COMP 4161

NICTA Advanced Course

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Toby Murray

HOL

Exercises from last time

- We said that $\mathcal{E}$ implies the Axiom of Choice:

  $\forall x.\ \exists y.\ Q\ x\ y \Longrightarrow \exists f.\ \forall x.\ Q\ x\ (f\ x)$

- Prove the axiom of choice as a lemma, using only the introduction and elimination rules for $\forall$ and $\exists$, namely `allI`, `exI`, `allE`, `exE`, and the introduction rule for $\mathcal{E}$, `someI`, using only the proof methods `rule`, `rule_tac`, `erule`, `erule_tac` and `assumption`.

Content

	Rough timeline
➜ Intro & motivation, getting started	[1]
➜ Foundations & Principles	
• Lambda Calculus, natural deduction	[2,3,4[a]]
• Higher Order Logic	[5,6[b],7]
• Term rewriting	[8,9,10[c]]
➜ Proof & Specification Techniques	
• Isar	[11,12[d]]
• Inductively defined sets, rule induction	[13[e],15]
• Datatypes, recursion, induction	[16,17[f],18,19]
• Calculational reasoning, mathematics style proofs	[20]
• Hoare logic, proofs about programs	[21[g],22,23]

[a] a1 out; [b] a1 due; [c] a2 out; [d] a2 due; [e] session break; [f] a3 out; [g] a3 due

DEFINING HIGHER ORDER LOGIC

What is Higher Order Logic?

NICTA

➜ **Propositional Logic:**

- no quantifiers
- all variables have type bool

➜ **First Order Logic:**

- quantification over values, but not over functions and predicates,
- terms and formulas syntactically distinct

➜ **Higher Order Logic:**

- quantification over everything, including predicates
- consistency by types
- formula = term of type bool
- definition built on $\lambda^{\rightarrow}$ with certain default types and constants

NICTA

Default types:

$$\text{bool} \qquad\qquad \_ \Rightarrow \_ \qquad\qquad \text{ind}$$

- **bool** sometimes called o
- $\Rightarrow$ sometimes called fun

Default Constants:

$$\begin{array}{lll} \longrightarrow & :: & bool \Rightarrow bool \Rightarrow bool \\ = & :: & \alpha \Rightarrow \alpha \Rightarrow bool \\ \epsilon & :: & (\alpha \Rightarrow bool) \Rightarrow \alpha \end{array}$$

Higher Order Abstract Syntax

Problem: Define syntax for binders like $\forall$, $\exists$, ε

One approach: $\forall :: var \Rightarrow term \Rightarrow bool$
Drawback: need to think about substitution, α conversion again.

But: Already have binder, substitution, α conversion in meta logic

$$\lambda$$

So: Use λ to encode all other binders.

Higher Order Abstract Syntax

Example:

$$\text{ALL} :: (\alpha \Rightarrow bool) \Rightarrow bool$$

HOAS	**usual syntax**
ALL $(\lambda x.\ x = 2)$	$\forall x.\ x = 2$
ALL P	$\forall x.\ P\ x$

Isabelle can translate usual binder syntax into HOAS.

Side Track: Syntax Declarations in Isabelle

NICTA

- **mixfix:**

  consts drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool \;\; (\text{"}\_, \_ \;\vdash\; \_\text{"})$

  Legal syntax now: $\Gamma, \Pi \vdash F$

- **priorities:**

  pattern can be annotated with priorities to indicate binding strength

  Example: drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool \;\; (\text{"}\_, \_ \;\vdash\; \_\text{"} \; [30, 0, 20] \; 60)$

- **infixl/infixr**: short form for left/right associative binary operators

  Example: or :: $bool \Rightarrow bool \Rightarrow bool \;\; (\text{infixr "} \lor \text{" } 30)$

- **binders:** declaration must be of the form

  $c :: (\tau_1 \Rightarrow \tau_2) \Rightarrow \tau_3 \;\; (\text{binder "}B\text{" } < p >)$

  $B\ x.\ P\ x$ translated into $c\ P$ (and vice versa)

  Example ALL :: $(\alpha \Rightarrow bool) \Rightarrow bool \;\; (\text{binder "}\forall\text{" } 10)$

More (including pretty printing) in Isabelle Reference Manual (7.3)

Back to HOL

Base: $bool$, $\Rightarrow$, ind $\quad$ $=$, $\longrightarrow$, ε

And the rest is definitions:

$$
\begin{array}{ll}
\mathsf{True} & \equiv (\lambda x :: bool.\ x) = (\lambda x.\ x) \\
\mathsf{All}\ P & \equiv P = (\lambda x.\ \mathsf{True}) \\
\mathsf{Ex}\ P & \equiv \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q \\
\mathsf{False} & \equiv \forall P.\ P \\
\neg P & \equiv P \longrightarrow \mathsf{False} \\
P \wedge Q & \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R \\
P \vee Q & \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R \\
\mathsf{If}\ P\ x\ y & \equiv \mathsf{SOME}\ z.\ (P = \mathsf{True} \longrightarrow z = x) \wedge (P = \mathsf{False} \longrightarrow z = y) \\
\mathsf{inj}\ f & \equiv \forall x\ y.\ f\ x = f\ y \longrightarrow x = y \\
\mathsf{surj}\ f & \equiv \forall y.\ \exists x.\ y = f\ x
\end{array}
$$

The Axioms of HOL

$$\frac{}{t = t}\ \textsf{refl} \qquad \frac{s = t \quad P\ s}{P\ t}\ \textsf{subst} \qquad \frac{\bigwedge x.\ f\ x = g\ x}{(\lambda x.\ f\ x) = (\lambda x.\ g\ x)}\ \textsf{ext}$$

$$\frac{P \Longrightarrow Q}{P \longrightarrow Q}\ \textsf{impI} \qquad \frac{P \longrightarrow Q \quad P}{Q}\ \textsf{mp}$$

$$\frac{}{(P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)}\ \textsf{iff}$$

$$\frac{}{P = \textsf{True} \vee P = \textsf{False}}\ \textsf{True\_or\_False}$$

$$\frac{P\ ?x}{P\ (\textsf{SOME}\ x.\ P\ x)}\ \textsf{someI}$$

$$\frac{}{\exists f :: ind \Rightarrow ind.\ \textsf{inj}\ f \wedge \neg \textsf{surj}\ f}\ \textsf{infty}$$

That's it.

- 3 basic constants
- 3 basic types
- 9 axioms

With this you can define and derive all the rest.

Isabelle knows 2 more axioms:

$$\frac{x = y}{x \equiv y}\ \text{eq\_reflection} \qquad \frac{}{(\text{THE}\ x.\ x = a) = a}\ \text{the\_eq\_trivial}$$

DEMO: THE DEFINITIONS IN ISABELLE

Deriving Proof Rules

In the following, we will

- ➜ look at the definitions in more detail
- ➜ derive the traditional proof rules from the axioms in Isabelle

Convenient for deriving rules: **named assumptions in lemmas**

lemma $[name :]$
assumes $[name_1 :]$ "$< prop >_1$"
assumes $[name_2 :]$ "$< prop >_2$"
$\vdots$
shows " $< prop >$ " $\quad < proof >$

proves: $[\![< prop >_1 ; < prop >_2 ; \ldots]\!] \Longrightarrow < prop >$

True

consts True :: $bool$

True $\equiv (\lambda x :: bool.\ x) = (\lambda x.\ x)$

Intuition:

right hand side is always true

Proof Rules:

$$\frac{}{\text{True}}\ \text{TrueI}$$

Proof:

$$\dfrac{\overline{(\lambda x :: bool.\ x) = (\lambda x.\ x)}\ \text{refl}}{\text{True}}\ \text{unfold True\_def}$$

DEMO

Universal Quantifier

consts ALL :: $(\alpha \Rightarrow bool) \Rightarrow bool$

ALL $P \quad \equiv \quad P = (\lambda x.\ \text{True})$

Intuition:

- ALL P is Higher Order Abstract Syntax for $\forall x.\ P\ x$.
- P is a function that takes an x and yields a truth values.
- ALL P should be true iff P yields true for all x, i.e.
  if it is equivalent to the function $\lambda x.$ True.

Proof Rules:

$$\frac{\bigwedge x.\ P\ x}{\forall x.\ P\ x}\ \text{allI} \qquad \frac{\forall x.\ P\ x \quad P\ ?x \Longrightarrow R}{R}\ \text{allE}$$

Proof: Isabelle Demo

False

consts False :: $bool$

False $\equiv$ $\forall P. P$

Intuition:

Everything can be derived from *False*.

Proof Rules:

$$\frac{\text{False}}{P}\ \text{FalseE} \qquad \frac{}{\text{True} \neq \text{False}}$$

Proof: Isabelle Demo

Negation

consts Not :: $bool \Rightarrow bool$ $(\neg\ \_)$

$\neg P \equiv P \longrightarrow$ False

Intuition:

Try $P = True$ and $P = False$ and the traditional truth table for $\longrightarrow$.

Proof Rules:

$$\frac{A \Longrightarrow False}{\neg A}\ \text{notI} \qquad \frac{\neg A \quad A}{P}\ \text{notE}$$

Proof: Isabelle Demo

Existential Quantifier

consts EX $:: (\alpha \Rightarrow bool) \Rightarrow bool$

EX $P \quad \equiv \quad \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

Intuition:

- EX P is HOAS for $\exists x.\ P\ x$. (like $\forall$)
- Right hand side is characterization of $\exists$ with $\forall$ and $\longrightarrow$
- Note that inner $\forall$ binds wide: $(\forall x.\ P\ x \longrightarrow Q)$
- Remember lemma from last time: $(\forall x.\ P\ x \longrightarrow Q) = ((\exists x.\ P\ x) \longrightarrow Q)$

Proof Rules:

$$\frac{P\ ?x}{\exists x.\ P\ x}\ \text{exI} \qquad \frac{\exists x.\ P\ x \qquad \bigwedge x.\ P\ x \Longrightarrow R}{R}\ \text{exE}$$

Proof: Isabelle Demo

Conjunction

consts And $:: bool \Rightarrow bool \Rightarrow bool \; (\_ \wedge \_)$

$P \wedge Q \equiv \forall R. \; (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

Intuition:

- ➜ Mirrors proof rules for $\wedge$
- ➜ Try truth table for P, Q, and R

Proof Rules:

$$\frac{A \quad B}{A \wedge B} \text{ conjI} \qquad \frac{A \wedge B \quad [\![A; B]\!] \Longrightarrow C}{C} \text{ conjE}$$

Proof: Isabelle Demo

Disjunction

consts Or $:: bool \Rightarrow bool \Rightarrow bool \; (\_ \vee \_)$

$P \vee Q \equiv \forall R. \; (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$

Intuition:

- Mirrors proof rules for $\vee$ (case distinction)
- Try truth table for P, Q, and R

Proof Rules:

$$\frac{A}{A \vee B} \quad \frac{B}{A \vee B} \; \text{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{C} \; \text{disjE}$$

Proof: Isabelle Demo

If-Then-Else

NICTA

consts If :: $bool \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$ $(\mathsf{if\_ \ then \ \_ \ else \ \_})$

If $P\ x\ y \ \equiv\ \mathsf{SOME}\ z.\ (P = \mathsf{True} \longrightarrow z = x) \wedge (P = \mathsf{False} \longrightarrow z = y)$

Intuition:

- ➜ for $P = \mathsf{True}$, right hand side collapses to SOME $z.\ z = x$
- ➜ for $P = \mathsf{False}$, right hand side collapses to SOME $z.\ z = y$

Proof Rules:

$$\frac{}{\mathsf{if\ True\ then}\ s\ \mathsf{else}\ t = s}\ \text{ifTrue} \qquad \frac{}{\mathsf{if\ False\ then}\ s\ \mathsf{else}\ t = t}\ \text{ifFalse}$$

Proof: Isabelle Demo

THAT WAS HOL

More on Automation

Last time: safe and unsafe rule, heuristics: use safe before unsafe

This can be automated

Syntax:

[<kind>!]	for safe rules (<kind> one of intro, elim, dest)
[<kind>]	for unsafe rules

Application (roughly):

do safe rules first, search/backtrack on unsafe rules only

Example:

declare attribute globally	**declare** conjI [intro!] allE [elim]
remove attribute gloabllay	**declare** allE [rule del]
use locally	**apply** (blast intro: someI)
delete locally	**apply** (blast del: conjI)

DEMO: AUTOMATION

We have learned today ...

NICTA

- ➜ Defining HOL
- ➜ Higher Order Abstract Syntax
- ➜ Deriving proof rules
- ➜ More automation