Chapter 3

Foundations of Higher-Order Logic

Overview of Chapter

Overview

1. Introduction to higher-order logic
2. Foundation of higher-order logic
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): Begriffssschrift (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 $S = \{x \mid x \notin x\}$, 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.

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

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:

$$\forall P.\forall m.\ P(0) \wedge (\forall n.\ P(n) \longrightarrow P(Suc(n))) \longrightarrow P(m)$$

$\vdots$

„higher order" $\leftrightsquigarrow$ 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., α *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 (no type parameters)
  - Terms of type *bool* are called (well-formed) formulas.

Basic HOL Syntax (2)

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

$$\begin{array}{lll} \mathit{True}, \mathit{False} :: \mathit{bool} & & \\ \_ = \_ :: \alpha \Rightarrow \alpha \Rightarrow \mathit{bool} & \text{(for all types } \alpha \in \tau\text{)} \\ \_ \longrightarrow \_ :: \mathit{bool} \Rightarrow \mathit{bool} \Rightarrow \mathit{bool} & \\ \iota \_ :: (\alpha \Rightarrow \mathit{bool}) \Rightarrow \alpha & \text{(for all types } \alpha \in \tau\text{)} \end{array}$$

- ι is called the description operator:
  $\iota\ p$ 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 exaclty 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 $\mathcal{D}_\alpha$ 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 $\langle (\mathcal{D}_\alpha)_{\alpha \in \tau}, \mathcal{J} \rangle$ consists of a frame $(\mathcal{D}_\alpha)_{\alpha \in \tau}$ and a function $\mathcal{J}$ mapping the constants of type α to elements of $\mathcal{D}_\alpha$:

- $\mathcal{J}(True) = T$ and $\mathcal{J}(False) = F$
- $\mathcal{J}(=_{\alpha \Rightarrow \alpha \Rightarrow bool})$ is the identity on $\mathcal{D}_\alpha$
- $\mathcal{J}(\longrightarrow_{bool \Rightarrow bool \Rightarrow bool})$ 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)

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

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

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 $\mathcal{D}_\alpha$ into $\mathcal{D}_\beta$ 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 $\mathcal{D}_\alpha$ to $\mathcal{D}_\alpha$ 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 $\mathcal{D}_\alpha$ to $\mathcal{D}_\beta$

Remarks

- A standard model is a general model, but not necessarily vice versa.
- Analogous definitions for satisfiability and validity w.r.t. standard models.

Isabelle/HOL

We introduce HOL in Isabelle's meta-logic:

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 "‘∃"' 10)
=	::	['a,'a] ⇒ bool	(infixl 50)
∧	::	[bool, bool] ⇒ bool	(infixr 35)
∨	::	[bool, bool] ⇒ bool	(infixr 30)
⟶	::	[bool, bool] ⇒ bool	(infixr 25)

Core definitions of HOL

defs

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

The axioms and rules of HOL

axioms/rules

refl:	"$t = t$"
subst:	"$[\![s = t\ ;\ P(s)]\!] \Longrightarrow P(t)$"
ext:	"$(\bigwedge x.\ f\ x = g\ x) \Longrightarrow (\lambda x.\ f\ x) = (\lambda x.\ g\ x)$"
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:	"$(P = \text{True}) \vee (P = \text{False})$"
the_eq_trivial:	"$(\text{THE } x.\ x = b) = (b :: 'a)$"

The axioms and rules of HOL (2)

Additionally, there is:

- universal α, β, and η congruence on terms (implicitly),
- the axiom of infinity, and
- the axiom of choice (Hilbert operator).
- This is the entire basis!

Properties of HOL

Theorem 1 (Soundness of HOL)

HOL is sound:

$\vdash \phi$ implies ϕ is valid in the general/standard sense

Theorem 2 (Incompleteness of HOL)

HOL is incomplete w.r.t. standard models:

There exist ϕ that are valid in the standard sense, but $\nvdash \phi$

Remark

[And86, Chap. 5-7] presents proofs for these theorems. Note, however, that [And86] does not restrict the semantics to models where $\mathcal{D}_{ind}$ is infinite.

Section 3.3

Conservative Extension of Theories

Basic ideas

- Theories are stepwise extension of the core theory of HOL
- Extensions may introduce new constants and new types
- Inconsistencies are avoided by construction
- Syntactical mechanisms are used to make extensions more convenient

Remark

Extensions only introduce names for “things” that already exist in the core theory.

Basic definitions

Terminology and basic definitions (cf. [GordonMelham93]):

Definition (Theory)

A (syntactic) *theory* T is a triple (χ, Σ, A) where

- χ is a set of type names
- Σ is a set of typed function/constant names using types of χ
- A is a set of axioms over Σ

Definition (Consistent)

A theory T is *consistent* iff *False* is not provable in T: $A \nvdash \mathit{False}$

Definition (Theory extension)

A theory $T' = (\chi', \Sigma', A')$ is an extension of a theory $T = (\chi, \Sigma, A)$ iff $\chi \subseteq \chi'$ and $\Sigma \subseteq \Sigma'$ and $A \subseteq A'$.

Basic definitions (cont.)

Definition (Conservative extension)

Let $T = (\chi, \Sigma, A)$ and $Th(T) = \{\phi \mid A \vdash \phi\}$;
a theory extension $T' = (\chi', \Sigma', A')$ of T is *conservative* iff

$$Th(T) = (Th(T') \mid_\Sigma)$$

where $\mid_\Sigma$ restricts sets of formulas to those containing only names in Σ.

Lemma (Consistency)

If T' is a conservative extension of a consistent theory T, then

$$\mathit{False} \notin Th(T')$$

Syntactic schemata for conservative extensions

Not every extension is conservative:

Counterexample

Let $T = (\chi, \Sigma, A)$ such that A includes the axioms of HOL and T is consistent.
$T' = (\chi, \Sigma, A \cup \{\forall f_{bool \Rightarrow bool}.x = f\ x\})$ is not a conservative extension of T.

We consider conservative extensions by:

- constant definitions
- type definitions

Remark

Cf. [GordonMelham93] for other extension schemata

Constant definitions

Definition (Constant definition)

A theory extension $T' = (\chi', \Sigma', A')$ of $T = (\chi, \Sigma, A)$ is called a constant definition iff

- $\chi' = \chi$ and $\Sigma' = \Sigma \cup \{c :: \alpha\}$ with $\alpha \in \chi$ and $c \notin \Sigma$
- $A' = A \cup \{c = E\}$
- E does not contain c (no recursion)
- E is closed (no free variables)
- (no subterm of E has a type containing a type variable that is not contained in the type of c)

Why side conditions?

- no recursion and closedness guarantee well-definedness
- Consider the following definition with a free type variable:

$$c = (\exists\, x :: 'a.\ \exists\, y :: 'a.\ x \neq y)$$

If the language allows to instantiate the type variables:

$$\begin{array}{ll} & c = c \qquad\qquad \text{(by refl)} \\ \Longrightarrow & (\exists\, x :: \mathit{bool}.\ \exists\, y :: \mathit{bool}.\ x \neq y) = (\exists\, x :: \mathit{Unit}.\ \exists\, y :: \mathit{Unit}.\ x \neq y) \\ \Longrightarrow & \mathit{True} = \mathit{False} \\ \Longrightarrow & \mathit{False} \end{array}$$

Constant definitions are conservative

Lemma (Constant definition)

A constant definition is a conservative extension.

Proof.

Proof sketch:

- $Th(T) \subseteq (Th(T')\,|_\Sigma)$: from definition of Th
- $(Th(T')\,|_\Sigma) \subseteq Th(T)$: let π' be a proof for $\phi \in (Th(T')\,|_\Sigma)$. We unfold any subterm in π' that contains c by $c = E$ into π. π is a proof in T, i.e., $\phi \in Th(T)$.

□

Constant definitions in Isabelle/HOL

Definitions of *True*, *False*, *All*, *Ex*, ¬, ∧, ∨, *if*, *let*:

True	::	bool	
False	::	bool	
Not	::	bool ⇒ bool	("‘¬_"' [40] 40)
If	::	[bool, 'a, 'a] ⇒ 'a	("‘if _ then _ else _)"')
Let	::	['a, 'a ⇒ 'b] ⇒ 'b	
The	::	('a ⇒ bool) ⇒ 'a	(binder "‘THE"' 10)
All	::	('a ⇒ bool) ⇒ bool	(binder "‘∀"' 10)
Ex	::	('a ⇒ bool) ⇒ bool	(binder "‘∃"' 10)
=	::	['a,'a] ⇒ bool	(infixl 50)
∧	::	[bool, bool] ⇒ bool	(infixr 35)
∨	::	[bool, bool] ⇒ bool	(infixr 30)
⟶	::	[bool, bool] ⇒ bool	(infixr 25)

Constant definitions in Isabelle/HOL (2)

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

Approaching type definitions

Idea

- Specify a subset of the elements of an existing type r
- "Copy" the subset and use the copy as value set of the new type t
- Link old and new type by two functions

More precisely, the definition of a new type t is based on:

- an existing type r
- a predicate $S :: r \Rightarrow bool$, defining a non-empty "subset" of r;
- an abstraction function $Abs_t :: r \Rightarrow t$
- a representation function $Rep_t :: t \Rightarrow r$
- axioms stating a bijection between the set characterized by S and the new type t.

Type definitions as theory extensions

Definition (Type definition)

Let $T = (\chi, \Sigma, A)$ be a theory and $r \in \chi$ and S a term of type $r \Rightarrow bool$.
A theory extension $T' = (\chi', \Sigma', A')$ of T is a type definition for t with $t \notin \chi$ iff

- $\chi' = \chi \cup \{t\}$
- $\Sigma' = \Sigma \cup \{\ Abs_t :: r \Rightarrow t,\ Rep_t :: t \Rightarrow r\ \}$
- $A' = A \cup \{\ \forall x.\ Abs_t(Rep_t\ x) = x,\ \ \forall y.\ S\ y \longrightarrow Rep_t(Abs_t\ y) = y\ \}$
- One has to prove $T \vdash\ \exists x.\ S\ x$ (using Isabelle/HOL)

Lemma (Type definition)

A type definition is a conservative extension.

For a proof see [GordonMelham93]

The nature of extensions

Remark

This may seem strange: if a new type is always isomorphic to a subset of an existing type, how is this construction going to lead to a "rich" collection of types for large-scale applications?

- But in fact, due to *ind* and $\Rightarrow$, the types in HOL are already very rich.
- Thus, extensions essentially give names to values and types that have already been "expressible" in the "old" theory.
- Extensions allow to formulate theorems in a more compact and readable way.

We now give two examples revealing the power of type definitions:

- Typed sets
- Pairs

Types for sets

We define the new type *natset* containing all sets of natural numbers:

- existing type: $(nat \Rightarrow bool)$
- predicate $S :: (nat \Rightarrow bool) \Rightarrow bool, S \equiv\ \lambda f.True$
- $\chi' = \chi \cup \{natset\}$
- $\Sigma' = \Sigma \cup \{\ Abs_{natset} :: (nat \Rightarrow bool) \Rightarrow natset,$
  $Rep_{natset} :: natset \Rightarrow (nat \Rightarrow bool)\ \}$
- $A' = A \cup \{\ \forall x.\ Abs_{natset}(Rep_{natset}\ x) = x,$
  $\forall y.\ True \longrightarrow Rep_{natset}(Abs_{natset}\ y) = y\ \}$
- One has to prove $T \vdash\ \exists x.(\lambda f.True)\ x$ (using Isabelle/HOL)

Remarks on the set type

Remarks

- Isabelle/HOL allows to define a parametric type α *set* where α is a type variable.
- Functions of type $\alpha \Rightarrow$ *bool* are used to represent sets, i.e., sets are represented by their characteristic function.
- In $(Abs_{\alpha\ set}\ f)$, the abstraction function $Abs_{\alpha\ set}$ can thus be read as "interpret f as a set".
- Here, sets are just an example to demonstrate type definitions. Later we study them for their own sake.

Approaching the types for pairs

Given some types α and β.
How can we represent pairs, i.e., define the type $\alpha \times \beta$?

Idea:

- Existing type: $\alpha \Rightarrow \beta \Rightarrow$ *bool*
- Represent pairs as functions of type $\alpha \Rightarrow \beta \Rightarrow$ *bool*)
- Use function $\lambda x :: \alpha.\ \lambda y :: \beta.\ x = a \wedge y = b$ to represent the pair (a, b)
- It is clear that there is exactly one function for each pair.
- There are also functions of type $\alpha \Rightarrow \beta \Rightarrow$ *bool* that do not represent a pair, i.e., we have to define a nontrivial S.

Types for pairs

We define the new type $\alpha \times \beta$:

- existing type: $\alpha \Rightarrow \beta \Rightarrow$ *bool*
- predicate $S \equiv \lambda f :: \alpha \Rightarrow \beta \Rightarrow$ *bool*.
  $$\exists a.\ \exists b.\ f = \lambda x :: \alpha.\ \lambda y :: \beta.\ x = a \wedge y = b$$
- $\chi' = \chi \cup \{\alpha \times \beta\}$

Remark

Isabelle/HOL provides a special syntax for type definitions.

Type definitions in Isabelle/HOL

Syntax for type definitions

```
typedef (typevars) T' = "{x. A(x)}"
```

Relation with explained schema:

- The new type is T'
- r is the type of x (inferred)
- S is $\lambda x.\ A\ x$
- Constants $Abs_{T'}$ and $Rep_{T'}$ are automatically generated.

Conservative extensions: Summary

- We have presented a method to safely build up larger theories:
  - Constant definitions
  - Type definitions
- Subtle side conditions
- New types must be isomorphic to a “subset” of an existing type.
- Isabelle/HOL uses these conservative extensions to
  - build up the theory `Main` from the core definitions of HOL (cf. Tutorials and manuals for Isabelle2011-1)
  - provide more convenient specialized syntax for conservative extensions (datatype, primrec, function, ...)

Conclusions of Chap. 3

- HOL generalizes semantics of FOL
  - *bool* serves as type of propositions
  - Syntax/semantics allows for higher-order functions
- Logic is rather minimal: 8 rules, more-or-less obvious
- Logic is very powerful in terms of what we can represent/derive.
  - Other “logical” syntax
  - Rich theories via conservative extensions

Questions

1. What is the foundational reason that HOL is typed? Are there other reasons w.r.t. an application in computer science?
2. What does “higher-order” mean?
3. Why is predicate logic not sufficient? Give an example?
4. What are the types in HOL?
5. What are the terms in HOL? Give examples of constants.
6. Explain the description operator.
7. What is a frame? What is an interpretation?
8. How is satisfiability defined?

Questions (2)

9. What is a standard model?
10. Give and explain one of the axioms of HOL?
11. Can the constants True and False be defined in HOL?
12. What does it mean that HOL+infinity is incomplete wrt. standard models?
13. What is a conservative extension?
14. What is the advantage of conservative extensions over axiomatic definitions?
15. Which syntactic schemata for conservative extensions were treated in the lecture?
16. Give examples of constant definitions.
17. Explain the definitions of new types?
18. Does a data type definition in Isabelle/HOL lead to a new type?