# Notable Examples in Isabelle/Pure

May 23, 2024

## 1 A simple formulation of First-Order Logic

The subsequent theory development illustrates single-sorted intuitionistic first-order logic with equality, formulated within the Pure framework.

**theory** *First_Order_Logic*
  **imports** *Pure*
**begin**

### 1.1 Abstract syntax

**typedecl** *i*
**typedecl** *o*

**judgment** *Trueprop* :: $o \Rightarrow prop$ (_ *5*)

### 1.2 Propositional logic

**axiomatization** *false* :: *o* (⊥)
  **where** *falseE* [*elim*]: $\bot \Longrightarrow A$

**axiomatization** *imp* :: $o \Rightarrow o \Rightarrow o$ (**infixr** ⟶ *25*)
  **where** *impI* [*intro*]: $(A \Longrightarrow B) \Longrightarrow A \longrightarrow B$
    **and** *mp* [*dest*]: $A \longrightarrow B \Longrightarrow A \Longrightarrow B$

**axiomatization** *conj* :: $o \Rightarrow o \Rightarrow o$ (**infixr** ∧ *35*)
  **where** *conjI* [*intro*]: $A \Longrightarrow B \Longrightarrow A \wedge B$
    **and** *conjD1*: $A \wedge B \Longrightarrow A$
    **and** *conjD2*: $A \wedge B \Longrightarrow B$

**theorem** *conjE* [*elim*]:
  **assumes** $A \wedge B$
  **obtains** *A* **and** *B*
⟨*proof*⟩

**axiomatization** *disj* :: $o \Rightarrow o \Rightarrow o$ (**infixr** $\vee$ *30*)
  **where** *disjE* [*elim*]: $A \vee B \Longrightarrow (A \Longrightarrow C) \Longrightarrow (B \Longrightarrow C) \Longrightarrow C$
    **and** *disjI1* [*intro*]: $A \Longrightarrow A \vee B$
    **and** *disjI2* [*intro*]: $B \Longrightarrow A \vee B$

**definition** *true* :: $o$ ($\top$)
  **where** $\top \equiv \bot \longrightarrow \bot$

**theorem** *trueI* [*intro*]: $\top$
  ⟨*proof*⟩

**definition** *not* :: $o \Rightarrow o$ ($\neg$ _ [*40*] *40*)
  **where** $\neg A \equiv A \longrightarrow \bot$

**theorem** *notI* [*intro*]: $(A \Longrightarrow \bot) \Longrightarrow \neg A$
  ⟨*proof*⟩

**theorem** *notE* [*elim*]: $\neg A \Longrightarrow A \Longrightarrow B$
  ⟨*proof*⟩

**definition** *iff* :: $o \Rightarrow o \Rightarrow o$ (**infixr** $\longleftrightarrow$ *25*)
  **where** $A \longleftrightarrow B \equiv (A \longrightarrow B) \wedge (B \longrightarrow A)$

**theorem** *iffI* [*intro*]:
  **assumes** $A \Longrightarrow B$
    **and** $B \Longrightarrow A$
  **shows** $A \longleftrightarrow B$
  ⟨*proof*⟩

**theorem** *iff1* [*elim*]:
  **assumes** $A \longleftrightarrow B$ **and** $A$
  **shows** $B$
⟨*proof*⟩

**theorem** *iff2* [*elim*]:
  **assumes** $A \longleftrightarrow B$ **and** $B$
  **shows** $A$
⟨*proof*⟩

## 1.3 Equality

**axiomatization** *equal* :: $i \Rightarrow i \Rightarrow o$ (**infixl** = *50*)
  **where** *refl* [*intro*]: $x = x$
    **and** *subst*: $x = y \Longrightarrow P\ x \Longrightarrow P\ y$

**theorem** *trans* [*trans*]: $x = y \Longrightarrow y = z \Longrightarrow x = z$
  ⟨*proof*⟩

**theorem** *sym* [*sym*]: $x = y \Longrightarrow y = x$
⟨*proof*⟩

### 1.4 Quantifiers

**axiomatization** *All* :: $(i \Rightarrow o) \Rightarrow o$ (**binder** ∀ *10*)
  **where** *allI* [*intro*]: $(\bigwedge x.\ P\ x) \Longrightarrow \forall x.\ P\ x$
    **and** *allD* [*dest*]: $\forall x.\ P\ x \Longrightarrow P\ a$

**axiomatization** *Ex* :: $(i \Rightarrow o) \Rightarrow o$ (**binder** ∃ *10*)
  **where** *exI* [*intro*]: $P\ a \Longrightarrow \exists x.\ P\ x$
    **and** *exE* [*elim*]: $\exists x.\ P\ x \Longrightarrow (\bigwedge x.\ P\ x \Longrightarrow C) \Longrightarrow C$

**lemma** $(\exists x.\ P\ (f\ x)) \longrightarrow (\exists y.\ P\ y)$
⟨*proof*⟩

**lemma** $(\exists x.\ \forall y.\ R\ x\ y) \longrightarrow (\forall y.\ \exists x.\ R\ x\ y)$
⟨*proof*⟩

**end**

# 2 Foundations of HOL

**theory** *Higher_Order_Logic*
  **imports** *Pure*
**begin**

The following theory development illustrates the foundations of Higher-Order Logic. The "HOL" logic that is given here resembles [2] and its predecessor [1], but the order of axiomatizations and defined connectives has be adapted to modern presentations of λ-calculus and Constructive Type Theory. Thus it fits nicely to the underlying Natural Deduction framework of Isabelle/Pure and Isabelle/Isar.

# 3 HOL syntax within Pure

**class** *type*
**default_sort** *type*

**typedecl** *o*
**instance** *o* :: *type* ⟨*proof*⟩
**instance** *fun* :: (*type*, *type*) *type* ⟨*proof*⟩

**judgment** *Trueprop* :: $o \Rightarrow prop$ (_ *5*)

# 4 Minimal logic (axiomatization)

**axiomatization** *imp* :: $o \Rightarrow o \Rightarrow o$ (**infixr** $\longrightarrow$ *25*)
  **where** *impI* [*intro*]: $(A \Longrightarrow B) \Longrightarrow A \longrightarrow B$
    **and** *impE* [*dest*, *trans*]: $A \longrightarrow B \Longrightarrow A \Longrightarrow B$

**axiomatization** *All* :: $('a \Rightarrow o) \Rightarrow o$ (**binder** $\forall$ *10*)
  **where** *allI* [*intro*]: $(\bigwedge x.\ P\ x) \Longrightarrow \forall x.\ P\ x$
    **and** *allE* [*dest*]: $\forall x.\ P\ x \Longrightarrow P\ a$

**lemma** *atomize_imp* [*atomize*]: $(A \Longrightarrow B) \equiv Trueprop\ (A \longrightarrow B)$
  ⟨*proof*⟩

**lemma** *atomize_all* [*atomize*]: $(\bigwedge x.\ P\ x) \equiv Trueprop\ (\forall x.\ P\ x)$
  ⟨*proof*⟩

### 4.0.1 Derived connectives

**definition** *False* :: *o*
  **where** $False \equiv \forall A.\ A$

**lemma** *FalseE* [*elim*]:
  **assumes** *False*
  **shows** *A*
⟨*proof*⟩

**definition** *True* :: *o*
  **where** $True \equiv False \longrightarrow False$

**lemma** *TrueI* [*intro*]: *True*
  ⟨*proof*⟩

**definition** *not* :: $o \Rightarrow o$ ($\neg$ _ [*40*] *40*)
  **where** $not \equiv \lambda A.\ A \longrightarrow False$

**lemma** *notI* [*intro*]:
  **assumes** $A \Longrightarrow False$
  **shows** $\neg\ A$
  ⟨*proof*⟩

**lemma** *notE* [*elim*]:
  **assumes** $\neg\ A$ **and** *A*
  **shows** *B*
⟨*proof*⟩

**lemma** *notE'*: $A \Longrightarrow \neg\ A \Longrightarrow B$
  ⟨*proof*⟩

**lemmas** *contradiction* = *notE notE′* — proof by contradiction in any order

**definition** *conj* :: $o \Rightarrow o \Rightarrow o$ (**infixr** ∧ *35*)
**where** $A \wedge B \equiv \forall C. (A \longrightarrow B \longrightarrow C) \longrightarrow C$

**lemma** *conjI* [*intro*]:
**assumes** $A$ **and** $B$
**shows** $A \wedge B$
⟨*proof*⟩

**lemma** *conjE* [*elim*]:
**assumes** $A \wedge B$
**obtains** $A$ **and** $B$
⟨*proof*⟩

**definition** *disj* :: $o \Rightarrow o \Rightarrow o$ (**infixr** ∨ *30*)
**where** $A \vee B \equiv \forall C. (A \longrightarrow C) \longrightarrow (B \longrightarrow C) \longrightarrow C$

**lemma** *disjI1* [*intro*]:
**assumes** $A$
**shows** $A \vee B$
⟨*proof*⟩

**lemma** *disjI2* [*intro*]:
**assumes** $B$
**shows** $A \vee B$
⟨*proof*⟩

**lemma** *disjE* [*elim*]:
**assumes** $A \vee B$
**obtains** $(a)$ $A$ | $(b)$ $B$
⟨*proof*⟩

**definition** *Ex* :: $('a \Rightarrow o) \Rightarrow o$ (**binder** ∃ *10*)
**where** $\exists x. P\ x \equiv \forall C. (\forall x. P\ x \longrightarrow C) \longrightarrow C$

**lemma** *exI* [*intro*]: $P\ a \Longrightarrow \exists x. P\ x$
⟨*proof*⟩

**lemma** *exE* [*elim*]:
**assumes** $\exists x. P\ x$
**obtains** (*that*) $x$ **where** $P\ x$
⟨*proof*⟩

### 4.0.2 Extensional equality

**axiomatization** $equal :: 'a \Rightarrow 'a \Rightarrow o$ (**infixl** $=$ *50*)
  **where** *refl* [*intro*]: $x = x$
    **and** *subst*: $x = y \Longrightarrow P\ x \Longrightarrow P\ y$

**abbreviation** *not_equal* $:: 'a \Rightarrow 'a \Rightarrow o$ (**infixl** $\neq$ *50*)
  **where** $x \neq y \equiv \neg\ (x = y)$

**abbreviation** *iff* $:: o \Rightarrow o \Rightarrow o$ (**infixr** $\longleftrightarrow$ *25*)
  **where** $A \longleftrightarrow B \equiv A = B$

**axiomatization**
  **where** *ext* [*intro*]: $(\bigwedge x.\ f\ x = g\ x) \Longrightarrow f = g$
    **and** *iff* [*intro*]: $(A \Longrightarrow B) \Longrightarrow (B \Longrightarrow A) \Longrightarrow A \longleftrightarrow B$
  **for** $f\ g :: 'a \Rightarrow 'b$

**lemma** *sym* [*sym*]: $y = x$ **if** $x = y$
  ⟨*proof*⟩

**lemma** [*trans*]: $x = y \Longrightarrow P\ y \Longrightarrow P\ x$
  ⟨*proof*⟩

**lemma** [*trans*]: $P\ x \Longrightarrow x = y \Longrightarrow P\ y$
  ⟨*proof*⟩

**lemma** *arg_cong*: $f\ x = f\ y$ **if** $x = y$
  ⟨*proof*⟩

**lemma** *fun_cong*: $f\ x = g\ x$ **if** $f = g$
  ⟨*proof*⟩

**lemma** *trans* [*trans*]: $x = y \Longrightarrow y = z \Longrightarrow x = z$
  ⟨*proof*⟩

**lemma** *iff1* [*elim*]: $A \longleftrightarrow B \Longrightarrow A \Longrightarrow B$
  ⟨*proof*⟩

**lemma** *iff2* [*elim*]: $A \longleftrightarrow B \Longrightarrow B \Longrightarrow A$
  ⟨*proof*⟩

## 4.1 Cantor's Theorem

Cantor's Theorem states that there is no surjection from a set to its powerset. The subsequent formulation uses elementary $\lambda$-calculus and predicate logic, with standard introduction and elimination rules.

**lemma** *iff_contradiction*:
  **assumes** $*$: $\neg\ A \longleftrightarrow A$
  **shows** $C$

⟨*proof*⟩

**theorem** *Cantor*: $\neg\ (\exists f :: 'a \Rightarrow 'a \Rightarrow o.\ \forall A.\ \exists x.\ A = f\ x)$
⟨*proof*⟩

### 4.2 Characterization of Classical Logic

The subsequent rules of classical reasoning are all equivalent.

**locale** *classical* =
**assumes** *classical*: $(\neg\ A \Longrightarrow A) \Longrightarrow A$
— predicate definition and hypothetical context
**begin**

**lemma** *classical_contradiction*:
**assumes** $\neg\ A \Longrightarrow False$
**shows** $A$
⟨*proof*⟩

**lemma** *double_negation*:
**assumes** $\neg\ \neg\ A$
**shows** $A$
⟨*proof*⟩

**lemma** *tertium_non_datur*: $A \vee \neg\ A$
⟨*proof*⟩

**lemma** *classical_cases*:
**obtains** $A \mid \neg\ A$
⟨*proof*⟩

**end**

**lemma** *classical_if_cases*: *classical*
**if** *cases*: $\bigwedge A\ C.\ (A \Longrightarrow C) \Longrightarrow (\neg\ A \Longrightarrow C) \Longrightarrow C$
⟨*proof*⟩

## 5 Peirce's Law

Peirce's Law is another characterization of classical reasoning. Its statement only requires implication.

**theorem** (**in** *classical*) *Peirce's_Law*: $((A \longrightarrow B) \longrightarrow A) \longrightarrow A$
⟨*proof*⟩

## 6 Hilbert's choice operator (axiomatization)

**axiomatization** *Eps* :: $('a \Rightarrow o) \Rightarrow 'a$
**where** *someI*: $P\ x \Longrightarrow P\ (Eps\ P)$

**syntax** *_Eps* :: *pttrn* ⇒ *o* ⇒ *'a* ((*3SOME* _./ _) [*0*, *10*] *10*)
**translations** *SOME x. P* ⇌ *CONST Eps* (*λx. P*)

It follows a derivation of the classical law of tertium-non-datur by means of Hilbert's choice operator (due to Berghofer, Beeson, Harrison, based on a proof by Diaconescu).

**theorem** *Diaconescu*: $A \vee \neg A$
⟨*proof*⟩

This means, the hypothetical predicate *classical* always holds unconditionally (with all consequences).

**interpretation** *classical*
⟨*proof*⟩

**thm** *classical*
*classical_contradiction*
*double_negation*
*tertium_non_datur*
*classical_cases*
*Peirce's_Law*

**end**

## References


[1] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[2] M. J. C. Gordon. HOL: A machine oriented formulation of higher order logic. Technical Report 68, University of Cambridge Computer Laboratory, 1985.