

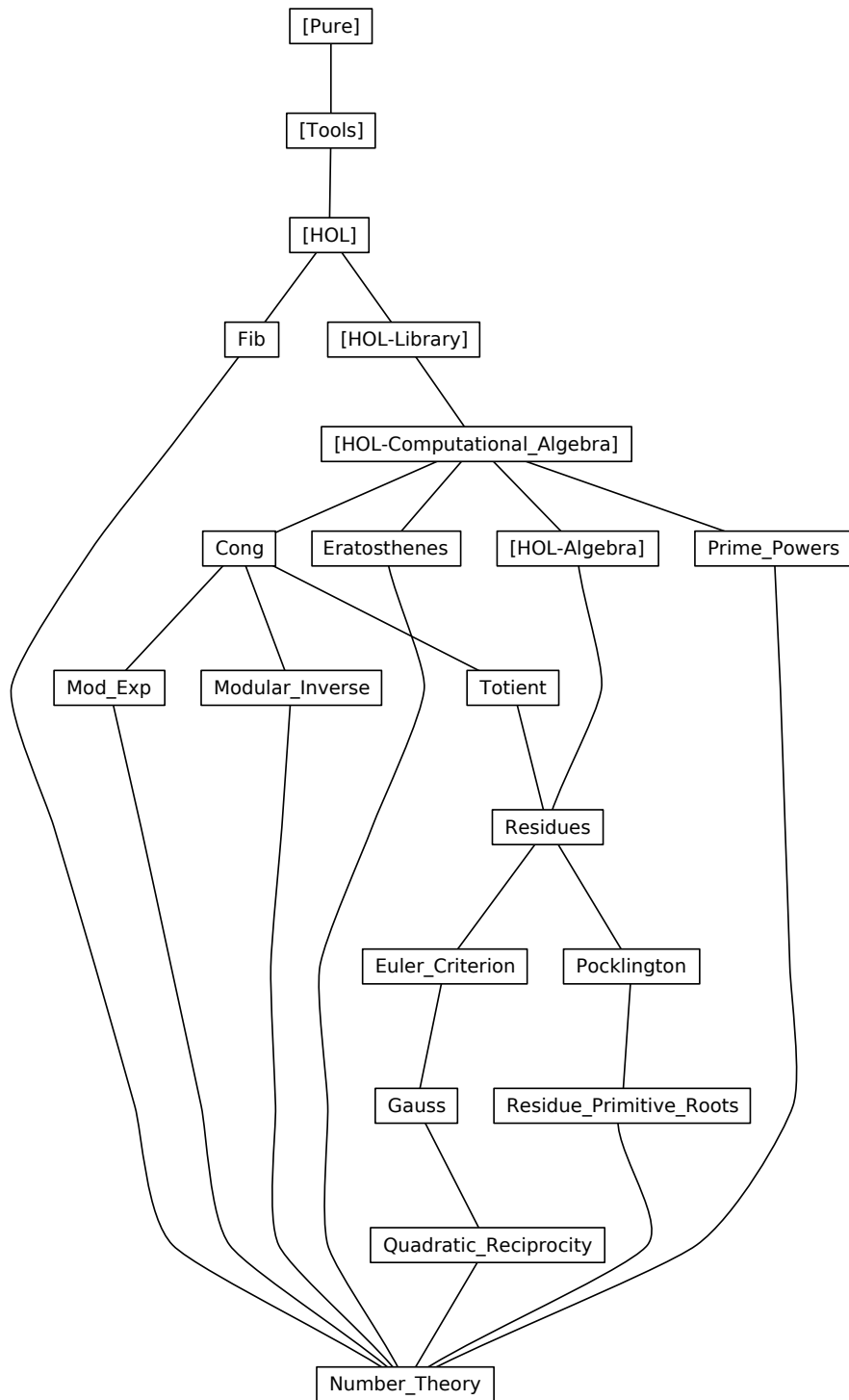
Various results of number theory

December 17, 2025

Contents

1	The fibonacci function	3
1.1	Fibonacci numbers	3
1.2	Basic Properties	3
1.3	More efficient code	3
1.4	A Few Elementary Results	4
1.5	Law 6.111 of Concrete Mathematics	4
1.6	Closed form	5
1.7	Divide-and-Conquer recurrence	5
1.8	Fibonacci and Binomial Coefficients	6
2	Congruence	6
2.1	Generic congruences	6
2.2	Congruences on <i>nat</i> and <i>int</i>	11
3	Fundamental facts about Euler's totient function	16
4	Residue rings	21
4.1	A locale for residue rings	22
4.2	Prime residues	24
5	Test cases: Euler's theorem and Wilson's theorem	25
5.1	Euler's theorem	25
5.2	Wilson's theorem	26
5.3	Upper bound for the number of n -th roots	26
6	The sieve of Eratosthenes	26
6.1	Preliminary: strict divisibility	27
6.2	Main corpus	27
6.3	Application: smallest prime beyond a certain number	29
7	Fast modular exponentiation	30

8	Gauss' Lemma	33
8.1	Basic properties of p	34
8.2	Basic Properties of the Gauss Sets	34
8.3	Relationships Between Gauss Sets	36
8.4	Gauss' Lemma	37
9	Pocklington's Theorem for Primes	42
9.1	Lemmas about previously defined terms	42
9.2	Some basic theorems about solving congruences	42
9.3	Lucas's theorem	43
9.4	Definition of the order of a number mod n	43
9.5	Another trivial primality characterization	46
9.6	Pocklington theorem	46
9.7	Prime factorizations	47
10	Prime powers	48
11	Primitive roots in residue rings and Carmichael's function	53
11.1	Primitive roots in residue rings	53
11.2	Primitive roots modulo a prime	54
11.3	Primitive roots modulo powers of an odd prime	55
11.4	Carmichael's function	56
11.5	Existence of primitive roots for general moduli	59
12	Modular Inverses	60
13	Comprehensive number theory	62



1 The fibonacci function

```
theory Fib
  imports Complex-Main
begin
```

1.1 Fibonacci numbers

```
fun fib :: nat  $\Rightarrow$  nat
  where
    fib0: fib 0 = 0
  | fib1: fib (Suc 0) = 1
  | fib2: fib (Suc (Suc n)) = fib (Suc n) + fib n
```

1.2 Basic Properties

```
lemma fib-1 [simp]: fib 1 = 1
  <proof>
```

```
lemma fib-2 [simp]: fib 2 = 1
  <proof>
```

```
lemma fib-plus-2: fib (n + 2) = fib (n + 1) + fib n
  <proof>
```

```
lemma fib-add: fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n
  <proof>
```

```
lemma fib-neq-0-nat: n > 0  $\implies$  fib n > 0
  <proof>
```

```
lemma fib-Suc-mono: fib m  $\leq$  fib (Suc m)
  <proof>
```

```
lemma fib-mono: m  $\leq$  n  $\implies$  fib m  $\leq$  fib n
  <proof>
```

1.3 More efficient code

The naive approach is very inefficient since the branching recursion leads to many values of *fib* being computed multiple times. We can avoid this by “remembering” the last two values in the sequence, yielding a tail-recursive version. This is far from optimal (it takes roughly $O(n \cdot M(n))$ time where $M(n)$ is the time required to multiply two n -bit integers), but much better than the naive version, which is exponential.

```
fun gen-fib :: nat  $\Rightarrow$  nat  $\Rightarrow$  nat  $\Rightarrow$  nat
  where
    gen-fib a b 0 = a
  | gen-fib a b (Suc 0) = b
```

$$| \text{gen-fib } a \ b \ (\text{Suc } (\text{Suc } n)) = \text{gen-fib } b \ (a + b) \ (\text{Suc } n)$$

lemma *gen-fib-recurrence*: $\text{gen-fib } a \ b \ (\text{Suc } (\text{Suc } n)) = \text{gen-fib } a \ b \ n + \text{gen-fib } a \ b \ (\text{Suc } n)$
 ⟨proof⟩

lemma *gen-fib-fib*: $\text{gen-fib } (\text{fib } n) \ (\text{fib } (\text{Suc } n)) \ m = \text{fib } (n + m)$
 ⟨proof⟩

lemma *fib-conv-gen-fib*: $\text{fib } n = \text{gen-fib } 0 \ 1 \ n$
 ⟨proof⟩

declare *fib-conv-gen-fib* [code]

1.4 A Few Elementary Results

Concrete Mathematics, page 278: Cassini's identity. The proof is much easier using integers, not natural numbers!

lemma *fib-Cassini-int*: $\text{int } (\text{fib } (\text{Suc } (\text{Suc } n)) * \text{fib } n) - \text{int } ((\text{fib } (\text{Suc } n))^2) = -((-1)^n)$
 ⟨proof⟩

lemma *fib-Cassini-nat*:
 $\text{fib } (\text{Suc } (\text{Suc } n)) * \text{fib } n =$
 (if even n then $(\text{fib } (\text{Suc } n))^2 - 1$ else $(\text{fib } (\text{Suc } n))^2 + 1$)
 ⟨proof⟩

1.5 Law 6.111 of Concrete Mathematics

lemma *coprime-fib-Suc-nat*: $\text{coprime } (\text{fib } n) \ (\text{fib } (\text{Suc } n))$
 ⟨proof⟩

lemma *gcd-fib-add*:
 $\text{gcd } (\text{fib } m) \ (\text{fib } (n + m)) = \text{gcd } (\text{fib } m) \ (\text{fib } n)$
 ⟨proof⟩

lemma *gcd-fib-diff*: $m \leq n \implies \text{gcd } (\text{fib } m) \ (\text{fib } (n - m)) = \text{gcd } (\text{fib } m) \ (\text{fib } n)$
 ⟨proof⟩

lemma *gcd-fib-mod*: $0 < m \implies \text{gcd } (\text{fib } m) \ (\text{fib } (n \bmod m)) = \text{gcd } (\text{fib } m) \ (\text{fib } n)$
 ⟨proof⟩

lemma *fib-gcd*: $\text{fib } (\text{gcd } m \ n) = \text{gcd } (\text{fib } m) \ (\text{fib } n)$ — Law 6.111
 ⟨proof⟩

theorem *fib-mult-eq-sum-nat*: $\text{fib } (\text{Suc } n) * \text{fib } n = (\sum k \in \{..n\}. \text{fib } k * \text{fib } k)$
 ⟨proof⟩

1.6 Closed form

lemma *fib-closed-form*:

```

fixes  $\varphi \ \psi :: \text{real}$ 
defines  $\varphi \equiv (1 + \text{sqrt } 5) / 2$ 
and  $\psi \equiv (1 - \text{sqrt } 5) / 2$ 
shows  $\text{of-nat } (\text{fib } n) = (\varphi^n - \psi^n) / \text{sqrt } 5$ 
<proof>

```

lemma *fib-closed-form'*:

```

fixes  $\varphi \ \psi :: \text{real}$ 
defines  $\varphi \equiv (1 + \text{sqrt } 5) / 2$ 
and  $\psi \equiv (1 - \text{sqrt } 5) / 2$ 
assumes  $n > 0$ 
shows  $\text{fib } n = \text{round } (\varphi^n / \text{sqrt } 5)$ 
<proof>

```

lemma *fib-asymptotics*:

```

fixes  $\varphi :: \text{real}$ 
defines  $\varphi \equiv (1 + \text{sqrt } 5) / 2$ 
shows  $(\lambda n. \text{real } (\text{fib } n) / (\varphi^n / \text{sqrt } 5)) \longrightarrow 1$ 
<proof>

```

1.7 Divide-and-Conquer recurrence

The following divide-and-conquer recurrence allows for a more efficient computation of Fibonacci numbers; however, it requires memoisation of values to be reasonably efficient, cutting the number of values to be computed to logarithmically many instead of linearly many. The vast majority of the computation time is then actually spent on the multiplication, since the output number is exponential in the input number.

lemma *fib-rec-odd*:

```

fixes  $\varphi \ \psi :: \text{real}$ 
defines  $\varphi \equiv (1 + \text{sqrt } 5) / 2$ 
and  $\psi \equiv (1 - \text{sqrt } 5) / 2$ 
shows  $\text{fib } (\text{Suc } (2 * n)) = \text{fib } n^2 + \text{fib } (\text{Suc } n)^2$ 
<proof>

```

lemma *fib-rec-even*: $\text{fib } (2 * n) = (\text{fib } (n - 1) + \text{fib } (n + 1)) * \text{fib } n$

<proof>

lemma *fib-rec-even'*: $\text{fib } (2 * n) = (2 * \text{fib } (n - 1) + \text{fib } n) * \text{fib } n$

<proof>

lemma *fib-rec*:

```

fib  $n =$ 
  (if  $n = 0$  then 0 else if  $n = 1$  then 1
   else if even  $n$  then let  $n' = n \text{ div } 2$ ;  $fn = \text{fib } n'$  in  $(2 * \text{fib } (n' - 1) + fn) * fn$ 
   else let  $n' = n \text{ div } 2$  in  $\text{fib } n'^2 + \text{fib } (\text{Suc } n')^2$ )

```

$\langle proof \rangle$

1.8 Fibonacci and Binomial Coefficients

lemma *sum-drop-zero*: $(\sum k = 0..Suc\ n. \text{if } 0 < k \text{ then } (f\ (k - 1)) \text{ else } 0) = (\sum j = 0..n. f\ j)$
 $\langle proof \rangle$

lemma *sum-choose-drop-zero*:
 $(\sum k = 0..Suc\ n. \text{if } k = 0 \text{ then } 0 \text{ else } (Suc\ n - k) \text{ choose } (k - 1)) =$
 $(\sum j = 0..n. (n-j) \text{ choose } j)$
 $\langle proof \rangle$

lemma *ne-diagonal-fib*: $(\sum k = 0..n. (n-k) \text{ choose } k) = fib\ (Suc\ n)$
 $\langle proof \rangle$

end

2 Congruence

theory *Cong*
imports *HOL-Computational-Algebra.Primes*
begin

2.1 Generic congruences

context *unique-euclidean-semiring*
begin

definition *cong* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
 $(\langle (\langle \text{indent}=1 \text{ notation}=\langle \text{mixfix } cong \rangle [- = -] '(\text{' mod -})) \rangle)$
where $[b = c] \text{ (mod } a) \longleftrightarrow b \text{ mod } a = c \text{ mod } a$

abbreviation *notcong* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
 $(\langle (\langle \text{indent}=1 \text{ notation}=\langle \text{mixfix } notcong \rangle [- \neq -] '(\text{' mod -})) \rangle)$
where $[b \neq c] \text{ (mod } a) \equiv \neg \text{cong } b\ c\ a$

lemma *cong-refl* [*simp*]:
 $[b = b] \text{ (mod } a)$
 $\langle proof \rangle$

lemma *cong-sym*:
 $[b = c] \text{ (mod } a) \implies [c = b] \text{ (mod } a)$
 $\langle proof \rangle$

lemma *cong-sym-eq*:
 $[b = c] \text{ (mod } a) \longleftrightarrow [c = b] \text{ (mod } a)$
 $\langle proof \rangle$

lemma *cong-trans* [*trans*]:
 $[b = c] \text{ (mod } a) \implies [c = d] \text{ (mod } a) \implies [b = d] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-mult-self-right*:
 $[b * a = 0] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-mult-self-left*:
 $[a * b = 0] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-mod-left* [*simp*]:
 $[b \text{ mod } a = c] \text{ (mod } a) \longleftrightarrow [b = c] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-mod-right* [*simp*]:
 $[b = c \text{ mod } a] \text{ (mod } a) \longleftrightarrow [b = c] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-0* [*simp*, *presburger*]:
 $[b = c] \text{ (mod } 0) \longleftrightarrow b = c$
 $\langle \text{proof} \rangle$

lemma *cong-1* [*simp*, *presburger*]:
 $[b = c] \text{ (mod } 1)$
 $\langle \text{proof} \rangle$

lemma *cong-dvd-iff*:
 $a \text{ dvd } b \longleftrightarrow a \text{ dvd } c \text{ if } [b = c] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-0-iff*: $[b = 0] \text{ (mod } a) \longleftrightarrow a \text{ dvd } b$
 $\langle \text{proof} \rangle$

lemma *cong-add*:
 $[b = c] \text{ (mod } a) \implies [d = e] \text{ (mod } a) \implies [b + d = c + e] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-mult*:
 $[b = c] \text{ (mod } a) \implies [d = e] \text{ (mod } a) \implies [b * d = c * e] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-scalar-right*:
 $[b = c] \text{ (mod } a) \implies [b * d = c * d] \text{ (mod } a)$
 $\langle \text{proof} \rangle$

lemma *cong-scalar-left*:
 $[b = c] \text{ (mod } a) \implies [d * b = d * c] \text{ (mod } a)$

$\langle proof \rangle$

lemma *cong-pow*:

$[b = c] \text{ (mod } a) \implies [b \wedge n = c \wedge n] \text{ (mod } a)$

$\langle proof \rangle$

lemma *cong-sum*:

$[sum\ f\ A = sum\ g\ A] \text{ (mod } a) \text{ if } \bigwedge x. x \in A \implies [f\ x = g\ x] \text{ (mod } a)$

$\langle proof \rangle$

lemma *cong-prod*:

$[prod\ f\ A = prod\ g\ A] \text{ (mod } a) \text{ if } (\bigwedge x. x \in A \implies [f\ x = g\ x] \text{ (mod } a))$

$\langle proof \rangle$

lemma *mod-mult-cong-right*:

$[c \text{ mod } (a * b) = d] \text{ (mod } a) \longleftrightarrow [c = d] \text{ (mod } a)$

$\langle proof \rangle$

lemma *mod-mult-cong-left*:

$[c \text{ mod } (b * a) = d] \text{ (mod } a) \longleftrightarrow [c = d] \text{ (mod } a)$

$\langle proof \rangle$

lemma *cong-mod-leftI* [simp]:

$[b = c] \text{ (mod } a) \implies [b \text{ mod } a = c] \text{ (mod } a)$

$\langle proof \rangle$

lemma *cong-mod-rightI* [simp]:

$[b = c] \text{ (mod } a) \implies [b = c \text{ mod } a] \text{ (mod } a)$

$\langle proof \rangle$

lemma *cong-cmult-leftI*: $[a = b] \text{ (mod } m) \implies [c * a = c * b] \text{ (mod } (c * m))$

$\langle proof \rangle$

lemma *cong-cmult-rightI*: $[a = b] \text{ (mod } m) \implies [a * c = b * c] \text{ (mod } (m * c))$

$\langle proof \rangle$

lemma *cong-dvd-mono-modulus*:

assumes $[a = b] \text{ (mod } m) \ m' \text{ dvd } m$

shows $[a = b] \text{ (mod } m')$

$\langle proof \rangle$

lemma *coprime-cong-transfer-left*:

assumes *coprime* $a\ b$ $[a = a'] \text{ (mod } b)$

shows *coprime* $a'\ b$

$\langle proof \rangle$

lemma *coprime-cong-transfer-right*:

assumes *coprime* $a\ b$ $[b = b'] \text{ (mod } a)$

shows *coprime* $a\ b'$

$\langle proof \rangle$

lemma *coprime-cong-cong-left*:

assumes $[a = a'] \pmod{b}$

shows $\text{coprime } a \ b \longleftrightarrow \text{coprime } a' \ b$

$\langle proof \rangle$

lemma *coprime-cong-cong-right*:

assumes $[b = b'] \pmod{a}$

shows $\text{coprime } a \ b \longleftrightarrow \text{coprime } a \ b'$

$\langle proof \rangle$

end

context *unique-euclidean-ring*

begin

lemma *cong-diff*:

$[b = c] \pmod{a} \implies [d = e] \pmod{a} \implies [b - d = c - e] \pmod{a}$

$\langle proof \rangle$

lemma *cong-diff-iff-cong-0*:

$[b - c = 0] \pmod{a} \longleftrightarrow [b = c] \pmod{a} \text{ (is } ?P \longleftrightarrow ?Q)$

$\langle proof \rangle$

lemma *cong-minus-minus-iff*:

$[- \ b = - \ c] \pmod{a} \longleftrightarrow [b = c] \pmod{a}$

$\langle proof \rangle$

lemma *cong-modulus-minus-iff [iff]*:

$[b = c] \pmod{-a} \longleftrightarrow [b = c] \pmod{a}$

$\langle proof \rangle$

lemma *cong-iff-dvd-diff*:

$[a = b] \pmod{m} \longleftrightarrow m \text{ dvd } (a - b)$

$\langle proof \rangle$

lemma *cong-iff-lin*:

$[a = b] \pmod{m} \longleftrightarrow (\exists k. \ b = a + m * k) \text{ (is } ?P \longleftrightarrow ?Q)$

$\langle proof \rangle$

lemma *cong-add-lcancel*:

$[a + x = a + y] \pmod{n} \longleftrightarrow [x = y] \pmod{n}$

$\langle proof \rangle$

lemma *cong-add-rcancel*:

$[x + a = y + a] \pmod{n} \longleftrightarrow [x = y] \pmod{n}$

$\langle proof \rangle$

lemma *cong-add-lcancel-0*:

$$[a + x = a] \text{ (mod } n) \longleftrightarrow [x = 0] \text{ (mod } n)$$

<proof>

lemma *cong-add-rcancel-0*:

$$[x + a = a] \text{ (mod } n) \longleftrightarrow [x = 0] \text{ (mod } n)$$

<proof>

lemma *cong-dvd-modulus*:

$$[x = y] \text{ (mod } n) \text{ if } [x = y] \text{ (mod } m) \text{ and } n \text{ dvd } m$$

<proof>

lemma *cong-modulus-mult*:

$$[x = y] \text{ (mod } m) \text{ if } [x = y] \text{ (mod } m * n)$$

<proof>

lemma *cong-uminus*: $[x = y] \text{ (mod } m) \implies [-x = -y] \text{ (mod } m)$

<proof>

end

lemma *cong-abs [simp]*:

$$[x = y] \text{ (mod } |m|) \longleftrightarrow [x = y] \text{ (mod } m)$$

for $x\ y :: 'a :: \{\text{unique-euclidean-ring, linordered-idom}\}$

<proof>

lemma *cong-square*:

$$\text{prime } p \implies 0 < a \implies [a * a = 1] \text{ (mod } p) \implies [a = 1] \text{ (mod } p) \vee [a = -1] \text{ (mod } p)$$

for $a\ p :: 'a :: \{\text{normalization-semidom, linordered-idom, unique-euclidean-ring}\}$

<proof>

lemma *cong-mult-rcancel*:

$$[a * k = b * k] \text{ (mod } m) \longleftrightarrow [a = b] \text{ (mod } m)$$

if *coprime* $k\ m$ **for** $a\ k\ m :: 'a :: \{\text{unique-euclidean-ring, ring-gcd}\}$

<proof>

lemma *cong-mult-lcancel*:

$$[k * a = k * b] \text{ (mod } m) = [a = b] \text{ (mod } m)$$

if *coprime* $k\ m$ **for** $a\ k\ m :: 'a :: \{\text{unique-euclidean-ring, ring-gcd}\}$

<proof>

lemma *coprime-cong-mult*:

$$[a = b] \text{ (mod } m) \implies [a = b] \text{ (mod } n) \implies \text{coprime } m\ n \implies [a = b] \text{ (mod } m * n)$$

for $a\ b :: 'a :: \{\text{unique-euclidean-ring, semiring-gcd}\}$

<proof>

lemma *cong-gcd-eq*:

$$\text{gcd } a\ m = \text{gcd } b\ m \text{ if } [a = b] \text{ (mod } m)$$

for $a\ b :: 'a :: \{\text{unique-euclidean-semiring}, \text{euclidean-semiring-gcd}\}$
 $\langle \text{proof} \rangle$

lemma *cong-imp-coprime*:

$[a = b] \ (\text{mod } m) \implies \text{coprime } a\ m \implies \text{coprime } b\ m$
for $a\ b :: 'a :: \{\text{unique-euclidean-semiring}, \text{euclidean-semiring-gcd}\}$
 $\langle \text{proof} \rangle$

lemma *cong-cong-prod-coprime*:

$[x = y] \ (\text{mod } (\prod_{i \in A} m\ i))$ **if**
 $(\forall i \in A. [x = y] \ (\text{mod } m\ i))$
 $(\forall i \in A. (\forall j \in A. i \neq j \longrightarrow \text{coprime } (m\ i) \ (m\ j)))$
for $x\ y :: 'a :: \{\text{unique-euclidean-ring}, \text{semiring-gcd}\}$
 $\langle \text{proof} \rangle$

2.2 Congruences on *nat* and *int*

lemma *cong-int-iff*:

$[\text{int } m = \text{int } q] \ (\text{mod } \text{int } n) \longleftrightarrow [m = q] \ (\text{mod } n)$
 $\langle \text{proof} \rangle$

lemma *cong-Suc-0* [*simp*, *presburger*]:

$[m = n] \ (\text{mod } \text{Suc } 0)$
 $\langle \text{proof} \rangle$

lemma *cong-diff-nat*:

$[a - c = b - d] \ (\text{mod } m)$ **if** $[a = b] \ (\text{mod } m)$ $[c = d] \ (\text{mod } m)$
and $a \geq c$ $b \geq d$ **for** $a\ b\ c\ d\ m :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-diff-iff-cong-0-nat*:

$[a - b = 0] \ (\text{mod } m) \longleftrightarrow [a = b] \ (\text{mod } m)$ **if** $a \geq b$ **for** $a\ b :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-diff-iff-cong-0-nat'*:

$[\text{nat } |\text{int } a - \text{int } b| = 0] \ (\text{mod } m) \longleftrightarrow [a = b] \ (\text{mod } m)$
 $\langle \text{proof} \rangle$

lemma *cong-altdef-nat*:

$a \geq b \implies [a = b] \ (\text{mod } m) \longleftrightarrow m \text{ dvd } (a - b)$
for $a\ b :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-altdef-nat'*:

$[a = b] \ (\text{mod } m) \longleftrightarrow m \text{ dvd } \text{nat } |\text{int } a - \text{int } b|$
 $\langle \text{proof} \rangle$

lemma *cong-mult-rcancel-nat*:

$[a * k = b * k] \ (\text{mod } m) \longleftrightarrow [a = b] \ (\text{mod } m)$

if *coprime* *k m* **for** *a k m :: nat*
 ⟨*proof*⟩

lemma *cong-mult-lcancel-nat*:
 $[k * a = k * b] \text{ (mod } m) = [a = b] \text{ (mod } m)$
if *coprime* *k m* **for** *a k m :: nat*
 ⟨*proof*⟩

lemma *coprime-cong-mult-nat*:
 $[a = b] \text{ (mod } m) \implies [a = b] \text{ (mod } n) \implies \text{coprime } m \ n \implies [a = b] \text{ (mod } m * n)$
for *a b :: nat*
 ⟨*proof*⟩

lemma *cong-less-imp-eq-nat*: $0 \leq a \implies a < m \implies 0 \leq b \implies b < m \implies [a = b] \text{ (mod } m) \implies a = b$
for *a b :: nat*
 ⟨*proof*⟩

lemma *cong-less-imp-eq-int*: $0 \leq a \implies a < m \implies 0 \leq b \implies b < m \implies [a = b] \text{ (mod } m) \implies a = b$
for *a b :: int*
 ⟨*proof*⟩

lemma *cong-less-unique-nat*: $0 < m \implies (\exists ! b. 0 \leq b \wedge b < m \wedge [a = b] \text{ (mod } m))$
for *a m :: nat*
 ⟨*proof*⟩

lemma *cong-less-unique-int*: $0 < m \implies (\exists ! b. 0 \leq b \wedge b < m \wedge [a = b] \text{ (mod } m))$
for *a m :: int*
 ⟨*proof*⟩

lemma *cong-iff-lin-nat*: $[a = b] \text{ (mod } m) \longleftrightarrow (\exists k1 \ k2. b + k1 * m = a + k2 * m)$
for *a b :: nat*
 ⟨*proof*⟩

lemma *cong-cong-mod-nat*: $[a = b] \text{ (mod } m) \longleftrightarrow [a \text{ mod } m = b \text{ mod } m] \text{ (mod } m)$
for *a b :: nat*
 ⟨*proof*⟩

lemma *cong-cong-mod-int*: $[a = b] \text{ (mod } m) \longleftrightarrow [a \text{ mod } m = b \text{ mod } m] \text{ (mod } m)$
for *a b :: int*
 ⟨*proof*⟩

lemma *cong-add-lcancel-nat*: $[a + x = a + y] \text{ (mod } n) \longleftrightarrow [x = y] \text{ (mod } n)$
for *a x y :: nat*
 ⟨*proof*⟩

lemma *cong-add-rcancel-nat*: $[x + a = y + a] \text{ (mod } n) \longleftrightarrow [x = y] \text{ (mod } n)$
for $a \ x \ y :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-add-lcancel-0-nat*: $[a + x = a] \text{ (mod } n) \longleftrightarrow [x = 0] \text{ (mod } n)$
for $a \ x :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-add-rcancel-0-nat*: $[x + a = a] \text{ (mod } n) \longleftrightarrow [x = 0] \text{ (mod } n)$
for $a \ x :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-dvd-modulus-nat*: $[x = y] \text{ (mod } m) \implies n \text{ dvd } m \implies [x = y] \text{ (mod } n)$
for $x \ y :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-to-1-nat*:
fixes $a :: \text{nat}$
assumes $[a = 1] \text{ (mod } n)$
shows $n \text{ dvd } (a - 1)$
 $\langle \text{proof} \rangle$

lemma *cong-0-1-nat'*: $[0 = \text{Suc } 0] \text{ (mod } n) \longleftrightarrow n = \text{Suc } 0$
 $\langle \text{proof} \rangle$

lemma *cong-0-1-nat*: $[0 = 1] \text{ (mod } n) \longleftrightarrow n = 1$
for $n :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-0-1-int*: $[0 = 1] \text{ (mod } n) \longleftrightarrow n = 1 \vee n = -1$
for $n :: \text{int}$
 $\langle \text{proof} \rangle$

lemma *cong-to-1'-nat*: $[a = 1] \text{ (mod } n) \longleftrightarrow a = 0 \wedge n = 1 \vee (\exists m. a = 1 + m * n)$
for $a :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-le-nat*: $y \leq x \implies [x = y] \text{ (mod } n) \longleftrightarrow (\exists q. x = q * n + y)$
for $x \ y :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-solve-nat*:
fixes $a :: \text{nat}$
shows $\exists x. [a * x = \text{gcd } a \ n] \text{ (mod } n)$
 $\langle \text{proof} \rangle$

lemma *cong-solve-int*:
fixes $a :: \text{int}$

shows $\exists x. [a * x = \text{gcd } a \ n] \ (\text{mod } n)$
 $\langle \text{proof} \rangle$

lemma *cong-solve-dvd-nat*:
fixes $a :: \text{nat}$
assumes $\text{gcd } a \ n \ \text{dvd } d$
shows $\exists x. [a * x = d] \ (\text{mod } n)$
 $\langle \text{proof} \rangle$

lemma *cong-solve-dvd-int*:
fixes $a :: \text{int}$
assumes $b: \text{gcd } a \ n \ \text{dvd } d$
shows $\exists x. [a * x = d] \ (\text{mod } n)$
 $\langle \text{proof} \rangle$

lemma *cong-solve-coprime-nat*:
 $\exists x. [a * x = \text{Suc } 0] \ (\text{mod } n)$ **if** *coprime* $a \ n$
 $\langle \text{proof} \rangle$

lemma *cong-solve-coprime-int*:
 $\exists x. [a * x = 1] \ (\text{mod } n)$ **if** *coprime* $a \ n$ **for** $a \ n \ x :: \text{int}$
 $\langle \text{proof} \rangle$

lemma *coprime-iff-invertible-nat*:
 $\text{coprime } a \ m \longleftrightarrow (\exists x. [a * x = \text{Suc } 0] \ (\text{mod } m))$ **(is** $?P \longleftrightarrow ?Q$ **)**
 $\langle \text{proof} \rangle$

lemma *coprime-iff-invertible-int*:
 $\text{coprime } a \ m \longleftrightarrow (\exists x. [a * x = 1] \ (\text{mod } m))$ **(is** $?P \longleftrightarrow ?Q$ **) for** $m :: \text{int}$
 $\langle \text{proof} \rangle$

lemma *coprime-iff-invertible'-nat*:
assumes $m > 0$
shows $\text{coprime } a \ m \longleftrightarrow (\exists x. 0 \leq x \wedge x < m \wedge [a * x = \text{Suc } 0] \ (\text{mod } m))$
 $\langle \text{proof} \rangle$

lemma *coprime-iff-invertible'-int*:
fixes $m :: \text{int}$
assumes $m > 0$
shows $\text{coprime } a \ m \longleftrightarrow (\exists x. 0 \leq x \wedge x < m \wedge [a * x = 1] \ (\text{mod } m))$
 $\langle \text{proof} \rangle$

lemma *cong-cong-lcm-nat*: $[x = y] \ (\text{mod } a) \implies [x = y] \ (\text{mod } b) \implies [x = y] \ (\text{mod } \text{lcm } a \ b)$
for $x \ y :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *cong-cong-lcm-int*: $[x = y] \ (\text{mod } a) \implies [x = y] \ (\text{mod } b) \implies [x = y] \ (\text{mod } \text{lcm } a \ b)$

```

for  $x\ y :: \text{int}$ 
   $\langle \text{proof} \rangle$ 

lemma cong-cong-prod-coprime-nat:
   $[x = y] \ (\text{mod } (\prod_{i \in A} m\ i))$  if
     $(\forall i \in A. [x = y] \ (\text{mod } m\ i))$ 
     $(\forall i \in A. (\forall j \in A. i \neq j \longrightarrow \text{coprime } (m\ i) (m\ j)))$ 
  for  $x\ y :: \text{nat}$ 
   $\langle \text{proof} \rangle$ 

lemma binary-chinese-remainder-nat:
  fixes  $m1\ m2 :: \text{nat}$ 
  assumes  $a: \text{coprime } m1\ m2$ 
  shows  $\exists x. [x = u1] \ (\text{mod } m1) \wedge [x = u2] \ (\text{mod } m2)$ 
   $\langle \text{proof} \rangle$ 

lemma binary-chinese-remainder-int:
  fixes  $m1\ m2 :: \text{int}$ 
  assumes  $a: \text{coprime } m1\ m2$ 
  shows  $\exists x. [x = u1] \ (\text{mod } m1) \wedge [x = u2] \ (\text{mod } m2)$ 
   $\langle \text{proof} \rangle$ 

lemma cong-modulus-mult-nat:  $[x = y] \ (\text{mod } m * n) \Longrightarrow [x = y] \ (\text{mod } m)$ 
  for  $x\ y :: \text{nat}$ 
   $\langle \text{proof} \rangle$ 

lemma cong-less-modulus-unique-nat:  $[x = y] \ (\text{mod } m) \Longrightarrow x < m \Longrightarrow y < m \Longrightarrow$ 
 $x = y$ 
  for  $x\ y :: \text{nat}$ 
   $\langle \text{proof} \rangle$ 

lemma binary-chinese-remainder-unique-nat:
  fixes  $m1\ m2 :: \text{nat}$ 
  assumes  $a: \text{coprime } m1\ m2$ 
  and  $nz: m1 \neq 0\ m2 \neq 0$ 
  shows  $\exists! x. x < m1 * m2 \wedge [x = u1] \ (\text{mod } m1) \wedge [x = u2] \ (\text{mod } m2)$ 
   $\langle \text{proof} \rangle$ 

lemma chinese-remainder-nat:
  fixes  $A :: 'a\ \text{set}$ 
  and  $m :: 'a \Rightarrow \text{nat}$ 
  and  $u :: 'a \Rightarrow \text{nat}$ 
  assumes  $\text{fin: finite } A$ 
  and  $\text{cop: } \forall i \in A. \forall j \in A. i \neq j \longrightarrow \text{coprime } (m\ i) (m\ j)$ 
  shows  $\exists x. \forall i \in A. [x = u\ i] \ (\text{mod } m\ i)$ 
   $\langle \text{proof} \rangle$ 

lemma coprime-cong-prod-nat:  $[x = y] \ (\text{mod } (\prod_{i \in A} m\ i))$ 
  if  $\bigwedge i\ j. [i \in A; j \in A; i \neq j] \Longrightarrow \text{coprime } (m\ i) (m\ j)$ 

```


and $\bigwedge i. i \in A \implies [x = y] \text{ (mod } m \text{ } i)$ **for** $x \ y :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *chinese-remainder-unique-nat*:
fixes $A :: 'a \text{ set}$
and $m :: 'a \Rightarrow \text{nat}$
and $u :: 'a \Rightarrow \text{nat}$
assumes *fin*: $\text{finite } A$
and *nz*: $\forall i \in A. m \ i \neq 0$
and *cop*: $\forall i \in A. \forall j \in A. i \neq j \longrightarrow \text{coprime } (m \ i) \ (m \ j)$
shows $\exists! x. x < (\prod i \in A. m \ i) \wedge (\forall i \in A. [x = u \ i] \text{ (mod } m \ i))$
 $\langle \text{proof} \rangle$

lemma (*in semiring-1-cancel*) *of-nat-eq-iff-cong-CHAR*:
 $\text{of-nat } x = (\text{of-nat } y :: 'a) \longleftrightarrow [x = y] \text{ (mod } \text{CHAR}('a))$
 $\langle \text{proof} \rangle$

lemma (*in ring-1*) *of-int-eq-iff-cong-CHAR*:
 $\text{of-int } x = (\text{of-int } y :: 'a) \longleftrightarrow [x = y] \text{ (mod int } \text{CHAR}('a))$
 $\langle \text{proof} \rangle$

Thanks to Manuel Eberl

lemma *prime-cong-4-nat-cases* [*consumes 1, case-names 2 cong-1 cong-3*]:
assumes *prime* ($p :: \text{nat}$)
obtains $p = 2 \mid [p = 1] \text{ (mod } 4) \mid [p = 3] \text{ (mod } 4)$
 $\langle \text{proof} \rangle$

end

3 Fundamental facts about Euler's totient function

theory *Totient*
imports
Complex-Main
HOL-Computational-Algebra.Primes
Cong
begin

definition *totatives* $:: \text{nat} \Rightarrow \text{nat set}$ **where**
 $\text{totatives } n = \{k \in \{0 <..n\}. \text{coprime } k \ n\}$

lemma *in-totatives-iff*: $k \in \text{totatives } n \longleftrightarrow k > 0 \wedge k \leq n \wedge \text{coprime } k \ n$
 $\langle \text{proof} \rangle$

lemma *totatives-code* [*code*]: $\text{totatives } n = \text{Set.filter } (\lambda k. \text{coprime } k \ n) \ \{0 <..n\}$
 $\langle \text{proof} \rangle$

lemma *finite-totatives* [simp]: *finite (totatives n)*
 ⟨proof⟩

lemma *totatives-subset*: *totatives n* \subseteq $\{0 <..n\}$
 ⟨proof⟩

lemma *zero-not-in-totatives* [simp]: $0 \notin \text{totatives } n$
 ⟨proof⟩

lemma *totatives-le*: $x \in \text{totatives } n \implies x \leq n$
 ⟨proof⟩

lemma *totatives-less*:
 assumes $x \in \text{totatives } n$ $n > 1$
 shows $x < n$
 ⟨proof⟩

lemma *totatives-0* [simp]: *totatives 0* = {}
 ⟨proof⟩

lemma *totatives-1* [simp]: *totatives 1* = {Suc 0}
 ⟨proof⟩

lemma *totatives-Suc-0* [simp]: *totatives (Suc 0)* = {Suc 0}
 ⟨proof⟩

lemma *one-in-totatives* [simp]: $n > 0 \implies \text{Suc } 0 \in \text{totatives } n$
 ⟨proof⟩

lemma *totatives-eq-empty-iff* [simp]: *totatives n* = {} $\longleftrightarrow n = 0$
 ⟨proof⟩

lemma *minus-one-in-totatives*:
 assumes $n \geq 2$
 shows $n - 1 \in \text{totatives } n$
 ⟨proof⟩

lemma *power-in-totatives*:
 assumes $m > 1$ *coprime m g*
 shows $g^i \bmod m \in \text{totatives } m$
 ⟨proof⟩

lemma *totatives-prime-power-Suc*:
 assumes *prime p*
 shows $\text{totatives } (p^i \wedge \text{Suc } n) = \{0 <..p^i \wedge \text{Suc } n\} - (\lambda m. p * m) ` \{0 <..p^i \wedge n\}$
 ⟨proof⟩

lemma *totatives-prime*: *prime p* $\implies \text{totatives } p = \{0 <..<p\}$
 ⟨proof⟩

lemma *bij-betw-totatives*:

assumes $m1 > 1$ $m2 > 1$ *coprime* $m1$ $m2$

shows $bij\text{-}betw\ (\lambda x. (x \bmod m1, x \bmod m2))\ (totatives\ (m1 * m2))$
 $(totatives\ m1 \times totatives\ m2)$

$\langle proof \rangle$

lemma *bij-betw-totatives-gcd-eq*:

fixes $n\ d :: nat$

assumes $d\ dvd\ n$ $n > 0$

shows $bij\text{-}betw\ (\lambda k. k * d)\ (totatives\ (n\ div\ d))\ \{k \in \{0 <..n\}. gcd\ k\ n = d\}$

$\langle proof \rangle$

definition *totient* $:: nat \Rightarrow nat$ **where**

$totient\ n = card\ (totatives\ n)$

primrec *totient-naive* $:: nat \Rightarrow nat \Rightarrow nat \Rightarrow nat$ **where**

$totient\text{-}naive\ 0\ acc\ n = acc$

| $totient\text{-}naive\ (Suc\ k)\ acc\ n =$

$(if\ coprime\ (Suc\ k)\ n\ then\ totient\text{-}naive\ k\ (acc + 1)\ n\ else\ totient\text{-}naive\ k\ acc$
 $n)$

lemma *totient-naive*:

$totient\text{-}naive\ k\ acc\ n = card\ \{x \in \{0 <..k\}. coprime\ x\ n\} + acc$
 $\langle proof \rangle$

lemma *totient-code-naive* [code]: $totient\ n = totient\text{-}naive\ n\ 0\ n$

$\langle proof \rangle$

lemma *totient-le*: $totient\ n \leq n$

$\langle proof \rangle$

lemma *totient-less*:

assumes $n > 1$

shows $totient\ n < n$

$\langle proof \rangle$

lemma *totient-0* [simp]: $totient\ 0 = 0$

$\langle proof \rangle$

lemma *totient-Suc-0* [simp]: $totient\ (Suc\ 0) = Suc\ 0$

$\langle proof \rangle$

lemma *totient-1* [simp]: $totient\ 1 = Suc\ 0$

$\langle proof \rangle$

lemma *totient-0-iff* [simp]: $totient\ n = 0 \longleftrightarrow n = 0$

$\langle proof \rangle$

lemma *totient-gt-0-iff* [simp]: $\text{totient } n > 0 \longleftrightarrow n > 0$
 <proof>

lemma *totient-gt-1*:
 assumes $n > 2$
 shows $\text{totient } n > 1$
 <proof>

lemma *card-gcd-eq-totient*:
 $n > 0 \implies d \text{ dvd } n \implies \text{card } \{k \in \{0 <..n\}. \text{gcd } k \ n = d\} = \text{totient } (n \text{ div } d)$
 <proof>

lemma *totient-divisor-sum*: $(\sum d \mid d \text{ dvd } n. \text{totient } d) = n$
 <proof>

lemma *totient-mult-coprime*:
 assumes *coprime* $m \ n$
 shows $\text{totient } (m * n) = \text{totient } m * \text{totient } n$
 <proof>

lemma *totient-prime-power-Suc*:
 assumes *prime* p
 shows $\text{totient } (p \wedge \text{Suc } n) = p \wedge n * (p - 1)$
 <proof>

lemma *totient-prime-power*:
 assumes *prime* $p \ n > 0$
 shows $\text{totient } (p \wedge n) = p \wedge (n - 1) * (p - 1)$
 <proof>

lemma *totient-imp-prime*:
 assumes $\text{totient } p = p - 1 \ p > 0$
 shows *prime* p
 <proof>

lemma *totient-prime*:
 assumes *prime* p
 shows $\text{totient } p = p - 1$
 <proof>

lemma *totient-2* [simp]: $\text{totient } 2 = 1$
and *totient-3* [simp]: $\text{totient } 3 = 2$
and *totient-5* [simp]: $\text{totient } 5 = 4$
and *totient-7* [simp]: $\text{totient } 7 = 6$
 <proof>

lemma *totient-4* [simp]: $\text{totient } 4 = 2$
and *totient-8* [simp]: $\text{totient } 8 = 4$
and *totient-9* [simp]: $\text{totient } 9 = 6$

$\langle \text{proof} \rangle$

lemma *totient-6* [*simp*]: $\text{totient } 6 = 2$
 $\langle \text{proof} \rangle$

lemma *totient-even*:
 assumes $n > 2$
 shows $\text{even } (\text{totient } n)$
 $\langle \text{proof} \rangle$

lemma *totient-prod-coprime*:
 assumes $\text{pairwise coprime } (f \text{ ` } A) \text{ inj-on } f \ A$
 shows $\text{totient } (\text{prod } f \ A) = (\prod_{a \in A. \text{totient } (f \ a)})$
 $\langle \text{proof} \rangle$

lemma *prime-power-eq-imp-eq*:
 fixes $p \ q :: 'a :: \text{factorial-semiring}$
 assumes $\text{prime } p \ \text{prime } q \ m > 0$
 assumes $p \wedge^m = q \wedge^n$
 shows $p = q$
 $\langle \text{proof} \rangle$

lemma *totient-formula1*:
 assumes $n > 0$
 shows $\text{totient } n = (\prod_{p \in \text{prime-factors } n. p \wedge^{(\text{multiplicity } p \ n - 1) * (p - 1)}}$
 $\langle \text{proof} \rangle$

lemma *totient-dvd*:
 assumes $m \ \text{dvd } n$
 shows $\text{totient } m \ \text{dvd } \text{totient } n$
 $\langle \text{proof} \rangle$

lemma *totient-dvd-mono*:
 assumes $m \ \text{dvd } n \ n > 0$
 shows $\text{totient } m \leq \text{totient } n$
 $\langle \text{proof} \rangle$

lemma *prime-factors-power*: $n > 0 \implies \text{prime-factors } (x \wedge^n) = \text{prime-factors } x$
 $\langle \text{proof} \rangle$

lemma *totient-formula2*:
 $\text{real } (\text{totient } n) = \text{real } n * (\prod_{p \in \text{prime-factors } n. 1 - 1 / \text{real } p})$
 $\langle \text{proof} \rangle$

lemma *totient-gcd*: $\text{totient } (a * b) * \text{totient } (\text{gcd } a \ b) = \text{totient } a * \text{totient } b * \text{gcd } a \ b$
 $\langle \text{proof} \rangle$

lemma *totient-mult*: $\text{totient } (a * b) = \text{totient } a * \text{totient } b * \text{gcd } a \ b \ \text{div } \text{totient } (\text{gcd } a \ b)$
 ⟨proof⟩

lemma *of-nat-eq-1-iff*: $\text{of-nat } x = (1 :: 'a :: \{\text{semiring-1}, \text{semiring-char-0}\}) \longleftrightarrow x = 1$
 ⟨proof⟩

lemma *odd-imp-coprime-nat*:
 assumes *odd* ($n :: \text{nat}$)
 shows *coprime* $n \ 2$
 ⟨proof⟩

lemma *totient-double*: $\text{totient } (2 * n) = (\text{if even } n \text{ then } 2 * \text{totient } n \text{ else } \text{totient } n)$
 ⟨proof⟩

lemma *totient-power-Suc*: $\text{totient } (n \wedge \text{Suc } m) = n \wedge m * \text{totient } n$
 ⟨proof⟩

lemma *totient-power*: $m > 0 \implies \text{totient } (n \wedge m) = n \wedge (m - 1) * \text{totient } n$
 ⟨proof⟩

lemma *totient-gcd-lcm*: $\text{totient } (\text{gcd } a \ b) * \text{totient } (\text{lcm } a \ b) = \text{totient } a * \text{totient } b$
 ⟨proof⟩

end

4 Residue rings

theory *Residues*

imports

Cong

HOL-Algebra.Multiplicative-Group

Totient

begin

lemma (*in ring-1*) *CHAR-dvd-CARD*: $\text{CHAR}('a) \ \text{dvd} \ \text{card } (\text{UNIV} :: 'a \ \text{set})$
 ⟨proof⟩

definition *QuadRes* :: $\text{int} \Rightarrow \text{int} \Rightarrow \text{bool}$
 where $\text{QuadRes } p \ a = (\exists y. ([y^2 = a] \ (\text{mod } p)))$

definition *Legendre* :: $\text{int} \Rightarrow \text{int} \Rightarrow \text{int}$
 where $\text{Legendre } a \ p =$
 (*if* $([a = 0] \ (\text{mod } p))$ *then* 0
 else if $\text{QuadRes } p \ a$ *then* 1

else -1)

4.1 A locale for residue rings

definition *residue-ring* :: *int* \Rightarrow *int ring*
where
residue-ring *m* =
 ($\text{carrier} = \{0..m - 1\}$,
 $\text{monoid.mult} = \lambda x y. (x * y) \text{ mod } m$,
 $\text{one} = 1$,
 $\text{zero} = 0$,
 $\text{add} = \lambda x y. (x + y) \text{ mod } m$)

locale *residues* =
fixes *m* :: *int* **and** *R* (**structure**)
assumes *m-gt-one*: $m > 1$
defines *R-m-def*: $R \equiv \text{residue-ring } m$
begin

lemma *abelian-group*: *abelian-group* *R*
 $\langle \text{proof} \rangle$

lemma *comm-monoid*: *comm-monoid* *R*
 $\langle \text{proof} \rangle$

interpretation *comm-monoid* *R*
 $\langle \text{proof} \rangle$

lemma *cring*: *cring* *R*
 $\langle \text{proof} \rangle$

end

sublocale *residues* < *cring*
 $\langle \text{proof} \rangle$

context *residues*
begin

These lemmas translate back and forth between internal and external concepts.

lemma *res-carrier-eq*: $\text{carrier } R = \{0..m - 1\}$
 $\langle \text{proof} \rangle$

lemma *res-add-eq*: $x \oplus y = (x + y) \text{ mod } m$
 $\langle \text{proof} \rangle$

lemma *res-mult-eq*: $x \otimes y = (x * y) \text{ mod } m$

$\langle proof \rangle$

lemma *res-zero-eq*: $\mathbf{0} = 0$

$\langle proof \rangle$

lemma *res-one-eq*: $\mathbf{1} = 1$

$\langle proof \rangle$

lemma *res-units-eq*: $Units\ R = \{x. 0 < x \wedge x < m \wedge coprime\ x\ m\}$ (**is** - = ?*rhs*)

$\langle proof \rangle$

lemma *res-neg-eq*: $\ominus x = (-\ x)\ mod\ m$

$\langle proof \rangle$

lemma *finite [iff]*: $finite\ (carrier\ R)$

$\langle proof \rangle$

lemma *finite-Units [iff]*: $finite\ (Units\ R)$

$\langle proof \rangle$

The function $a \mapsto a\ mod\ m$ maps the integers to the residue classes. The following lemmas show that this mapping respects addition and multiplication on the integers.

lemma *mod-in-carrier [iff]*: $a\ mod\ m \in carrier\ R$

$\langle proof \rangle$

lemma *add-cong*: $(x\ mod\ m) \oplus (y\ mod\ m) = (x + y)\ mod\ m$

$\langle proof \rangle$

lemma *mult-cong*: $(x\ mod\ m) \otimes (y\ mod\ m) = (x * y)\ mod\ m$

$\langle proof \rangle$

lemma *zero-cong*: $\mathbf{0} = 0$

$\langle proof \rangle$

lemma *one-cong*: $\mathbf{1} = 1\ mod\ m$

$\langle proof \rangle$

lemma *pow-cong*: $(x\ mod\ m) [\wedge] n = x^{\wedge}n\ mod\ m$

$\langle proof \rangle$

lemma *neg-cong*: $\ominus (x\ mod\ m) = (-\ x)\ mod\ m$

$\langle proof \rangle$

lemma (**in residues**) *prod-cong*: $finite\ A \implies (\bigotimes_{i \in A}. (f\ i)\ mod\ m) = (\prod_{i \in A}. f\ i)\ mod\ m$

$\langle proof \rangle$

lemma (in *residues*) *sum-cong*: $\text{finite } A \implies (\bigoplus_{i \in A} (f \ i) \bmod m) = (\sum_{i \in A} f \ i) \bmod m$
 <proof>

lemma *mod-in-res-units* [*simp*]:
 assumes $1 < m$ and *coprime* $a \ m$
 shows $a \bmod m \in \text{Units } R$
 <proof>

lemma *res-eq-to-cong*: $(a \bmod m) = (b \bmod m) \longleftrightarrow [a = b] \bmod m$
 <proof>

Simplifying with these will translate a ring equation in R to a congruence.

lemmas *res-to-cong-simps* =
add-cong mult-cong pow-cong one-cong
prod-cong sum-cong neg-cong res-eq-to-cong

Other useful facts about the residue ring.

lemma *one-eq-neg-one*: $1 = \ominus 1 \implies m = 2$
 <proof>

end

4.2 Prime residues

locale *residues-prime* =
 fixes $p :: \text{nat}$ and R (**structure**)
 assumes *p-prime* [*intro*]: *prime* p
 defines $R \equiv \text{residue-ring } (\text{int } p)$

sublocale *residues-prime* < *residues* p
 <proof>

context *residues-prime*
begin

lemma *p-coprime-left*:
coprime $p \ a \longleftrightarrow \neg p \ \text{dvd } a$
 <proof>

lemma *p-coprime-right*:
coprime $a \ p \longleftrightarrow \neg p \ \text{dvd } a$
 <proof>

lemma *p-coprime-left-int*:
coprime $(\text{int } p) \ a \longleftrightarrow \neg \text{int } p \ \text{dvd } a$
 <proof>

lemma *p-coprime-right-int*:

coprime a (int p) $\longleftrightarrow \neg \text{int } p \text{ dvd } a$
 $\langle \text{proof} \rangle$

lemma *is-field*: *field R*
 $\langle \text{proof} \rangle$

lemma *res-prime-units-eq*: *Units R = {1..p - 1}*
 $\langle \text{proof} \rangle$

end

sublocale *residues-prime* < *field*
 $\langle \text{proof} \rangle$

5 Test cases: Euler's theorem and Wilson's theorem

5.1 Euler's theorem

lemma (*in residues*) *totatives-eq*:
totatives (nat m) = nat ' Units R
 $\langle \text{proof} \rangle$

lemma (*in residues*) *totient-eq*:
totient (nat m) = card (Units R)
 $\langle \text{proof} \rangle$

lemma (*in residues-prime*) *prime-totient-eq*: *totient p = p - 1*
 $\langle \text{proof} \rangle$

lemma (*in residues*) *euler-theorem*:
assumes *coprime a m*
shows $[a^{\wedge} \text{totient (nat m)} = 1] \text{ (mod m)}$
 $\langle \text{proof} \rangle$

lemma *euler-theorem*:
fixes *a m :: nat*
assumes *coprime a m*
shows $[a^{\wedge} \text{totient m} = 1] \text{ (mod m)}$
 $\langle \text{proof} \rangle$

lemma *fermat-theorem*:
fixes *p a :: nat*
assumes *prime p and $\neg p \text{ dvd } a$*
shows $[a^{\wedge} (p - 1) = 1] \text{ (mod p)}$
 $\langle \text{proof} \rangle$

5.2 Wilson's theorem

lemma (in *field*) *inv-pair-lemma*: $x \in \text{Units } R \implies y \in \text{Units } R \implies \{x, \text{inv } x\} \neq \{y, \text{inv } y\} \implies \{x, \text{inv } x\} \cap \{y, \text{inv } y\} = \{\}$
 <proof>

lemma (in *residues-prime*) *wilson-theorem1*:
 assumes $a: p > 2$
 shows $[fact (p - 1) = (-1::int)] \pmod p$
 <proof>

lemma *wilson-theorem*:
 assumes *prime* p
 shows $[fact (p - 1) = - 1] \pmod p$
 <proof>

This result can be transferred to the multiplicative group of $\mathbb{Z}/p\mathbb{Z}$ for p prime.

lemma *mod-nat-int-pow-eq*:
 fixes $n :: nat$ and $p a :: int$
 shows $a \geq 0 \implies p \geq 0 \implies (nat a ^ n) \pmod (nat p) = nat ((a ^ n) \pmod p)$
 <proof>

theorem *residue-prime-mult-group-has-gen*:
 fixes $p :: nat$
 assumes *prime-p* : *prime* p
 shows $\exists a \in \{1 .. p - 1\}. \{1 .. p - 1\} = \{a^i \pmod p | i . i \in UNIV\}$
 <proof>

5.3 Upper bound for the number of n -th roots

lemma *roots-mod-prime-bound*:
 fixes $n c p :: nat$
 assumes *prime* p $n > 0$
 defines $A \equiv \{x \in \{..<p\}. [x ^ n = c] \pmod p\}$
 shows $\text{card } A \leq n$
 <proof>

end

6 The sieve of Eratosthenes

theory *Eratosthenes*
 imports *Main HOL-Computational-Algebra.Primes*
 begin

6.1 Preliminary: strict divisibility

context *dvd*

begin

abbreviation *dvd-strict* :: 'a \Rightarrow 'a \Rightarrow bool (**infixl** \langle dvd'-strict \rangle 50)

where

b dvd-strict a \equiv *b dvd a* \wedge \neg *a dvd b*

end

6.2 Main corpus

The sieve is modelled as a list of booleans, where *False* means *marked out*.

type-synonym *marks* = bool list

definition *numbers-of-marks* :: nat \Rightarrow marks \Rightarrow nat set

where

numbers-of-marks n bs = fst ' {*x* \in set (enumerate *n bs*). snd *x*}

lemma *numbers-of-marks-simps* [simp, code]:

numbers-of-marks n [] = {}

numbers-of-marks n (True # bs) = insert *n* (*numbers-of-marks* (Suc *n*) *bs*)

numbers-of-marks n (False # bs) = *numbers-of-marks* (Suc *n*) *bs*

\langle proof \rangle

lemma *numbers-of-marks-Suc*:

numbers-of-marks (Suc *n*) *bs* = Suc ' *numbers-of-marks n bs*

\langle proof \rangle

lemma *numbers-of-marks-replicate-False* [simp]:

numbers-of-marks n (replicate *m* False) = {}

\langle proof \rangle

lemma *numbers-of-marks-replicate-True* [simp]:

numbers-of-marks n (replicate *m* True) = {*n*..*n+m*}

\langle proof \rangle

lemma *in-numbers-of-marks-eq*:

m \in *numbers-of-marks n bs* \longleftrightarrow *m* \in {*n*..*n + length bs*} \wedge *bs* ! (*m* - *n*)

\langle proof \rangle

lemma *sorted-list-of-set-numbers-of-marks*:

sorted-list-of-set (*numbers-of-marks n bs*) = map fst (filter snd (enumerate *n bs*))

\langle proof \rangle

Marking out multiples in a sieve

definition *mark-out* :: nat \Rightarrow marks \Rightarrow marks

where

$mark-out\ n\ bs = map\ (\lambda(q, b). b \wedge \neg\ Suc\ n\ dvd\ Suc\ (Suc\ q))\ (enumerate\ n\ bs)$

lemma *mark-out-Nil* [simp]: $mark-out\ n\ [] = []$
 ⟨proof⟩

lemma *length-mark-out* [simp]: $length\ (mark-out\ n\ bs) = length\ bs$
 ⟨proof⟩

lemma *numbers-of-marks-mark-out*:
 $numbers-of-marks\ n\ (mark-out\ m\ bs) = \{q \in numbers-of-marks\ n\ bs. \neg\ Suc\ m\ dvd\ Suc\ q - n\}$
 ⟨proof⟩

Auxiliary operation for efficient implementation

definition *mark-out-aux* :: $nat \Rightarrow nat \Rightarrow marks \Rightarrow marks$
where

$mark-out-aux\ n\ m\ bs =$
 $map\ (\lambda(q, b). b \wedge (q < m + n \vee \neg\ Suc\ n\ dvd\ Suc\ (Suc\ q) + (n - m\ mod\ Suc\ n)))\ (enumerate\ n\ bs)$

lemma *mark-out-code* [code]: $mark-out\ n\ bs = mark-out-aux\ n\ n\ bs$
 ⟨proof⟩

lemma *mark-out-aux-simps* [simp, code]:
 $mark-out-aux\ n\ m\ [] = []$
 $mark-out-aux\ n\ 0\ (b \# bs) = False \# mark-out-aux\ n\ n\ bs$
 $mark-out-aux\ n\ (Suc\ m)\ (b \# bs) = b \# mark-out-aux\ n\ m\ bs$
 ⟨proof⟩

Main entry point to sieve

fun *sieve* :: $nat \Rightarrow marks \Rightarrow marks$
where

$sieve\ n\ [] = []$
 $| sieve\ n\ (False \# bs) = False \# sieve\ (Suc\ n)\ bs$
 $| sieve\ n\ (True \# bs) = True \# sieve\ (Suc\ n)\ (mark-out\ n\ bs)$

There are the following possible optimisations here:

- *sieve* can abort as soon as n is too big to let *mark-out* have any effect.
- Search for further primes can be given up as soon as the search position exceeds the square root of the maximum candidate.

This is left as an constructive exercise to the reader.

lemma *numbers-of-marks-sieve*:
 $numbers-of-marks\ (Suc\ n)\ (sieve\ n\ bs) =$
 $\{q \in numbers-of-marks\ (Suc\ n)\ bs. \forall m \in numbers-of-marks\ (Suc\ n)\ bs. \neg\ m\ dvd\ strict\ q\}$

$\langle \text{proof} \rangle$

Relation of the sieve algorithm to actual primes

definition *primes-upto* :: $\text{nat} \Rightarrow \text{nat list}$

where

primes-upto $n = \text{sorted-list-of-set } \{m. m \leq n \wedge \text{prime } m\}$

lemma *set-primes-upto*: $\text{set } (\text{primes-upto } n) = \{m. m \leq n \wedge \text{prime } m\}$

$\langle \text{proof} \rangle$

lemma *sorted-primes-upto* [iff]: $\text{sorted } (\text{primes-upto } n)$

$\langle \text{proof} \rangle$

lemma *distinct-primes-upto* [iff]: $\text{distinct } (\text{primes-upto } n)$

$\langle \text{proof} \rangle$

lemma *set-primes-upto-sieve*:

$\text{set } (\text{primes-upto } n) = \text{numbers-of-marks } 2 \text{ (sieve } 1 \text{ (replicate } (n - 1) \text{ True))}$

$\langle \text{proof} \rangle$

lemma *primes-upto-sieve* [code]:

$\text{primes-upto } n = \text{map fst } (\text{filter snd } (\text{enumerate } 2 \text{ (sieve } 1 \text{ (replicate } (n - 1) \text{ True))}))$

$\langle \text{proof} \rangle$

lemma *prime-in-primes-upto*: $\text{prime } n \longleftrightarrow n \in \text{set } (\text{primes-upto } n)$

$\langle \text{proof} \rangle$

6.3 Application: smallest prime beyond a certain number

definition *smallest-prime-beyond* :: $\text{nat} \Rightarrow \text{nat}$

where

smallest-prime-beyond $n = (\text{LEAST } p. \text{prime } p \wedge p \geq n)$

lemma *prime-smallest-prime-beyond* [iff]: $\text{prime } (\text{smallest-prime-beyond } n) \text{ (is ?P)}$

and *smallest-prime-beyond-le* [iff]: $\text{smallest-prime-beyond } n \geq n \text{ (is ?Q)}$

$\langle \text{proof} \rangle$

lemma *smallest-prime-beyond-smallest*: $\text{prime } p \implies p \geq n \implies \text{smallest-prime-beyond } n \leq p$

$\langle \text{proof} \rangle$

lemma *smallest-prime-beyond-eq*:

$\text{prime } p \implies p \geq n \implies (\bigwedge q. \text{prime } q \implies q \geq n \implies q \geq p) \implies \text{smallest-prime-beyond } n = p$

$\langle \text{proof} \rangle$

definition *smallest-prime-between* :: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat option}$

where

smallest-prime-between $m\ n =$
 (if $(\exists p. \text{prime } p \wedge m \leq p \wedge p \leq n)$ then *Some* (*smallest-prime-beyond* m) else *None*)

lemma *smallest-prime-between-None*:

smallest-prime-between $m\ n = \text{None} \longleftrightarrow (\forall q. m \leq q \wedge q \leq n \longrightarrow \neg \text{prime } q)$
 $\langle \text{proof} \rangle$

lemma *smallest-prime-between-Some*:

smallest-prime-between $m\ n = \text{Some } p \longleftrightarrow \text{smallest-prime-beyond } m = p \wedge p \leq n$
 $\langle \text{proof} \rangle$

lemma [code]: *smallest-prime-between* $m\ n = \text{List.find } (\lambda p. p \geq m) (\text{primes-upto } n)$
 $\langle \text{proof} \rangle$

definition *smallest-prime-beyond-aux* :: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat}$

where

smallest-prime-beyond-aux $k\ n = \text{smallest-prime-beyond } n$

lemma [code]:

smallest-prime-beyond-aux $k\ n =$
 (case *smallest-prime-between* $n\ (k * n)$ of
 Some $p \Rightarrow p$
 | *None* $\Rightarrow \text{smallest-prime-beyond-aux } (\text{Suc } k)\ n$)
 $\langle \text{proof} \rangle$

lemma [code]: *smallest-prime-beyond* $n = \text{smallest-prime-beyond-aux } 2\ n$
 $\langle \text{proof} \rangle$

end

7 Fast modular exponentiation

theory *Mod-Exp*

imports *Cong HOL-Library.Power-By-Squaring*

begin

context *euclidean-semiring-cancel*

begin

definition *mod-exp-aux* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow \text{nat} \Rightarrow 'a$

where *mod-exp-aux* $m = \text{efficient-funpow } (\lambda x\ y. x * y \text{ mod } m)$

lemma *mod-exp-aux-code* [code]:

mod-exp-aux $m\ y\ x\ n =$
 (if $n = 0$ then y

else if $n = 1$ then $(x * y) \bmod m$
 else if even n then $\text{mod-exp-aux } m \ y \ ((x * x) \bmod m) \ (n \text{ div } 2)$
 else $\text{mod-exp-aux } m \ ((x * y) \bmod m) \ ((x * x) \bmod m) \ (n \text{ div } 2)$
 $\langle \text{proof} \rangle$

lemma *mod-exp-aux-correct*:
 $\text{mod-exp-aux } m \ y \ x \ n \bmod m = (x \wedge^n * y) \bmod m$
 $\langle \text{proof} \rangle$

definition *mod-exp* :: $'a \Rightarrow \text{nat} \Rightarrow 'a \Rightarrow 'a$
where $\text{mod-exp } b \ e \ m = (b \wedge^e) \bmod m$

lemma *mod-exp-code* [code]: $\text{mod-exp } b \ e \ m = \text{mod-exp-aux } m \ 1 \ b \ e \bmod m$
 $\langle \text{proof} \rangle$

end

lemmas [code-abbrev] = *mod-exp-def*[**where** $?a = \text{nat}$] *mod-exp-def*[**where** $?a = \text{int}$]

lemma *cong-power-nat-code* [code-unfold]:
 $[b \wedge^e = (x :: \text{nat})] \ (\bmod m) \longleftrightarrow \text{mod-exp } b \ e \ m = x \bmod m$
 $\langle \text{proof} \rangle$

lemma *cong-power-int-code* [code-unfold]:
 $[b \wedge^e = (x :: \text{int})] \ (\bmod m) \longleftrightarrow \text{mod-exp } b \ e \ m = x \bmod m$
 $\langle \text{proof} \rangle$

The following rules allow the simplifier to evaluate *mod-exp* efficiently.

lemma *eval-mod-exp-aux* [simp]:
 $\text{mod-exp-aux } m \ y \ x \ 0 = y$
 $\text{mod-exp-aux } m \ y \ x \ (\text{Suc } 0) = (x * y) \bmod m$
 $\text{mod-exp-aux } m \ y \ x \ (\text{numeral } (\text{num.Bit0 } n)) =$
 $\text{mod-exp-aux } m \ y \ (x^2 \bmod m) \ (\text{numeral } n)$
 $\text{mod-exp-aux } m \ y \ x \ (\text{numeral } (\text{num.Bit1 } n)) =$
 $\text{mod-exp-aux } m \ ((x * y) \bmod m) \ (x^2 \bmod m) \ (\text{numeral } n)$
 $\langle \text{proof} \rangle$

lemma *eval-mod-exp* [simp]:
 $\text{mod-exp } b' \ 0 \ m' = 1 \bmod m'$
 $\text{mod-exp } b' \ 1 \ m' = b' \bmod m'$
 $\text{mod-exp } b' \ (\text{Suc } 0) \ m' = b' \bmod m'$
 $\text{mod-exp } b' \ e' \ 0 = b' \wedge^{e'}$
 $\text{mod-exp } b' \ e' \ 1 = 0$
 $\text{mod-exp } b' \ e' \ (\text{Suc } 0) = 0$
 $\text{mod-exp } 0 \ 1 \ m' = 0$
 $\text{mod-exp } 0 \ (\text{Suc } 0) \ m' = 0$
 $\text{mod-exp } 0 \ (\text{numeral } e) \ m' = 0$


```

    mod-exp 1 e' m' = 1 mod m'
    mod-exp (Suc 0) e' m' = 1 mod m'
    mod-exp (numeral b) (numeral e) (numeral m) =
      mod-exp-aux (numeral m) 1 (numeral b) (numeral e) mod numeral m
  <proof>

```

end

```

theory Euler-Criterion
imports Residues
begin

```

context

```

  fixes p :: nat
  fixes a :: int

```

```

  assumes p-prime: prime p
  assumes p-ge-2: 2 < p
  assumes p-a-relprime: [a ≠ 0](mod p)
begin

```

```

private lemma odd-p: odd p
  <proof> lemma p-minus-1-int:
    int (p - 1) = int p - 1
  <proof> lemma p-not-eq-Suc-0 [simp]:
    p ≠ Suc 0
  <proof> lemma one-mod-int-p-eq [simp]:
    1 mod int p = 1
  <proof> lemma E-1:
    assumes QuadRes (int p) a
    shows [a ^ ((p - 1) div 2) = 1] (mod int p)
  <proof> definition S1 :: int set where S1 = {0 <.. int p - 1}

```

```

private definition P :: int ⇒ int ⇒ bool where
  P x y ⟷ [x * y = a] (mod p) ∧ y ∈ S1

```

```

private definition f-1 :: int ⇒ int where
  f-1 x = (THE y. P x y)

```

```

private definition f :: int ⇒ int set where
  f x = {x, f-1 x}

```

```

private definition S2 :: int set set where S2 = f ` S1

```

```

private lemma P-lemma: assumes x ∈ S1
  shows ∃! y. P x y
  <proof> lemma f-1-lemma-1: assumes x ∈ S1
    shows P x (f-1 x) <proof> lemma f-1-lemma-2: assumes x ∈ S1

```

```

shows  $f^{-1} (f^{-1} x) = x$ 
 $\langle \text{proof} \rangle$  lemma f-lemma-1: assumes  $x \in S1$ 
shows  $f x = f (f^{-1} x)$   $\langle \text{proof} \rangle$  lemma l1: assumes  $\neg \text{QuadRes } p \ a \ x \in S1$ 
shows  $x \neq f^{-1} x$ 
 $\langle \text{proof} \rangle$  lemma l2: assumes  $\neg \text{QuadRes } p \ a \ x \in S1$ 
shows  $\prod (f x) = a \pmod{p}$ 
 $\langle \text{proof} \rangle$  lemma l3: assumes  $x \in S2$ 
shows finite  $x$   $\langle \text{proof} \rangle$  lemma l4:  $S1 = \bigcup S2$   $\langle \text{proof} \rangle$  lemma l5: assumes  $x$ 
 $\in S2 \ y \in S2 \ x \neq y$ 
shows  $x \cap y = \{\}$ 
 $\langle \text{proof} \rangle$  lemma l6:  $\text{prod Prod } S2 = \prod S1$ 
 $\langle \text{proof} \rangle$  lemma l7:  $\text{fact } n = \prod \{0 <.. \text{int } n\}$ 
 $\langle \text{proof} \rangle$  lemma l8:  $\text{fact } (p - 1) = \prod S1$   $\langle \text{proof} \rangle$  lemma l9:  $[\text{prod Prod } S2 = -1]$ 
 $\pmod{p}$ 
 $\langle \text{proof} \rangle$  lemma l10: assumes  $\text{card } S = n \wedge x. x \in S \implies [g x = a] \pmod{p}$ 
shows  $[\text{prod } g S = a \wedge n] \pmod{p}$   $\langle \text{proof} \rangle$  lemma l11: assumes  $\neg \text{QuadRes } p \ a$ 
shows  $\text{card } S2 = (p - 1) \text{ div } 2$ 
 $\langle \text{proof} \rangle$  lemma l12: assumes  $\neg \text{QuadRes } p \ a$ 
shows  $[\text{prod Prod } S2 = a \wedge ((p - 1) \text{ div } 2)] \pmod{p}$ 
 $\langle \text{proof} \rangle$  lemma E-2: assumes  $\neg \text{QuadRes } p \ a$ 
shows  $[a \wedge ((p - 1) \text{ div } 2) = -1] \pmod{p}$   $\langle \text{proof} \rangle$ 

lemma euler-criterion-aux:  $[(\text{Legendre } a \ p) = a \wedge ((p - 1) \text{ div } 2)] \pmod{p}$ 
 $\langle \text{proof} \rangle$ 

end

theorem euler-criterion: assumes  $\text{prime } p \ 2 < p$ 
shows  $[(\text{Legendre } a \ p) = a \wedge ((p - 1) \text{ div } 2)] \pmod{p}$ 
 $\langle \text{proof} \rangle$ 

hide-fact euler-criterion-aux

end

```

8 Gauss' Lemma

```

theory Gauss
imports Euler-Criterion
begin

```

```

lemma cong-prime-prod-zero-nat:
 $[a * b = 0] \pmod{p} \implies \text{prime } p \implies [a = 0] \pmod{p} \vee [b = 0] \pmod{p}$ 
for  $a :: \text{nat}$ 
 $\langle \text{proof} \rangle$ 

```

```

lemma cong-prime-prod-zero-int:
 $[a * b = 0] \pmod{p} \implies \text{prime } p \implies [a = 0] \pmod{p} \vee [b = 0] \pmod{p}$ 
for  $a :: \text{int}$ 

```

$\langle proof \rangle$

```

locale GAUSS =
  fixes  $p :: nat$ 
  fixes  $a :: int$ 
  assumes  $p\text{-prime}$ :  $prime\ p$ 
  assumes  $p\text{-ge-2}$ :  $2 < p$ 
  assumes  $p\text{-a-relprime}$ :  $[a \neq 0](mod\ p)$ 
  assumes  $a\text{-nonzero}$ :  $0 < a$ 
begin

definition  $A = \{0::int <.. ((int\ p - 1)\ div\ 2)\}$ 
definition  $B = (\lambda x. x * a) \text{ ` } A$ 
definition  $C = (\lambda x. x\ mod\ p) \text{ ` } B$ 
definition  $D = C \cap \{.. (int\ p - 1)\ div\ 2\}$ 
definition  $E = C \cap \{(int\ p - 1)\ div\ 2 <..\}$ 
definition  $F = (\lambda x. (int\ p - x)) \text{ ` } E$ 

```

8.1 Basic properties of p

lemma $odd\text{-}p$: $odd\ p$
 $\langle proof \rangle$

lemma $p\text{-minus-one-l}$: $(int\ p - 1)\ div\ 2 < p$
 $\langle proof \rangle$

lemma $p\text{-eq2}$: $int\ p = (2 * ((int\ p - 1)\ div\ 2)) + 1$
 $\langle proof \rangle$

lemma $p\text{-odd-int}$: **obtains** $z :: int$ **where** $int\ p = 2 * z + 1$ $0 < z$
 $\langle proof \rangle$

8.2 Basic Properties of the Gauss Sets

lemma $finite\text{-}A$: $finite\ A$
 $\langle proof \rangle$

lemma $finite\text{-}B$: $finite\ B$
 $\langle proof \rangle$

lemma $finite\text{-}C$: $finite\ C$
 $\langle proof \rangle$

lemma $finite\text{-}D$: $finite\ D$
 $\langle proof \rangle$

lemma $finite\text{-}E$: $finite\ E$
 $\langle proof \rangle$

lemma *finite-F*: *finite F*
 ⟨*proof*⟩

lemma *C-eq*: $C = D \cup E$
 ⟨*proof*⟩

lemma *A-card-eq*: $\text{card } A = \text{nat } ((\text{int } p - 1) \text{ div } 2)$
 ⟨*proof*⟩

lemma *inj-on-xa-A*: *inj-on* $(\lambda x. x * a)$ *A*
 ⟨*proof*⟩

definition *ResSet* :: *int* \Rightarrow *int set* \Rightarrow *bool*
 where *ResSet* *m X* $\longleftrightarrow (\forall y1\ y2. y1 \in X \wedge y2 \in X \wedge [y1 = y2] \text{ (mod } m) \longrightarrow y1 = y2)$

lemma *ResSet-image*:
 $0 < m \implies \text{ResSet } m\ A \implies \forall x \in A. \forall y \in A. ([f\ x = f\ y] \text{ (mod } m) \longrightarrow x = y) \implies \text{ResSet } m\ (f\ ' A)$
 ⟨*proof*⟩

lemma *A-res*: *ResSet* *p A*
 ⟨*proof*⟩

lemma *B-res*: *ResSet* *p B*
 ⟨*proof*⟩

lemma *SR-B-inj*: *inj-on* $(\lambda x. x \text{ mod } p)$ *B*
 ⟨*proof*⟩

lemma *nonzero-mod-p*: $0 < x \implies x < \text{int } p \implies [x \neq 0] \text{ (mod } p)$
 for *x* :: *int*
 ⟨*proof*⟩

lemma *A-ncong-p*: $x \in A \implies [x \neq 0] \text{ (mod } p)$
 ⟨*proof*⟩

lemma *A-greater-zero*: $x \in A \implies 0 < x$
 ⟨*proof*⟩

lemma *B-ncong-p*: $x \in B \implies [x \neq 0] \text{ (mod } p)$
 ⟨*proof*⟩

lemma *B-greater-zero*: $x \in B \implies 0 < x$
 ⟨*proof*⟩

lemma *B-mod-greater-zero*:
 $0 < x \text{ mod int } p$ **if** $x \in B$
 ⟨*proof*⟩

lemma *C-greater-zero*: $y \in C \implies 0 < y$
 $\langle \text{proof} \rangle$

lemma *F-subset*: $F \subseteq \{x. 0 < x \wedge x \leq ((\text{int } p - 1) \text{ div } 2)\}$
 $\langle \text{proof} \rangle$

lemma *D-subset*: $D \subseteq \{x. 0 < x \wedge x \leq ((p - 1) \text{ div } 2)\}$
 $\langle \text{proof} \rangle$

lemma *F-eq*: $F = \{x. \exists y \in A. (x = p - ((y * a) \bmod p) \wedge (\text{int } p - 1) \text{ div } 2 < (y * a) \bmod p)\}$
 $\langle \text{proof} \rangle$

lemma *D-eq*: $D = \{x. \exists y \in A. (x = (y * a) \bmod p \wedge (y * a) \bmod p \leq (\text{int } p - 1) \text{ div } 2)\}$
 $\langle \text{proof} \rangle$

lemma *all-A-relprime*:
coprime x p **if** $x \in A$
 $\langle \text{proof} \rangle$

lemma *A-prod-relprime*: *coprime* (*prod id* A) p
 $\langle \text{proof} \rangle$

8.3 Relationships Between Gauss Sets

lemma *StandardRes-inj-on-ResSet*: $\text{ResSet } m \ X \implies \text{inj-on } (\lambda b. b \bmod m) \ X$
 $\langle \text{proof} \rangle$

lemma *B-card-eq-A*: $\text{card } B = \text{card } A$
 $\langle \text{proof} \rangle$

lemma *B-card-eq*: $\text{card } B = \text{nat } ((\text{int } p - 1) \text{ div } 2)$
 $\langle \text{proof} \rangle$

lemma *F-card-eq-E*: $\text{card } F = \text{card } E$
 $\langle \text{proof} \rangle$

lemma *C-card-eq-B*: $\text{card } C = \text{card } B$
 $\langle \text{proof} \rangle$

lemma *D-E-disj*: $D \cap E = \{\}$
 $\langle \text{proof} \rangle$

lemma *C-card-eq-D-plus-E*: $\text{card } C = \text{card } D + \text{card } E$
 $\langle \text{proof} \rangle$

lemma *C-prod-eq-D-times-E*: $\text{prod id } E * \text{prod id } D = \text{prod id } C$

$\langle proof \rangle$

lemma *C-B-zcong-prod*: $[prod\ id\ C = prod\ id\ B] \ (mod\ p)$
 $\langle proof \rangle$

lemma *F-Un-D-subset*: $(F \cup D) \subseteq A$
 $\langle proof \rangle$

lemma *F-D-disj*: $(F \cap D) = \{\}$
 $\langle proof \rangle$

lemma *F-Un-D-card*: $card\ (F \cup D) = nat\ ((p - 1)\ div\ 2)$
 $\langle proof \rangle$

lemma *F-Un-D-eq-A*: $F \cup D = A$
 $\langle proof \rangle$

lemma *prod-D-F-eq-prod-A*: $prod\ id\ D * prod\ id\ F = prod\ id\ A$
 $\langle proof \rangle$

lemma *prod-F-zcong*: $[prod\ id\ F = ((-1) \wedge (card\ E)) * prod\ id\ E] \ (mod\ p)$
 $\langle proof \rangle$

8.4 Gauss' Lemma

lemma *aux*: $prod\ id\ A * (-1) \wedge card\ E * a \wedge card\ A * (-1) \wedge card\ E = prod\ id\ A * a \wedge card\ A$
 $\langle proof \rangle$

theorem *pre-gauss-lemma*: $[a \wedge nat((int\ p - 1)\ div\ 2) = (-1) \wedge (card\ E)] \ (mod\ p)$
 $\langle proof \rangle$

theorem *gauss-lemma*: *Legendre* $a\ p = (-1) \wedge (card\ E)$
 $\langle proof \rangle$

end

end

theory *Quadratic-Reciprocity*
imports *Gauss*
begin

The proof is based on Gauss's fifth proof, which can be found at <https://www.lehigh.edu/~shw2/q-recip/gauss5.pdf>.

locale *QR* =
 fixes $p :: nat$

```

fixes  $q :: \text{nat}$ 
assumes  $p\text{-prime}$ :  $\text{prime } p$ 
assumes  $p\text{-ge-2}$ :  $2 < p$ 
assumes  $q\text{-prime}$ :  $\text{prime } q$ 
assumes  $q\text{-ge-2}$ :  $2 < q$ 
assumes  $pq\text{-neq}$ :  $p \neq q$ 
begin

lemma  $\text{odd-}p$ :  $\text{odd } p$ 
   $\langle \text{proof} \rangle$ 

lemma  $p\text{-ge-0}$ :  $0 < \text{int } p$ 
   $\langle \text{proof} \rangle$ 

lemma  $p\text{-eq2}$ :  $\text{int } p = (2 * ((\text{int } p - 1) \text{ div } 2)) + 1$ 
   $\langle \text{proof} \rangle$ 

lemma  $\text{odd-}q$ :  $\text{odd } q$ 
   $\langle \text{proof} \rangle$ 

lemma  $q\text{-ge-0}$ :  $0 < \text{int } q$ 
   $\langle \text{proof} \rangle$ 

lemma  $q\text{-eq2}$ :  $\text{int } q = (2 * ((\text{int } q - 1) \text{ div } 2)) + 1$ 
   $\langle \text{proof} \rangle$ 

lemma  $pq\text{-eq2}$ :  $\text{int } p * \text{int } q = (2 * ((\text{int } p * \text{int } q - 1) \text{ div } 2)) + 1$ 
   $\langle \text{proof} \rangle$ 

lemma  $pq\text{-coprime}$ :  $\text{coprime } p \ q$ 
   $\langle \text{proof} \rangle$ 

lemma  $pq\text{-coprime-int}$ :  $\text{coprime } (\text{int } p) (\text{int } q)$ 
   $\langle \text{proof} \rangle$ 

lemma  $qp\text{-ineq}$ :  $\text{int } p * k \leq (\text{int } p * \text{int } q - 1) \text{ div } 2 \longleftrightarrow k \leq (\text{int } q - 1) \text{ div } 2$ 
   $\langle \text{proof} \rangle$ 

lemma  $QRqp$ :  $QR \ q \ p$ 
   $\langle \text{proof} \rangle$ 

lemma  $pq\text{-commute}$ :  $\text{int } p * \text{int } q = \text{int } q * \text{int } p$ 
   $\langle \text{proof} \rangle$ 

lemma  $pq\text{-ge-0}$ :  $\text{int } p * \text{int } q > 0$ 
   $\langle \text{proof} \rangle$ 

definition  $r = ((p - 1) \text{ div } 2) * ((q - 1) \text{ div } 2)$ 
definition  $m = \text{card } (\text{GAUSS.E } p \ q)$ 

```

definition $n = \text{card } (\text{GAUSS}.E \ q \ p)$

abbreviation $\text{Res } k \equiv \{0 \dots k - 1\} \text{ for } k :: \text{int}$

abbreviation $\text{Res-ge-0 } k \equiv \{0 <.. k - 1\} \text{ for } k :: \text{int}$

abbreviation $\text{Res-0 } k \equiv \{0::\text{int}\} \text{ for } k :: \text{int}$

abbreviation $\text{Res-l } k \equiv \{0 <.. (k - 1) \text{ div } 2\} \text{ for } k :: \text{int}$

abbreviation $\text{Res-h } k \equiv \{(k - 1) \text{ div } 2 <.. k - 1\} \text{ for } k :: \text{int}$

abbreviation $\text{Sets-pq } r0 \ r1 \ r2 \equiv$

$\{(x::\text{int}). x \in r0 \ (\text{int } p * \text{int } q) \wedge x \bmod p \in r1 \ (\text{int } p) \wedge x \bmod q \in r2 \ (\text{int } q)\}$

definition $A = \text{Sets-pq } \text{Res-l } \text{Res-l } \text{Res-h}$

definition $B = \text{Sets-pq } \text{Res-l } \text{Res-h } \text{Res-l}$

definition $C = \text{Sets-pq } \text{Res-h } \text{Res-h } \text{Res-l}$

definition $D = \text{Sets-pq } \text{Res-l } \text{Res-h } \text{Res-h}$

definition $E = \text{Sets-pq } \text{Res-l } \text{Res-0 } \text{Res-h}$

definition $F = \text{Sets-pq } \text{Res-l } \text{Res-h } \text{Res-0}$

definition $a = \text{card } A$

definition $b = \text{card } B$

definition $c = \text{card } C$

definition $d = \text{card } D$

definition $e = \text{card } E$

definition $f = \text{card } F$

lemma $Gpq: \text{GAUSS } p \ q$

$\langle \text{proof} \rangle$

lemma $Gqp: \text{GAUSS } q \ p$

$\langle \text{proof} \rangle$

lemma $QR\text{-lemma-01}: (\lambda x. x \bmod q) \ ' E = \text{GAUSS}.E \ q \ p$

$\langle \text{proof} \rangle$

lemma $QR\text{-lemma-02}: e = n$

$\langle \text{proof} \rangle$

lemma $QR\text{-lemma-03}: f = m$

$\langle \text{proof} \rangle$

definition $f\text{-1} :: \text{int} \Rightarrow \text{int} \times \text{int}$

where $f\text{-1 } x = ((x \bmod p), (x \bmod q))$

definition $P\text{-1} :: \text{int} \times \text{int} \Rightarrow \text{int} \Rightarrow \text{bool}$

where $P\text{-1 } \text{res } x \longleftrightarrow x \bmod p = \text{fst } \text{res} \wedge x \bmod q = \text{snd } \text{res} \wedge x \in \text{Res } (\text{int } p * \text{int } q)$

definition $g\text{-1} :: \text{int} \times \text{int} \Rightarrow \text{int}$

where $g\text{-1 } \text{res} = (\text{THE } x. P\text{-1 } \text{res } x)$

lemma *P-1-lemma*:

fixes $res :: int \times int$
 assumes $0 \leq fst\ res\ fst\ res < p\ 0 \leq snd\ res\ snd\ res < q$
 shows $\exists!x. P-1\ res\ x$
 $\langle proof \rangle$

lemma *g-1-lemma*:

fixes $res :: int \times int$
 assumes $0 \leq fst\ res\ fst\ res < p\ 0 \leq snd\ res\ snd\ res < q$
 shows $P-1\ res\ (g-1\ res)$
 $\langle proof \rangle$

definition $BuC = Sets-pq\ Res-ge-0\ Res-h\ Res-l$

lemma *finite-BuC* [simp]:

finite BuC
 $\langle proof \rangle$

lemma *QR-lemma-04*: $card\ BuC = card\ (Res-h\ p \times Res-l\ q)$
 $\langle proof \rangle$

lemma *QR-lemma-05*: $card\ (Res-h\ p \times Res-l\ q) = r$
 $\langle proof \rangle$

lemma *QR-lemma-06*: $b + c = r$
 $\langle proof \rangle$

definition *f-2*: $int \Rightarrow int$

where $f-2\ x = (int\ p * int\ q) - x$

lemma *f-2-lemma-1*: $f-2\ (f-2\ x) = x$
 $\langle proof \rangle$

lemma *f-2-lemma-2*: $[f-2\ x = int\ p - x] \ (mod\ p)$
 $\langle proof \rangle$

lemma *f-2-lemma-3*: $f-2\ x \in S \implies x \in f-2\ 'S$
 $\langle proof \rangle$

lemma *QR-lemma-07*:

$f-2\ 'Res-l\ (int\ p * int\ q) = Res-h\ (int\ p * int\ q)$
 $f-2\ 'Res-h\ (int\ p * int\ q) = Res-l\ (int\ p * int\ q)$
 $\langle proof \rangle$

lemma *QR-lemma-08*:

$f-2\ x\ mod\ p \in Res-l\ p \longleftrightarrow x\ mod\ p \in Res-h\ p$
 $f-2\ x\ mod\ p \in Res-h\ p \longleftrightarrow x\ mod\ p \in Res-l\ p$
 $\langle proof \rangle$

lemma *QR-lemma-09*:

$f-2 \ x \bmod q \in \text{Res-}l \ q \longleftrightarrow x \bmod q \in \text{Res-}h \ q$
 $f-2 \ x \bmod q \in \text{Res-}h \ q \longleftrightarrow x \bmod q \in \text{Res-}l \ q$
 $\langle \text{proof} \rangle$

lemma *QR-lemma-10*: $a = c$

$\langle \text{proof} \rangle$

definition $BuD = \text{Sets-}pq \ \text{Res-}l \ \text{Res-}h \ \text{Res-ge-}0$

definition $BuD uF = \text{Sets-}pq \ \text{Res-}l \ \text{Res-}h \ \text{Res}$

definition $f-3 :: \text{int} \Rightarrow \text{int} \times \text{int}$

where $f-3 \ x = (x \bmod p, x \text{ div } p + 1)$

definition $g-3 :: \text{int} \times \text{int} \Rightarrow \text{int}$

where $g-3 \ x = \text{fst } x + (\text{snd } x - 1) * p$

lemma *QR-lemma-11*: $\text{card } BuDuF = \text{card } (\text{Res-}h \ p \times \text{Res-}l \ q)$

$\langle \text{proof} \rangle$

lemma *QR-lemma-12*: $b + d + m = r$

$\langle \text{proof} \rangle$

lemma *QR-lemma-13*: $a + d + n = r$

$\langle \text{proof} \rangle$

lemma *QR-lemma-14*: $(-1::\text{int}) \wedge (m + n) = (-1) \wedge r$

$\langle \text{proof} \rangle$

lemma *Quadratic-Reciprocity*:

$\text{Legendre } p \ q * \text{Legendre } q \ p = (-1::\text{int}) \wedge ((p - 1) \text{ div } 2 * ((q - 1) \text{ div } 2))$

$\langle \text{proof} \rangle$

end

theorem *Quadratic-Reciprocity*:

assumes $\text{prime } p \ 2 < p \ \text{prime } q \ 2 < q \ p \neq q$

shows $\text{Legendre } p \ q * \text{Legendre } q \ p = (-1::\text{int}) \wedge ((p - 1) \text{ div } 2 * ((q - 1) \text{ div } 2))$

$\langle \text{proof} \rangle$

theorem *Quadratic-Reciprocity-int*:

assumes $\text{prime } (\text{nat } p) \ 2 < p \ \text{prime } (\text{nat } q) \ 2 < q \ p \neq q$

shows $\text{Legendre } p \ q * \text{Legendre } q \ p = (-1::\text{int}) \wedge (\text{nat } ((p - 1) \text{ div } 2 * ((q - 1) \text{ div } 2)))$

$\langle \text{proof} \rangle$

end

9 Pocklington's Theorem for Primes

```
theory Pocklington
imports Residues
begin
```

9.1 Lemmas about previously defined terms

```
lemma prime-nat-iff'': prime (p::nat)  $\longleftrightarrow$   $p \neq 0 \wedge p \neq 1 \wedge (\forall m. 0 < m \wedge m < p \longrightarrow \text{coprime } p \ m)$ 
<proof>
```

```
lemma finite-number-segment: card { m. 0 < m  $\wedge$  m < n } = n - 1
<proof>
```

9.2 Some basic theorems about solving congruences

```
lemma cong-solve:
  fixes n :: nat
  assumes an: coprime a n
  shows  $\exists x. [a * x = b] \pmod n$ 
<proof>
```

```
lemma cong-solve-unique:
  fixes n :: nat
  assumes an: coprime a n and nz:  $n \neq 0$ 
  shows  $\exists! x. x < n \wedge [a * x = b] \pmod n$ 
<proof>
```

```
lemma cong-solve-unique-nontrivial:
  fixes p :: nat
  assumes p: prime p
    and pa: coprime p a
    and x0:  $0 < x$ 
    and xp:  $x < p$ 
  shows  $\exists! y. 0 < y \wedge y < p \wedge [x * y = a] \pmod p$ 
<proof>
```

```
lemma cong-unique-inverse-prime:
  fixes p :: nat
  assumes prime p and  $0 < x$  and  $x < p$ 
  shows  $\exists! y. 0 < y \wedge y < p \wedge [x * y = 1] \pmod p$ 
<proof>
```

```
lemma chinese-remainder-coprime-unique:
  fixes a :: nat
  assumes ab: coprime a b and az:  $a \neq 0$  and bz:  $b \neq 0$ 
    and ma: coprime m a and nb: coprime n b
  shows  $\exists! x. \text{coprime } x \ (a * b) \wedge x < a * b \wedge [x = m] \pmod a \wedge [x = n] \pmod b$ 
<proof>
```

9.3 Lucas's theorem

lemma *lucas-coprime-lemma*:

fixes $n :: \text{nat}$

assumes $m: m \neq 0$ **and** $am: [a \wedge^m = 1] \pmod n$

shows *coprime* $a\ n$

$\langle \text{proof} \rangle$

lemma *lucas-weak*:

fixes $n :: \text{nat}$

assumes $n: n \geq 2$

and $an: [a \wedge^{(n-1)} = 1] \pmod n$

and $nm: \forall m. 0 < m \wedge m < n - 1 \longrightarrow \neg [a \wedge^m = 1] \pmod n$

shows *prime* n

$\langle \text{proof} \rangle$

theorem *lucas*:

assumes $n2: n \geq 2$ **and** $an1: [a \wedge^{(n-1)} = 1] \pmod n$

and $pn: \forall p. \text{prime } p \wedge p \text{ dvd } n - 1 \longrightarrow [a \wedge^{((n-1) \text{ div } p)} \neq 1] \pmod n$

shows *prime* n

$\langle \text{proof} \rangle$

9.4 Definition of the order of a number mod n

definition $\text{ord } n\ a = (\text{if } \text{coprime } n\ a \text{ then } \text{Least } (\lambda d. d > 0 \wedge [a \wedge^d = 1] \pmod n) \text{ else } 0)$

This has the expected properties.

lemma *coprime-ord*:

fixes $n::\text{nat}$

assumes *coprime* $n\ a$

shows $\text{ord } n\ a > 0 \wedge [a \wedge^{\text{ord } n\ a} = 1] \pmod n \wedge (\forall m. 0 < m \wedge m < \text{ord } n\ a \longrightarrow [a \wedge^m \neq 1] \pmod n)$

$\langle \text{proof} \rangle$

With the special value 0 for non-coprime case, it's more convenient.

lemma *ord-works*: $[a \wedge^{(\text{ord } n\ a)} = 1] \pmod n \wedge (\forall m. 0 < m \wedge m < \text{ord } n\ a \longrightarrow \neg [a \wedge^m = 1] \pmod n)$

for $n :: \text{nat}$

$\langle \text{proof} \rangle$

lemma *ord*: $[a \wedge^{\text{ord } n\ a} = 1] \pmod n$

for $n :: \text{nat}$

$\langle \text{proof} \rangle$

lemma *ord-minimal*: $0 < m \implies m < \text{ord } n\ a \implies \neg [a \wedge^m = 1] \pmod n$

for $n :: \text{nat}$

$\langle \text{proof} \rangle$

lemma *ord-eq-0*: $\text{ord } n\ a = 0 \longleftrightarrow \neg \text{coprime } n\ a$

for $n :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *divides-rexp*: $x \text{ dvd } y \implies x \text{ dvd } (y \wedge \text{Suc } n)$
for $x \ y :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *ord-divides*: $[a \wedge d = 1] \pmod n \longleftrightarrow \text{ord } n \ a \text{ dvd } d$
(is ?lhs \longleftrightarrow ?rhs)
for $n :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *order-divides-totient*:
 $\text{ord } n \ a \text{ dvd } \text{totient } n$ **if** *coprime* $n \ a$
 $\langle \text{proof} \rangle$

lemma *order-divides-expdiff*:
fixes $n::\text{nat}$ **and** $a::\text{nat}$ **assumes** $na: \text{coprime } n \ a$
shows $[a \wedge d = a \wedge e] \pmod n \longleftrightarrow [d = e] \pmod{(\text{ord } n \ a)}$
 $\langle \text{proof} \rangle$

lemma *ord-not-coprime* [*simp*]: $\neg \text{coprime } n \ a \implies \text{ord } n \ a = 0$
 $\langle \text{proof} \rangle$

lemma *ord-1* [*simp*]: $\text{ord } 1 \ n = 1$
 $\langle \text{proof} \rangle$

lemma *ord-1-right* [*simp*]: $\text{ord } (n::\text{nat}) \ 1 = 1$
 $\langle \text{proof} \rangle$

lemma *ord-Suc-0-right* [*simp*]: $\text{ord } (n::\text{nat}) \ (\text{Suc } 0) = 1$
 $\langle \text{proof} \rangle$

lemma *ord-0-nat* [*simp*]: $\text{ord } 0 \ (n :: \text{nat}) = (\text{if } n = 1 \text{ then } 1 \text{ else } 0)$
 $\langle \text{proof} \rangle$

lemma *ord-0-right-nat* [*simp*]: $\text{ord } (n :: \text{nat}) \ 0 = (\text{if } n = 1 \text{ then } 1 \text{ else } 0)$
 $\langle \text{proof} \rangle$

lemma *ord-divides'*: $[a \wedge d = \text{Suc } 0] \pmod n = (\text{ord } n \ a \text{ dvd } d)$
 $\langle \text{proof} \rangle$

lemma *ord-Suc-0* [*simp*]: $\text{ord } (\text{Suc } 0) \ n = 1$
 $\langle \text{proof} \rangle$

lemma *ord-mod* [*simp*]: $\text{ord } n \ (k \bmod n) = \text{ord } n \ k$
 $\langle \text{proof} \rangle$

lemma *ord-gt-0-iff* [*simp*]: $\text{ord } (n::\text{nat}) \ x > 0 \longleftrightarrow \text{coprime } n \ x$

$\langle \text{proof} \rangle$

lemma *ord-eq-Suc-0-iff*: $\text{ord } n \ (x :: \text{nat}) = \text{Suc } 0 \longleftrightarrow [x = 1] \ (\text{mod } n)$
 $\langle \text{proof} \rangle$

lemma *ord-cong*:
 assumes $[k1 = k2] \ (\text{mod } n)$
 shows $\text{ord } n \ k1 = \text{ord } n \ k2$
 $\langle \text{proof} \rangle$

lemma *ord-nat-code* [*code-unfold*]:
 $\text{ord } n \ a =$
 $(\text{if } n = 0 \text{ then if } a = 1 \text{ then } 1 \text{ else } 0 \text{ else}$
 $\text{if coprime } n \ a \text{ then Min (Set.filter } (\lambda k. [a \wedge k = 1] \ (\text{mod } n)) \ \{0 <..n\}) \text{ else}$
 $0)$
 $\langle \text{proof} \rangle$

theorem *ord-modulus-mult-coprime*:
 fixes $x :: \text{nat}$
 assumes *coprime* $m \ n$
 shows $\text{ord } (m * n) \ x = \text{lcm } (\text{ord } m \ x) \ (\text{ord } n \ x)$
 $\langle \text{proof} \rangle$

corollary *ord-modulus-prod-coprime*:
 assumes $\text{finite } A \ \bigwedge i \ j. i \in A \implies j \in A \implies i \neq j \implies \text{coprime } (f \ i) \ (f \ j)$
 shows $\text{ord } (\prod i \in A. f \ i :: \text{nat}) \ x = (\text{LCM } i \in A. \text{ord } (f \ i) \ x)$
 $\langle \text{proof} \rangle$

lemma *ord-power-aux*:
 fixes $m \ x \ k \ a :: \text{nat}$
 defines $l \equiv \text{ord } m \ a$
 shows $\text{ord } m \ (a \wedge k) * \text{gcd } k \ l = l$
 $\langle \text{proof} \rangle$

theorem *ord-power*: $\text{coprime } m \ a \implies \text{ord } m \ (a \wedge k :: \text{nat}) = \text{ord } m \ a \ \text{div } \text{gcd } k \ (\text{ord } m \ a)$
 $\langle \text{proof} \rangle$

lemma *inj-power-mod*:
 assumes *coprime* $n \ (a :: \text{nat})$
 shows $\text{inj-on } (\lambda k. a \wedge k \ \text{mod } n) \ \{..<\text{ord } n \ a\}$
 $\langle \text{proof} \rangle$

lemma *ord-eq-2-iff*: $\text{ord } n \ (x :: \text{nat}) = 2 \longleftrightarrow [x \neq 1] \ (\text{mod } n) \wedge [x^2 = 1] \ (\text{mod } n)$
 $\langle \text{proof} \rangle$

lemma *square-mod-8-eq-1-iff*: $[x^2 = 1] \ (\text{mod } 8) \longleftrightarrow \text{odd } (x :: \text{nat})$
 $\langle \text{proof} \rangle$

lemma *ord-twopow-aux*:

assumes $k \geq 3$ **and** $\text{odd } (x :: \text{nat})$
shows $[x \wedge (2 \wedge (k - 2)) = 1] \pmod{(2 \wedge k)}$
 $\langle \text{proof} \rangle$

lemma *ord-twopow-3-5*:

assumes $k \geq 3$ $x \bmod 8 \in \{3, 5 :: \text{nat}\}$
shows $\text{ord } (2 \wedge k) x = 2 \wedge (k - 2)$
 $\langle \text{proof} \rangle$

lemma *ord-4-3 [simp]*: $\text{ord } 4 \ (3 :: \text{nat}) = 2$
 $\langle \text{proof} \rangle$

lemma *elements-with-ord-1*: $n > 0 \implies \{x \in \text{totatives } n. \text{ord } n x = \text{Suc } 0\} = \{1\}$
 $\langle \text{proof} \rangle$

lemma *residue-prime-has-primroot*:

fixes $p :: \text{nat}$
assumes $\text{prime } p$
shows $\exists a \in \text{totatives } p. \text{ord } p a = p - 1$
 $\langle \text{proof} \rangle$

9.5 Another trivial primality characterization

lemma *prime-prime-factor*: $\text{prime } n \longleftrightarrow n \neq 1 \wedge (\forall p. \text{prime } p \wedge p \text{ dvd } n \longrightarrow p = n)$
(is ?lhs \longleftrightarrow ?rhs)
for $n :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *prime-divisor-sqrt*: $\text{prime } n \longleftrightarrow n \neq 1 \wedge (\forall d. d \text{ dvd } n \wedge d^2 \leq n \longrightarrow d = 1)$
for $n :: \text{nat}$
 $\langle \text{proof} \rangle$

lemma *prime-prime-factor-sqrt*:

$\text{prime } (n :: \text{nat}) \longleftrightarrow n \neq 0 \wedge n \neq 1 \wedge (\nexists p. \text{prime } p \wedge p \text{ dvd } n \wedge p^2 \leq n)$
(is ?lhs \longleftrightarrow ?rhs)
 $\langle \text{proof} \rangle$

9.6 Pocklington theorem

lemma *pocklington-lemma*:

fixes $p :: \text{nat}$
assumes $n: n \geq 2$ **and** $nqr: n - 1 = q * r$
and $an: [a \wedge (n - 1) = 1] \pmod{n}$
and $aq: \forall p. \text{prime } p \wedge p \text{ dvd } q \longrightarrow \text{coprime } (a \wedge ((n - 1) \text{ div } p) - 1) n$
and $pp: \text{prime } p$ **and** $pn: p \text{ dvd } n$
shows $[p = 1] \pmod{q}$

$\langle \text{proof} \rangle$

theorem *pocklington*:

assumes $n: n \geq 2$ **and** $nqr: n - 1 = q * r$ **and** $sqr: n \leq q^2$
and $an: [a^{n-1} = 1] \pmod n$
and $aq: \forall p. \text{prime } p \wedge p \text{ dvd } q \longrightarrow \text{coprime } (a^{(n-1) \text{ div } p} - 1) \ n$
shows $\text{prime } n$
 $\langle \text{proof} \rangle$

Variant for application, to separate the exponentiation.

lemma *pocklington-alt*:

assumes $n: n \geq 2$ **and** $nqr: n - 1 = q * r$ **and** $sqr: n \leq q^2$
and $an: [a^{n-1} = 1] \pmod n$
and $aq: \forall p. \text{prime } p \wedge p \text{ dvd } q \longrightarrow (\exists b. [a^{(n-1) \text{ div } p} = b] \pmod n \wedge \text{coprime } (b - 1) \ n)$
shows $\text{prime } n$
 $\langle \text{proof} \rangle$

9.7 Prime factorizations

definition *primefact* $ps \ n \longleftrightarrow \text{foldr } (*) \ ps \ 1 = n \wedge (\forall p \in \text{set } ps. \text{prime } p)$

lemma *primefact*:

fixes $n :: \text{nat}$
assumes $n: n \neq 0$
shows $\exists ps. \text{primefact } ps \ n$
 $\langle \text{proof} \rangle$

lemma *primefact-contains*:

fixes $p :: \text{nat}$
assumes $pf: \text{primefact } ps \ n$
and $p: \text{prime } p$
and $pn: p \text{ dvd } n$
shows $p \in \text{set } ps$
 $\langle \text{proof} \rangle$

lemma *primefact-variant*: $\text{primefact } ps \ n \longleftrightarrow \text{foldr } (*) \ ps \ 1 = n \wedge \text{list-all prime } ps$

$\langle \text{proof} \rangle$

Variant of Lucas theorem.

lemma *lucas-primefact*:

assumes $n: n \geq 2$ **and** $an: [a^{n-1} = 1] \pmod n$
and $psn: \text{foldr } (*) \ ps \ 1 = n - 1$
and $psp: \text{list-all } (\lambda p. \text{prime } p \wedge \neg [a^{(n-1) \text{ div } p} = 1] \pmod n) \ ps$
shows $\text{prime } n$
 $\langle \text{proof} \rangle$

Variant of Pocklington theorem.


```

lemma pocklington-primefact:
  assumes  $n: n \geq 2$  and  $qrn: q * r = n - 1$  and  $nq2: n \leq q^2$ 
  and  $arnb: (a \hat{r}) \bmod n = b$  and  $psq: \text{foldr } (*) \text{ ps } 1 = q$ 
  and  $bqn: (b \hat{q}) \bmod n = 1$ 
  and  $psp: \text{list-all } (\lambda p. \text{prime } p \wedge \text{coprime } ((b \hat{q} (q \text{ div } p)) \bmod n - 1) \text{ } n) \text{ ps}$ 
  shows prime n
  <proof>

end

```

10 Prime powers

```

theory Prime-Powers
  imports Complex-Main HOL-Computational-Algebra.Primes HOL-Library.FuncSet
begin

```

```

definition aprimedivisor :: 'a :: normalization-semidom  $\Rightarrow$  'a where
  aprimedivisor q = (SOME p. prime p  $\wedge$  p dvd q)

```

```

definition primepow :: 'a :: normalization-semidom  $\Rightarrow$  bool where
  primepow n  $\longleftrightarrow (\exists p \ k. \text{prime } p \wedge k > 0 \wedge n = p \hat{k})$ 

```

```

definition primepow-factors :: 'a :: normalization-semidom  $\Rightarrow$  'a set where
  primepow-factors n = {x. primepow x  $\wedge$  x dvd n}

```

```

lemma primepow-gt-Suc-0: primepow n  $\implies n > \text{Suc } 0$ 
  <proof>

```

```

lemma
  assumes prime p p dvd n
  shows prime-aprimedivisor: prime (aprimedivisor n)
  and aprimedivisor-dvd: aprimedivisor n dvd n
  <proof>

```

```

lemma
  assumes  $n \neq 0$   $\neg \text{is-unit } (n :: 'a :: \text{factorial-semiring})$ 
  shows prime-aprimedivisor': prime (aprimedivisor n)
  and aprimedivisor-dvd': aprimedivisor n dvd n
  <proof>

```

```

lemma aprimedivisor-of-prime [simp]:
  assumes prime p
  shows aprimedivisor p = p
  <proof>

```

```

lemma aprimedivisor-pos-nat:  $(n :: \text{nat}) > 1 \implies \text{aprimedivisor } n > 0$ 
  <proof>

```

```

lemma aprimedivisor-primepow-power:

```

assumes $\text{primepow } n \ k > 0$
shows $\text{aprimedivisor } (n \wedge k) = \text{aprimedivisor } n$
 <proof>

lemma *aprimedivisor-prime-power*:
assumes $\text{prime } p \ k > 0$
shows $\text{aprimedivisor } (p \wedge k) = p$
 <proof>

lemma *prime-factorization-primelow*:
assumes $\text{primepow } n$
shows $\text{prime-factorization } n =$
 $\text{replicate-mset } (\text{multiplicity } (\text{aprimedivisor } n) \ n) \ (\text{aprimedivisor } n)$
 <proof>

lemma *primepow-decompose*:
fixes $n :: 'a :: \text{factorial-semiring-multiplicative}$
assumes $\text{primepow } n$
shows $\text{aprimedivisor } n \wedge \text{multiplicity } (\text{aprimedivisor } n) \ n = n$
 <proof>

lemma *prime-power-not-one*:
assumes $\text{prime } p \ k > 0$
shows $p \wedge k \neq 1$
 <proof>

lemma *zero-not-primelow* [simp]: $\neg \text{primepow } 0$
 <proof>

lemma *one-not-primelow* [simp]: $\neg \text{primepow } 1$
 <proof>

lemma *primelow-not-unit* [simp]: $\text{primepow } p \implies \neg \text{is-unit } p$
 <proof>

lemma *not-primelow-Suc-0-nat* [simp]: $\neg \text{primepow } (\text{Suc } 0)$
 <proof>

lemma *primelow-gt-0-nat*: $\text{primepow } n \implies n > (0 :: \text{nat})$
 <proof>

lemma *unit-factor-primelow*:
fixes $p :: 'a :: \text{factorial-semiring-multiplicative}$
shows $\text{primepow } p \implies \text{unit-factor } p = 1$
 <proof>

lemma *aprimedivisor-primelow*:
assumes $\text{prime } p \ p \ \text{dvd } n \ \text{primepow } (n :: 'a :: \text{factorial-semiring-multiplicative})$
shows $\text{aprimedivisor } (p * n) = p \ \text{aprimedivisor } n = p$

$\langle \text{proof} \rangle$

lemma *power-eq-prime-powerD*:

fixes $p :: 'a :: \text{factorial-semiring}$

assumes $\text{prime } p \ n > 0 \ x \wedge n = p \wedge k$

shows $\exists i. \text{normalize } x = \text{normalize } (p \wedge i)$

$\langle \text{proof} \rangle$

lemma *primepow-power-iff*:

fixes $p :: 'a :: \text{factorial-semiring-multiplicative}$

assumes $\text{unit-factor } p = 1$

shows $\text{primepow } (p \wedge n) \longleftrightarrow \text{primepow } p \wedge n > 0$

$\langle \text{proof} \rangle$

lemma *primepow-power-iff-nat*:

$p > 0 \implies \text{primepow } (p \wedge n) \longleftrightarrow \text{primepow } (p :: \text{nat}) \wedge n > 0$

$\langle \text{proof} \rangle$

lemma *primepow-prime [simp]*: $\text{prime } n \implies \text{primepow } n$

$\langle \text{proof} \rangle$

lemma *primepow-prime-power [simp]*:

$\text{prime } (p :: 'a :: \text{factorial-semiring-multiplicative}) \implies \text{primepow } (p \wedge n) \longleftrightarrow n > 0$

$\langle \text{proof} \rangle$

lemma *aprimedivisor-vimage*:

assumes $\text{prime } (p :: 'a :: \text{factorial-semiring-multiplicative})$

shows $\text{aprimedivisor } -' \{p\} \cap \text{primepow-factors } n = \{p \wedge k \mid k. k > 0 \wedge p \wedge k \text{ dvd } n\}$

$\langle \text{proof} \rangle$

lemma *aprimedivisor-nat*:

assumes $n \neq (\text{Suc } 0 :: \text{nat})$

shows $\text{prime } (\text{aprimedivisor } n) \ \text{aprimedivisor } n \text{ dvd } n$

$\langle \text{proof} \rangle$

lemma *aprimedivisor-gt-Suc-0*:

assumes $n \neq \text{Suc } 0$

shows $\text{aprimedivisor } n > \text{Suc } 0$

$\langle \text{proof} \rangle$

lemma *aprimedivisor-le-nat*:

assumes $n > \text{Suc } 0$

shows $\text{aprimedivisor } n \leq n$

$\langle \text{proof} \rangle$

lemma *bij-betw-primewows*:

$\text{bij_betw } (\lambda(p,k). p \wedge \text{Suc } k :: 'a :: \text{factorial-semiring-multiplicative})$
 $(\text{Collect prime} \times \text{UNIV}) (\text{Collect primepow})$
 $\langle \text{proof} \rangle$

lemma *primepow-multD*:
assumes *primepow* $(a * b :: \text{nat})$
shows $a = 1 \vee \text{primepow } a \wedge b = 1 \vee \text{primepow } b$
 $\langle \text{proof} \rangle$

lemma *primepow-mult-aprime divisorI*:
assumes *primepow* $(n :: 'a :: \text{factorial-semiring-multiplicative})$
shows *primepow* $(\text{aprime divisor } n * n)$
 $\langle \text{proof} \rangle$

lemma *primepow-factors-altdef*:
fixes $x :: 'a :: \text{factorial-semiring-multiplicative}$
assumes $x \neq 0$
shows $\text{primepow-factors } x = \{p \wedge k \mid p \in \text{prime-factors } x \wedge k \in \{0 < .. \text{multiplicity } p \ x\}\}$
 $\langle \text{proof} \rangle$

lemma *finite-primepow-factors*:
assumes $x \neq (0 :: 'a :: \text{factorial-semiring-multiplicative})$
shows *finite* $(\text{primepow-factors } x)$
 $\langle \text{proof} \rangle$

lemma *aprime divisor-primepow-factors-conv-prime-factorization*:
assumes $[simp]: n \neq (0 :: 'a :: \text{factorial-semiring-multiplicative})$
shows $\text{image-mset } \text{aprime divisor } (\text{mset-set } (\text{primepow-factors } n)) = \text{prime-factorization } n$
 $(\text{is } ?lhs = ?rhs)$
 $\langle \text{proof} \rangle$

lemma *prime-elem-aprime divisor-nat*: $d > \text{Suc } 0 \implies \text{prime-elem } (\text{aprime divisor } d)$
 $\langle \text{proof} \rangle$

lemma *aprime divisor-gt-0-nat* $[simp]$: $d > \text{Suc } 0 \implies \text{aprime divisor } d > 0$
 $\langle \text{proof} \rangle$

lemma *aprime divisor-gt-Suc-0-nat* $[simp]$: $d > \text{Suc } 0 \implies \text{aprime divisor } d > \text{Suc } 0$
 $\langle \text{proof} \rangle$

lemma *aprime divisor-not-Suc-0-nat* $[simp]$: $d > \text{Suc } 0 \implies \text{aprime divisor } d \neq \text{Suc } 0$
 $\langle \text{proof} \rangle$

lemma *multiplicity-aprime divisor-gt-0-nat* [simp]:
 $d > \text{Suc } 0 \implies \text{multiplicity } (\text{aprime divisor } d) \ d > 0$
 <proof>

lemma *primepowI*:
 $\text{prime } p \implies k > 0 \implies p^k = n \implies \text{primepow } n \wedge \text{aprime divisor } n = p$
 <proof>

lemma *not-primepowI*:
assumes $\text{prime } p \ \text{prime } q \ p \neq q \ p \text{ dvd } n \ q \text{ dvd } n$
shows $\neg \text{primepow } n$
 <proof>

lemma *sum-prime-factorization-conv-sum-primepow-factors*:
fixes $n :: 'a :: \text{factorial-semiring-multiplicative}$
assumes $n \neq 0$
shows $(\sum_{q \in \text{primepow-factors } n} f(\text{aprime divisor } q)) = (\sum_{p \in \# \text{prime-factorization } n} f p)$
 <proof>

lemma *multiplicity-aprime divisor-Suc-0-iff*:
assumes $\text{primepow } (n :: 'a :: \text{factorial-semiring-multiplicative})$
shows $\text{multiplicity } (\text{aprime divisor } n) \ n = \text{Suc } 0 \longleftrightarrow \text{prime } n$
 <proof>

definition *mangoldt* :: $\text{nat} \Rightarrow 'a :: \text{real-algebra-1}$ **where**
 $\text{mangoldt } n = (\text{if } \text{primepow } n \text{ then } \text{of-real } (\ln (\text{real } (\text{aprime divisor } n))) \text{ else } 0)$

lemma *mangoldt-0* [simp]: $\text{mangoldt } 0 = 0$
 <proof>

lemma *mangoldt-Suc-0* [simp]: $\text{mangoldt } (\text{Suc } 0) = 0$
 <proof>

lemma *of-real-mangoldt* [simp]: $\text{of-real } (\text{mangoldt } n) = \text{mangoldt } n$
 <proof>

lemma *mangoldt-sum*:
assumes $n \neq 0$
shows $(\sum d \mid d \text{ dvd } n. \text{mangoldt } d :: 'a :: \text{real-algebra-1}) = \text{of-real } (\ln (\text{real } n))$
 <proof>

lemma *mangoldt-primepow*:
 $\text{prime } p \implies \text{mangoldt } (p^k) = (\text{if } k > 0 \text{ then } \text{of-real } (\ln (\text{real } p)) \text{ else } 0)$
 <proof>

lemma *mangoldt-primepow'* [simp]: $\text{prime } p \implies k > 0 \implies \text{mangoldt } (p^k) = \text{of-real } (\ln (\text{real } p))$

```

    <proof>

lemma mangoldt-prime [simp]: prime p  $\implies$  mangoldt p = of-real (ln (real p))
    <proof>

lemma mangoldt-nonneg: 0  $\leq$  (mangoldt d :: real)
    <proof>

lemma norm-mangoldt [simp]:
    norm (mangoldt n :: 'a :: real-normed-algebra-1) = mangoldt n
    <proof>

lemma Re-mangoldt [simp]: Re (mangoldt n) = mangoldt n
    and Im-mangoldt [simp]: Im (mangoldt n) = 0
    <proof>

lemma abs-mangoldt [simp]: abs (mangoldt n :: real) = mangoldt n
    <proof>

lemma mangoldt-le:
    assumes n > 0
    shows mangoldt n  $\leq$  ln n
    <proof>

end

```

11 Primitive roots in residue rings and Carmichael's function

```

theory Residue-Primitive-Roots
  imports Pocklington
begin

```

This theory develops the notions of primitive roots (generators) in residue rings. It also provides a definition and all the basic properties of Carmichael's function $\lambda(n)$, which is strongly related to this. The proofs mostly follow Apostol's presentation

11.1 Primitive roots in residue rings

A primitive root of a residue ring modulo n is an element g that *generates* the ring, i. e. such that for each x coprime to n there exists an i such that $x = g^i$. A simpler definition is that g must have the same order as the cardinality of the multiplicative group, which is $\varphi(n)$.

Note that for convenience, this definition does *not* demand $g < n$.

```

inductive residue-primroot :: nat  $\Rightarrow$  nat  $\Rightarrow$  bool where

```

$n > 0 \implies \text{coprime } n \ g \implies \text{ord } n \ g = \text{totient } n \implies \text{residue-primroot } n \ g$

lemma *residue-primroot-def* [code]:

$\text{residue-primroot } n \ x \longleftrightarrow n > 0 \wedge \text{coprime } n \ x \wedge \text{ord } n \ x = \text{totient } n$
 $\langle \text{proof} \rangle$

lemma *not-residue-primroot-0* [simp]: $\sim \text{residue-primroot } 0 \ x$

$\langle \text{proof} \rangle$

lemma *residue-primroot-mod* [simp]: $\text{residue-primroot } n \ (x \bmod n) = \text{residue-primroot } n \ x$

$\langle \text{proof} \rangle$

lemma *residue-primroot-cong*:

assumes $[x = x'] \ (mod \ n)$

shows $\text{residue-primroot } n \ x = \text{residue-primroot } n \ x'$

$\langle \text{proof} \rangle$

lemma *not-residue-primroot-0-right* [simp]: $\text{residue-primroot } n \ 0 \longleftrightarrow n = 1$

$\langle \text{proof} \rangle$

lemma *residue-primroot-1-iff*: $\text{residue-primroot } n \ (\text{Suc } 0) \longleftrightarrow n \in \{1, 2\}$

$\langle \text{proof} \rangle$

11.2 Primitive roots modulo a prime

For prime p , we now analyse the number of elements in the ring $\mathbb{Z}/p\mathbb{Z}$ whose order is precisely d for each d .

context

fixes $n :: nat$ **and** ψ

assumes $n: n > 1$

defines $\psi \equiv (\lambda d. \text{card } \{x \in \text{totatives } n. \text{ord } n \ x = d\})$

begin

lemma *elements-with-ord-restrict-totatives*:

$d > 0 \implies \{x \in \{..<n\}. \text{ord } n \ x = d\} = \{x \in \text{totatives } n. \text{ord } n \ x = d\}$

$\langle \text{proof} \rangle$

lemma *prime-elements-with-ord*:

assumes $\psi \ d \neq 0$ **and** *prime* n

and $a: a \in \text{totatives } n \ \text{ord } n \ a = d \ a \neq 1$

shows $\text{inj-on } (\lambda k. a \wedge^k \bmod n) \ \{..<d\}$

and $\{x \in \{..<n\}. [x \wedge^d = 1] \ (mod \ n)\} = (\lambda k. a \wedge^k \bmod n) \ ^\circ \ \{..<d\}$

and $\text{bij-betw } (\lambda k. a \wedge^k \bmod n) \ (\text{totatives } d) \ \{x \in \{..<n\}. \text{ord } n \ x = d\}$

$\langle \text{proof} \rangle$

lemma *prime-card-elements-with-ord*:

assumes $\psi \ d \neq 0$ **and** *prime* n

shows $\psi \ d = \text{totient } d$

<proof>

lemma *prime-sum-card-elements-with-ord-eq-totient:*

$(\sum d \mid d \text{ dvd } \text{totient } n. \psi d) = \text{totient } n$

<proof>

We can now show that the number of elements of order d is $\varphi(d)$ if $d \mid p - 1$ and 0 otherwise.

theorem *prime-card-elements-with-ord-eq-totient:*

assumes *prime* n

shows $\psi d = (\text{if } d \text{ dvd } n - 1 \text{ then } \text{totient } d \text{ else } 0)$

<proof>

As a corollary, we get that the number of primitive roots modulo a prime p is $\varphi(p - 1)$. Since this number is positive, we also get that there is at least one primitive root modulo p .

lemma

assumes *prime* n

shows *prime-card-primitive-roots:* $\text{card } \{x \in \text{totatives } n. \text{ord } n x = n - 1\} = \text{totient } (n - 1)$

$\text{card } \{x \in \{..<n\}. \text{ord } n x = n - 1\} = \text{totient } (n - 1)$

and *prime-primitive-root-exists:* $\exists x. \text{residue-primroot } n x$

<proof>

end

11.3 Primitive roots modulo powers of an odd prime

Any primitive root g modulo an odd prime p is also a primitive root modulo p^k for all $k > 0$ if $[g^{p-1} \neq 1] \pmod{p^2}$. To show this, we first need the following lemma.

lemma *residue-primroot-power-prime-power-neq-1:*

assumes $k \geq 2$

assumes p : *prime* p *odd* p **and** *residue-primroot* $p g$ **and** $[g^{p-1} \neq 1] \pmod{p^2}$

shows $[g^{p^{k-1}} \neq 1] \pmod{p^k}$

<proof>

We can now show that primitive roots modulo p with the above condition are indeed also primitive roots modulo p^k .

proposition *residue-primroot-prime-lift-iff:*

assumes p : *prime* p *odd* p **and** *residue-primroot* $p g$

shows $(\forall k > 0. \text{residue-primroot } (p^k) g) \longleftrightarrow [g^{p-1} \neq 1] \pmod{p^2}$

<proof>

If p is an odd prime, there is always a primitive root g modulo p , and if g does not fulfil the above assumption required for it to be liftable to p^k , we

can use $g + p$, which is also a primitive root modulo p and *does* fulfil the assumption.

This shows that any modulus that is a power of an odd prime has a primitive root.

theorem *residue-primroot-odd-prime-power-exists:*

assumes p : prime p odd p

obtains g **where** $\forall k > 0. \text{ residue-primroot } (p \wedge k) g$
 $\langle \text{proof} \rangle$

11.4 Carmichael's function

Carmichael's function $\lambda(n)$ gives the LCM of the orders of all elements in the residue ring modulo n – or, equivalently, the maximum order, as we will show later. Algebraically speaking, it is the exponent of the multiplicative group $(\mathbb{Z}/n\mathbb{Z})^*$.

It is not to be confused with Liouville's function, which is also denoted by $\lambda(n)$.

definition *Carmichael where*

$\text{Carmichael } n = (\text{LCM } a \in \text{ totatives } n. \text{ ord } n a)$

lemma *Carmichael-0 [simp]: Carmichael 0 = 1*

$\langle \text{proof} \rangle$

lemma *Carmichael-1 [simp]: Carmichael 1 = 1*

$\langle \text{proof} \rangle$

lemma *Carmichael-Suc-0 [simp]: Carmichael (Suc 0) = 1*

$\langle \text{proof} \rangle$

lemma *ord-dvd-Carmichael:*

assumes $n > 1$ coprime $n k$

shows $\text{ord } n k \text{ dvd } \text{Carmichael } n$

$\langle \text{proof} \rangle$

lemma *Carmichael-divides:*

assumes $\text{Carmichael } n \text{ dvd } k$ coprime $n a$

shows $[a \wedge k = 1] \pmod{n}$

$\langle \text{proof} \rangle$

lemma *Carmichael-dvd-totient: Carmichael n dvd totient n*

$\langle \text{proof} \rangle$

lemma *Carmichael-dvd-mono-coprime:*

assumes coprime $m n$ $m > 1$ $n > 1$

shows $\text{Carmichael } m \text{ dvd } \text{Carmichael } (m * n)$

$\langle \text{proof} \rangle$

λ distributes over the product of coprime numbers similarly to φ , but with LCM instead of multiplication:

lemma *Carmichael-mult-coprime*:

assumes *coprime m n*

shows $\text{Carmichael } (m * n) = \text{lcm } (\text{Carmichael } m) (\text{Carmichael } n)$

$\langle \text{proof} \rangle$

lemma *Carmichael-pos* [*simp, intro*]: $\text{Carmichael } n > 0$

$\langle \text{proof} \rangle$

lemma *Carmichael-nonzero* [*simp*]: $\text{Carmichael } n \neq 0$

$\langle \text{proof} \rangle$

lemma *power-Carmichael-eq-1*:

assumes $n > 1$ *coprime n x*

shows $[x ^ \text{Carmichael } n = 1] \pmod n$

$\langle \text{proof} \rangle$

lemma *Carmichael-2* [*simp*]: $\text{Carmichael } 2 = 1$

$\langle \text{proof} \rangle$

lemma *Carmichael-4* [*simp*]: $\text{Carmichael } 4 = 2$

$\langle \text{proof} \rangle$

lemma *residue-primroot-Carmichael*:

assumes *residue-primroot n g*

shows $\text{Carmichael } n = \text{totient } n$

$\langle \text{proof} \rangle$

lemma *Carmichael-odd-prime-power*:

assumes *prime p odd p k > 0*

shows $\text{Carmichael } (p ^ k) = p ^{(k-1)} * (p-1)$

$\langle \text{proof} \rangle$

lemma *Carmichael-prime*:

assumes *prime p*

shows $\text{Carmichael } p = p-1$

$\langle \text{proof} \rangle$

lemma *Carmichael-twopow-ge-8*:

assumes $k \geq 3$

shows $\text{Carmichael } (2 ^ k) = 2 ^{(k-2)}$

$\langle \text{proof} \rangle$

lemma *Carmichael-twopow*:

$\text{Carmichael } (2 ^ k) = (\text{if } k \leq 2 \text{ then } 2 ^{(k-1)} \text{ else } 2 ^{(k-2)})$

$\langle \text{proof} \rangle$

lemma *Carmichael-prime-power*:

assumes *prime* p $k > 0$
shows $\text{Carmichael } (p \wedge k) =$
 $(\text{if } p = 2 \wedge k > 2 \text{ then } 2 \wedge (k - 2) \text{ else } p \wedge (k - 1) * (p - 1))$
 $\langle \text{proof} \rangle$

lemma *Carmichael-prod-coprime*:

assumes *finite* $A \wedge i j. i \in A \implies j \in A \implies i \neq j \implies \text{coprime } (f i) (f j)$
shows $\text{Carmichael } (\prod_{i \in A} f i) = (\text{LCM } i \in A. \text{Carmichael } (f i))$
 $\langle \text{proof} \rangle$

Since λ distributes over coprime factors and we know the value of $\lambda(p^k)$ for prime p , we can now give a closed formula for $\lambda(n)$ in terms of the prime factorisation of n :

theorem *Carmichael-closed-formula*:

$\text{Carmichael } n =$
 $(\text{LCM } p \in \text{prime-factors } n. \text{let } k = \text{multiplicity } p \text{ in } \text{if } p = 2 \wedge k > 2 \text{ then } 2 \wedge (k - 2) \text{ else } p \wedge (k - 1) * (p - 1))$
 $(\text{is } - = \text{Lcm } ?A)$
 $\langle \text{proof} \rangle$

corollary *even-Carmichael*:

assumes $n > 2$
shows $\text{even } (\text{Carmichael } n)$
 $\langle \text{proof} \rangle$

lemma *eval-Carmichael*:

assumes *prime-factorization* $n = A$
shows $\text{Carmichael } n = (\text{LCM } p \in \text{set-mset } A. \text{let } k = \text{count } A \text{ } p \text{ in } \text{if } p = 2 \wedge k > 2 \text{ then } 2 \wedge (k - 2) \text{ else } p \wedge (k - 1) * (p - 1))$
 $\langle \text{proof} \rangle$

Any residue ring always contains a λ -root, i.e. an element whose order is $\lambda(n)$.

theorem *Carmichael-root-exists*:

assumes $n > (0::\text{nat})$
obtains g **where** $g \in \text{totatives } n$ **and** $\text{ord } n \text{ } g = \text{Carmichael } n$
 $\langle \text{proof} \rangle$

This also means that the Carmichael number is not only the LCM of the orders of the elements of the residue ring, but indeed the maximum of the orders.

lemma *Carmichael-altdef*:

$\text{Carmichael } n = (\text{if } n = 0 \text{ then } 1 \text{ else } \text{Max } (\text{ord } n \text{ ' totatives } n))$
 $\langle \text{proof} \rangle$

11.5 Existence of primitive roots for general moduli

We now related Carmichael's function to the existence of primitive roots and, in the end, use this to show precisely which moduli have primitive roots and which do not.

The first criterion for the existence of a primitive root is this: A primitive root modulo n exists iff $\lambda(n) = \varphi(n)$.

lemma *Carmichael-eq-totient-imp-primroot:*
assumes $n > 0$ **and** *Carmichael* $n = \text{totient } n$
shows $\exists g. \text{residue-primroot } n \ g$
<proof>

theorem *residue-primroot-iff-Carmichael:*
 $(\exists g. \text{residue-primroot } n \ g) \longleftrightarrow \text{Carmichael } n = \text{totient } n \wedge n > 0$
<proof>

Any primitive root modulo mn for coprime m, n is also a primitive root modulo m and n . The converse does not hold in general.

lemma *residue-primroot-modulus-mult-coprimeD:*
assumes *coprime* $m \ n$ **and** *residue-primroot* $(m * n) \ g$
shows *residue-primroot* $m \ g$ *residue-primroot* $n \ g$
<proof>

If a primitive root modulo mn exists for coprime m, n , then $\lambda(m)$ and $\lambda(n)$ must also be coprime. This is helpful in establishing that there are no primitive roots modulo mn by showing e.g. that $\lambda(m)$ and $\lambda(n)$ are both even.

lemma *residue-primroot-modulus-mult-coprime-imp-Carmichael-coprime:*
assumes *coprime* $m \ n$ **and** *residue-primroot* $(m * n) \ g$
shows *coprime* $(\text{Carmichael } m) (\text{Carmichael } n)$
<proof>

The following moduli are precisely those that have primitive roots.

definition *cyclic-moduli* :: *nat set* **where**
 $\text{cyclic-moduli} = \{1, 2, 4\} \cup \{p^k \mid p \text{ prime } p \wedge \text{odd } p \wedge k > 0\} \cup$
 $\{2 * p^k \mid p \text{ prime } p \wedge \text{odd } p \wedge k > 0\}$

theorem *residue-primroot-iff-in-cyclic-moduli:*
 $(\exists g. \text{residue-primroot } m \ g) \longleftrightarrow m \in \text{cyclic-moduli}$
<proof>

lemma *residue-primroot-is-generator:*
assumes $m > 1$ **and** *residue-primroot* $m \ g$
shows *bij-betw* $(\lambda i. g^i \text{ mod } m) \{..<\text{totient } m\} (\text{totatives } m)$
<proof>

Given one primitive root g , all the primitive roots are powers g^i for $1 \leq i \leq \varphi(n)$ with $\gcd(i, \varphi(n)) = 1$.

theorem *residue-primroot-bij-betw-primroots:*
assumes $m > 1$ **and** *residue-primroot* m g
shows *bij-betw* $(\lambda i. g \wedge i \bmod m)$ $(\text{totatives } (\text{totient } m))$
 $\{g \in \text{totatives } m. \text{residue-primroot } m \ g\}$
 $\langle \text{proof} \rangle$

It follows from the above statement that any residue ring modulo n that has primitive roots has exactly $\varphi(\varphi(n))$ of them.

corollary *card-residue-primroots:*
assumes $\exists g. \text{residue-primroot } m \ g$
shows $\text{card } \{g \in \text{totatives } m. \text{residue-primroot } m \ g\} = \text{totient } (\text{totient } m)$
 $\langle \text{proof} \rangle$

corollary *card-residue-primroots':*
 $\text{card } \{g \in \text{totatives } m. \text{residue-primroot } m \ g\} =$
 $(\text{if } m \in \text{cyclic-moduli} \text{ then } \text{totient } (\text{totient } m) \text{ else } 0)$
 $\langle \text{proof} \rangle$

As an example, we evaluate $\lambda(122200)$ using the prime factorisation.

lemma *Carmichael* $122200 = 1380$
 $\langle \text{proof} \rangle$

end

12 Modular Inverses

theory *Modular-Inverse*
imports *Cong HOL-Library.FuncSet*
begin

The following returns the unique number m such that $mn \equiv 1 \pmod{p}$ if there is one, i.e. if n and p are coprime, and otherwise 0 by convention.

definition *modular-inverse* **where**
 $\text{modular-inverse } p \ n = (\text{if } \text{coprime } p \ n \text{ then } \text{fst } (\text{bezout-coefficients } n \ p) \bmod p$
 $\text{else } 0)$

lemma *cong-modular-inverse1:*
assumes $\text{coprime } n \ p$
shows $[n * \text{modular-inverse } p \ n = 1] \pmod{p}$
 $\langle \text{proof} \rangle$

lemma *cong-modular-inverse2:*
assumes $\text{coprime } n \ p$
shows $[\text{modular-inverse } p \ n * n = 1] \pmod{p}$
 $\langle \text{proof} \rangle$

lemma *coprime-modular-inverse* [simp, intro]:
fixes $n :: 'a :: \{\text{euclidean-ring-gcd}, \text{unique-euclidean-semiring}\}$
assumes *coprime* n p
shows *coprime* (*modular-inverse* p n) p
 $\langle \text{proof} \rangle$

lemma *modular-inverse-int-nonneg*: $p > 0 \implies \text{modular-inverse } p (n :: \text{int}) \geq 0$
 $\langle \text{proof} \rangle$

lemma *modular-inverse-int-less*: $p > 0 \implies \text{modular-inverse } p (n :: \text{int}) < p$
 $\langle \text{proof} \rangle$

lemma *modular-inverse-int-eqI*:
fixes $x y :: \text{int}$
assumes $y \in \{0..<m\} [x * y = 1] \pmod m$
shows *modular-inverse* m $x = y$
 $\langle \text{proof} \rangle$

lemma *modular-inverse-1* [simp]:
assumes $m > (1 :: \text{int})$
shows *modular-inverse* m $1 = 1$
 $\langle \text{proof} \rangle$

lemma *modular-inverse-int-mult*:
fixes $x y :: \text{int}$
assumes *coprime* x m *coprime* y m $m > 0$
shows *modular-inverse* m ($x * y$) = (*modular-inverse* m y * *modular-inverse* m x) *mod* m
 $\langle \text{proof} \rangle$

lemma *bij-betw-int-remainders-mult*:
fixes $a n :: \text{int}$
assumes a : *coprime* a n
shows *bij-betw* $(\lambda m. a * m \text{ mod } n)$ $\{1..<n\}$ $\{1..<n\}$
 $\langle \text{proof} \rangle$

lemma *mult-modular-inverse-of-not-coprime* [simp]: $\neg \text{coprime } a \ m \implies \text{modular-inverse } m \ a = 0$
 $\langle \text{proof} \rangle$

lemma *mult-modular-inverse-eq-0-iff*:
fixes $a :: 'a :: \{\text{unique-euclidean-semiring}, \text{euclidean-ring-gcd}\}$
shows $\neg \text{is-unit } m \implies \text{modular-inverse } m \ a = 0 \longleftrightarrow \neg \text{coprime } a \ m$
 $\langle \text{proof} \rangle$

lemma *mult-modular-inverse-int-pos*: $m > 1 \implies \text{coprime } a \ m \implies \text{modular-inverse } m \ a > (0 :: \text{int})$
 $\langle \text{proof} \rangle$

```

lemma abs-mult-modular-inverse-int-less:  $m \neq 0 \implies |\text{modular-inverse } m \ a :: \text{int}|$ 
<  $|m|$ 
  <proof>

lemma modular-inverse-int-less':  $m \neq 0 \implies (\text{modular-inverse } m \ a :: \text{int}) < |m|$ 
  <proof>

end

```

13 Comprehensive number theory

```

theory Number-Theory
imports
  Fib
  Residues
  Eratosthenes
  Mod-Exp
  Quadratic-Reciprocity
  Pocklington
  Prime-Powers
  Residue-Primitive-Roots
  Modular-Inverse
begin

end

```