

# Notable Examples in Isabelle/Pure

September 11, 2023

## 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 ( $\perp$ )  
  where falseE [elim]:  $\perp \Longrightarrow A$ 
```

```
axiomatization imp :: o  $\Rightarrow$  o  $\Rightarrow$  o (infixr  $\longrightarrow$  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  $\wedge$  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  $\top \equiv \perp \longrightarrow \perp$

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

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

**theorem** *notI* [*intro*]:  $(A \Longrightarrow \perp) \Longrightarrow \neg A$   
 ⟨*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*]:  $x = y \implies y = z \implies x = z$   
⟨*proof*⟩

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

## 1.4 Quantifiers

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

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

**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 been adapted to modern presentations of  $\lambda$ -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* ::  $(\lambda a \Rightarrow o) \Rightarrow o$  (**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*]:  $(A \Longrightarrow B) \equiv \text{Trueprop } (A \longrightarrow B)$   
   $\langle \text{proof} \rangle$

**lemma** *atomize\_all* [*atomize*]:  $(\bigwedge x. P x) \equiv \text{Trueprop } (\forall x. P x)$   
   $\langle \text{proof} \rangle$

### 4.0.1 Derived connectives

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

**lemma** *FalseE* [*elim*]:  
  **assumes** *False*  
  **shows**  $A$   
   $\langle \text{proof} \rangle$

**definition** *True* ::  $o$   
  **where** *True*  $\equiv \text{False} \longrightarrow \text{False}$

**lemma** *TrueI* [*intro*]: *True*  
   $\langle \text{proof} \rangle$

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

**lemma** *notI* [*intro*]:  
  **assumes**  $A \Longrightarrow \text{False}$   
  **shows**  $\neg A$   
   $\langle \text{proof} \rangle$

**lemma** *notE* [*elim*]:  
  **assumes**  $\neg A$  **and**  $A$   
  **shows**  $B$   
   $\langle \text{proof} \rangle$

**lemma** *notE'*:  $A \Longrightarrow \neg A \Longrightarrow B$   
   $\langle \text{proof} \rangle$

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

**definition** *conj* ::  $o \Rightarrow o \Rightarrow o$  (**infixr**  $\wedge$  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**  $\vee$  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* ::  $(\lambda a. a \Rightarrow o) \Rightarrow o$  (**binder**  $\exists$  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 ⇒ 'a ⇒ o (infixl = 50)

**where** *refl* [*intro*]: x = x  
  **and** *subst*: x = y ⇒ P x ⇒ P y

**abbreviation** *not\_equal* :: 'a ⇒ 'a ⇒ o (infixl ≠ 50)

**where** x ≠ y ≡ ¬ (x = y)

**abbreviation** *iff* :: o ⇒ o ⇒ o (infixr ↔ 25)

**where** A ↔ B ≡ A = B

**axiomatization**

**where** *ext* [*intro*]: (∧x. f x = g x) ⇒ f = g  
  **and** *iff* [*intro*]: (A ⇒ B) ⇒ (B ⇒ A) ⇒ A ↔ B  
  **for** f g :: 'a ⇒ 'b

**lemma** *sym* [*sym*]: y = x **if** x = y

  ⟨*proof*⟩

**lemma** [*trans*]: x = y ⇒ P y ⇒ P x

  ⟨*proof*⟩

**lemma** [*trans*]: P x ⇒ x = y ⇒ 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 ⇒ y = z ⇒ x = z

  ⟨*proof*⟩

**lemma** *iff1* [*elim*]: A ↔ B ⇒ A ⇒ B

  ⟨*proof*⟩

**lemma** *iff2* [*elim*]: A ↔ B ⇒ B ⇒ 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** \*: ¬ A ↔ 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**  $\neg A \Longrightarrow \text{False}$

**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*  $\Rightarrow$  *o*  $\Rightarrow$  'a ((*3SOME* \_./ \_) [0, 10] 10)  
**translations** *SOME* *x. P*  $\equiv$  *CONST Eps* ( $\lambda x. P$ )

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.