Chapter 3

Foundations of Higher-Order Logic

Overview

1. Introduction to higher-order logic
2. Foundation of higher-order logic
3. Conservative extension of theories

Introduction
3. Foundations of Higher-Order Logic

3.1 Introduction

A bit of history and context

- Gottlob Frege proposed a system on which (he thought) all mathematics could be derived (in principle): Begriffsschrift (1879)
- Bertrand Russell found paradox in Frege's system and proposed the Ramified Theory of Types
- Wrote Principia Mathematica with Whitehead, an attempt at developing basic mathematics completely formally ("My intellect never recovered from the strain")

Russel's paradox

Theorem

Let \( S = \{ x \mid x \not\in x \} \), then \( S \in S \) if and only if \( S \not\in S \)

Proof.

- If \( S \in S \), then \( S \not\in S \).
- If \( S \not\in S \), then \( S \in S \).

Remark

- Thus, we found a mathematical contradiction.
- Logical point of view: we derived \( F \leftrightarrow \neg F \) where \( F \equiv (S \in S) \); thus, we can derive \textit{False}, and consequently, every formula.
- To solve the problem, it is not sufficient to

Approaches to avoid inconsistencies

- Type theory:
  - Russel: Use a hierarchy of types to avoid self-referential expressions
  - A. Church proposed a simple type theory (1940)
  - many approaches extend Church’s type theory (HOL, Calculus of constructions, etc.)
- Set theory is often seen as the basis for mathematics.
  - Zermelo-Fraenkel, Bernays-Goedel, ...
  - Set theories distinguish between sets and classes.
  - Consistency maintained as some collections are „too big“ to be sets, e.g., class of all sets is not a set. A class cannot belong to another class (let alone a set)! Set theory

Aspects of HOL

- Higher-order logic (HOL) is an expressive foundation for
  - mathematics: analysis, algebra, ...
  - computer science: program correctness, hardware verification, ...
- Reasoning in HOL is classical.
- Still important: modeling of problems (now in HOL).
- Still important: deriving relevant reasoning principles.

Remark

Web-page listing approaches to formalize mathematics and logics:
3. Foundations of Higher-Order Logic

3.1 Introduction

Aspects of HOL (2)

• HOL offers safety through strength:
  ▶ small kernel of constants and axioms
  ▶ safety via conservative (definitional) extensions

• Contrast with
  ▶ weaker logics (e.g., propositional logic, FOL): can’t define much
  ▶ axiomatic extensions: can lead to inconsistency

Bertrand Russell:
“[The method of “postulating” what we want has many advantages; they are the same as the advantages of theft over honest toil.]”
(Introduction to Mathematical Philosophy, 1919)

Rationale for Isabelle/HOL

We use Isabelle/HOL, the HOL specialization of the generic proof assistant Isabelle:

• HOL vs. set theory:
  ▶ types are helpful for computer science applications
  ▶ HOL is sufficiently expressive for most applications (in general, ZF set theory is more expressive)
  ▶ “If you prefer ML to Lisp, you will probably prefer HOL to ZF” (quote by Larry Paulson)

• Isabelle/HOL vs. other HOL systems: pragmatic advantages over the HOL system or PVS

• Constructive alternatives for HOL: Coq or Nuprl, classical reasoning not supported

About the term „higher-order logic“

1st-order: supports functions and predicates over individuals (0th-order objects) and quantification of individuals:

\[ \forall x, y. R(x, y) \rightarrow R(y, x) \]

2nd-order: supports functions and predicates that have first-order functions as arguments or results and allow quantification over first-order predicates and functions:

\[ \forall P. \forall m. P^0 \land (\forall n. P^n \rightarrow P(Sucn)) \rightarrow P^m \]

„higher order“ \(\leftrightarrow\) union of all finite orders

Foundation of HOL

Section 3.2
Starting remarks

Simplification
In the rest of this chapter, we only consider
• a core syntax of HOL (not the rich syntax of Isabelle/HOL)
• a version of HOL without parameterized types (not the richer type system of Isabelle/HOL; cf. [GordonMelham93] for a version with parametric polymorphism)

Goals:
• Learn the semantics and axiomatic foundation of HOL
• Learn some meta-level properties about HOL
• Deepen the understanding of what verification is about

Basic HOL Syntax (1)

Types:
\[ \tau ::= \text{bool} \mid \text{ind} \mid \tau \Rightarrow \tau \]
- \text{bool} and \text{ind} are also called \( o \) and \( i \) in the literature [Chu40, And86]
- no user-defined type constructors, e.g., \text{bool list}
- no polymorphic type definitions, e.g., \( \alpha \text{ list} \)

Terms: Let \( \mathcal{V} \) be a set of variables and \( C \) a set of constants:
\[ \mathcal{T} ::= \mathcal{V} \mid C \mid (\mathcal{T} \mathcal{T}) \mid \lambda \mathcal{V}. \mathcal{T} \]
- Terms are simply-typed.
- Terms of type \text{bool} are called (well-formed) formulas.

Basic HOL Syntax (2)

The constants of HOL are typed and include at least:
- \( \text{True}, \text{False} :: \text{bool} \)
- \( _=_{=} :: \alpha \Rightarrow \alpha \Rightarrow \text{bool} \) (for all types \( \alpha \in \tau \))
- \( _\leadsto_{=} :: \text{bool} \Rightarrow \text{bool} \Rightarrow \text{bool} \)
- \( \iota \cdot :: (\alpha \Rightarrow \text{bool}) \Rightarrow \alpha \) (for all types \( \alpha \in \tau \))

\( \iota \cdot p \) yields the unique element \( x \) for which \( (p \ x) \) is \text{True}, if such a unique \( x \) exists. Otherwise, it yields an arbitrary value (of type \( \alpha \)).

Note that in Isabelle/HOL, the provisos „for all types \( \alpha \in \tau \)“ can be expressed by type variables.

HOL Semantics

Intuitively an extension of many-sorted semantics with functions
- FOL (w/o sorts): formulas are interpreted in a structure consisting of a domain/universe and functions/predicates
\[ (\mathcal{D}, (f_i)_{i \in \mathcal{F}}, (p_j)_{j \in \mathcal{P}}) \]
- Many-sorted FOL: there is a domain for each sort \( s \in S \) where \( S \) is finite; functions/predicates have a sorted signature:
\[ ((\mathcal{D}_s)_{s \in S}, (f_i)_{i \in \mathcal{F}}, (p_j)_{j \in \mathcal{P}}) \]
- HOL: domain \( \mathcal{D} \) is indexed by (infinitely many) types

Our presentation ignores polymorphism on the object-logical level, it is treated on the meta-level, though (for a version covering object-level parametric polymorphism cf. [GordonMelham93]).
Universes are prerequisite for HOL models

**Definition (Universe)**
A collection of sets \( \mathcal{U} \) is called a **universe**, if it satisfies the following closure conditions:

- **Inhab**: Each \( X \in \mathcal{U} \) is a nonempty set
- **Sub**: If \( X \in \mathcal{U} \) and \( Y \neq \emptyset \subseteq X \), then \( Y \in \mathcal{U} \)
- **Prod**: If \( X, Y \in \mathcal{U} \) then \( X \times Y \in \mathcal{U} \) where \( X \times Y \) is the Cartesian product (\( \{ \{ x \}, \{ x, y \} \} \) encodes \( (x, y) \))
- **Pow**: If \( X \in \mathcal{U} \) then \( \mathcal{P}(X) = \{ Y : Y \subseteq X \} \in \mathcal{U} \)
- **Infty**: \( \mathcal{U} \) contains an infinite set of individuals

Remarks on universes \( \mathcal{U} \)

- **Representation of function spaces in universes**: \( X \Rightarrow Y \) is the set of all (total) functions from \( X \) to \( Y \) where a function is represented by its graph

- Universes have two distinguished sets:
  - **Unit**: A distinguished set \( \{1\} \) with exactly one element
  - **Bool**: A distinguished set \( \{T, F\} \) with exactly two element sets (existence follows from Infty and Sub)

Frames

**Definition (frame)**
Let \( \mathcal{U} \) be a universe.
A frame is a collection \( (\mathcal{D}_\alpha)_{\alpha \in \tau} \) with \( \mathcal{D}_\alpha \in \mathcal{U} \) for all \( \alpha \in \tau \) and
- \( \mathcal{D}_{\text{bool}} = \{T, F\} \)
- \( \mathcal{D}_{\text{ind}} = X \) where \( X \) is some infinite set of individuals
- \( \mathcal{D}_{\alpha \Rightarrow \beta} \subseteq \mathcal{D}_\alpha \Rightarrow \mathcal{D}_\beta \), i.e. some collection of functions from \( \mathcal{D}_\alpha \) to \( \mathcal{D}_\beta \)

Examples
Some of the subsets \( \mathcal{D}_{\alpha \Rightarrow \beta} \) might contain, e.g.,
- the identity function, others do not
- only the computable functions

Interpretations

**Definition (Interpretation)**
An **interpretation** \( (\mathcal{D}_\alpha, \mathcal{J}) \) consists of a frame \( (\mathcal{D}_\alpha)_{\alpha \in \tau} \) and a function \( \mathcal{J} \) mapping the constants of type \( \alpha \) to elements of \( \mathcal{D}_\alpha \):
- \( \mathcal{J}(\text{True}) = T \) and \( \mathcal{J}(\text{False}) = F \)
- \( \mathcal{J}(=_{\alpha \Rightarrow \text{bool}}) \) is the identity on \( \mathcal{D}_\alpha \)
- \( \mathcal{J}(\rightarrow_{\text{bool}}{\text{bool}}) \) denotes the implication function over \( \mathcal{D}_{\text{bool}} \), i.e.,

\[
    b \rightarrow b' = \begin{cases} 
        F & \text{if } b = T \text{ and } b' = F \\
        T & \text{otherwise}
    \end{cases}
\]
Generalized Models - Facts (2)

- If $\mathcal{M}$ is a general model and $\rho$ a variable assignment, then $\mathcal{V}^\mathcal{M}(\rho, t)$ is uniquely determined, for every term $t$. $\mathcal{V}^\mathcal{M}(\rho, t)$ is the value of $t$ in $\mathcal{M}$ w.r.t. $\rho$.
- Gives rise to the standard notion of satisfiability/validity:
  - We write $\mathcal{V}^\mathcal{M}, \rho \models \phi$ for $\mathcal{V}^\mathcal{M}(\rho, \phi) = T$.
  - $\phi$ is satisfiable in $\mathcal{M}$ if $\mathcal{V}^\mathcal{M}, \rho \models \phi$ for some variable assignment $\rho$.
  - $\phi$ is valid in $\mathcal{M}$ if $\mathcal{V}^\mathcal{M}, \rho \models \phi$, for every variable assignment $\rho$.
  - $\phi$ is valid (in the general sense) if $\phi$ is valid in every general model $\mathcal{M}$.

Generalized models

Definition (Generalized models)
An interpretation $\mathcal{M} = (\langle D_\alpha \rangle_{\alpha \in \mathbb{N}}, \mathcal{F})$ is a (general) model for HOL if there is a binary function $\mathcal{V}^\mathcal{M}$ such that for all type-indexed families of variable assignments $\rho = (\rho_\alpha)_{\alpha \in \mathbb{N}}$:

1. $\mathcal{V}^\mathcal{M}(\rho, x_0) = \rho_0(\mathsf{x}_0)$
2. $\mathcal{V}^\mathcal{M}(\rho, c) = \mathcal{F}(c)$, for constants $c$
3. $\mathcal{V}^\mathcal{M}(\rho, s_{\alpha=0} \mathsf{t}_I) = \mathcal{V}^\mathcal{M}(\rho, s)\mathcal{V}^\mathcal{M}(\rho, t)$
   - i.e., the value of the function $\mathcal{V}^\mathcal{M}(\rho, s)$ at the argument $\mathcal{V}^\mathcal{M}(\rho, t)$
4. $\mathcal{V}^\mathcal{M}(\lambda \mathsf{x}_0. \mathsf{t}_I) = \mathsf{t}_I$ where $\mathcal{V}^\mathcal{M}(\lambda \mathsf{x}_0. \mathsf{t}_I, \rho) = 5$.
5. $\mathcal{V}^\mathcal{M}(\rho, \lambda \mathsf{x}_0. \mathsf{t}_I) = \mathcal{V}^\mathcal{M}(\rho, \mathsf{t}_I)$ where $\mathcal{V}^\mathcal{M}(\rho, \mathsf{t}_I) = 5$.

- Not all interpretations are general models.
- Closure conditions guarantee that every well-formed term has a value under every assignment, e.g.,
  - closure under functions: identity function from $D_\alpha$ to $D_\alpha$ must belong to $D_{\alpha\rightarrow \alpha}$ so that $\mathcal{V}^\mathcal{M}(\rho, \lambda \mathsf{x}_0. \mathsf{t}_I)$ is defined.
  - closure under application:
    - if $D_N$ is set of natural numbers and
    - $D_{N \times N \rightarrow N}$ contains addition function $\mathsf{p}$ where $\mathsf{p} x y = x + y$
    - then $D_{N \times N \rightarrow N}$ must contain $k x = 2x + 5$
    - since $k = \mathcal{V}^\mathcal{M}(\rho, \lambda \mathsf{x}. \mathsf{t}_I(\mathsf{t} x y) \mathsf{t}_I(\mathsf{t} x y))$ where $\rho(\mathsf{t}) = \mathsf{p}$ and $\rho(y) = 5$.
Standard models

Definition (Standard Models:)
A general model is a standard model iff for all $\alpha, \beta \in \tau$, $D_{\alpha} \Rightarrow D_{\beta}$ is the set of all functions from $D_{\alpha}$ to $D_{\beta}$.

Remarks
- A standard model is a general model, but not necessarily vice versa.
- Analogous definitions for satisfiability and validity w.r.t. standard models.
- We can now re-introduce HOL in Isabelle's meta-logic.

Isabelle/HOL

The syntax of the core language is introduced by:

- **consts**
  - `True :: bool`
  - `False :: bool`
  - `Not :: bool => bool` ("\~\_\_" [40] 40)
  - `If :: [bool, 'a, 'a] => 'a` ("if _ then _ else _")
  - `The :: ('a => bool) => 'a` (binder "THE" 10)
  - `All :: ('a => bool) => bool` (binder "\_" 10)
  - `Ex :: ('a => bool) => bool` (binder "\exists\" 10)
  - `= :: ['a,'a]` (infixl 50)
  - `\& :: [bool, bool] => bool` (infixr 35)
  - `\| :: [bool, bool] => bool` (infixr 30)
  - `\rightarrow :: [bool, bool] => bool` (infixr 25)

- **defs**
  - `True_def: True ≡ (\(\lambda x :: bool.x\) = (\(\lambda x.x\)))`
  - `All_def: All(P) ≡ (P = (\(\lambda x. True\)))`
  - `Ex_def: Ex(P) ≡ \& Q.(\forall x.Px \rightarrow Q) \rightarrow Q`
  - `False_def: False ≡ (\forall P.P)`
  - `not_def: \neg P ≡ P \rightarrow False`
  - `and_def: P \& Q ≡ \forall R.(P \rightarrow Q \rightarrow R) \rightarrow R`
  - `or_def: P \| Q ≡ \forall R.(P \rightarrow R) \rightarrow (Q \rightarrow R) \rightarrow R`
  - `if_def: If P x y ≡ THE z :: 'a.(P = True \rightarrow z = x)\&
  (P = False \rightarrow z = y)`

The axioms and rules of HOL

- **axioms/rules**
  - `refl: "t = t"
  - `subst: [s = t ; P(s) \rightarrow P(t)"
  - `ext: "(\& x. f x = g x) \rightarrow (\forall x.f x) = (\forall x.g x)"
  - `impl: "(P \rightarrow Q) \rightarrow P \rightarrow Q"
  - `mp: [P \rightarrow Q ; P] \rightarrow Q"
  - `iff: "(P \rightarrow Q) \rightarrow (Q \rightarrow P) \rightarrow (P = Q)"
  - `True_or_False: "(P = True) \& (P = False)"
  - `the_eq_trivial: "(THE x.x = b) = (b :: 'a)""