

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

Overview of Chapter

- 3. Foundations of Higher-Order Logic
 - 3.1 Introduction
 - 3.2 Foundation of HOL
 - 3.3 Conservative Extension of Theories

Section 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")

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

Remark

Web-page listing approaches to formalize mathematics and logics:

<http://www.cs.ru.nl/~freek/digimath/index.html>

Russel's paradox

Theorem

Let $S = \{x \mid x \notin x\}$, then $S \in S$ if and only if $S \notin S$

Proof.

- If $S \in S$, then $S \notin S$.
- If $S \notin S$, 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 *False*, and consequently, every formula.
- To solve the problem, it is not sufficient to

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.

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)

Choice of Isabelle/HOL

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:

$$\forall x, y. R(x, y) \longrightarrow R(y, x)$$

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

$$\forall P. \forall m. P0 \wedge (\forall n. Pn \longrightarrow P(\text{Suc}n)) \longrightarrow Pm$$

⋮

„higher order“ \leftrightarrow union of all finite orders

Section 3.2

Foundation of HOL

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:**

$$\tau ::= \mathit{bool} \mid \mathit{ind} \mid \tau \Rightarrow \tau$$

- ▶ *bool* and *ind* are also called *o* and *i* in the literature [Chu40, And86]
- ▶ no user-defined type constructors, e.g., *bool list*
- ▶ no polymorphic type definitions, e.g., $\alpha \mathit{list}$

- **Terms:** Let \mathcal{V} be a set of variables and \mathcal{C} a set of constants:

$$\mathcal{T} ::= \mathcal{V} \mid \mathcal{C} \mid (\mathcal{T}\mathcal{T}) \mid \lambda\mathcal{V}.\mathcal{T}$$

- ▶ **Terms are simply-typed.**
- ▶ Terms of type *bool* are called **(well-formed) formulas**.

Basic HOL Syntax (2)

- The **constants** of HOL are typed and include at least:

$$\mathit{True}, \mathit{False} ::= \mathit{bool}$$

$$_ = _ ::= \alpha \Rightarrow \alpha \Rightarrow \mathit{bool} \quad (\text{for all types } \alpha \in \tau)$$

$$_ \longrightarrow _ ::= \mathit{bool} \Rightarrow \mathit{bool} \Rightarrow \mathit{bool}$$

$$\iota _ ::= (\alpha \Rightarrow \mathit{bool}) \Rightarrow \alpha \quad (\text{for all types } \alpha \in \tau)$$

- ι is called the **description operator**:
 ιp yields the unique element x for which $(p x)$ is *True*, if such a unique x exists. Otherwise, it yields an arbitrary value (of type α).
- 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

$$\langle \mathcal{D}, (f_i)_{i \in F}, (p_i)_{i \in P} \rangle$$

- ▶ Many-sorted FOL: there is a domain for each sort $s \in S$ where S is finite; functions/predicates have a sorted signature:

$$\langle (\mathcal{D}_s)_{s \in S}, (f_i)_{i \in F}, (p_i)_{i \in P} \rangle$$

- ▶ 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 $Y \neq 0 \subseteq X$, 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
 - For X and Y nonempty, $X \Rightarrow Y$ is a nonempty subset of $\mathcal{P}(X \times Y)$
 - From closure conditions: If $X, Y \in \mathcal{U}$, then $X \Rightarrow Y \in \mathcal{U}$.
- 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

- $\mathcal{D}_{bool} = \{T, F\}$
- $\mathcal{D}_{ind} = X$ 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}_α to \mathcal{D}_β

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** $\langle (\mathcal{D}_\alpha)_{\alpha \in \tau}, \mathcal{J} \rangle$ consists of a frame $(\mathcal{D}_\alpha)_{\alpha \in \tau}$ and a function \mathcal{J} mapping the constants of type α to elements of \mathcal{D}_α :

- $\mathcal{J}(True) = T$ and $\mathcal{J}(False) = F$
- $\mathcal{J}(=_{\alpha \Rightarrow \alpha \Rightarrow bool})$ is the identity on \mathcal{D}_α
- $\mathcal{J}(\longrightarrow_{bool \Rightarrow bool \Rightarrow bool})$ denotes the implication function over \mathcal{D}_{bool} , i.e.,

$$b \longrightarrow b' = \begin{cases} F & \text{if } b = T \text{ and } b' = F \\ T & \text{otherwise} \end{cases}$$

Interpretations (2)

- $\mathcal{J}(t_{(\alpha \Rightarrow \text{bool}) \Rightarrow \alpha}) \in (\mathcal{D}_\alpha \Rightarrow \mathcal{D}_{\text{bool}}) \Rightarrow \mathcal{D}_\alpha$ denotes the function

$$\text{the}(\rho) = \begin{cases} a & \text{if } \rho = (\lambda x. x = a) \\ y & \text{otherwise, where } y \text{ is some element of } \mathcal{D}_\alpha \end{cases}$$

Remark

We have to make sure that

- the interpretations of the constants are elements of the frame
- all definable functions are elements of the frame

Generalized models

Definition (Generalized models)

An interpretation $\mathfrak{M} = \langle (\mathcal{D}_\alpha)_{\alpha \in \tau}, \mathcal{J} \rangle$ is a **(general) model for HOL** iff there is a binary function $\mathcal{V}^{\mathfrak{M}}$ such that for all type-indexed families of variable assignments $\rho = (\rho_\alpha)_{\alpha \in \tau}$:

- (a) $\mathcal{V}^{\mathfrak{M}}(\rho, x_\alpha) = \rho_\alpha(x_\alpha)$
- (b) $\mathcal{V}^{\mathfrak{M}}(\rho, c) = \mathcal{J}(c)$, for constants c
- (c) $\mathcal{V}^{\mathfrak{M}}(\rho, s_{\alpha \Rightarrow \beta} t_\alpha) = \mathcal{V}^{\mathfrak{M}}(\rho, s) \mathcal{V}^{\mathfrak{M}}(\rho, t)$
i.e., the value of the function $\mathcal{V}^{\mathfrak{M}}(\rho, s)$ at the argument $\mathcal{V}^{\mathfrak{M}}(\rho, t)$
- (d) $\mathcal{V}^{\mathfrak{M}}(\lambda x_\alpha. t_\beta) =$ “the function from \mathcal{D}_α into \mathcal{D}_β whose value for each $z \in \mathcal{D}_\alpha$ is $\mathcal{V}^{\mathfrak{M}}(\rho[x \leftarrow z], t)$ ”
- If t is a term of type α , then $\mathcal{V}^{\mathfrak{M}}(\rho, t) \in \mathcal{D}_\alpha$.

Generalized Models - Facts (1)

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

Generalized Models - Facts (2)

- 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 \mathcal{D}_α to \mathcal{D}_α must belong to $\mathcal{D}_{\alpha \Rightarrow \alpha}$ so that $\mathcal{V}^{\mathfrak{M}}(\rho, \lambda x_\alpha. x)$ is defined.

closure under application:

- ▶ if \mathcal{D}_N is set of natural numbers and
- ▶ $\mathcal{D}_{N \Rightarrow N \Rightarrow N}$ contains addition function p where $p \ x \ y = x + y$
- ▶ then $\mathcal{D}_{N \Rightarrow N}$ must contain $k \ x = 2x + 5$
since $k = \mathcal{V}^{\mathfrak{M}}(\rho, \lambda x. f (f \ x \ x) \ y)$ where $\rho(f) = p$ and $\rho(y) = 5$.

Standard models

Definition (Standard Models:)

A **general model** is a **standard model** iff for all $\alpha, \beta \in \tau$, $\mathcal{D}_{\alpha \Rightarrow \beta}$ is the set of **all** functions from \mathcal{D}_α to \mathcal{D}_β

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  $\Rightarrow$  bool      ("¬_" [40] 40)
If    :: [bool, 'a, 'a]  $\Rightarrow$  'a  ("if _ then _ else _")
The   :: ('a  $\Rightarrow$  bool)  $\Rightarrow$  'a   (binder "THE" 10)
All   :: ('a  $\Rightarrow$  bool)  $\Rightarrow$  bool (binder "∀" 10)
Ex    :: ('a  $\Rightarrow$  bool)  $\Rightarrow$  bool (binder "∃" 10)
=     :: ['a,'a]  $\Rightarrow$  bool       (infixl 50)
 $\wedge$  :: [bool, bool]  $\Rightarrow$  bool (infixr 35)
 $\vee$    :: [bool, bool]  $\Rightarrow$  bool (infixr 30)
 $\longrightarrow$  :: [bool, bool]  $\Rightarrow$  bool (infixr 25)

```

Core definitions of HOL

defs

```

True_def:  True       $\equiv$  (( $\lambda x :: \text{bool}.x$ ) = ( $\lambda x.x$ ))
All_def:   All(P)     $\equiv$  (P = ( $\lambda x.$  True))
Ex_def:    Ex(P)      $\equiv$   $\forall Q.(\forall x.Px \longrightarrow Q) \longrightarrow Q$ 
False_def: False     $\equiv$  ( $\forall P.P$ )
not_def:    $\neg P$       $\equiv$  P  $\longrightarrow$  False
and_def:   P  $\wedge$  Q   $\equiv$   $\forall R.(P \longrightarrow Q \longrightarrow R) \longrightarrow R$ 
or_def:    P  $\vee$  Q     $\equiv$   $\forall R.(P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$ 
if_def:    If P  $\times$  y  $\equiv$  THE z :: 'a.(P = True  $\longrightarrow$  z = x)  $\wedge$ 
                               (P = False  $\longrightarrow$  z = y)

```

The axioms and rules of HOL

axioms/rules

```

refl:      "t = t"
subst:    "[| s = t ; P(s) |]  $\Longrightarrow$  P(t)"

ext:      "( $\wedge x. f x = g x$ )  $\Longrightarrow$  ( $\lambda x. f x$ ) = ( $\lambda x. g x$ )"

impl:     "(P  $\Longrightarrow$  Q)  $\Longrightarrow$  P  $\longrightarrow$  Q"
mp:       "[| P  $\longrightarrow$  Q ; P |]  $\Longrightarrow$  Q"

iff:      "(P  $\longrightarrow$  Q)  $\longrightarrow$  (Q  $\longrightarrow$  P)  $\longrightarrow$  (P = Q)"
True_or_False: "(P = True)  $\vee$  (P = False)"

the_eq_trivial: "(THE x. x = b) = (b :: 'a)"

```