Preuves Interactives et Applications

Christine Paulin & Burkhart Wolff

http://www.lri.fr/ ~paulin/PreuvesInteractives

Université Paris-Saclay

HOL and its Specification Constructs

Revisions

- What is „typed λ-calculus"
- What is „β-reduction"
- Using typed λ-calculus to represent logical systems
- What is „natural deduction" ? (from another perspective)

Revisions: Typed λ-calculus

- Rules over a global signature Σ:

$$\frac{(c, \tau_c) \in \Sigma}{\rho \vdash c : \tau_c} \qquad \frac{}{x_1 : \tau_1, \ldots, x_p : \tau_p \vdash x_i : \tau_i}$$

$$\frac{\rho, x : \tau \vdash t : \sigma}{\rho \vdash \lambda x.t : \tau \to \sigma} \qquad \frac{\rho \vdash f : \tau \to \sigma \quad \rho \vdash t : \tau}{\rho \vdash f\,t : \sigma}$$

- We assume Σ =

{("_+_", nat→nat→nat), ("0", nat), ("Suc _", nat→nat),
("_=_", nat→nat→bool), ("True", bool), ("False", bool),
("_=_", bool→bool→bool)}

Revisions: Typed λ-calculus

- Examples: Are there variable environments ρ such that the following terms are typable in Σ: (note that we use infix notation: we write "0 + x" instead of "_+_ 0 x")

  - (_+_ 0) = (Suc x)
  - ((x + y) = (y + x)) = False
  - f(_+_ 0) = (λc. g c) x
  - _+_ z (_+_ (Suc 0)) = (0 + f False)
  - a + b = (True = c)

Revisions: β-reduction

- Assume that we want to find typed solutions for ?X, ?Y, ?Z such that the following terms become equivalent modulo α-conversion and β-reduction:
  - ?X a =?= a + ?Y
  - (λc. g c) =?= (λx. ?Y x)
  - (λc. ?X c) a =?= ?Y
  - λa. (λc. X c) a =?= (λx. ?Y)
- Note: Variables like ?X, ?Y, ?Z are called schematic variables; they play a major role in Isabelles Rule-Instantiation Mechanism
- Are the solutions for schematic variables always unique ?

Deduction

- Logic Whirl-Pool of the 20ies (Girard) as response to foundational problems in Mathematics

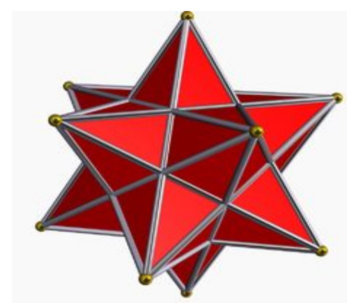
  - growing uneasiness over the question:

    What is a proof ?

    Are there limits of provability ?

Deduction

- Historical context in the 20ies:
  - 1500 false proofs of
    „all parallels do not intersect in infinity“
  - lots of proofs and refutations of
    „all polyhedrons are eularian“ (Lakatosz)



$E = F + K - 2$???

  - Frege`s axiomatic set theory proven inconsistent by Russel
  - Science vs. Marxism debate (Popper)

Deduction

- Historical context in the 20ies:
  - this seemed quite far away from Leipnitz vision of

    „Calculemus !“ (We don‘t agree ? Let‘s calculate ...)

    of what constitutes, well,

    Science ...

Deduction

- Historical context in the 20ies:
  - attempts to formalize the intuition of „deduction" by Frege, Hilbert, Russel, Lukasiewics, ...
  - 2 Calculi presented by Gerhard Gentzen in 1934.
    - „natürliches Schliessen" (natural deduction):

$$\frac{\begin{array}{c}[P] \\ \vdots \\ Q\end{array}}{R}$$

    - „Sequenzkalkül" (sequent calculus)

$$\frac{\Gamma \vdash A \vee B \quad \Gamma \cup \{A\} \vdash C \quad \Gamma \cup \{B\} \vdash C}{\Gamma \vdash C}$$

Deduction

- An Inference System (or Logical Calculus) allows to infer formulas from a set of elementary judgements (axioms) and inferred judgements by rules:

$$\frac{A_1 \quad \ldots \quad A_n}{A_{n+1}}$$

"from the assumptions A_1 to A_n, you can infer the conclusion A_{n+1}." A rule with n=0 is an elementary fact. Variables occurring in the formulas A_n can be arbitrarily substituted.

Deduction

- judgements discussed in this course (or elsewhere):

  $t : \tau$ "term t has type τ"

  $\Gamma \vdash \varphi$ "formula φ is valid under assumptions Γ"

  $\vdash \{P\}\ x := x+1\ \{Q\}$ "Hoare Triple"

  φ prop "φ is a property"

  φ valid "φ is a valid (true) property"

  x mortal $\Longrightarrow$ sokrates mortal --- judgements with free variable

  etc ...

Natural Deduction

- An Inference System for the equality operator (or "HO Equational Logic") looks like this:

$$\frac{}{(s = s) prop} \qquad \frac{(s = t) prop}{(t = s) prop} \qquad \frac{(r = s) prop \quad (s = t) prop}{(r = t) prop}$$

$$\frac{(s(x) = t(x)) prop}{(s = t) prop} \; where\ x\ is\ fresh \qquad \frac{(s = t) prop \quad (P(s)) prop}{(P(t)) prop}$$

(where the first rule is an elementary fact).

Natural Deduction

- the same thing presented a bit more neatly (without prop):

$$\frac{}{x = x} \qquad \frac{s = t}{t = s} \qquad \frac{r = s \quad s = t}{r = t}$$

$$\frac{\bigwedge x.\ s\ x = t\ x}{s = t} \qquad \frac{s = t \quad P\ s}{P\ t}$$

(equality on functions as above ("extensional equality") is an HO principle, and it is a classical principle).

Representing logical systems in the typed λ- calculus

- It is straight-forward to use the typed λ-terms as a syntactic means to represent logics; including binding issues related to quantifiers like ∀, ∃, …
- Example: The Isabelle language „Pure“:
  It consists of typed λ-terms with constants:
  - foundational types "prop" and "_ => _" ("_ ⇒ _")
  - the Pure (universal) quantifier

    all :: "(α → Prop) → Prop"

    ("⋀x. P x","\<And> x. P x"   "!!x. P x")
  - the Pure implication "A ==> B" ("_ ⟹ _")
  - the Pure equality "A == B"   "A ≡ B"

„Pure": A (Meta)-Language for Deductive Systems

- Pure is a language to write logical rules.
- Wrt. Isabelle, it is the meta-language, i.e. the built-in formula language.
- Equivalent notations for natural deduction rules:

$A_1 \Longrightarrow (\ldots \Longrightarrow (A_n \Longrightarrow A_{n+1})\ldots),$

$[\![A_1; \ldots; A_n]\!] \Longrightarrow A_{n+1},$

$$\frac{A_1 \quad \ldots \quad A_n}{A_{n+1}}$$

theorem
assumes A_1
and ...
and A_n
shows A_{n+1}

„Pure": A (Meta)-Language for Deductive Systems

- Some more complex rules involving the concept of "Discharge" of (formerly hypothetical) assumptions:

(P ⟹ Q) ⟹ R :

theorem
assumes "P ⟹ Q"
shows "R"

$$\frac{\begin{array}{c}[P]\\ \vdots\\ Q\end{array}}{R}$$

Propositional Logic as ND calculus

- Some (almost) basic rules in HOL

$$\frac{Q}{\neg\neg Q}$$

$$\frac{\neg\neg Q}{Q}\,\text{notnotE}$$

$$\frac{\begin{matrix}[A]\\ \vdots \\ B\end{matrix}}{A \rightarrow B}\,\text{impI}$$

$$\frac{A \rightarrow B \quad A}{B}\,\text{mp}$$

$$\frac{A}{A \vee B}\,\text{disjI1}$$

$$\frac{B}{A \vee B}\,\text{disjI2}$$

$$\frac{A \vee B \quad \begin{matrix}[A]\\ \vdots \\ Q\end{matrix} \quad \begin{matrix}[B]\\ \vdots \\ Q\end{matrix}}{Q}\,\text{disjE}$$

Propositional Logic as ND calculus

- Some (almost) basic rules in HOL

$$\frac{A \wedge B \qquad \begin{matrix}[A, B] \\ \vdots \\ Q\end{matrix}}{Q}\,\text{conjE} \qquad\qquad \frac{A \quad B}{A \wedge B}\,\text{conjI}$$

Key Concepts: Rule-Instances

- A Rule-Instance is a rule where the free variables in its judgements were substituted by a common substitution σ:

$$\frac{A \quad B}{A \wedge B}\,\text{conjI} \quad \xrightarrow{\ \sigma\ } \quad \frac{3 < x \quad x \leq y}{3 < x \wedge x \leq y}$$

where σ is {A ↦ 3<x, B ↦ x≤y}.

Key Concepts: Formal Proofs

- A series of inference rule instances is usually displayed as a Proof Tree (or : Derivation or: Formal Proof)

$$
\text{subst}\;\dfrac{\dfrac{\text{sym}\;\dfrac{f(a,b) = a}{a = f(a,b)} \qquad \dfrac{f(a,b) = a \qquad f(f(a,b),b) = c}{f(a,b) = c}\;\text{subst}}{a = c}\;\text{trans} \qquad \dfrac{}{g(a) = g(a)}\;\text{refl}}{g(a) = g(c)}
$$

- The hypothetical facts at the leaves are called the assumptions of the proof (here $f(a,b) = a$ and $f(f(a,b),b) = c$).

Key Concepts: Discharge

$$\dfrac{\begin{matrix}[A]\\ \vdots \\ B\end{matrix}}{A \rightarrow B}$$

- A key requisite of ND is the concept of discharge of assumptions allowed by some rules (like impI)

$$\dfrac{\dfrac{\dfrac{\dfrac{[f(a,b)=a]}{a=f(a,b)}\,\text{sym} \qquad \dfrac{[f(a,b)=a] \quad f(f(a,b),b)=c}{f(a,b)=c}\,\text{subst}}{a=c}\,\text{trans} \qquad \dfrac{}{g(a)=g(a)}\,\text{refl}}{g(a)=g(c)}\,\text{subst}}{f(a,b)=a \rightarrow g(a)=g(c)}$$

- The set of assumptions is diminished by the discharged hypothetical facts of the proof (remaining: $f(f(a,b),b) = c$).

Key Concepts: Global Assumptions

- The set of (proof-global) assumptions gives rise to the notation:

$$\{f(a,b) = a, f(f(a,b),b) = c\} \vdash g(a) = g(c)$$

written:

$$A \vdash \phi$$

or when emphasising the global theory (also called: global context):

$$A \vdash_E \phi$$

Sequent-style calculus

- Gentzen introduced and alternative "style" to natural deduction: Sequent style rules.
  - Idea: using the tuples $A \vdash \phi$ as basic judgments of the rules.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

Sequent-style calculus

- in contrast to:

$$\dfrac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \to B} \qquad\qquad \dfrac{A \to B \quad A}{B}$$

Sequent-style vs. ND calculus

- Both styles are linked by two transformations called "lifting over assumptions" Lifting over assumptions transforms:

$$\frac{A_1 \quad \ldots \quad A_n}{A_{n+1}}$$

$\Updownarrow$

$$\frac{\Gamma \vdash A_1 \quad \ldots \quad \Gamma \vdash A_n}{\Gamma \vdash A_{n+1}}$$

where we consider for the moment $\vdash$ just equivalent to meta implication $\Longrightarrow$

Quantifiers

- When reasoning over logics with quantifiers (such as FOL, set-theory, TLA, …, and of course: HOL), the additional concept of "parameters" of a rule is necessary. We assume that there is an infinite set of variables and that it is always possible to find a "fresh" unused one …

  – Consider:

$$\frac{\forall x. P(x)}{P(t)}\ for\ any\ term\ t \qquad \frac{\forall x. P(x) \qquad \begin{array}{c}[P(y)]_y \\ \vdots \\ Q\end{array}}{Q}$$

$$\frac{P(u)}{\forall x. P(x)}\ for\ any\ fresh\ variable\ u \qquad \frac{P(0) \qquad \begin{array}{c}[P(n)]_n \\ \vdots \\ P(Suc\ n)\end{array}}{\forall x. P(x)}$$

Quantifiers

- For allI, Isabelle allows certain free variables ?X, ?Y, ?Z that represent „wholes“ in a term that can be filled in later by substitution; Coq requires the instantiation when applying the rule.

- Isabelle uses a built-in (“meta”)-quantifier $\bigwedge$x. P x already seen on page 13; Coq uses internally a similar concept not explicitly revealed to the user.

Introduction to Isabelle/HOL

Basic HOL Syntax

- HOL (= Higher-Order Logic) goes back to Alonzo Church who invented this in the 30ies …
- "Classical" Logic over the λ-calculus with Curry-style typing (in contrast to Coq)
- Logical type: "bool" injects to "prop". i.e

  Trueprop :: "bool $\Rightarrow$ prop"

  is wrapped around any HOL-Term without being printed:

  Trueprop A $\Longrightarrow$ Trueprop B is printed: A $\Longrightarrow$ B but A::bool!

Basic HOL Syntax

- Logical connective syntax (Unicode + ASCII):

input:	print:	alt-ascii input
– “_ \<and> _”	“_∧_”	“_ & _”
– “_ \<or> _”	“_ ∨_”	“_ | _
– “_\<longrightarrow>_”	“_ → _”	“_ --> _”
– “_ \<not> _”	“¬_"	“~ _”
– “\<forall> x. P“	“∀x. P”	“! x. P x”
– “\<exists> x. P“	“∃x. P”	“? x. P x”

Basic HOL Rules

- HOL is an equational logic, i.e. a system with the constant "_=_::'a 'a bool" and the rules:

$$\frac{}{x = x}\ \text{refl} \qquad \frac{s = t}{t = s}\ \text{sym} \qquad \frac{r = s \quad s = t}{r = t}\ \text{trans}$$

$$\frac{\bigwedge x.\ s\ x = t\ x}{s = t}\ \text{ext} \qquad \frac{s = t \quad P\ s}{P\ t}\ \text{subst}$$

Basic HOL Rules

- HOL is an equational logic, i.e. a system with the constant "_=_::'a 'a bool" and the rules:

$$\frac{}{x = x}\ \text{refl} \qquad \frac{s = t}{t = s}\ \text{sym} \qquad \frac{r = s \quad s = t}{r = t}\ \text{trans}$$

$$\frac{\bigwedge x.\ s\ x = t\ x}{s = t}\ \text{ext} \qquad \frac{s = t \quad P\ s}{P\ t}\ \text{subst}$$

which rule makes HOL „higher-order" ???

Basic HOL Rules

- Some (almost) basic rules in HOL

$$\frac{A \wedge B \qquad \begin{matrix}[A, B] \\ \vdots \\ Q\end{matrix}}{Q}\,\text{conjE} \qquad\qquad \frac{A \quad B}{A \wedge B}\,\text{conjI}$$

HOL Rules

- The quantifier rules of HOL:

$$\frac{\bigwedge x.\ P\ x}{\forall x. P\ x}\ \text{allI}$$

$$\frac{\forall x. P\ x \qquad \begin{matrix}[P\ ?t] \\ \vdots \\ Q\end{matrix}}{Q}\ \text{allE}$$

allE
(safe, but
incomplete)

again: what makes theses HOL „higher-order“ ???

HOL Rules

- The quantifier rules of HOL:

$$\frac{\forall x. P\ x \qquad \begin{array}{c}[P\ ?t; \forall x. P\ x] \\ \vdots \\ Q\end{array}}{Q}$$

alldupE
(unsafe, but
complete)

HOL Rules

- The quantifier rules of HOL:

$$\dfrac{\forall x. P\ x \qquad \begin{array}{c}[P\ ?t; \forall x. P\ x]\\ \vdots \\ Q\end{array}}{Q}$$

alldupE
(unsafe, but
complete)

HOL Rules

- The quantifier rules of HOL:

$$\frac{P\ ?t}{\exists x. P\ x}\ \text{exI}$$

$$\frac{\exists x.\ P(x) \qquad \begin{matrix}[P(x)]_x \\ \vdots \\ Q\end{matrix}}{Q}\ \text{exE}$$

HOL Rules

- From these rules (which were defined actually slightly differently), a large body of other rules can be DERIVED (formally proven, and introduced as new rule in the proof environment).

  Examples: see exercises.

Typed Set-theory in HOL

- The HOL Logic comes immediately with a typed set - theory: The type

  α set $\cong$ $\alpha \Rightarrow$ bool, that's it !

  can be defined isomorphically to its type of characteristic functions !

- THIS GIVES RISE TO A RICH SET THEORY DEVELOPPED IN THE LIBRARY (Set.thy).

Typed Set Theory: Syntax

- Logical connective syntax (Unicode + ASCII):

input:	print:	alt-ascii input
"_ \<in> _"	"_ ∈ _"	"_ : _"
"{_ . _}"	*{x. True ∧ x = x}*	*for example*
"_ \<union> _"	"_∪_"	"_ Un _"
"_ \<inter> _"	"_ ∩_"	"_ Int _"
"_\<subseteq>_"	"_ ⊆ _"	"_ <= _"
. . .		

Conclusion

- Typed λ-calculus is a rich term language for the representation of logics, logical rules, and logical derivations (proofs)
- On the basis of typed λ-calculus, Higher-order logic (HOL) is fairly easy to represent
- … the differences to first-order logic (FOL) are actually tiny.