forall_F x \text{ in at } z. f \ x \neq 0$ )
  proof
    assume ( $\forall_F x \text{ in at } z. f \ x \neq 0$ )
    thus ( $\forall_F x \text{ in at } z. \text{remove\_sings } f \ x \neq 0$ )
      using eventually_remove_sings_eq_at[OF assms]
      by eventually_elim auto
  next
    assume ( $\forall_F x \text{ in at } z. \text{remove\_sings } f \ x \neq 0$ )
    thus ( $\forall_F x \text{ in at } z. f \ x \neq 0$ )
      using eventually_remove_sings_eq_at[OF assms]
      by eventually_elim auto
  qed
  show ?thesis
  unfolding isolated_zero_def using assms * by simp
qed

lemma zorder_isolated_zero_pos:
  assumes isolated_zero f z f_analytic_on {z}
  shows zorder f z > 0
proof (subst zorder_pos_iff' [OF assms(2)])
  show f z = 0
  using assms by (simp add: zero_isolated_zero_analytic)

```

```

next
  have  $\forall_F z \text{ in at } z. f z \neq 0$ 
    using assms by (auto simp: isolated_zero_def)
  thus  $\exists_F z \text{ in at } z. f z \neq 0$ 
    by (simp add: eventually_frequently)
qed

```

```

lemma zorder_isolated_zero_pos':
  assumes isolated_zero f z isolated_singularity_at f z
  shows zorder f z > 0
proof -
  from assms(1) have f  $\rightarrow 0$ 
    by (simp add: isolated_zero_def)
  with assms(2) have remove_sings f analytic_on {z}
    by (intro remove_sings_analytic_at)
  hence zorder (remove_sings f) z > 0
    using assms by (intro zorder_isolated_zero_pos) auto
  thus ?thesis
    using assms by simp
qed

```

```

lemma zero_isolated_zero_nicely_meromorphic:
  assumes isolated_zero f z f nicely_meromorphic_on {z}
  shows f z = 0
proof -
  have  $\neg \text{is\_pole } f z$ 
    using assms pole_is_not_zero by blast
  with assms(2) have f analytic_on {z}
    by (simp add: nicely_meromorphic_on_imp_analytic_at)
  with zero_isolated_zero_analytic assms(1) show ?thesis
    by blast
qed

```

```

lemma meromorphic_on_imp_not_zero_cosparse:
  assumes f meromorphic_on A
  shows eventually ( $\lambda z. \neg \text{is\_pole } f z$ ) (cosparse A)
proof -
  have eventually ( $\lambda z. \neg \text{is\_pole } (\lambda z. \text{inverse } (f z)) z$ ) (cosparse A)
    by (intro meromorphic_on_imp_not_pole_cosparse meromorphic_intros assms)
  thus ?thesis
    by (simp add: is_pole_inverse_iff)
qed

```

```

lemma nicely_meromorphic_on_inverse [meromorphic_intros]:
  assumes f nicely_meromorphic_on A
  shows ( $\lambda x. \text{inverse } (f x)$ ) nicely_meromorphic_on A
  unfolding nicely_meromorphic_on_def
proof (intro conjI ballI)
  fix z assume z:  $z \in A$ 

```

```

have is_pole f z  $\wedge$  f z = 0  $\vee$  f -z  $\rightarrow$  f z
  using assms z by (auto simp: nicely_meromorphic_on_def)
thus is_pole ( $\lambda x$ . inverse (f x)) z  $\wedge$  inverse (f z) = 0  $\vee$ 
  ( $\lambda x$ . inverse (f x)) -z  $\rightarrow$  inverse (f z)
proof
  assume is_pole f z  $\wedge$  f z = 0
  hence isolated_zero ( $\lambda z$ . inverse (f z)) z  $\wedge$  inverse (f z) = 0
    by (auto simp: isolated_zero_inverse_iff)
  hence ( $\lambda x$ . inverse (f x)) -z  $\rightarrow$  inverse (f z)
    by (simp add: isolated_zero_def)
  thus ?thesis ..
next
  assume lim: f -z  $\rightarrow$  f z
  hence ana: f analytic_on {z}
    using assms is_pole_def nicely_meromorphic_on_imp_analytic_at
      not_tendsto_and_filterlim_at_infinity trivial_limit_at z by blast
  show ?thesis
  proof (cases isolated_zero f z)
    case True
    with lim have f z = 0
      using continuous_within zero_isolated_zero by blast
    with True have is_pole ( $\lambda z$ . inverse (f z)) z  $\wedge$  inverse (f z) = 0
      by (auto simp: is_pole_inverse_iff)
    thus ?thesis ..
  next
    case False
    hence f z  $\neq$  0  $\vee$  (f z = 0  $\wedge$  eventually ( $\lambda z$ . f z = 0) (at z))
      using non_isolated_zero_imp_eventually_zero[OF ana] by blast
    thus ?thesis
    proof (elim disjE conjE)
      assume f z  $\neq$  0
      hence ( $\lambda z$ . inverse (f z)) -z  $\rightarrow$  inverse (f z)
        by (intro tendsto_intros lim)
      thus ?thesis ..
    next
      assume *: f z = 0 eventually ( $\lambda z$ . f z = 0) (at z)
      have eventually ( $\lambda z$ . inverse (f z) = 0) (at z)
        using *(2) by eventually_elim auto
      hence ( $\lambda z$ . inverse (f z)) -z  $\rightarrow$  0
        by (simp add: tendsto_eventually)
      with *(1) show ?thesis
        by auto
    qed
  qed
qed
qed
qed (use assms in  $\langle$ auto simp: nicely_meromorphic_on_def intro!: meromorphic_intros $\rangle$ )

lemma is_pole_zero_at_nicely_mero:
  assumes f nicely_meromorphic_on A is_pole f z z  $\in$  A

```

```

shows  $f z = 0$ 
by (meson assms at_neq_bot
    is_pole_def nicely_meromorphic_on_def
    not_tendsto_and_filterlim_at_infinity)

lemma zero_or_pole:
  assumes mero:  $f$  nicely_meromorphic_on  $A$ 
    and  $z \in A$   $f z = 0$  and event:  $\forall_F x$  in at  $z$ .  $f x \neq 0$ 
  shows isolated_zero  $f z \vee$  is_pole  $f z$ 
proof -
  from mero  $\langle z \in A \rangle$ 
  have (is_pole  $f z \wedge f z = 0$ )  $\vee f -z \rightarrow f z$ 
    unfolding nicely_meromorphic_on_def by simp
  moreover have isolated_zero  $f z$  if  $f -z \rightarrow f z$ 
    unfolding isolated_zero_def
    using  $\langle f z = 0 \rangle$  that event tendsto_nhds_iff by auto
  ultimately show ?thesis by auto
qed

lemma isolated_zero_fls_subdegree_iff:
  assumes  $(\lambda x. f (z + x))$  has_laurent_expansion  $F$ 
  shows isolated_zero  $f z \longleftrightarrow$  fls_subdegree  $F > 0$ 
  using assms unfolding isolated_zero_def
  by (metis Lim_at_zero_fls_zero_subdegree has_laurent_expansion_eventually_nonzero_iff
    not_le
    order.refl tendsto_0_subdegree_iff_0)

lemma zorder_pos_imp_isolated_zero:
  assumes  $f$  meromorphic_on  $\{z\}$  eventually  $(\lambda z. f z \neq 0)$  (at  $z$ )  $\text{zorder } f z > 0$ 
  shows isolated_zero  $f z$ 
  using assms isolated_zero_fls_subdegree_iff
  by (metis has_laurent_expansion_eventually_nonzero_iff
    has_laurent_expansion_zorder insertI1
    meromorphic_on_def)

lemma zorder_neg_imp_is_pole:
  assumes  $f$  meromorphic_on  $\{z\}$  eventually  $(\lambda z. f z \neq 0)$  (at  $z$ )  $\text{zorder } f z < 0$ 
  shows is_pole  $f z$ 
  using assms is_pole_fls_subdegree_iff at_neq_bot eventually_frequently_meromorphic_at_iff
  neg_zorder_imp_is_pole by blast

lemma not_pole_not_isolated_zero_imp_zorder_eq_0:
  assumes  $f$  meromorphic_on  $\{z\}$   $\neg$ is_pole  $f z \neg$ isolated_zero  $f z$  frequently  $(\lambda z. f z \neq 0)$  (at  $z$ )
  shows  $\text{zorder } f z = 0$ 
proof -
  have remove_sings  $f$  analytic_on  $\{z\}$ 
    using assms meromorphic_at_iff not_essential_def remove_sings_analytic_at

```

```

by blast
  moreover from this and assms have remove_sings f z  $\neq$  0
  using isolated_zero_def meromorphic_at_iff non_zero_neighbour remove_sings_eq_0_iff
by blast
  moreover have frequently ( $\lambda z$ . remove_sings f z  $\neq$  0) (at z)
  using assms analytic_at_neq_imp_eventually_neq calculation(1,2)
  eventually_frequently trivial_limit_at by blast
  ultimately have zorder (remove_sings f) z = 0
  using zorder_eq_0_iff by blast
  thus ?thesis
  using assms(1) meromorphic_at_iff by auto
qed

```

lemma not_essential_compose:

```

  assumes not_essential f (g z) g analytic_on {z}
  shows not_essential ( $\lambda x$ . f (g x)) z
proof (cases isolated_zero ( $\lambda w$ . g w - g z) z)
case False
  hence eventually ( $\lambda w$ . g w - g z = 0) (nhds z)
  by (intro non_isolated_zero_imp_eventually_zero' analytic_intros assms) auto
  hence not_essential ( $\lambda x$ . f (g x)) z  $\longleftrightarrow$  not_essential ( $\lambda \_$ . f (g z)) z
  by (intro not_essential_cong refl)
  (auto elim!: eventually_mono simp: eventually_at_filter)
  thus ?thesis
  by (simp add: not_essential_const)
next
case True
  hence ev: eventually ( $\lambda w$ . g w  $\neq$  g z) (at z)
  by (auto simp: isolated_zero_def)
  from assms consider c where f -g z  $\rightarrow$  c | is_pole f (g z)
  by (auto simp: not_essential_def)
  have isCont g z
  by (rule analytic_at_imp_isCont) fact
  hence lim: g -z  $\rightarrow$  g z
  using isContD by blast

```

```

from assms(1) consider c where f -g z  $\rightarrow$  c | is_pole f (g z)

```

```

  unfolding not_essential_def by blast

```

```

thus ?thesis

```

```

proof cases

```

```

  fix c assume f -g z  $\rightarrow$  c

```

```

  hence ( $\lambda x$ . f (g x)) -z  $\rightarrow$  c

```

```

  by (rule filterlim_compose) (use lim ev in <auto simp: filterlim_at>)

```

```

  thus ?thesis

```

```

  by (auto simp: not_essential_def)

```

```

next

```

```

  assume is_pole f (g z)

```

```

  hence is_pole ( $\lambda x$ . f (g x)) z

```

```

  by (rule is_pole_compose) fact+

```

```

    thus ?thesis
      by (auto simp: not_essential_def)
  qed
qed

lemma isolated_singularity_at_compose:
  assumes isolated_singularity_at f (g z) g analytic_on {z}
  shows isolated_singularity_at ( $\lambda x. f (g x)$ ) z
proof (cases isolated_zero ( $\lambda w. g w - g z$ ) z)
  case False
    hence eventually ( $\lambda w. g w - g z = 0$ ) (nhds z)
      by (intro non_isolated_zero_imp_eventually_zero') (use assms in <auto intro!: analytic_intros>)
    hence isolated_singularity_at ( $\lambda x. f (g x)$ ) z  $\longleftrightarrow$  isolated_singularity_at ( $\lambda_. f (g z)$ ) z
      by (intro isolated_singularity_at_cong_refl)
      (auto elim!: eventually_mono simp: eventually_at_filter)
    thus ?thesis
      by (simp add: isolated_singularity_at_const)
  next
  case True
    from assms(1) obtain r where r:  $r > 0$  f analytic_on ball (g z) r - {g z}
      by (auto simp: isolated_singularity_at_def)
    hence holo_f: f holomorphic_on ball (g z) r - {g z}
      by (subst (asm) analytic_on_open) auto
    from assms(2) obtain r' where r':  $r' > 0$  g holomorphic_on ball z r'
      by (auto simp: analytic_on_def)

    have continuous_on (ball z r') g
      using holomorphic_on_imp_continuous_on r' by blast
    hence isCont g z
      using r' by (subst (asm) continuous_on_eq_continuous_at) auto
    hence  $g - z \rightarrow g z$ 
      using isContD by blast
    hence eventually ( $\lambda w. g w \in \text{ball } (g z) r$ ) (at z)
      using <r > 0> unfolding tendsto_def by force
    moreover have eventually ( $\lambda w. g w \neq g z$ ) (at z) using True
      by (auto simp: isolated_zero_def elim!: eventually_mono)
    ultimately have eventually ( $\lambda w. g w \in \text{ball } (g z) r - \{g z\}$ ) (at z)
      by eventually_elim auto
    then obtain r'' where r'':  $r'' > 0 \ \forall w \in \text{ball } z \ r'' - \{z\}. g w \in \text{ball } (g z) r - \{g z\}$ 
      by
        unfolding eventually_at_filter eventually_nhds_metric ball_def
        by (auto simp: dist_commute)
    have f  $\circ$  g holomorphic_on ball z (min r' r'') - {z}
    proof (rule holomorphic_on_compose_gen)
      show g holomorphic_on ball z (min r' r'') - {z}
        by (rule holomorphic_on_subset[OF r'(2)]) auto
    end
  end

```



```

    show f holomorphic_on ball (g z) r - {g z}
      by fact
    show g ' (ball z (min r' r'') - {z})  $\subseteq$  ball (g z) r - {g z}
      using r'' by force
  qed
  hence f  $\circ$  g analytic_on ball z (min r' r'') - {z}
    by (subst analytic_on_open) auto
  thus ?thesis using <r' > 0> <r'' > 0>
    by (auto simp: isolated_singularity_at_def o_def intro!: exI[of _ min r' r''])
  qed

lemma is_pole_power_int_0:
  assumes f analytic_on {x} isolated_zero f x n < 0
  shows is_pole ( $\lambda x. f x \text{ powi } n$ ) x
proof -
  have f  $-x \rightarrow f x$ 
    using assms(1) by (simp add: analytic_at_imp_isCont isContD)
  with assms show ?thesis
    unfolding is_pole_def
    by (intro filterlim_power_int_neg_at_infinity) (auto simp: isolated_zero_def)
  qed

lemma isolated_zero_imp_not_constant_on:
  fixes f :: 'a :: perfect_space  $\Rightarrow$  'b :: real_normed_div_algebra
  assumes isolated_zero f x x  $\in$  A open A
  shows  $\neg$ f constant_on A
proof
  assume f constant_on A
  then obtain c where c:  $\bigwedge x. x \in A \Rightarrow f x = c$ 
    by (auto simp: constant_on_def)
  have eventually ( $\lambda z. z \in A - \{x\}$ ) (at x)
    by (intro eventually_at_in_open assms)
  hence eventually ( $\lambda z. f z = c$ ) (at x)
    by eventually_elim (use c in auto)
  hence f  $-x \rightarrow c$ 
    using tendsto_eventually by blast
  moreover from assms have f  $-x \rightarrow 0$ 
    by (simp add: isolated_zero_def)
  ultimately have [simp]: c = 0
    using tendsto_unique[of at x f c 0] by (simp add: at_neq_bot)

  have eventually ( $\lambda x. f x \neq 0$ ) (at x)
    using assms by (auto simp: isolated_zero_def)
  moreover have eventually ( $\lambda x. x \in A$ ) (at x)
    using assms by (intro eventually_at_in_open') auto
  ultimately have eventually ( $\lambda x. \text{False}$ ) (at x)
    by eventually_elim (use c in auto)
  thus False
    by simp

```

qed

end

9 The Weierstraß Factorisation Theorem

```
theory Weierstrass_Factorization
  imports Meromorphic
begin
```

9.1 The elementary factors

The Weierstraß elementary factors are the family of entire functions

$$E_n(z) = (1 - z) \exp \left(z + \frac{z^2}{2} + \dots + \frac{z^n}{n} \right)$$

with the key properties that they have a single zero at $z = 1$ and satisfy $E_n(z) = 1 + O(z^{n+1})$ around the origin.

definition *weierstrass_factor* :: *nat* \Rightarrow *complex* \Rightarrow *complex* **where**
weierstrass_factor *n* *z* = $(1 - z) * \exp (\sum_{k=1..n}. z^k / of_nat\ k)$

lemma *weierstrass_factor_continuous_on* [*continuous_intros*]:
continuous_on *A* *f* \implies *continuous_on* *A* $(\lambda z. \text{weierstrass_factor } n\ (f\ z))$
by (*auto simp: weierstrass_factor_def intro!: continuous_intros*)

lemma *weierstrass_factor_holomorphic* [*holomorphic_intros*]:
f *holomorphic_on* *A* $\implies (\lambda z. \text{weierstrass_factor } n\ (f\ z))$ *holomorphic_on* *A*
by (*auto simp: weierstrass_factor_def intro!: holomorphic_intros*)

lemma *weierstrass_factor_analytic* [*analytic_intros*]:
f *analytic_on* *A* $\implies (\lambda z. \text{weierstrass_factor } n\ (f\ z))$ *analytic_on* *A*
by (*auto simp: weierstrass_factor_def intro!: analytic_intros*)

lemma *weierstrass_factor_0* [*simp*]: *weierstrass_factor* *n* 0 = 1
by (*auto simp: weierstrass_factor_def power_0_left*)

lemma *weierstrass_factor_1* [*simp*]: *weierstrass_factor* *n* 1 = 0
by (*simp add: weierstrass_factor_def*)

lemma *weierstrass_factor_eq_0_iff* [*simp*]: *weierstrass_factor* *n* *z* = 0 \longleftrightarrow *z* = 1
by (*simp add: weierstrass_factor_def*)

lemma *zorder_weierstrass_factor* [*simp*]: *zorder* (*weierstrass_factor* *n*) 1 = 1
proof (*rule zorder_eqI*)
show $(\lambda z. -\exp (\sum_{k=1..n}. z^k / of_nat\ k))$ *holomorphic_on* *UNIV*
by (*intro holomorphic_intros auto*)

qed (auto simp: weierstrass_factor_def algebra_simps)

lemma

fixes $z :: 'a :: \{\text{banach}, \text{real_normed_field}\}$
assumes $\text{norm } z \leq 1 / 2$
shows $\text{norm_exp_bounds_lemma: norm } (\exp z - 1 - z) \leq \text{norm } z / 2$
and $\text{norm_exp_bounds: norm } (\exp z - 1) \geq 1 / 2 * \text{norm } z$
 $\text{norm } (\exp z - 1) \leq 3 / 2 * \text{norm } z$
proof –
show *: $\text{norm } (\exp z - 1 - z) \leq \text{norm } z / 2$
proof (cases $z = 0$)
case False
have $\text{sums: } (\lambda k. z^{(k+2)} / \text{fact } (k+2)) \text{ sums } (\exp z - (\sum_{k<2}. z^k / \text{fact } k))$
by (intro sums_split_initial_segment exp_converges)

have $\text{summable } (\lambda k. \text{norm } z^{(k+2)} / \text{fact } (k+2))$
using summable_norm_exp[of z]
by (intro summable_norm summable_ignore_initial_segment)
(auto simp: norm_power norm_divide divide_simps)
also have $?this \longleftrightarrow \text{summable } (\lambda k. \text{norm } z^2 * (\text{norm } z^k / \text{fact } (k+2)))$
by (simp add: power_add mult_ac divide_simps power2_eq_square del: of_nat_Suc of_nat_add)
also have $\dots \longleftrightarrow \text{summable } (\lambda k. \text{norm } z^k / \text{fact } (k+2))$
by (subst summable_cmult_iff) (use $\langle z \neq 0 \rangle$ in auto)
finally have $\text{summable: summable } (\lambda k. \text{norm } z^k / \text{fact } (k+2))$.

have $\exp z - 1 - z = (\sum k. z^{(k+2)} / \text{fact } (k+2))$
using sums **by** (simp add: sums_iff scaleR_conv_of_real divide_simps eval_nat_numeral)
also have $\text{norm } \dots \leq (\sum k. \text{norm } (z^{(k+2)} / \text{fact } (k+2)))$
using summable_norm_exp[of z]
by (intro summable_norm summable_ignore_initial_segment)
(auto simp: norm_power norm_divide divide_simps)
also have $\dots = (\sum k. \text{norm } z^2 * (\text{norm } z^k / \text{fact } (k+2)))$
by (simp add: power_add norm_power norm_divide mult_ac norm_mult power2_eq_square del: of_nat_Suc)
also have $\dots = \text{norm } z^2 * (\sum k. \text{norm } z^k / \text{fact } (k+2))$
using summable **by** (rule suminf_mult)
also have $\dots \leq \text{norm } z^2 * (1 / (1 - \text{norm } z) / 2)$
proof (intro mult_left_mono, rule sums_le)
show $(\lambda k. \text{norm } z^k / \text{fact } (k+2)) \text{ sums } (\sum k. \text{norm } z^k / \text{fact } (k+2))$
using summable **by** blast
show $(\lambda k. \text{norm } z^k / 2) \text{ sums } (1 / (1 - \text{norm } z) / 2)$
using assms **by** (intro geometric_sums sums_divide) auto
next
fix $k :: \text{nat}$
have $\text{norm } z^k / \text{fact } (k+2) \leq \text{norm } z^k / \text{fact } 2$
by (intro divide_left_mono fact_mono) auto

```

      thus  $\text{norm } z^k / \text{fact } (k + 2) \leq \text{norm } z^k / 2$ 
      by simp
    qed (auto simp: divide_simps)
  also have  $\dots = \text{norm } z * (\text{norm } z / (1 - \text{norm } z)) / 2$ 
  by (simp add: power2_eq_square)
  also have  $\dots \leq \text{norm } z * ((1 / 2) / (1 - (1 / 2))) / 2$ 
  using assms by (intro mult_mono frac_le diff_mono) auto
  also have  $\dots = \text{norm } z / 2$ 
  by simp
  finally show  $\text{norm } (\exp z - 1 - z) \leq \text{norm } z / 2$  .
qed auto

have  $\text{norm } (\exp z - 1) \leq \text{norm } z + \text{norm } (\exp z - 1 - z)$ 
by (rule norm_triangle_sub)
with * show  $\text{norm } (\exp z - 1) \leq 3 / 2 * \text{norm } z$ 
by simp

have  $\text{norm } z - \text{norm } (\exp z - 1 - z) \leq \text{norm } (\exp z - 1)$ 
using norm_triangle_ineq3[of  $\exp z - 1 - z - z$ ] by simp
with * show  $\text{norm } (\exp z - 1) \geq 1 / 2 * \text{norm } z$ 
by simp
qed

lemma weierstrass_factor_bound:
  assumes  $\text{norm } z \leq 1 / 2$ 
  shows  $\text{norm } (\text{weierstrass\_factor } n \ z - 1) \leq 3 * \text{norm } z^{\text{Suc } n}$ 
proof (cases  $n = 0 \vee z = 0$ )
  case True
  thus ?thesis
  proof
    assume  $n = 0$ 
    thus ?thesis by (auto simp: weierstrass_factor_def)
  qed auto
next
  case False
  with assms have  $z \neq 1 \wedge n > 0 \wedge z \neq 0$ 
  by auto

  have summable  $(\lambda k. \text{cmod } z^{(k + \text{Suc } n)} / \text{real } (k + \text{Suc } n))$ 
  using ln_series'[of  $-\text{norm } z$ ] assms
  by (intro summable_norm summable_ignore_initial_segment)
  (simp_all add: sums_iff summable_minus_iff power_minus' norm_divide
  norm_power)
  also have  $?this \longleftrightarrow \text{summable } (\lambda k. \text{norm } z^{\text{Suc } n} * (\text{norm } z^k / \text{real } (k + \text{Suc } n)))$ 
  by (simp add: power_add mult_ac)
  also have  $\dots \longleftrightarrow \text{summable } (\lambda k. \text{norm } z^k / \text{real } (k + \text{Suc } n))$ 
  by (subst summable_cmult_iff) (use  $\langle z \neq 0 \rangle$  in auto)
  finally have summable: summable  $(\lambda k. \text{norm } z^k / \text{real } (k + \text{Suc } n))$  .

```

```

have ( $\lambda k. z^k / \text{of\_nat } k$ ) sums  $- \text{Ln } (1 - z)$ 
  using sums_minus[OF Ln_series[of  $-z$ ]] assms by (simp add: power_minus')
hence ( $\lambda k. z^{(k + \text{Suc } n)} / \text{of\_nat } (k + \text{Suc } n)$ ) sums  $(- \text{Ln } (1 - z) -$ 
 $(\sum_{k < \text{Suc } n} z^k / \text{of\_nat } k))$ 
  by (intro sums_split_initial_segment)
also have  $(\sum_{k < \text{Suc } n} z^k / \text{of\_nat } k) = (\sum_{k=1..n} z^k / \text{of\_nat } k)$ 
  by (intro sum.mono_neutral_right) auto
finally have norm  $(\ln (1 - z) + (\sum_{k=1..n} z^k / \text{of\_nat } k)) =$ 
  norm  $(\sum_{k. z^{(k + \text{Suc } n)} / \text{of\_nat } (k + \text{Suc } n)})$ 
  by (simp add: sums_iff_norm_uminus_minus)

also have  $\dots \leq (\sum_{k. \text{norm } (z^{(k + \text{Suc } n)} / \text{of\_nat } (k + \text{Suc } n)))$ 
  using ln_series'[of  $-\text{norm } z$ ] assms
  by (intro summable_norm summable_ignore_initial_segment)
  (simp_all add: sums_iff_summable_minus_iff_power_minus' norm_divide
norm_power)
also have  $\dots = (\sum_{k. \text{norm } z^{\text{Suc } n} * (\text{norm } z^k / \text{real } (k + \text{Suc } n)))$ 
  by (simp add: algebra_simps norm_mult norm_power norm_divide power_add
del: of_nat_add of_nat_Suc)
also have  $\dots = \text{norm } z^{\text{Suc } n} * (\sum_{k. \text{norm } z^k / \text{real } (k + \text{Suc } n)})$ 
  by (intro suminf_mult summable)
also have  $\dots \leq \text{norm } z^{\text{Suc } n} * (1 / (1 - \text{norm } z))$ 
proof (intro mult_left_mono[OF sums_le])
  show  $(\lambda k. \text{norm } z^k / \text{real } (k + \text{Suc } n))$  sums  $(\sum_{k. \text{norm } z^k / \text{real } (k +$ 
 $\text{Suc } n))$ 
    using summable by blast
  show  $(\lambda k. \text{norm } z^k)$  sums  $(1 / (1 - \text{norm } z))$ 
    by (rule geometric_sums) (use assms in auto)
qed (auto simp: field_simps)
also have  $\text{norm } z^{\text{Suc } n} * (1 / (1 - \text{norm } z)) \leq \text{norm } z^{\text{Suc } n} * (1 / (1 -$ 
 $(1 / 2)))$ 
  using assms by (intro mult_mono power_mono divide_left_mono diff_mono
mult_pos_pos) auto
also have  $\dots = 2 * \text{norm } z^{\text{Suc } n}$ 
  by simp
finally have norm_le: norm  $(\ln (1 - z) + (\sum_{k=1..n} z^k / \text{of\_nat } k)) \leq 2$ 
 $* \text{norm } z^{\text{Suc } n} .$ 

also have  $\dots \leq 2 * \text{norm } z^2$ 
  using  $\langle n > 0 \rangle$  assms by (intro mult_left_mono power_decreasing) auto
also have  $\dots \leq 2 * (1 / 2)^2$ 
  by (intro mult_left_mono assms power_mono) auto
finally have norm_le': norm  $(\ln (1 - z) + (\sum_{k=1..n} z^k / \text{of\_nat } k)) \leq 1$ 
 $/ 2$ 
  by (simp add: power2_eq_square)

have weierstrass_factor  $n \ z = \exp (\ln (1 - z) + (\sum_{k=1..n} z^k / \text{of\_nat } k))$ 
  using  $\langle z \neq 1 \rangle$  by (simp add: exp_add weierstrass_factor_def)

```

```

  also have  $\text{norm } (\dots - 1) \leq (3 / 2) * \text{norm } (\ln (1 - z) + (\sum_{k=1..n} z^k / \text{of\_nat } k))$ 
    by (intro norm_exp_bounds norm_le')
  also have  $\dots \leq (3 / 2) * (2 * \text{norm } z^{\wedge} \text{Suc } n)$ 
    by (intro mult_left_mono norm_le) auto
  finally show ?thesis
    by simp
qed

```

9.2 Infinite products of elementary factors

We now show that the elementary factors can be used to construct an entire function with its zeros at a certain set of points (given by a sequence of non-zero numbers a_n with no accumulation point).

```

locale weierstrass_product =
  fixes  $a :: \text{nat} \Rightarrow \text{complex}$ 
  fixes  $p :: \text{nat} \Rightarrow \text{nat}$ 
  assumes  $a\_nonzero: \bigwedge n. a\ n \neq 0$ 
  assumes  $\text{filterlim\_}a: \text{filterlim } a \text{ at\_infinity at\_top}$ 
  assumes  $\text{summable\_}a\_p: \bigwedge r. r > 0 \implies \text{summable } (\lambda n. (r / \text{norm } (a\ n))^{\wedge} \text{Suc } (p\ n))$ 
begin

```

```

definition  $f :: \text{complex} \Rightarrow \text{complex}$  where
   $f\ z = (\prod n. \text{weierstrass\_factor } (p\ n) (z / a\ n))$ 

```

```

lemma  $\text{abs\_convergent}: \text{abs\_convergent\_prod } (\lambda n. \text{weierstrass\_factor } (p\ n) (z / a\ n))$ 

```

```

  unfolding  $\text{abs\_convergent\_prod\_conv\_summable}$ 
proof (rule summable_comparison_test_ev)
  have  $\text{eventually } (\lambda n. \text{norm } (a\ n) > 2 * \text{norm } z) \text{ at\_top}$ 
    using  $\text{filterlim\_}a$  by (metis filterlim_at_infinity_imp_norm_at_top filterlim_at_top_dense)
  thus  $\text{eventually } (\lambda n. \text{norm } (\text{norm } (\text{weierstrass\_factor } (p\ n) (z / a\ n) - 1)) \leq 3 * \text{norm } (z / a\ n)^{\wedge} \text{Suc } (p\ n)) \text{ at\_top}$ 
proof eventually_elim
  case (elim n)
  hence  $\text{norm } (z / a\ n) \leq 1 / 2$ 
    by (auto simp: norm_divide divide_simps)
  thus ?case using  $\text{weierstrass\_factor\_bound}$ [of  $z / a\ n\ p\ n$ ]
    by simp
qed
next
  show  $\text{summable } (\lambda n. 3 * \text{norm } (z / a\ n)^{\wedge} \text{Suc } (p\ n))$ 
    using  $\text{summable\_mult}$ [OF  $\text{summable\_}a\_p$ [of  $\text{norm } z$ ], of 3]
    by (cases  $z = 0$ ) (auto simp: norm_divide)
qed

```

```

lemma  $\text{convergent}: \text{convergent\_prod } (\lambda n. \text{weierstrass\_factor } (p\ n) (z / a\ n))$ 

```

using *abs_convergent*[of *z*] *abs_convergent_prod_imp_convergent_prod* **by** *blast*

lemma *has_prod*: $(\lambda n. \text{weierstrass_factor } (p \ n) \ (z / a \ n)) \text{ has_prod } f \ z$
using *convergent*[of *z*] **unfolding** *f_def* **by** *auto*

lemma *finite_occs_a*: *finite* $(a - \{z\})$

proof –

have *eventually* $(\lambda n. \text{norm } (a \ n) > \text{norm } z) \text{ at_top}$
using *filterlim_a* **by** $(\text{metis } \text{filterlim_at_infinity_imp_norm_at_top } \text{filterlim_at_top_dense})$
then obtain *N* **where** $N: \bigwedge n. n \geq N \implies \text{norm } (a \ n) > \text{norm } z$
by $(\text{auto } \text{simp: } \text{eventually_at_top_linorder})$
have $n < N$ **if** $n \in a - \{z\}$ **for** *n*
using $N[\text{of } n]$ **that** **by** $(\text{cases } n < N) \text{ auto}$
hence $a - \{z\} \subseteq \{.. < N\}$ *finite* $\{.. < N\}$
by *auto*
thus *?thesis*
using *finite_subset* **by** *blast*
qed

context

fixes *P*

defines $P \equiv (\lambda N \ z. \prod_{n < N. \text{weierstrass_factor } (p \ n) \ (z / a \ n)})$

begin

lemma *uniformly_convergent*:

assumes $R > 0$

shows *uniformly_convergent_on* $(\text{cball } 0 \ R) \ P$

unfolding *P_def*

proof $(\text{rule } \text{uniformly_convergent_on_prod})$

show *uniformly_convergent_on* $(\text{cball } 0 \ R) \ (\lambda N \ z. \sum_{n < N. \text{norm } (\text{weierstrass_factor } (p \ n) \ (z / a \ n) - 1))$

proof $(\text{rule } \text{Weierstrass_m_test_ev})$

have *eventually* $(\lambda n. \text{norm } (a \ n) \geq 2 * R) \text{ sequentially}$

using *filterlim_a* **by** $(\text{metis } \text{filterlim_at_infinity_imp_norm_at_top } \text{filterlim_at_top})$

thus $\forall_F n \text{ in sequentially. } \forall z \in \text{cball } 0 \ R. \text{norm } (\text{norm } (\text{weierstrass_factor } (p \ n) \ (z / a \ n) - 1)) \leq$

$3 * (R / \text{norm } (a \ n)) \wedge \text{Suc } (p \ n)$

proof *eventually_elim*

case $(\text{elim } n)$

show *?case*

proof *safe*

fix *z* :: *complex* **assume** *z*: $z \in \text{cball } 0 \ R$

have $2 * \text{norm } z \leq 2 * R$

using *z* **by** *auto*

also have $\dots \leq \text{norm } (a \ n)$

using *elim* **by** *simp*

finally have $\text{norm } (a \ n) \geq 2 * \text{norm } z$.

```

    hence norm (weierstrass_factor (p n) (z / a n) - 1) ≤ 3 * norm (z / a
n) ^ Suc (p n)
    by (intro weierstrass_factor_bound) (auto simp: norm_divide divide_simps)
    also have ... = 3 * (norm z / norm (a n)) ^ Suc (p n)
    by (simp add: norm_divide)
    also have ... ≤ 3 * (R / norm (a n)) ^ Suc (p n)
    by (intro mult_left_mono power_mono divide_right_mono) (use z in
auto)
    finally show norm (norm (weierstrass_factor (p n) (z / a n) - 1)) ≤
3 * (R / norm (a n)) ^ Suc (p n) by simp
  qed
next
show summable (λn. 3 * (R / norm (a n)) ^ Suc (p n))
by (intro summable_mult summable_a_p assms)
qed
qed (auto intro!: continuous_intros simp: a_nonzero)

lemma uniform_limit:
  assumes R > 0
  shows uniform_limit (cball 0 R) P f at_top
proof -
  obtain g where g: uniform_limit (cball 0 R) P g at_top
  using uniformly_convergent[OF assms] by (auto simp: uniformly_convergent_on_def)
  also have ?this ⟷ uniform_limit (cball 0 R) P f at_top
  proof (intro uniform_limit_cong)
    fix z :: complex assume z ∈ cball 0 R
    with g have (λn. P (Suc n) z) ⟶ g z
    by (metis tendsto_uniform_limitI filterlim_sequentially_Suc)
    moreover have (λn. P (Suc n) z) ⟶ f z
    using convergent_prod_LIMSEQ[OF convergent[of z]] unfolding P_def
lessThan_Suc_atMost
    by (simp add: f_def)
    ultimately show g z = f z
    using tendsto_unique by force
  qed auto
  finally show ?thesis .
qed

```

```

lemma holomorphic [holomorphic_intros]: f holomorphic_on A
proof (rule holomorphic_on_subset)
  show f holomorphic_on UNIV
proof (rule holomorphic_uniform_sequence)
  fix z :: complex
  have *: uniform_limit (cball 0 (norm z + 1)) P f sequentially
  by (rule uniform_limit) (auto intro: add_nonneg_pos)
  hence uniform_limit (cball z 1) P f sequentially
  by (rule uniform_limit_on_subset) (simp add: cball_subset_cball_iff)
  thus ∃ d>0. cball z d ⊆ UNIV ∧ uniform_limit (cball z d) P f sequentially

```



```

    by (intro exI[of _ 1]) auto
  qed (auto intro!: holomorphic_intros simp: P_def)
qed auto

lemma analytic [analytic_intros]: f analytic_on A
  using holomorphic[of UNIV] analytic_on_holomorphic by blast

end

lemma zero:  $f z = 0 \iff z \in \text{range } a$ 
  using has_prod_eq_0_iff[OF has_prod, of z] by (auto simp: a_nonzero)

lemma not_islimpt_range_a:  $\neg z \text{ islimpt } (\text{range } a)$ 
proof
  assume  $z \text{ islimpt } (\text{range } a)$ 
  then obtain  $r :: \text{nat} \Rightarrow \text{nat}$  where  $r$ :  $\text{strict\_mono } r \ (a \circ r) \longrightarrow z$ 
    using islimpt_range_imp_convergent_subsequence by metis
  moreover have  $\text{filterlim } (a \circ r) \text{ at\_infinity sequentially}$ 
    unfolding o_def by (rule filterlim_compose[OF filterlim_a filterlim_subseq[OF
 $r(1)$ ]])
  ultimately show False
    by (meson not_tendsto_and_filterlim_at_infinity trivial_limit_sequentially)
qed

lemma isolated_zero:
  assumes  $z \in \text{range } a$ 
  shows  $\text{isolated\_zero } f z$ 
proof -
  have eventually  $(\lambda z. f z \neq 0) \text{ (at } z)$ 
    using not_islimpt_range_a[of z] by (auto simp: islimpt_iff_eventually zero)
  moreover have  $f - z \rightarrow f z$ 
    by (intro isContD analytic_at_imp_isCont analytic)
  hence  $f - z \rightarrow 0$ 
    using assms zero[of z] by auto
  ultimately show ?thesis
    by (auto simp: isolated_zero_def)
qed

lemma zorder:  $\text{zorder } f z = \text{card } (a - \{z\})$ 
proof -
  obtain  $N$  where  $N$ :  $a - \{z\} \subseteq \{..N\}$ 
    using finite_occurs_a[of z] by (meson finite_nat_iff_bounded_le)
  define  $g$  where  $g = (\lambda z n. \text{weierstrass\_factor } (p n) (z / a n))$ 
  define  $h1$  where  $h1 = (\lambda w. (\prod n \in \{..N\} - a - \{z\}. g w n) * (\prod n. g w (n + \text{Suc } N)))$ 
  define  $h2$  where  $h2 = (\lambda w. (\prod n \in \{..N\} \cap a - \{z\}. g w n))$ 

  have  $\text{has\_prod } h1'$ :  $(\lambda n. g w (n + \text{Suc } N)) \text{ has\_prod } (\prod n. g w (n + \text{Suc } N))$ 

```

```

for w
  unfolding g_def
  by (intro convergent_prod_has_prod convergent_prod_ignore_initial_segment
    convergent)

  have f_eq: f w = h1 w * h2 w for w
  proof -
    have f w = ( $\prod n < \text{Suc } N. g \ w \ n$ ) * ( $\prod n. g \ w \ (n + \text{Suc } N)$ )
    proof (rule has_prod_unique2)
      show ( $\lambda n. g \ w \ n$ ) has_prod (( $\prod n < \text{Suc } N. g \ w \ n$ ) * ( $\prod n. g \ w \ (n + \text{Suc } N)$ ))
      unfolding g_def by (intro has_prod_ignore_initial_segment' convergent)
      show g w has_prod f w
      unfolding g_def by (rule has_prod)
    qed
    also have  $\{.. < \text{Suc } N\} = (\{..N\} - a - \{z\}) \cup (\{..N\} \cap a - \{z\})$ 
    by auto
    also have ( $\prod k \in \dots. g \ w \ k$ ) = ( $\prod k \in \{..N\} - a - \{z\}. g \ w \ k$ ) * ( $\prod k \in \{..N\} \cap$ 
     $a - \{z\}. g \ w \ k$ )
    by (intro prod.union_disjoint) auto
    finally show ?thesis
    by (simp add: h1_def h2_def mult_ac)
  qed

  have ana_h1: h1 analytic_on {z}
  proof -
    interpret h1: weierstrass_product  $\lambda n. a \ (n + \text{Suc } N) \ \lambda n. p \ (n + \text{Suc } N)$ 
    proof
      have filterlim ( $\lambda n. n + \text{Suc } N$ ) at_top at_top
      by (rule filterlim_add_const_nat_at_top)
      thus filterlim ( $\lambda n. a \ (n + \text{Suc } N)$ ) at_infinity at_top
      by (intro filterlim_compose[OF filterlim_a])
      show summable ( $\lambda n. (r / \text{cmod } (a \ (n + \text{Suc } N))) \wedge \text{Suc } (p \ (n + \text{Suc } N)))$  if
       $r > 0$  for r
      by (intro summable_ignore_initial_segment summable_a_p that)
    qed (auto simp: a_nonzero)

    show ?thesis using h1.analytic
    unfolding h1_def g_def h1.f_def by (intro analytic_intros) (auto simp:
    a_nonzero)
  qed

  have ana_h2: h2 analytic_on {z}
  unfolding h2_def g_def by (intro analytic_intros) (auto simp: a_nonzero)

  have zorder f z = zorder ( $\lambda w. h1 \ w * h2 \ w$ ) z
  by (simp add: f_eq [abs_def])
  also have ... = zorder h1 z + zorder h2 z
  proof (rule zorder_times_analytic)
    have eventually ( $\lambda w. f \ w \neq 0$ ) (at z)

```

```

    using not_islimpt_range_a[of z] by (auto simp: islimpt_conv_frequently_at
frequently_def zero)
    thus eventually ( $\lambda w. h1\ w * h2\ w \neq 0$ ) (at z)
    by (simp add: f_eq)
  qed fact+
  also have zorder h2 z =  $(\sum n \in \{..N\} \cap a - \{z\}. zorder (\lambda w. g\ w\ n)\ z)$ 
  unfolding h2_def
  by (intro zorder_prod_analytic)
    (auto simp: a_nonzero g_def eventually_at_filter intro!: analytic_intros)
  also have h1 z  $\neq 0$  using N_has_prod_eq_0_iff[OF has_prod_h1[of z]]
  by (auto simp: h1_def g_def)
  hence zorder h1 z = 0
  by (intro zorder_eq_0I ana_h1)
  also have  $(\sum n \in \{..N\} \cap a - \{z\}. zorder (\lambda w. g\ w\ n)\ z) = (\sum n \in \{..N\} \cap a - \{z\}. 1)$ 
  proof (intro sum.cong refl)
    fix n :: nat
    assume n:  $n \in \{..N\} \cap a - \{z\}$ 
    have zorder ( $\lambda w. weierstrass\_factor\ (p\ n)\ (1 / a\ n * w)$ ) z =
      zorder (weierstrass_factor (p n)) (1 / a n * z)
    using a_nonzero[of n] eventually_neq_at_within[of 1 z / a n UNIV]
    by (intro zorder_scale analytic_intros analytic_on_imp_meromorphic_on)
  auto
    hence zorder ( $\lambda w. g\ w\ n$ ) z = zorder (weierstrass_factor (p n)) 1
    using n a_nonzero[of n] by (auto simp: g_def)
  thus zorder ( $\lambda w. g\ w\ n$ ) z = 1
  by simp
  qed
  also have ... = card ( $\{..N\} \cap a - \{z\}$ )
  by simp
  also have  $\{..N\} \cap a - \{z\} = a - \{z\}$ 
  using N by blast
  finally show ?thesis
  by simp
qed
end

```

The following locale is the most common case of $p(n) = n$.

```

locale weierstrass_product' =
  fixes a :: nat  $\Rightarrow$  complex
  assumes a_nonzero:  $\bigwedge n. a\ n \neq 0$ 
  assumes filterlim_a: filterlim a at_infinity at_top
  assumes finite_occs_a':  $\bigwedge z. z \in \text{range } a \implies \text{finite } (a - \{z\})$ 
begin

lemma finite_occs_a: finite (a - {z})
proof (cases z  $\in$  range a)
  case False

```

```

    hence  $a - \{z\} = \{\}$ 
    by auto
    thus ?thesis by simp
qed (use finite_occs_a'[of z] in auto)

sublocale weierstrass_product  $a \lambda n. n$ 
proof
  fix  $r :: \text{real}$  assume  $r: r > 0$ 
  show summable  $(\lambda n. (r / \text{norm } (a \ n)) ^ \text{Suc } n)$ 
  proof (rule summable_comparison_test_ev)
    have eventually  $(\lambda n. \text{norm } (a \ n) > 2 * r)$  at_top
    using filterlim_a by (metis filterlim_at_infinity_imp_norm_at_top filterlim_at_top_dense)
    thus eventually  $(\lambda n. \text{norm } ((r / \text{norm } (a \ n)) ^ \text{Suc } n) \leq (1 / 2) ^ \text{Suc } n)$  at_top
  proof eventually_elim
    case (elim n)
    have  $\text{norm } ((r / \text{norm } (a \ n)) ^ \text{Suc } n) = (r / \text{norm } (a \ n)) ^ \text{Suc } n$ 
    using  $\langle r > 0 \rangle$  by (simp add: abs_mult)
    also have  $\dots \leq (1 / 2) ^ \text{Suc } n$ 
    using  $\langle r > 0 \rangle$  elim by (intro power_mono) (auto simp: divide_simps)
    finally show ?case .
  qed
  qed
next
  show summable  $(\lambda n. (1 / 2) ^ \text{Suc } n :: \text{real})$ 
  unfolding summable_Suc_iff by (intro summable_geometric) auto
  qed
qed (use a_nonzero filterlim_a finite_occs_a in auto)

end

```

9.3 Writing a quotient as an exponential

If two holomorphic functions f and g on a simply connected domain have the same zeros with the same multiplicities, they can be written as $g(x) = e^{h(x)}f(x)$ for some holomorphic function $h(x)$.

lemma *holomorphic_zorder_factorization:*

```

  assumes  $g \text{ holomorphic\_on } A$  open  $A$  connected  $A$ 
  assumes  $f \text{ holomorphic\_on } A \wedge z. z \in A \implies f \ z = 0 \longleftrightarrow g \ z = 0$ 
     $\wedge z. z \in A \implies \text{zorder } f \ z = \text{zorder } g \ z$ 
  obtains  $h$  where  $h \text{ holomorphic\_on } A \wedge z. z \in A \implies h \ z \neq 0 \wedge z. z \in A \implies$ 
 $g \ z = h \ z * f \ z$ 
proof (cases  $\exists z \in A. g \ z \neq 0$ )
  case False
  show ?thesis
    by (rule that[of  $\lambda \_ . 1$ ]) (use False assms in auto)
next
  case True
  define  $F$  where  $F = \text{fps\_expansion } f$ 

```

```

define  $G$  where  $G = \text{fps\_expansion } g$ 
define  $c$  where  $c = (\lambda z. \text{fps\_nth } (G \ z) \ (\text{subdegree } (G \ z)) \ / \ \text{fps\_nth } (F \ z) \ (\text{subdegree } (F \ z)))$ 
define  $h$  where  $h = (\lambda z. \text{if } f \ z = 0 \text{ then } c \ z \text{ else } g \ z \ / \ f \ z)$ 

have  $ev\_nonzero$ :  $\text{eventually } (\lambda w. g \ w \neq 0 \wedge w \in A) \ (\text{at } z) \text{ if } z \in A \text{ for } z$ 
proof –
  from  $True$  obtain  $z0$  where  $z0$ :  $z0 \in A \ g \ z0 \neq 0$ 
  by  $auto$ 
  show  $?thesis$ 
  by  $(\text{rule } non\_zero\_neighbour\_alt[\text{where } ?\beta = z0])$ 
   $(\text{use } \text{assms that } z0 \text{ in } \langle \text{auto intro; simply\_connected\_imp\_connected} \rangle)$ 
qed

have  $g\_ana$ :  $g \text{ analytic\_on } \{z\} \text{ if } z \in A \text{ for } z$ 
  using  $\text{assms } \langle \text{open } A \rangle \text{ analytic\_at that by blast}$ 
have  $f\_ana$ :  $f \text{ analytic\_on } \{z\} \text{ if } z \in A \text{ for } z$ 
  using  $\text{assms } \langle \text{open } A \rangle \text{ analytic\_at that by blast}$ 

have  $F$ :  $(\lambda w. f \ (z + w)) \text{ has\_fps\_expansion } F \ z \text{ if } z \in A \text{ for } z$ 
  unfolding  $F\_def$  by  $(\text{rule } analytic\_at\_imp\_has\_fps\_expansion[OF \ f\_ana[OF \ that]])$ 
have  $G$ :  $(\lambda w. g \ (z + w)) \text{ has\_fps\_expansion } G \ z \text{ if } z \in A \text{ for } z$ 
  unfolding  $G\_def$  by  $(\text{rule } analytic\_at\_imp\_has\_fps\_expansion[OF \ g\_ana[OF \ that]])$ 

have  $[simp]$ :  $G \ z \neq 0 \text{ if } z \in A \text{ for } z$ 
proof
  assume  $G \ z = 0$ 
  hence  $\text{eventually } (\lambda w. g \ w = 0) \ (\text{at } z) \text{ using } G[OF \ that]$ 
  by  $(\text{auto simp: has\_fps\_expansion\_0\_iff at\_to\_0' eventually\_filtermap add\_ac eventually\_at\_filter nhds\_to\_0' elim: eventually\_mono})$ 
  hence  $\text{eventually } (\lambda_. False) \ (\text{at } z)$ 
  using  $ev\_nonzero[OF \ that] \text{ unfolding eventually\_at\_filter by eventually\_elim auto}$ 
  thus  $False$ 
  by  $simp$ 
qed
have  $[simp]$ :  $F \ z \neq 0 \text{ if } z \in A \text{ for } z$ 
proof
  assume  $F \ z = 0$ 
  hence  $\text{eventually } (\lambda w. f \ w = 0) \ (\text{at } z) \text{ using } F[of \ z] \text{ that}$ 
  by  $(\text{auto simp: has\_fps\_expansion\_0\_iff at\_to\_0' eventually\_filtermap add\_ac eventually\_at\_filter nhds\_to\_0' elim: eventually\_mono})$ 
  hence  $\text{eventually } (\lambda_. False) \ (\text{at } z)$ 
  using  $ev\_nonzero[OF \ that] \text{ unfolding eventually\_at\_filter by eventually\_elim (use assms in auto)}$ 

```

```

    thus False
    by simp
qed
have [simp]:  $c \cdot z \neq 0$  if  $z \in A$  for  $z$ 
  using that by (simp add: c_def)

have h:  $h$  analytic_on  $\{z\} \wedge h \cdot z \neq 0$  if  $z \in A$  for  $z$ 
proof -
  show ?thesis
  proof (cases  $f \cdot z = 0$ )
    case False
    from False that have  $(\lambda z. g \cdot z / f \cdot z)$  analytic_on  $\{z\}$ 
      by (intro analytic_intros g_ana f_ana) auto
    also have  $?this \longleftrightarrow h$  analytic_on  $\{z\}$ 
    proof (rule analytic_at_cong)
      have eventually  $(\lambda w. f \cdot w \neq 0)$  (nhds  $z$ )
        using ev_nonzero[OF  $\langle z \in A \rangle$ ] unfolding eventually_at_filter
        by eventually_elim (use False  $\langle z \in A \rangle$  assms in auto)
      thus eventually  $(\lambda z. g \cdot z / f \cdot z = h \cdot z)$  (nhds  $z$ )
        by eventually_elim (auto simp: h_def)
    qed auto
    finally have  $h$  analytic_on  $\{z\}$  .
    moreover have  $h \cdot z \neq 0$ 
      using that assms by (simp add: h_def)
    ultimately show ?thesis by blast
  next
    case True
    with that have  $z: z \in A \cdot f \cdot z = 0$ 
      by auto
    have zorder  $f \cdot z = \text{int}(\text{subdegree}(F \cdot z))$ 
      using F by (rule has_fps_expansion_zorder) (use  $z$  in auto)
    also have zorder  $f \cdot z = \text{zorder } g \cdot z$ 
      using  $z$  assms by auto
    also have zorder  $g \cdot z = \text{subdegree}(G \cdot z)$ 
      using G by (rule has_fps_expansion_zorder) (use  $z$  in auto)
    finally have subdegree_eq:  $\text{subdegree}(F \cdot z) = \text{subdegree}(G \cdot z)$ 
      by simp

    have  $(\lambda w. \text{if } w = 0 \text{ then } c \cdot z \text{ else } g \cdot (z + w) / f \cdot (z + w))$  has_fps_expansion
       $G \cdot z / F \cdot z$  (is ?P)
      using subdegree_eq  $z$  by (intro has_fps_expansion_divide F G) (auto simp:
c_def)
    also have  $?this \longleftrightarrow (\lambda w. h \cdot (z + w))$  has_fps_expansion  $G \cdot z / F \cdot z$ 
    proof (intro has_fps_expansion_cong)
      have eventually  $(\lambda w. w \neq z \longrightarrow f \cdot w \neq 0)$  (nhds  $z$ )
        using ev_nonzero[OF  $\langle z \in A \rangle$ ] unfolding eventually_at_filter
        by eventually_elim (use  $\langle z \in A \rangle$  assms in auto)
      hence eventually  $(\lambda w. w \neq 0 \longrightarrow f \cdot (z + w) \neq 0)$  (nhds  $0$ )
        by (simp add: nhds_to_0' eventually_filtermap)
    qed
  end

```

```

      thus eventually ( $\lambda w. (if\ w = 0\ then\ c\ z\ else\ g\ (z + w) / f\ (z + w)) = h\ (z + w)$ ) (nhds 0)
    unfolding h_def by eventually_elim (use z in ⟨auto simp: c_def h_def⟩)
  qed auto
  finally have h_analytic_on {z}
    using has_fps_expansion_imp_analytic by blast
  moreover have h_z ≠ 0
    using that z by (auto simp: h_def c_def)
  ultimately show ?thesis by blast
qed
qed

from h have h_ana: h analytic_on A and h_nz:  $\forall z \in A. h\ z \neq 0$ 
  using analytic_on_analytic_at by blast+
moreover have g_z = h_z * f_z if z ∈ A for z
  using assms that by (auto simp: h_def)
ultimately show ?thesis
  using ⟨open A⟩ by (intro that[of h]) (auto simp: analytic_on_open)
qed

```

9.4 Constructing the sequence of zeros

The form of the Weierstraß Factorisation Theorem that we derived above requires an explicit sequence of the zeros that tends to infinity. We will now show that under mild conditions, such a sequence always exists.

More precisely: if A is an infinite closed set that is sparse in the sense that its intersection with any compact set is finite, then there exists an injective sequence f enumerating the values of A in ascending order by absolute value, and f tends to infinity for $n \rightarrow \infty$.

lemma *sequence_of_sparse_set_exists:*

```

fixes A :: complex set
assumes infinite A closed A  $\wedge r. r \geq 0 \implies finite (A \cap cball\ 0\ r)$ 
obtains f :: nat  $\Rightarrow$  complex
  where mono (norm  $\circ$  f) inj f range f = A filterlim f at_infinity at_top
proof -
  have  $\exists f :: nat \Rightarrow complex. \forall n.
    f\ n \in A \wedge
    f\ n \notin f\ ' \{..
  proof (rule dependent_wf_choice[OF wf], goal_cases)
    case (1 f g n r)
    thus ?case by auto
  next
    case (2 n f)
    have f:  $f\ k \in A \ \{z \in A. norm\ z < norm\ (f\ k)\} \subseteq f\ ' \{..
      if k < n for k$$ 
```

```

    using 2[of k] that by simp_all

  have infinite (A - f ' {.. $n$ })
    using assms(1) by (intro Diff_infinite_finite) auto
  then obtain z0 where z0:  $z0 \in A - f ' \{.. $n$ \}$ 
    by (meson finite.intros(1) finite_subset subsetI)
  have finite (A  $\cap$  cball 0 (norm z0))
    by (intro assms(3)) auto
  hence finite (A  $\cap$  cball 0 (norm z0) - f ' {.. $n$ })
    using finite_subset by blast
  moreover from z0 have A  $\cap$  cball 0 (norm z0) - f ' {.. $n$ }  $\neq \{\}$ 
    by auto
  ultimately obtain z where is_arg_min norm ( $\lambda z. z \in A \cap \text{cball } 0 \text{ (norm } z0) - f ' \{.. $n$ \}$ ) z
    using ex_is_arg_min_if_finite by blast
  hence z:  $z \in A \text{ norm } z \leq \text{norm } z0 \text{ } z \notin f ' \{.. $n$ \}$ 
     $\wedge w. w \in A \implies \text{norm } w \leq \text{norm } z0 \implies w \notin f ' \{.. $n$ \} \implies \text{norm } w \geq$ 
    norm z
    by (auto simp: is_arg_min_def)

  show ?case
  proof (rule exI[of _ z], safe)
    fix w assume w:  $w \in A \text{ norm } w < \text{norm } z$ 
    with z(4)[of w] z w show  $w \in f ' \{.. $n$ \}$ 
      by linarith
  next
    fix k assume k:  $k < n$ 
    show norm (f k)  $\leq$  norm z
      using f(2)[of k] z(1,3) k by auto
  qed (use z in auto)
qed

then obtain f :: nat  $\Rightarrow$  complex where f:
   $\bigwedge n. f n \in A$ 
   $\bigwedge n. f n \notin f ' \{.. $n$ \}$ 
   $\bigwedge n. \{z \in A. \text{norm } z < \text{norm } (f n)\} \subseteq f ' \{.. $n$ \}$ 
   $\bigwedge k n. k < n \implies \text{norm } (f k) \leq \text{norm } (f n)$ 
  by meson
from f(2) have f_neq:  $f n \neq f k$  if  $k < n$  for k n
  using that by blast

have inj: inj f
proof (rule injI)
  fix m n :: nat
  assume f m = f n
  thus m = n
    using f_neq[of m n] f_neq[of n m] by (cases m n rule: linorder_cases) auto
qed

have range: range f = A

```



```

proof safe
  fix  $z$  assume  $z: z \in A$ 
  show  $z \in \text{range } f$ 
  proof (rule ccontr)
    assume  $z \notin \text{range } f$ 
    hence  $\text{norm } (f\ n) \leq \text{norm } z$  for  $n$ 
      using  $f(3)[\text{of } n]\ z$  by auto
    hence  $\text{range } f \subseteq A \cap \text{cball } 0\ (\text{norm } z)$ 
      using  $f(1)$  by auto
    moreover have  $\text{finite } (A \cap \text{cball } 0\ (\text{norm } z))$ 
      by (intro assms) auto
    ultimately have  $\text{finite } (\text{range } f)$ 
      using finite_subset by blast
    moreover have  $\text{infinite } (\text{range } f)$ 
      using inj by (subst finite_image_iff) auto
    ultimately show False by contradiction
  qed
qed (use f(1) in auto)

have mono:  $\text{mono } (\text{norm} \circ f)$ 
proof (rule monoI, unfold o_def)
  fix  $m\ n :: \text{nat}$ 
  assume  $m \leq n$ 
  thus  $\text{norm } (f\ m) \leq \text{norm } (f\ n)$ 
    using  $f(4)[\text{of } m\ n]$  by (cases m < n) auto
qed

have  $\neg \text{bounded } A$ 
proof
  assume  $\text{bounded } A$ 
  hence  $\text{bdd\_above } (\text{norm } ` A)$ 
    by (meson bdd_above_norm)
  hence  $\text{norm } z \leq \text{Sup } (\text{norm } ` A)$  if  $z \in A$  for  $z$ 
    using that by (meson cSUP_upper)
  hence  $A \subseteq \text{cball } 0\ (\text{Sup } (\text{norm } ` A))$ 
    by auto
  also have  $\dots \subseteq \text{cball } 0\ (\text{max } 1\ (\text{Sup } (\text{norm } ` A)))$ 
    by auto
  finally have  $A \subseteq A \cap \text{cball } 0\ (\text{max } 1\ (\text{Sup } (\text{norm } ` A)))$ 
    by blast
  moreover have  $\text{finite } (A \cap \text{cball } 0\ (\text{max } 1\ (\text{Sup } (\text{norm } ` A))))$ 
    by (intro assms) auto
  ultimately have  $\text{finite } A$ 
    using finite_subset by blast
  hence  $\text{finite } (\text{range } f)$ 
    by (simp add: range)
  thus False
    using inj by (subst (asm) finite_image_iff) auto
qed

```

```

have lim: filterlim f at_infinity at_top
  unfolding filterlim_at_infinity_conv_norm_at_top filterlim_at_top
proof
  fix C :: real
  from ⟨¬bounded A⟩ obtain z where z ∈ A norm z > C
  unfolding bounded_iff by (auto simp: not_le)
  obtain n where [simp]: z = f n
  using range ⟨z ∈ A⟩ by auto
  show eventually (λn. norm (f n) ≥ C) at_top
  using eventually_ge_at_top[of n]
proof eventually_elim
  case (elim k)
  have C ≤ norm (f n)
    using ⟨norm z > C⟩ by simp
  also have ... ≤ norm (f k)
    using monoD[OF ⟨mono (norm ∘ f)⟩, of n k] elim by auto
  finally show ?case .
qed
qed

show ?thesis
  by (intro that[of f] inj range mono lim)
qed

lemma strict_mono_sequence_partition:
  assumes strict_mono (f :: nat ⇒ 'a :: {linorder, no_top})
  assumes x ≥ f 0
  assumes filterlim f at_top at_top
  shows ∃k. x ∈ {f k..

```

ultimately show *?thesis* by auto
qed

lemma *strict_mono_sequence_partition'*:
 assumes *strict_mono* ($f :: \text{nat} \Rightarrow 'a :: \{\text{linorder}, \text{no_top}\}$)
 assumes $x \geq f\ 0$
 assumes *filterlim* f at_top at_top
 shows $\exists! k. x \in \{f\ k..<f\ (Suc\ k)\}$
proof (*rule ex_ex1I*)
 show $\exists k. x \in \{f\ k..<f\ (Suc\ k)\}$
 using *strict_mono_sequence_partition*[*OF* *assms*] .
 fix $k1\ k2$ **assume** $x \in \{f\ k1..<f\ (Suc\ k1)\} \ x \in \{f\ k2..<f\ (Suc\ k2)\}$
 thus $k1 = k2$
proof (*induction* $k1\ k2$ *rule: linorder_wlog*)
 case (*le* $k1\ k2$)
 hence $f\ k2 < f\ (Suc\ k1)$
 by *auto*
 hence $k2 < Suc\ k1$
 using *assms*(1) *strict_mono_less* **by** *blast*
 with *le* **show** $k1 = k2$
 by *linarith*
qed *auto*
qed

lemma *sequence_of_sparse_set_exists'*:
 fixes $A :: \text{complex set}$ **and** $c :: \text{complex} \Rightarrow \text{nat}$
 assumes *infinite* A *closed* $A \ \bigwedge r. r \geq 0 \implies \text{finite } (A \cap \text{cball } 0\ r)$
 assumes *c_pos*: $\bigwedge x. x \in A \implies c\ x > 0$
 obtains $f :: \text{nat} \Rightarrow \text{complex}$ **where**
 mono ($\text{norm} \circ f$) *range* $f = A$ *filterlim* f at_infinity at_top
 $\bigwedge z. z \in A \implies \text{finite } (f^{-1}\{z\}) \wedge \text{card } (f^{-1}\{z\}) = c\ z$
proof –
 obtain $f :: \text{nat} \Rightarrow \text{complex}$ **where** f :
 mono ($\text{norm} \circ f$) *inj* f *range* $f = A$ *filterlim* f at_infinity at_top
 using *assms* *sequence_of_sparse_set_exists* **by** *blast*
 have *f_eq_iff* [*simp*]: $f\ m = f\ n \iff m = n$ **for** $m\ n$
 using $\langle \text{inj } f \rangle$ **by** (*auto simp: inj_def*)

define $h :: \text{nat} \Rightarrow \text{nat}$ **where** $h = (\lambda n. \sum_{k < n. c\ (f\ k)})$

 have [*simp*]: $h\ 0 = 0$
 by (*simp add: h_def*)
 have *h_ge*: $h\ n \geq n$ **for** n
proof –
 have $h\ n \geq (\sum_{k < n. 1})$
 unfolding *h_def* **by** (*intro sum_mono*) (*use c_pos f in* $\langle \text{auto simp: Suc_le_eq} \rangle$)
 thus *?thesis* **by** *simp*
qed

```

have strict_mono h
  unfolding strict_mono_Suc_iff using f by (auto simp: h_def c_pos)
moreover from this have filterlim h at_top at_top
  using filterlim_subseq by blast
ultimately have Ex1:  $\exists! k. n \in \{h k..<h (Suc k)\}$  for n
  by (intro strict_mono_sequence_partition') auto

define g :: nat  $\Rightarrow$  nat where g = ( $\lambda n. THE k. n \in \{h k..<h (Suc k)\}$ )
have g:  $n \in \{h (g n)..<h (Suc (g n))\}$  for n
  using theI'[OF Ex1[of n]] by (simp add: g_def)
have g_eqI:  $g n = k$  if  $n \in \{h k..<h (Suc k)\}$  for n k
  using the1_equality[OF Ex1 that] by (simp add: g_def)
have g_h:  $g (h n) = n$  for n
  by (rule g_eqI) (auto intro: strict_monoD[OF  $\langle$ strict_mono h $\rangle$ ])

have mono g
  unfolding incseq_Suc_iff
proof safe
  fix n :: nat
  have h (g n) + 1  $\leq$  Suc n
    using g[of n] by auto
  also have Suc n < h (Suc (g (Suc n)))
    using g[of Suc n] by auto
  finally show g n  $\leq$  g (Suc n)
    by (metis  $\langle$ strict_mono h $\rangle$  add_lessD1 less_Suc_eq_le strict_mono_less)
qed

have filterlim g at_top at_top
  unfolding filterlim_at_top
proof
  fix n :: nat
  show eventually ( $\lambda m. g m \geq n$ ) at_top
    using eventually_ge_at_top[of h n]
  proof eventually_elim
    case (elim m)
    have n  $\leq$  g (h n)
      by (simp add: g_h)
    also have g (h n)  $\leq$  g m
      by (intro monoD[OF  $\langle$ mono g $\rangle$ ] elim)
    finally show ?case .
  qed
qed

have vimage:  $(f \circ g) - \{f n\} = \{h n..<h (Suc n)\}$  for n
  using g by (auto intro!: g_eqI)

show ?thesis
proof (rule that[of f  $\circ$  g])

```

```

  have incseq ( $\lambda n. (norm \circ f) (g\ n)$ )
    by (intro monoI monoD[OF f(1)] monoD[OF  $\langle incseq\ g \rangle$ ])
  thus incseq ( $norm \circ (f \circ g)$ )
    by (simp add: o_def)
next
  have range ( $f \circ g$ )  $\subseteq A$ 
    using f(3) by auto
  moreover have  $A \subseteq range\ (f \circ g)$ 
  proof
    fix z assume  $z \in A$ 
    then obtain n where [simp]:  $z = f\ n$ 
      using f(3) by auto
    have  $f\ (g\ (h\ n)) \in range\ (f \circ g)$ 
      unfolding o_def by blast
    thus  $z \in range\ (f \circ g)$ 
      by (simp add: g_h)
  qed
  ultimately show  $range\ (f \circ g) = A$  by blast
next
  fix z assume  $z \in A$ 
  then obtain n where [simp]:  $z = f\ n$ 
    using f(3) by auto
  have finite { $h\ n..<h\ (Suc\ n)$ }
    by auto
  moreover have  $card\ \{h\ n..<h\ (Suc\ n)\} = c\ z$ 
    by (simp add: h_def)
  ultimately show finite (( $f \circ g$ ) - '{z})  $\wedge card\ ((f \circ g) - '{z}) = c\ z$ 
    using vimage[of n] by simp
next
  show filterlim ( $f \circ g$ ) at_infinity at_top
    unfolding o_def by (rule filterlim_compose[OF f(4)  $\langle filterlim\ g\ at\_top \rangle$ ])
  qed
qed

```

9.5 The factorisation theorem for holomorphic functions

If g is a holomorphic function on an open connected domain whose zeros do not have an accumulation point on the frontier of A , then we can write g as a product of a function h holomorphic on A and an entire function f such that h is non-zero everywhere in A and the zeros of f are precisely the zeros of A with the same multiplicity.

In other words, we can get rid of all the zeros of g by dividing it with a suitable entire function f .

theorem *weierstrass_factorization:*

assumes g holomorphic_on A open A connected A

assumes $\bigwedge z. z \in frontier\ A \implies \neg z\ islimpt\ \{w \in A. g\ w = 0\}$

obtains $h\ f$ **where**

```

  h holomorphic_on A f holomorphic_on UNIV
   $\forall z. f z = 0 \longleftrightarrow (\forall z \in A. g z = 0) \vee (z \in A \wedge g z = 0)$ 
   $\forall z \in A. \text{zorder } f z = \text{zorder } g z$ 
   $\forall z \in A. h z \neq 0$ 
   $\forall z \in A. g z = h z * f z$ 
proof (cases  $\forall z \in A. g z = 0$ )
case True
show ?thesis
proof (rule that[of  $\lambda_. 1 \ \lambda_. 0$ ]; (intro ballI allI impI)?)
  fix z assume z:  $z \in A$ 
  have ev: eventually  $(\lambda w. w \in A)$  (at z)
    using z assms by (intro eventually_at_in_open') auto
  show  $\text{zorder } (\lambda :: \text{complex}. 0 :: \text{complex}) z = \text{zorder } g z$ 
    by (intro zorder_cong eventually_mono[OF ev] refl) (use True in auto)
qed (use assms True in auto)
next
case not_identically_zero: False
define Z where  $Z = \{z \in A. g z = 0\}$ 
have freq_nz: frequently  $(\lambda z. g z \neq 0)$  (at z) if  $z \in A$  for z
proof -
  have  $\forall_F w$  in at z.  $g w \neq 0 \wedge w \in A$ 
    using non_zero_neighbour_alt[OF assms(1,2,3) that(1)] not_identically_zero
by auto
  hence  $\forall_F w$  in at z.  $g w \neq 0$ 
    by eventually_elim auto
  thus ?thesis
    using eventually_frequently by force
qed

have zorder_pos_iff:  $\text{zorder } g z > 0 \longleftrightarrow g z = 0$  if  $z \in A$  for z
  by (subst zorder_pos_iff[OF assms(1,2) that]) (use freq_nz[of z] that in auto)

show ?thesis
proof (cases finite Z)
case True
  define f where  $f = (\lambda z. \prod_{w \in Z. (z - w)^{\text{powi } (\text{zorder } g w)}}$ 

  have eq_zero_iff:  $f z = 0 \longleftrightarrow z \in A \wedge g z = 0$  for z
    using True local.zorder_pos_iff
    unfolding f_def Z_def by fastforce
  have zorder_eq:  $\text{zorder } f z = \text{zorder } g z$  if  $z \in A$  for z
proof (cases  $g z = 0$ )
case False
  have g analytic_on {z}
    using that assms analytic_at by blast
  hence [simp]:  $\text{zorder } g z = 0$ 
    using False by (intro zorder_eq_0I) auto
  moreover have f analytic_on {z}
    unfolding f_def by (auto intro!: analytic_intros)

```

```

    hence zorder f z = 0
      using False by (intro zorder_eq_0I) (auto simp: eq_zero_iff)
    ultimately show ?thesis
      by simp
  next
  case zero: True
  have g analytic_on {z}
    using that assms(1,2) analytic_at by blast
  hence zorder g z ≥ 0
    using that by (intro zorder_ge_0 freq_nz) auto
  define f' where f' = (λz'. (∏ w∈Z- {z}. (z' - w) powi (zorder g w)))
  have zorder (λz'. (z' - z) powi (zorder g z) * f' z') z = zorder g z
  proof (rule zorder_eqI)
    show open (UNIV :: complex set) f' holomorphic_on UNIV z ∈ UNIV
      using local.zorder_pos_iff
      by (fastforce intro!: holomorphic_intros simp: f'_def Z_def)+
    show f' z ≠ 0
      using True unfolding f'_def by (subst prod_zero_iff) auto
  qed (use ⟨zorder g z ≥ 0⟩ in ⟨auto simp: powr_of_int⟩)
  also have (λz'. (z' - z) powi (zorder g z) * f' z') = f
  proof
    fix z' :: complex
    have Z = insert z (Z - {z})
      using that zero by (auto simp: Z_def)
    also have (∏ w∈... (z' - w) powi (zorder g w)) = (z' - z) powi (zorder
g z) * f' z'
      using True by (subst prod.insert) (auto simp: f'_def)
    finally show (z' - z) powi (zorder g z) * f' z' = f z'
      by (simp add: f'_def)
  qed
  finally show ?thesis .
qed

obtain h :: complex ⇒ complex where h:
  h holomorphic_on A ∧ z. z ∈ A ⇒ h z ≠ 0 ∧ z. z ∈ A ⇒ g z = h z * f z
proof (rule holomorphic_zorder_factorization[OF assms(1-3)])
  show f holomorphic_on A
    using local.zorder_pos_iff
    unfolding f_def Z_def by (fastforce intro: holomorphic_intros)
  show f z = 0 ⟷ g z = 0 if z ∈ A for z
    using that by (subst eq_zero_iff) auto
  show zorder f z = zorder g z if z ∈ A for z
    by (rule zorder_eq) fact
  qed metis

show ?thesis
proof (rule that[of h f]; (intro ballI)?)
  show h holomorphic_on A
    by fact

```

```

    show f holomorphic_on UNIV
    using local.zorder_pos_iff
    unfolding f_def Z_def by (fastforce intro: holomorphic_intros)
  qed (use True not_identically_zero in ⟨auto simp: eq_zero_iff zorder_eq h(2)
h(3)[symmetric]⟩)

next
case False
note infinite_zeroes = not_identically_zero False
define c where c = (λz. nat (zorder g z))

have ev_nz: eventually (λw. g w ≠ 0) (at z) if z ∈ A for z
proof -
  from infinite_zeroes(1) obtain z0 where z0: z0 ∈ A g z0 ≠ 0
  by auto
  have eventually (λw. g w ≠ 0 ∧ w ∈ A) (at z)
  by (rule non_zero_neighbour_alt[where ?β = z0]) (use assms z0 that in
auto)
  thus ?thesis
  by eventually_elim auto
qed

have no_limpt_Z: ¬z islimpt Z for z
proof
  assume z islimpt Z
  show False
  proof (cases z ∈ A)
    case False
    have z islimpt A
    by (rule islimpt_subset[OF ⟨z islimpt Z⟩]) (auto simp: Z_def)
    hence z ∈ closure A
    by (simp add: closure_def)
    with ⟨z ∉ A⟩ have z ∈ frontier A
    by (simp add: closure_Un_frontier)
    with assms and ⟨z islimpt Z⟩ show False
    by (auto simp: Z_def)
  next
  case True
  from True have eventually (λw. g w ≠ 0) (at z)
  using ev_nz by blast
  hence ¬z islimpt Z
  by (auto simp: islimpt_iff_eventually Z_def elim!: eventually_mono)
  with ⟨z islimpt Z⟩ show False by blast
  qed
qed
have closed Z
  using no_limpt_Z unfolding closed_limpt by blast

obtain a where a:

```



```

    incseq (norm ∘ a) range a = Z - {0}
    ∧ z. z ∈ Z - {0} ⇒ finite (a - ' {z}) ∧ card (a - ' {z}) = c z
    filterlim a at_infinity at_top
  proof (rule sequence_of_sparse_set_exists')
    show infinite (Z - {0})
      using infinite_zeroes(2) by auto
  next
    show closed (Z - {0})
      unfolding closed_limpt using no_limpt_Z islimpt_subset by blast
  next
    show finite ((Z - {0}) ∩ cball 0 R) if R ≥ 0 for R
    proof (rule ccontr)
      assume *: infinite ((Z - {0}) ∩ cball 0 R)
      have ∃ z ∈ cball 0 R. z islimpt ((Z - {0}) ∩ cball 0 R)
        by (rule Heine_Borel_imp_Bolzano_Weierstrass) (use * in auto)
      then obtain z where z islimpt ((Z - {0}) ∩ cball 0 R)
        by blast
      hence z islimpt Z
        using islimpt_subset by blast
      thus False
        using no_limpt_Z by blast
    qed
  next
    show c z > 0 if z ∈ Z - {0} for z
      using zorder_pos_iff[of z] that by (auto simp: c_def Z_def)
    qed metis

  interpret f: weierstrass_product' a
  proof
    show a n ≠ 0 for n
      using a(2) by auto
    show finite (a - ' {z}) if z ∈ range a for z
      using a(3)[of z] a(2) that by simp
    qed fact+

  define m where m = (if 0 ∈ A then nat (zorder g 0) else 0)

  have zorder_eq: zorder (λz. z ^ m * f.f z) z = zorder g z if z ∈ A for z
  proof (cases g z = 0)
    case False
      have g analytic_on {z}
        using ⟨z ∈ A⟩ analytic_at assms by blast
      hence zorder g z = 0
        by (intro zorder_eq_0I False)
      have z ∉ range a
        using False Z_def a(2) by blast
      hence zorder (λz. z ^ m * f.f z) z = 0
        using False ⟨zorder g z = 0⟩
        by (intro zorder_eq_0I analytic_intros) (auto simp: f.zero m_def)
  
```

```

    with ⟨zorder g z = 0⟩ show ?thesis
      by simp
  next
    case True
    define F where F = fps_expansion f.f z
    have frequently (λw. g w ≠ 0) (at z)
      using ev_nz[OF that] eventually_frequently by force
    hence zorder g z ≥ 0
      by (intro zorder_ge_0) (use assms that in ⟨auto simp: analytic_at⟩)

    have ev: eventually (λz. z ∈ A) (nhds z)
      using that assms by (intro eventually_nhds_in_open) auto
    have exp1: (λw. f.f (z + w)) has_fps_expansion F
      unfolding F_def by (intro analytic_at_imp_has_fps_expansion[OF
f.analytic])
    have exp2: (λw. (z + w) ^ m * f.f (z + w)) has_fps_expansion (fps_const
z + fps_X) ^ m * F
      by (intro fps_expansion_intros exp1)
    have [simp]: F ≠ 0
    proof
      assume F = 0
      hence eventually (λz. f.f z = 0) (nhds z)
        using exp1 by (auto simp: has_fps_expansion_def nhds_to_0' eventu-
ally_filtermap)
      hence eventually (λz. g z = 0) (at z)
        by (auto simp: f.zero a Z_def eventually_at_filter elim!: eventually_mono)
      hence eventually (λz::complex. False) (at z)
        using ev_nz[OF ⟨z ∈ A⟩] by eventually_elim auto
      thus False by simp
    qed

    have zorder (λw. w ^ m * f.f w) z = int (subdegree ((fps_const z + fps_X)
^ m * F))
      using has_fps_expansion_zorder[OF exp2] by simp
    also have ... = int (subdegree F) + (if z = 0 then m else 0)
      by auto
    also have int (subdegree F) = zorder f.f z
      using has_fps_expansion_zorder[OF exp1] by simp
    also have ... = int (card (a - {z}))
      by (rule f.zorder)
    also have card (a - {z}) = (if z = 0 then 0 else c z)
    proof (cases z = 0)
      case True
      hence a - {z} = {}
        using a(2) by auto
      thus ?thesis using True by simp
    next
      case False
      thus ?thesis

```

```

      by (subst a(3)) (use True that in ⟨auto simp: Z_def⟩)
    qed
    also have int ... + (if z = 0 then m else 0) = zorder g z
      using ⟨zorder g z ≥ 0⟩ that by (auto simp: c_def m_def)
    finally show ?thesis .
  qed

  have eq_zero_iff: z ^ m * f.f z = 0 ⟷ g z = 0 if z ∈ A for z
    using that by (auto simp add: f.zero a m_def zorder_pos_iff Z_def)

  obtain h :: complex ⇒ complex where h:
    h holomorphic_on A ∧ z. z ∈ A ⟹ h z ≠ 0 ∧ z. z ∈ A ⟹ g z = h z * (z
    ^ m * f.f z)
  proof (rule holomorphic_zorder_factorization[OF assms(1-3)])
    show (λz. z ^ m * f.f z) holomorphic_on A
      by (intro holomorphic_intros)
    show z ^ m * f.f z = 0 ⟷ g z = 0 if z ∈ A for z
      by (rule eq_zero_iff) fact+
    show zorder (λz. z ^ m * f.f z) z = zorder g z if z ∈ A for z
      by (rule zorder_eq) fact+
  qed metis

  show ?thesis
  proof (rule that[of h λz. z ^ m * f.f z]; (intro ballI allI impI)?)
    show h holomorphic_on A
      by fact
    show (λz. z ^ m * f.f z) holomorphic_on UNIV
      by (intro holomorphic_intros)
  next
    fix z :: complex
    show (z ^ m * f.f z = 0) = ((∀ z ∈ A. g z = 0) ∨ z ∈ A ∧ g z = 0)
      using infinite_zeroes(1) a(2)
      by (auto simp: m_def zorder_eq eq_zero_iff zorder_pos_iff Z_def f.zero)
  qed (use zorder_eq eq_zero_iff h in auto)
  qed
  qed

```

The following is a simpler version for entire functions.

```

theorem weierstrass_factorization_UNIV:
  assumes g holomorphic_on UNIV
  obtains h f where
    h holomorphic_on UNIV f holomorphic_on UNIV
    ∀ z. f z = 0 ⟷ g z = 0
    ∀ z. zorder f z = zorder g z
    ∀ z. h z ≠ 0
    ∀ z. g z = h z * f z
  using assms by (rule weierstrass_factorization, goal_cases) auto

```

9.6 The factorisation theorem for meromorphic functions

Let g be a meromorphic function on a connected open domain A . Assume that the poles and zeros of g have no accumulation point on the border of A . Then g can be written in the form $g(z) = h(z)f_1(z)/f_2(z)$ where h is holomorphic on A with no zeroes and f_1 and f_2 are entire.

Moreover, as direct consequences of that, the zeroes of f_1 are precisely the zeroes of g and the zeros of f_2 are precisely the poles of g (with the corresponding multiplicity).

theorem *weierstrass_factorization_meromorphic:*

assumes *mero*: g *nicely_meromorphic_on* A **and** A : *open* A *connected* A

assumes *no_limpt*: $\bigwedge z. z \in \text{frontier } A \implies \neg z \text{ islimpt } \{w \in A. g \ w = 0 \vee \text{is_pole } g \ w\}$

obtains $h \ f1 \ f2$ **where**

h *holomorphic_on* A $f1$ *holomorphic_on* UNIV $f2$ *holomorphic_on* UNIV

$\forall z \in A. f1 \ z = 0 \longleftrightarrow \neg \text{is_pole } g \ z \wedge g \ z = 0$

$\forall z \in A. f2 \ z = 0 \longleftrightarrow \text{is_pole } g \ z$

$\forall z \in A. \neg \text{is_pole } g \ z \longrightarrow \text{zorder } f1 \ z = \text{zorder } g \ z$

$\forall z \in A. \text{is_pole } g \ z \longrightarrow \text{zorder } f2 \ z = -\text{zorder } g \ z$

$\forall z \in A. h \ z \neq 0$

$\forall z \in A. g \ z = h \ z * f1 \ z / f2 \ z$

proof –

have *mero'*: g *meromorphic_on* A

using *mero* **unfolding** *nicely_meromorphic_on_def* **by** *auto*

define *pts* **where** *pts* = $\{z \in A. \text{is_pole } g \ z\}$

have $\{z. \text{is_pole } g \ z\}$ *sparse_in* A

using *meromorphic_on_imp_not_pole_cosparse*[*OF mero*]

by (*auto simp: eventually_cosparse*)

hence *pts* *sparse_in* A

unfolding *pts_def* **by** (*rule sparse_in_subset2*) *auto*

have *open_diff_pts*: *open* ($A - \text{pts}'$) **if** $\text{pts}' \subseteq \text{pts}$ **for** pts'

proof (*rule open_diff_sparse_pts*)

show pts' *sparse_in* A

using $\langle \text{pts} \text{ sparse_in } A \rangle$ **by** (*rule sparse_in_subset2*) *fact*

qed (*use* $\langle \text{open } A \rangle$ **in** *auto*)

have *ev*: *eventually* $(\lambda w. w \in A - \text{pts})$ (*at* z) **if** $z \in A$ **for** z

proof (*cases* $z \in \text{pts}$)

case *False*

thus *?thesis*

using *that open_diff_pts*[*of pts*] **by** (*intro eventually_at_in_open'*) *auto*

next

case *True*

have *eventually* $(\lambda w. w \in (A - (\text{pts} - \{z\})) - \{z\})$ (*at* z)

using *that* **by** (*intro eventually_at_in_open open_diff_pts*) *auto*

also have $A - (\text{pts} - \{z\}) - \{z\} = A - \text{pts}$

using *True* **by** *auto*

finally show *?thesis* .

qed

```

show ?thesis
proof (cases  $\forall z \in A - \text{pts}. g\ z = 0$ )
  case True
  have no_poles:  $\neg \text{is\_pole } g\ z$  if  $z \in A$  for  $z$ 
  proof -
    have  $\text{is\_pole } g\ z \longleftrightarrow \text{is\_pole } (\lambda \_::\text{complex}. 0 :: \text{complex})\ z$ 
    by (intro is_pole_cong[OF eventually_mono[OF ev]]) (use that True in
auto)
    thus ?thesis
    by simp
  qed
  hence [simp]:  $\text{pts} = \{\}$ 
  by (auto simp: pts_def)
  have [simp]:  $\text{zorder } g\ z = \text{zorder } (\lambda \_::\text{complex}. 0 :: \text{complex})\ z$  if  $z \in A$  for  $z$ 
  by (intro zorder_cong[OF eventually_mono[OF ev]]) (use True that in auto)
  show ?thesis
  by (rule that[of  $\lambda \_. 1$   $\lambda \_. 0$   $\lambda \_. 1$ ]) (use True in  $\langle \text{auto simp: no\_poles} \rangle$ )

```

next

```

case False
have is_pole_iff:  $\text{is\_pole } g\ z \longleftrightarrow z \in \text{pts}$  if  $z \in A$  for  $z$ 
  using that by (auto simp: pts_def)

obtain h f1 where h_f1:
  h holomorphic_on  $A - \text{pts}$  f1 holomorphic_on UNIV
   $\forall z. f1\ z = 0 \longleftrightarrow (\forall z \in A - \text{pts}. g\ z = 0) \vee (z \in A - \text{pts} \wedge g\ z = 0)$ 
   $\forall z \in A - \text{pts}. \text{zorder } f1\ z = \text{zorder } g\ z$ 
   $\forall z \in A - \text{pts}. h\ z \neq 0$ 
   $\forall z \in A - \text{pts}. g\ z = h\ z * f1\ z$ 
proof (rule weierstrass_factorization)
  have g_analytic_on  $A - \text{pts}$ 
  by (rule nicely_meromorphic_without_singularities)
  (use mero in  $\langle \text{auto simp: pts\_def dest: nicely\_meromorphic\_on\_subset} \rangle$ )
  thus holo:  $g$  holomorphic_on  $A - \text{pts}$ 
  by (rule analytic_imp_holomorphic)
  show open  $(A - \text{pts})$ 
  by (rule open_diff_pts) auto
  show connected  $(A - \text{pts})$ 
  by (rule sparse_imp_connected) (use A  $\langle \text{pts sparse\_in } A \rangle$  in auto)
  show  $\neg z$  islimpt  $\{w \in A - \text{pts}. g\ w = 0\}$  if  $z \in \text{frontier } (A - \text{pts})$  for  $z$ 
  proof -
    from that have  $z \in \text{frontier } A - \text{pts} \cup \text{pts}$ 
    using  $\langle \text{open } (A - \text{pts}) \rangle \langle \text{open } A \rangle \text{closure\_mono[of } A - \text{pts } A]$ 
    by (auto simp: frontier_def interior_open)
    thus ?thesis
  proof

```

```

    assume  $z \in pts$ 
    hence  $is\_pole\ g\ z$ 
      by (auto simp: pts_def)
    hence eventually  $(\lambda z. g\ z \neq 0)$  (at  $z$ )
      using non_zero_neighbour_pole by blast
    hence  $\neg z\ islimpt\ \{w. g\ w = 0\}$ 
      by (auto simp: islimpt_iff_eventually)
    thus ?thesis
      using islimpt_subset[of  $z\ \{w \in A - pts. g\ w = 0\}\ \{w. g\ w = 0\}$ ] by blast
  next
    assume  $z: z \in frontier\ A - pts$ 
    show  $\neg z\ islimpt\ \{w \in A - pts. g\ w = 0\}$ 
    proof
      assume  $z\ islimpt\ \{w \in A - pts. g\ w = 0\}$ 
      hence  $z\ islimpt\ \{w \in A. g\ w = 0 \vee is\_pole\ g\ w\}$ 
        by (rule islimpt_subset) (auto simp: pts_def)
      thus False using no_limpt  $z$  by blast
    qed
  qed
qed
qed
qed

have  $f1\_eq\_0\_iff: f1\ z = 0 \longleftrightarrow (z \in A - pts \wedge g\ z = 0)$  for  $z$ 
  using  $h\_f1(3)$  False by auto

define  $h'$  where  $h' = (\lambda z. if\ z \in pts\ then\ 0\ else\ inverse\ (h\ z))$ 

have isolated_h: isolated_singularity_at  $h\ z$  if  $z \in pts$  for  $z$ 
proof -
  have open  $(A - (pts - \{z\}))$ 
    by (rule open_diff_pts) auto
  moreover have  $z \in (A - (pts - \{z\}))$ 
    using that by (auto simp: pts_def)
  moreover have  $h$  holomorphic_on  $(A - (pts - \{z\})) - \{z\}$ 
    by (rule holomorphic_on_subset[OF  $h\_f1(1)$ ]) (use that in auto)
  ultimately show isolated_singularity_at  $h\ z$ 
    using isolated_singularity_at_holomorphic by blast
qed

have is_pole_h: is_pole  $h\ z$  if  $z \in A$  is_pole  $g\ z$  for  $z$ 
proof -
  have  $f1$ :  $f1$  analytic_on  $\{z\}$ 
    by (meson analytic_on_holomorphic  $h\_f1(2)$  open_UNIV top_greatest)
  have eventually  $(\lambda w. g\ w \neq 0)$  (at  $z$ )
    using  $\langle is\_pole\ g\ z \rangle$  non_zero_neighbour_pole by blast
  with  $ev[OF\ that(1)]$  have  $ev'$ : eventually  $(\lambda w. g\ w * f1\ w \neq 0)$  (at  $z$ )
    by eventually_elim (use  $h\_f1(3)$  in auto)

  have is_pole  $(\lambda w. g\ w / f1\ w)\ z$ 

```

```

proof (rule is_pole_divide_zorder)
  show isolated_singularity_at f1 z not_essential f1 z
using f1 by (simp_all add: isolated_singularity_at_analytic not_essential_analytic)
show isolated_singularity_at g z not_essential g z
  using mero' that unfolding meromorphic_on_altdef by blast+
show freq: frequently ( $\lambda w. g w * f1 w \neq 0$ ) (at z)
  using ev' by (rule eventually_frequently[rotated]) auto
from freq have freq': frequently ( $\lambda w. f1 w \neq 0$ ) (at z)
  using frequently_elim1 by fastforce
have zorder g z < 0
using ⟨is_pole g z⟩ ⟨isolated_singularity_at g z⟩ isolated_pole_imp_neg_zorder
by auto
  also have  $0 \leq \text{zorder } f1 z$ 
    by (rule zorder_ge_0[OF f1 freq'])
  finally show zorder g z < zorder f1 z .
qed
also have ?this  $\longleftrightarrow$  is_pole h z
  proof (intro is_pole_cong refl eventually_mono[OF eventually_conj[OF
    ev[OF that(1)] ev']])
    fix w assume  $w \in A - \text{pts} \wedge g w * f1 w \neq 0$ 
    thus  $g w / f1 w = h w$  using h_f1(6)
    by (auto simp: divide_simps)
  qed
finally show is_pole h z .
qed

have h' analytic_on {z} if  $z \in A$  for z
proof (cases  $z \in \text{pts}$ )
  case False
    moreover have open (A - pts)
      by (rule open_diff_pts) auto
    ultimately have ( $\lambda z. \text{inverse } (h z)$ ) analytic_on {z}
      using that h_f1(1,2,5) ⟨open (A - pts)⟩ analytic_at False
      by (intro analytic_intros) (auto simp: f1_eq_0_iff)
    also have eventually ( $\lambda z. z \in A - \text{pts}$ ) (nhds z)
      using that False ⟨open (A - pts)⟩ by (intro eventually_nhds_in_open) auto
    hence ( $\lambda z. \text{inverse } (h z)$ ) analytic_on {z}  $\longleftrightarrow$  h' analytic_on {z}
      by (intro analytic_at_cong) (auto elim!: eventually_mono simp: h'_def)
    finally show ?thesis .
  next
    case True
    have ( $\lambda w. \text{if } w = z \text{ then } 0 \text{ else } \text{inverse } (h w)$ ) holomorphic_on (A - (pts - {z}))
      proof (rule is_pole_inverse_holomorphic)
        from True have  $A - (\text{pts} - \{z\}) - \{z\} = A - \text{pts}$ 
          by auto
        thus h holomorphic_on A - (pts - {z}) - {z}
          using h_f1(1) by simp
      next

```

```

    show open (A - (pts - {z}))
      by (rule open_diff_pts) auto
  next
    have is_pole g z
      using True that by (simp add: is_pole_iff)
    thus is_pole h z
      using is_pole_h that by auto
  next
    show  $\forall w \in A - (pts - \{z\}) - \{z\}. h\ w \neq 0$ 
      using h_f1(5) by auto
  qed
  also have ?this  $\longleftrightarrow$  h' holomorphic_on A - (pts - {z})
  proof (intro holomorphic_cong refl)
    fix w assume w:  $w \in A - (pts - \{z\})$ 
    show (if  $w = z$  then 0 else inverse (h w)) = h' w
      using True w by (cases w = z) (auto simp: h'_def)
  qed
  finally have h' holomorphic_on A - (pts - {z}) .
  moreover have  $z \in A - (pts - \{z\})$  open (A - (pts - {z}))
    using True that by (auto intro!: open_diff_pts)
  ultimately show h' analytic_on {z}
    using analytic_at by blast
  qed
  hence h': h' analytic_on A
    using analytic_on_analytic_at by blast

  have h'_eq_0_iff:  $h' w = 0 \longleftrightarrow \text{is\_pole } g\ w$  if  $w \in A$  for w
    using that h_f1(5) is_pole_iff[of w] by (auto simp: h'_def)

  obtain h'' f2 where h''_f2:
    h'' holomorphic_on A f2 holomorphic_on UNIV
     $\forall z. f2\ z = 0 \longleftrightarrow (\forall z \in A. h'\ z = 0) \vee (z \in A \wedge h'\ z = 0)$ 
     $\forall z \in A. h'\ z = 0 \longrightarrow \text{zorder } f2\ z = \text{zorder } h'\ z$ 
     $\forall z \in A. h''\ z \neq 0 \ \forall z \in A. h'\ z = h''\ z * f2\ z$ 
  proof (rule weierstrass_factorization[of h' A])
    show open A connected A
      by fact+
    show h' holomorphic_on A
      using h' <open A> by (simp add: analytic_on_open)
    show  $\neg z \text{ islimpt } \{w \in A. h'\ w = 0\}$  if  $z \in \text{frontier } A$  for z
  proof
    assume z islimpt {w ∈ A. h' w = 0}
    also have {w ∈ A. h' w = 0} = pts
      by (auto simp: h'_eq_0_iff pts_def)
    finally have z islimpt {w ∈ A. g w = 0 ∨ is_pole g w}
      by (rule islimpt_subset) (auto simp: pts_def)
    thus False using no_limpt[of z] that
      by blast
  qed
  qed

```



```

qed blast

show ?thesis
proof (rule that[of  $\lambda w. \text{inverse } (h'' w) f1 f2$ ]; (intro ballI allI impI)?)
  show  $(\lambda w. \text{inverse } (h'' w)) \text{ holomorphic\_on } A$ 
    using  $h''\_f2(1,2,5)$  by (intro holomorphic_intros) auto
next
  show  $f1 \text{ holomorphic\_on } UNIV \ f2 \text{ holomorphic\_on } UNIV$ 
    by fact+
next
  show  $f1 z = 0 \longleftrightarrow \neg \text{is\_pole } g z \wedge g z = 0 \text{ if } z \in A \text{ for } z$ 
    using that by (subst  $f1\_eq\_0\_iff$ ) (auto simp: pts_def)
next
  show  $f2 z = 0 \longleftrightarrow \text{is\_pole } g z \text{ if } z \in A \text{ for } z$ 
  proof -
    have  $\neg(\forall z \in A. h' z = 0)$ 
      using  $False \ h''\_f2(6) \ h\_f1(6) \ h'\_eq\_0\_iff \text{ is\_pole\_iff}$  by auto
    hence  $f2 z = 0 \longleftrightarrow h' z = 0$ 
      using  $h''\_f2(3)$  that by auto
    also have  $\dots \longleftrightarrow \text{is\_pole } g z$ 
      using that by (simp add:  $\text{is\_pole\_iff } h'\_eq\_0\_iff$ )
    finally show ?thesis .
  qed
next
  show  $\text{zorder } f1 z = \text{zorder } g z \text{ if } z \in A \neg \text{is\_pole } g z \text{ for } z$ 
    using  $h\_f1(4)$  that by (auto simp: pts_def)
next
  show  $\text{zorder } f2 z = -\text{zorder } g z \text{ if } z \in A \text{ is\_pole } g z \text{ for } z$ 
  proof -
    have  $\text{zorder } f2 z = \text{zorder } h' z$ 
      using  $h''\_f2(4)$  that  $h'\_eq\_0\_iff[of z] \text{ is\_pole\_iff}[of z]$  by auto
    also have  $\dots = \text{zorder } (\lambda w. \text{inverse } (h w)) z$ 
      using that by (intro zorder_cong eventually_mono[OF ev]) (auto simp:
 $h'\_def$ )
    also have  $\dots = -\text{zorder } h z$ 
  proof (intro zorder_inverse)
    have  $\text{is\_pole } h z$ 
      using that  $\text{is\_pole\_iff}[of z] \text{ is\_pole\_h}[of z]$  by auto
    thus  $\text{not\_essential } h z$ 
      by force
    show frequently  $(\lambda w. h w \neq 0) \text{ (at } z)$ 
      using  $\text{non\_zero\_neighbour\_pole}[OF \langle \text{is\_pole } h z \rangle]$  eventually_frequently
  by force
  qed (use that in  $\langle \text{auto intro!; isolated\_h simp: pts\_def} \rangle$ )
  also have  $\text{zorder } f1 z = 0$ 
  proof (rule zorder_eq_0I)
    show  $f1 \text{ analytic\_on } \{z\}$ 
      using that  $h\_f1(2) \text{ holomorphic\_on\_imp\_analytic\_at}$  by blast
    show  $f1 z \neq 0$ 

```

```

      using that h_f1(3) False by (auto simp: pts_def)
    qed
  hence zorder h z = zorder f1 z + zorder h z
    by simp
  also have ... = zorder ( $\lambda w. f1\ w * h\ w$ ) z
  proof (rule zorder_times [symmetric])
    have f1 analytic_on {z}
      using that h_f1(2) holomorphic_on_imp_analytic_at by blast
    thus isolated_singularity_at f1 z not_essential f1 z
      using isolated_singularity_at_analytic not_essential_analytic by blast+
    show isolated_singularity_at h z
      using that by (intro isolated_h) (auto simp: pts_def)
    have is_pole h z
      using is_pole_iff[of z] that by (intro is_pole_h) auto
    thus not_essential h z
      by force
    have z  $\in A$ 
      using that by auto
    have eventually ( $\lambda w. g\ w \neq 0$ ) (at z)
      using non_zero_neighbour_pole[of g z] that by auto
    hence eventually ( $\lambda w. f1\ w * h\ w \neq 0$ ) (at z)
      using ev[OF  $\langle z \in A \rangle$ ] by eventually_elim (use h_f1(6) in auto)
    thus frequently ( $\lambda w. f1\ w * h\ w \neq 0$ ) (at z)
      using eventually_frequently by force
  qed
  also have ... = zorder g z
  proof (rule zorder_cong)
    have eventually ( $\lambda w. w \in A - pts$ ) (at z)
      using ev[of z] that by auto
    thus eventually ( $\lambda w. f1\ w * h\ w = g\ w$ ) (at z)
      by eventually_elim (use h_f1(6) in auto)
  qed auto
  finally show ?thesis .
next
  show inverse (h'' z)  $\neq 0$  if z  $\in A$  for z
    using h''_f2(5) that by auto
next
  show g z = inverse (h'' z) * f1 z / f2 z if z: z  $\in A$  for z
  proof (cases is_pole g z)
    case False
    have *: g z = h z * f1 z h' z = h'' z * f2 z h'' z  $\neq 0$  h z  $\neq 0$ 
      using that h''_f2(5,6) h_f1(5,6) False unfolding pts_def by blast+
    have g z = h z * f1 z
      by fact
    also have ... = f1 z / h' z
      using * that False by (simp add: h'_def field_simps pts_def)
    also have h' z = h'' z * f2 z
      by fact

```

```

    also have  $f1\ z / (h''\ z * f2\ z) = \text{inverse } (h''\ z) * f1\ z / f2\ z$ 
      by (simp add: divide_inverse_commute)
    finally show ?thesis .
  next
  case True
  have  $\neg g\ -z \rightarrow g\ z$ 
  using True at_neq_bot is_pole_def not_tendsto_and_filterlim_at_infinity
by blast
  with mero and z and True have  $g\ z = 0$ 
    by (auto simp: nicely_meromorphic_on_def)
  moreover have  $f2\ z = 0$ 
    using True z by (simp add: h''_f2(3) h'_eq_0_iff)
  ultimately show ?thesis by simp
qed
qed
qed
qed

```

Again, we derive an easier version for functions meromorphic on the entire complex plane.

theorem *weierstrass_factorization_meromorphic_UNIV*:

assumes g nicely_meromorphic_on UNIV

obtains $h\ f1\ f2$ **where**

h holomorphic_on UNIV $f1$ holomorphic_on UNIV $f2$ holomorphic_on UNIV

$\forall z. f1\ z = 0 \longleftrightarrow \neg \text{is_pole } g\ z \wedge g\ z = 0$

$\forall z. f2\ z = 0 \longleftrightarrow \text{is_pole } g\ z$

$\forall z. \neg \text{is_pole } g\ z \longrightarrow \text{zorder } f1\ z = \text{zorder } g\ z$

$\forall z. \text{is_pole } g\ z \longrightarrow \text{zorder } f2\ z = -\text{zorder } g\ z$

$\forall z. h\ z \neq 0$

$\forall z. g\ z = h\ z * f1\ z / f2\ z$

proof (rule weierstrass_factorization_meromorphic)

show g nicely_meromorphic_on UNIV

by fact

show connected (UNIV :: complex set)

by (simp add: Convex.connected_UNIV)

show $\neg z$ islimpt $\{w \in \text{UNIV}. g\ w = 0 \vee \text{is_pole } g\ w\}$ **if** $z \in \text{frontier UNIV}$ **for**

using that **by** simp

show open (UNIV :: complex set)

by simp

qed auto

end

theory Complex_Analysis

imports

Riemann_Mapping

Residue_Theorem

Weierstrass_Factorization

begin

end

References

[1]