Foundations of Higher-Order Logic with Classical Reasoning and Hilbert-Choice

Makarius Wenzel

December 2016

http://sketis.net/2016/release-candidates-for-isabelle2016-1

Introduction

History of Higher-Order Logic

Simple Type Theory

Alonzo Church (1940): *A Formulation of the Simple Theory of Types.* Journal of Symbolic Logic, volume 5, 1940

Peter Andrews (1986): *An Introduction to Mathematical Logic and Type Theory: to Truth through Proof.* Academic Press, 1986

William Farmer (2008): *The Seven Virtues of Simple Type Theory.* Journal of Applied Logic, volume 6, number 3, 2008
http://imps.mcmaster.ca/doc/seven-virtues.pdf

HOL

Michael Gordon (1985): *HOL: A machine oriented formulation of higher order logic.* University of Cambridge Computer Laboratory, technical report 68, 1985

Implementations of HOL

Cambridge HOL family

HOL4 (SML; M. Norrish et al): successor of HOL90, HOL88, HOL

ProofPower (SML; R. Arthan): commercially supported fork

HOL-Light (OCaml; J. Harrison): important re-implementation

HOL Zero (OCaml; M. Adams): another re-implementation

Isabelle/HOL

- alien to HOL family, equidistant to Coq
- Isabelle/HOL = Isabelle/Pure + HOL library and tools
- Isabelle/HOL ≈ Functional Programming + Logic
  see T. Nipkow: *Programming and Proving in Isabelle/HOL*
  http://isabelle.in.tum.de/doc/prog-prove.pdf

Quasi-programming in Isabelle/HOL

Approach:

1. define conventional types: tuples, records, recursive datatypes
2. define recursive functions over types (with well-formedness proofs)
3. simulate computation via equational reasoning:
   (a) term rewriting within the logic (Simplifier)
   (b) symbolic normalization by evaluation (NBE)
   (c) actual evaluation via code-generator:
   HOL subset is translated to SML, OCaml, Scala, Haskell

Example: `$ISABELLE_HOME/src/HOL/ex/Seq.thy`

export_code *conc* **in** *SML*
export_code *conc* **in** *OCaml*
export_code *conc* **in** *Scala*
export_code *conc* **in** *Haskell*

Isabelle foundations

Primitive inferences

Syntax (types and terms):

$fun :: (type,\ type)type$	function space $'a \Rightarrow\ 'b$
$all :: ('a \Rightarrow prop) \Rightarrow prop$	universal quantification $\bigwedge x{::}'a.\ B\ x$
$imp :: prop \Rightarrow prop \Rightarrow prop$	implication $A \Longrightarrow B$

Derivations (theorems): implicit theory Θ

$$\frac{A \in \Theta}{\vdash A}\,(axiom) \qquad \frac{}{A \vdash A}\,(assume)$$

$$\frac{\Gamma \vdash B[x] \quad x \notin \Gamma}{\Gamma \vdash \bigwedge x.\ B[x]}\,(\bigwedge\text{-}intro) \qquad \frac{\Gamma \vdash \bigwedge x.\ B[x]}{\Gamma \vdash B[a]}\,(\bigwedge\text{-}elim)$$

$$\frac{\Gamma \vdash B}{\Gamma - A \vdash A \Longrightarrow B}\,(\Longrightarrow\text{-}intro) \qquad \frac{\Gamma_1 \vdash A \Longrightarrow B \quad \Gamma_2 \vdash A}{\Gamma_1 \cup \Gamma_2 \vdash B}\,(\Longrightarrow\text{-}elim)$$

Object-logic rules

Main principles: (Paulson 1989)

- Pure syntax is notation for rules in Natural Deduction

  logical entailment: $A_1 \Longrightarrow \ldots A_n \Longrightarrow B$ represents $\dfrac{A_1 \ \ldots \ A_n}{B}$
  local parameter: $\bigwedge x.\ B\ x$ represents "eigen-variable" condition
  global parameter: $B\ ?x$ schematic variable for outermost $\bigwedge x$.

- Declarative rule composition via:
  - back-chaining
  - lifting into subgoal context
  - higher-order unification (G. Huet, D. Miller).

$\longrightarrow$ Isabelle/Pure reasoning similar to λ-Prolog

Inferences for rule composition

$$\frac{\overline{A} \Longrightarrow B \quad B' \Longrightarrow C \quad B\,\theta = B'\theta}{\overline{A}\,\theta \Longrightarrow C\,\theta}\ (compose)$$

$$\frac{\overline{A} \Longrightarrow B}{(\overline{H} \Longrightarrow \overline{A}) \Longrightarrow (\overline{H} \Longrightarrow B)}\ (\Longrightarrow\text{-}lift)$$

$$\frac{\overline{A}\ \overline{a} \Longrightarrow B\ \overline{a}}{(\bigwedge \overline{x}.\ \overline{A}\ (\overline{a}\ \overline{x})) \Longrightarrow (\bigwedge \overline{x}.\ B\ (\overline{a}\ \overline{x}))}\ (\bigwedge\text{-}lift)$$

$$\frac{\begin{array}{rl} rule: & \overline{A}\ \overline{a} \Longrightarrow B\ \overline{a} \\ goal: & (\bigwedge \overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow B'\ \overline{x}) \Longrightarrow C \\ goal\ unifier: & (\lambda \overline{x}.\ B\ (\overline{a}\ \overline{x}))\,\theta = B'\theta \end{array}}{(\bigwedge \overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow \overline{A}\ (\overline{a}\ \overline{x}))\,\theta \Longrightarrow C\,\theta}\ (resolution)$$

$$\frac{\begin{array}{rl} goal: & (\bigwedge \overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow A\ \overline{x}) \Longrightarrow C \\ assm\ unifier: & A\,\theta = H_i\,\theta\ \ \text{(for some } H_i\text{)} \end{array}}{C\,\theta}\ (assumption)$$

Structured Proofs

Natural Deduction: (Gentzen 1935)

$$\frac{A \longrightarrow B \quad A}{B} \qquad \frac{\begin{matrix}[A] \\ \vdots \\ B\end{matrix}}{A \longrightarrow B}$$

Isabelle/Pure rules: (Paulson 1989)

$$(A \longrightarrow B) \Longrightarrow A \Longrightarrow B \qquad (A \Longrightarrow B) \Longrightarrow A \longrightarrow B$$

Isabelle/Isar proofs: (Wenzel 1999)

have $A \longrightarrow B$ $\langle proof \rangle$
also have A $\langle proof \rangle$
finally have B **.**

have $A \longrightarrow B$
proof
  assume A
  then show B $\langle proof \rangle$
qed

Isar proof decomposition as Pure inference

have $\bigwedge\overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow B'\,\overline{x}$
proof $-$
  fix $\overline{a}$
  assume $\overline{G}\ \overline{a}$
  show $B\ \overline{a}\ \ \langle proof \rangle$
qed

$$\frac{\begin{array}{rll} subgoal\colon & (\bigwedge\overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow B'\,\overline{x}) \Longrightarrow C & \\ subproof\colon & \overline{G}\ \overline{a} \Longrightarrow B\ \overline{a} & \text{for schematic } \overline{a} \\ concl\ unifier\colon & (\lambda\overline{x}.\ B\ (\overline{a}\ \overline{x}))\,\theta = B'\theta & \\ assm\ unifiers\colon & (\lambda\overline{x}.\ G_j\ (\overline{a}\ \overline{x}))\,\theta = H_i\,\theta & \text{for each } G_j \text{ some } H_i \end{array}}{C\,\theta}$$

Example: rules for predicate logic (1)

theorem $impI$: $(A \Longrightarrow B) \Longrightarrow A \longrightarrow B$
theorem mp: $A \longrightarrow B \Longrightarrow A \Longrightarrow B$

theorem $allI$: $(\bigwedge x.\ B\ x) \Longrightarrow \forall\, x.\ B\ x$
theorem $spec$: $\forall\, x.\ B\ x \Longrightarrow B\ a$

theorem exI: $B\ a \Longrightarrow \exists\, x.\ B\ x$
theorem exE: $\exists\, x.\ B\ x \Longrightarrow (\bigwedge a.\ B\ a \Longrightarrow C) \Longrightarrow C$

theorem $conjI$: $A \Longrightarrow B \Longrightarrow A \wedge B$
theorem $conjE$: $A \wedge B \Longrightarrow (A \Longrightarrow B \Longrightarrow C) \Longrightarrow C$

theorem $disjI1$: $A \Longrightarrow A \vee B$
theorem $disjI2$: $B \Longrightarrow A \vee B$
theorem $disjE$: $A \vee B \Longrightarrow (A \Longrightarrow C) \Longrightarrow (B \Longrightarrow C) \Longrightarrow C$

Example: rules for predicate logic (2)

theorem TrueI: True
theorem FalseE: $\mathit{False} \Longrightarrow C$

theorem notI: $(A \Longrightarrow \mathit{False}) \Longrightarrow \neg\ A$
theorem notE: $\neg\ A \Longrightarrow A \Longrightarrow C$

theorem iffI: $(A \Longrightarrow B) \Longrightarrow (B \Longrightarrow A) \Longrightarrow A \longleftrightarrow B$
theorem $\mathit{iffD1}$: $A \longleftrightarrow B \Longrightarrow A \Longrightarrow B$
theorem $\mathit{iffD2}$: $A \longleftrightarrow B \Longrightarrow B \Longrightarrow A$

Example: structured proofs (1)

theorem $curry$: $(A \wedge B \longrightarrow C) \longrightarrow (A \longrightarrow B \longrightarrow C)$
proof
  assume $*$: $A \wedge B \longrightarrow C$
  show $A \longrightarrow B \longrightarrow C$
  proof
    assume A
    show $B \longrightarrow C$
    proof
      assume B
      from $\langle A \rangle$ $\langle B \rangle$ **have** $A \wedge B$ **..**
      with $*$ **show** C **..**
    qed
  qed
qed

Example: structured proofs (1)

```
theorem iff_contradiction: C if ¬ A ⟷ A
proof (rule notE)
  show ¬ A
  proof
    assume A
    with ‹¬ A ⟷ A› have ¬ A ..
    from this and ‹A› show False ..
  qed
  with ‹¬ A ⟷ A› show A ..
qed

theorem A ⟹ ¬ ¬ A
  by iprover  — intuitionistic prover

theorem ¬ ¬ A ⟹ A
  by blast  — classical tableau prover
```

Foundations of Higher-Order Logic

Actual Isabelle/HOL

- `$ISABELLE_HOME/src/Tools/Code_Generator.thy`
  - tool setup for quasi-programming in HOL
- `$ISABELLE_HOME/src/HOL/HOL.thy`
  - classic axiomatic basis (see Church, Gordon)
- `$ISABELLE_HOME/src/HOL/Nat.thy`
  - axiom of infinity
  - definition of natural numbers
- `$ISABELLE_HOME/src/HOL/Hilbert_Choice.thy`
  - Hilbert's epsilon operator (universal choice function)
- tons of definitions, statements, proofs
- tons of add-on tools

Pure bootstrap of HOL

1. HOL syntax within Pure
2. Minimal logic (axiomatization)
3. Derived connectives
4. Extensional equality (axiomatization)
5. Example: Cantor's Theorem
6. Example: characterization of classical logic
7. Hilbert's choice operator (axiomatization)
   $\leadsto$ proof of $tertium\_non\_datur$ using Hilbert's choice