Computer Supported Modeling and Reasoning

David Basin, Achim D. Brucker, Jan-Georg Smaus, and Burkhart Wolff

April 2005

http://www.infsec.ethz.ch/education/permanent/csmr/

Higer-Order Logic: Derived Rules

David Basin

Outline

Last lecture: Introduction to HOL

- Basic syntax and semantics
- Basic eight (or nine) axioms
- Definitions of $True$, $False$, $\wedge$, $\vee$, $\forall$ ...

Today:

- Deriving rules for the defined constants
- Outlook on the rest of this course

Reminder: Different Syntaxes

Conceptual	vs.	Isabelle/PG notation
$\lambda x^{bool}.P(x)$		λ`x` :: `bool. P`
$\forall x.\, P(x)$		"$All(\lambda \mathtt{x}.P\, x)$" = "$\forall x.P(x)$"
$\iota x.\, P(x)$		"$The(\lambda \mathtt{x}.P\, x)$" = "$\mathtt{THE} x.P(x)$"

We will be using all those forms as convenient.

Reminder: Definitions

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

Derived Rules

The definitions can be understood as syntactic abbreviations.

Later, we will see that they are in fact conservative constant definitions.

We usually proceed as follows: first show a rule involving a constant, then replace the constant with its definition (if applicable), then show the derivation.

Equality

- Rule *sym* and ND derivation

$$\dfrac{s = t \qquad \dfrac{}{s = s}\,\textit{refl}}{t = s}\,\textit{subst}$$

- HOL rule s=t $\Longrightarrow$ t=s: Proof:

lemma sym : "s=t $\Longrightarrow$t=s";
apply (*erule subst*); $(*\ P\ is\ \lambda x.x{=}s\ *)$
apply (*rule* refl 1); $(*\ s{=}s\ *)$
done

Equality: Transitivity and Congruences

- Rule *trans* and ND derivation

$$\frac{s = t \quad r = s}{r = t}\ \textit{trans}$$

  HOL rule $\llbracket$ r=s; s=t $\rrbracket \Longrightarrow$ r=t

- Congruences (only HOL forms):
  - (f ::' a⇒'b) = g $\Longrightarrow$ f(x)=g(x) (funcong)
  - x=y $\Longrightarrow$f(x)=f(y) (argcong)

  HOL proofs using *subst* and *refl*.

Equality of Booleans (*iffI*)

Rule *iffI* and ND derivation

$$\dfrac{\dfrac{\dfrac{}{(P \to Q) \to (Q \to P) \to (P = Q)}\ \textit{iff} \qquad \dfrac{\begin{matrix}[P]\\ \vdots \\ Q\end{matrix}}{P \to Q}\ \textit{impI}}{(Q \to P) \to P = Q}\ \textit{mp} \qquad \dfrac{\begin{matrix}[Q]\\ \vdots \\ P\end{matrix}}{Q \to P}\ \textit{impI}}{P = Q}\ \textit{mp}$$

HOL rule $[\![\ \mathsf{P} \Longrightarrow \mathsf{Q};\ \ \mathsf{Q} \Longrightarrow \mathsf{P}\ ]\!] \Longrightarrow \mathsf{P{=}Q}$.

Equality of Booleans (*iffD2*)

Rule *iffD2* and ND derivation

$$\dfrac{\dfrac{P = Q}{Q = P}\,\textit{sym} \qquad Q}{P}\,\textit{subst}\ \textit{iffD2}$$

HOL rule $[\![\ P{=}Q;\ Q\ ]\!] \Longrightarrow P$.

True

$$True = ((\lambda x^{bool}.x) = (\lambda x.x))$$

- Rule *TrueI* and ND derivation

$$\dfrac{}{(\lambda x.x) = (\lambda x.x)}\,refl \qquad \dfrac{}{True}\,TrueI$$

- Rule *eqTrueE* and ND derivation

$$\dfrac{P = True \quad \dfrac{}{True}\,TrueI}{P}\,eqTrueE \;/\; iffD2$$

HOL rule P=True $\Longrightarrow$P.

True (Cont.)

- Rule *eqTrueI* and ND derivation

$$\dfrac{\dfrac{}{True}\,\mathit{TrueI} \qquad P}{P = True}\,\mathit{iffI}\ \mathit{eqTrueI}$$

Note that 0 assumptions were discharged.

HOL rule P $\Longrightarrow$ P=True.

Universal Quantification

$$\forall P = (P = (\lambda x. True))$$

- Rule *allI* and ND derivation

$$\dfrac{\dfrac{\bigwedge x.\, P(x)}{\bigwedge x.\, P(x) = True}\ \textit{eqTrueI}}{P = \lambda x. True \;/\; \forall P}\ \textit{ext} \,/\, \textit{allI}$$

HOL rule $(\bigwedge x.\ P(x)) \Longrightarrow \forall\, x.\ P(x)$.

Universal Quantification (Cont.)

- Rule *spec* and ND derivation

$$\cfrac{\cfrac{P = \lambda x. True}{P(x) = True}\ \textit{fun\_cong}}{P(x)}\ \textit{spec}\ \textit{eqTrueE}$$

HOL rule ∀ x ::' a. P(x) $\Longrightarrow$ P(x).

Note: Need universal quantification to reason about $False$ (since $False = (\forall P. P)$).

False

$False = (\forall P.P)$

- FalseI: No rule!
- Rule *FalseE* and ND derivation

$$\frac{False}{P}\ FalseE \qquad \frac{\forall P.P}{P}\ spec$$

HOL rule False $\Longrightarrow$ P.

False (Cont.)

- Rule *False_neq_True* and ND derivation

$$\dfrac{\dfrac{False = True}{False}\ eqTrueE}{P}\ FalseE\ neq\_True$$

HOL rule False=True $\Longrightarrow$P.

-

Similar:

$$\dfrac{True = False}{P}\ True\_neq\_False$$

Negation

$$\neg P = P \rightarrow False$$

- Rule *notI* and ND derivation

$$\dfrac{\begin{array}{c}[P]\\ \vdots \\ False\end{array}}{P \rightarrow \neg P False}\ \textit{impI notI}$$

HOL rule (P $\Longrightarrow$ False) $\Longrightarrow$ ¬ P.

Negation (Cont.)

- Rule *notE* and ND derivation

$$\dfrac{\dfrac{\neg P \,/\, P \to False \qquad P}{False}\ \mathit{mp}}{R}\ \mathit{FalseE}\,/\,\mathit{notE}$$

HOL rule $[\![\neg P;\ \ P]\!] \Longrightarrow R$.

Negation (Cont.)

- Rule *True_Not_False* and ND derivation

$$\dfrac{\dfrac{[True = False]^1}{False}\ \textit{True\_neq\_False}}{(True = False) \to False}\ \textit{notI}^1\ \textit{True\_Not\_False}$$

HOL rule True $\neq$ False.

Existential Quantification

- Ex(P) $\equiv \forall$ Q. ($\forall$ x. P x$\longrightarrow$Q)$\longrightarrow$Q
- P(x) $\Longrightarrow \exists$ x ::' a . P(x) (*exI*)

$$\dfrac{Px \qquad \dfrac{[\forall y.P(y) \rightarrow Q]}{P(x) \rightarrow Q}\ \textit{spec}}{\dfrac{Q}{\dfrac{(\forall y.P(y) \rightarrow Q) \rightarrow Q}{\forall Q.(\forall x.P(x) \rightarrow Q) \rightarrow Q}\ \textit{allI}}\ \textit{impI}}\ \textit{mp}$$

- $[\![\ \exists\, x ::' a.\, P(x);\ \bigwedge x.\ P(x) \Longrightarrow Q\ ]\!] \Longrightarrow Q$ (*exE*)

$$\dfrac{\dfrac{\forall Q.((\forall y.P(y) \to Q) \to Q)}{(\forall y.P(y) \to Q) \to Q}\ \textit{spec} \qquad \dfrac{\bigwedge x.\ \dfrac{[P(x)]}{Q}\ \textit{impI}}{\forall y.P(y) \to Q}\ \textit{allI}}{Q}\ \textit{mp}$$

Conjunction

$$P \wedge Q = \forall R.(P \to Q \to R) \to R$$

- Rule *conjI* and ND derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{[P \to Q \to R]^1 \quad P}{Q \to R}\,mp \qquad Q}{R}\,mp}{(P \to Q \to R) \to R}\,impI^1}{\forall R.(P \to Q \to R) \to R \;\;/\;\; P \wedge Q}\,def$$

HOL rule $[\![P; Q]\!] \Longrightarrow P \wedge Q$.

Conjunction (Cont.)

- Rule *conjEL* and ND derivation

$$\dfrac{\dfrac{\forall R.(P \to Q \to R) \to R}{(P \to Q \to P) \to P}\ \textit{spec} \qquad \dfrac{\dfrac{[P]^1}{Q \to P}\ \textit{impI}}{P \to Q \to P}\ \textit{impI}^1}{P}\ \textit{conjEL}$$

HOL rule P ∧ Q $\Longrightarrow$ P.

Conjunction (Cont.)

- $P \wedge Q \Longrightarrow Q$" (*conjER*)
- $[\![P \wedge Q; [\![P; Q]\!] \Longrightarrow R]\!] \Longrightarrow R$ (*conjE*) (rule analogous to *disjE*)

Disjunction

$$P \vee Q = \forall R.(P \to R) \to (Q \to R) \to R$$

- Rule *disjIL* and ND derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{[P \to R]^1 \quad P}{R}\;mp}{(Q \to R) \to R}\;impI}{(P \to R) \to (Q \to R) \to R}\;impI^1}{\forall R.(P \to R) \to (Q \to R) \to R}\;disjIL$$

HOL rule $P \Longrightarrow P \vee Q$.

Disjunction (Cont.)

- $Q \Longrightarrow P \vee Q$ (*disjIR*) similar
- Rule *disjE* and ND derivation

$$
\dfrac{\dfrac{\dfrac{\forall R.(P \to R) \to (Q \to R) \to R}{(P \to R) \to (Q \to R) \to R}\,\textit{spec} \qquad \dfrac{\begin{matrix} P \\ \vdots \\ R \end{matrix}}{P \to R}\,\textit{impI}}{(Q \to R) \to R}\,\textit{mp} \qquad \dfrac{\begin{matrix} Q \\ \vdots \\ R \end{matrix}}{Q \to R}\,\textit{impI}}{R}\,\textit{mp}\ \textit{disjE}
$$

HOL rule $[\![\text{P} \vee \text{Q}; \text{P} \Longrightarrow \text{R}; \text{Q} \Longrightarrow \text{R}]\!] \Longrightarrow \text{R}$.

- $P \vee \neg P$ (*excluded middle*). Follows using *tof*.

Miscellaneous Definitions

Typical example (if-then-else):

If P x y $\equiv$ THE z. (P=True $\longrightarrow$z=x) $\wedge$
(P=False $\longrightarrow$z=y)

The way rules are derived should now be clear. E.g.,

$$\frac{P = True}{If\ P\ x\ y = x} \qquad \frac{P = False}{If\ P\ x\ y = y}$$

Summary on Deriving Rules

HOL is very powerful in terms of what we can represent/derive:

- All well-known inference rules can be derived.
- Other "logical" syntax (e.g. if-then-else) can be defined.
- Rich theories can be obtained by a method we see next lecture.

Mathematics and Software Engineering in HOL

In the weeks to come, we will see how Isabelle/HOL can be used as foundation for mathematics and software engineering.

Outline:

- The central method for making HOL scale up: conservative extensions (< 1 week)
- How the different parts of mathematics are encoded in the Isabelle/HOL library (several weeks)
- How software systems are embedded in Isabelle/HOL

(several weeks)

Outlook on Mathematics

After some historical background, we will look at how central parts of mathematics are encoded as Isabelle/HOL theories:

- Orders and sets
- Fixpoints, induction, and recursion
- Arithmetic
- Datatypes

Outlook on Software Engineering

Some weeks from now, we will look at case studies of how HOL can be applied in software engineering, i.e. how software systems can be embedded in Isabelle/HOL:

- Foundations, functional languages and denotational semantics
- Imperative languages, Hoare logic
- Z and data-refinement, CSP and process-refinement
- Object-oriented languages (Java-Light . . .)

Of the last three items, we want to treat only one in depth, depending on the audience's preferences.

Conservative Extensions: Motivation

But first, conservative extensions.

Stage of our course before studying HOL:

- fairly small theories,
- "intuitive" models, (e.g. naïve set theory),
- but inconsistent (due to foundational problems).

How can we ever hope to apply these techniques to software engineering?

What Is Needed for Scaling up?

Let's try to apply well-known structuring techniques:

Known mechanisms; of which Isabelle implements:

Modularization	⇒	~~(Parameterized)~~ theories, (class) polymorphism
Reuse	⇒	Libraries, retrieval utilities
Safe, well-understood integration mechanisms	⇒	~~Persistent parametric theories,~~ Conservative theory extensions

Topic of next lecture.

More Detailed Explanations

RC

RC stands for refinement calculus.

Z, CSP

Z and CSP are specification languages. CSP stands for communicating sequential processes.

Persistence

Persistent theories play a role in the prover PVS.

References

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proofs*. Academic Press, 1986.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[GM93] Michael J. C. Gordon and Tom F. Melham, editors. *Introduction to HOL*. Cambridge University Press, 1993.

[WR25] Alfred N. Whitehead and Bertrand Russell. *Principia Mathematica*, volume 1. Cambridge University Press, 1925. 2nd edition.