



COMP 4161
NICTA Advanced Course

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Toby Murray, Rafal Kolanski

HOL

Slide 1



Content

- Intro & motivation, getting started [1]
- Foundations & Principles
 - Lambda Calculus, natural deduction [1,2]
 - Higher Order Logic [3]
 - Term rewriting [4^a]
- Proof & Specification Techniques
 - Inductively defined sets, rule induction [5]
 - Datatypes, recursion, induction [6^b, 7]
 - Code generation, type classes [7]
 - Hoare logic, proofs about programs, refinement [8,9^c, 10^d]
 - Isar, locales [11,12]

^aa1 due; ^ba2 due; ^csession break; ^da3 due

Slide 2



DEFINING HIGHER ORDER LOGIC

Slide 3



What is Higher Order Logic?

- **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 λ^{\rightarrow} with certain default types and constants

Slide 4

Defining Higher Order Logic



Default types:

$bool$ $_ \Rightarrow _$ ind

→ **bool** sometimes called o

→ \Rightarrow sometimes called fun

Default Constants:

\longrightarrow :: $bool \Rightarrow bool \Rightarrow bool$

$=$:: $\alpha \Rightarrow \alpha \Rightarrow bool$

ϵ :: $(\alpha \Rightarrow bool) \Rightarrow \alpha$

Slide 5

Higher Order Abstract Syntax



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

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

λ

So: Use λ to encode all other binders.

Slide 6

Higher Order Abstract Syntax



Example:

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

HOAS

$ALL (\lambda x. x = 2)$

$ALL P$

usual syntax

$\forall x. x = 2$

$\forall x. P x$

Isabelle can translate usual binder syntax into HOAS.

Slide 7

Side Track: Syntax Declarations in Isabelle



→ mixfix:

consts $drvbl :: ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ ("_,_ \vdash _")

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

→ priorities:

pattern can be annotated with priorities to indicate binding strength

Example: $drvbl :: ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ ("_,_ \vdash_" [30, 0, 20] 60)

→ infix/infixr:

short form for left/right associative binary operators

Example: $or :: bool \Rightarrow bool \Rightarrow bool$ (infixr " \vee " 30)

→ binders:

declaration must be of the form

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

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

Example $ALL :: (\alpha \Rightarrow bool) \Rightarrow bool$ (binder " \forall " 10)

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

Slide 8

Back to HOL

Base: $bool, \Rightarrow, ind, =, \longrightarrow, \varepsilon$

And the rest is definitions:

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

$All\ P \equiv P = (\lambda x. True)$

$Ex\ P \equiv \forall Q. (\forall x. P\ x \longrightarrow Q) \longrightarrow Q$

$False \equiv \forall P. P$

$\neg P \equiv P \longrightarrow False$

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

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

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

$inj\ f \equiv \forall x\ y. f\ x = f\ y \longrightarrow x = y$

$surj\ f \equiv \forall y. \exists x. y = f\ x$

Slide 9



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:

$$\frac{x = y}{x \equiv y} \text{ eq_reflection} \quad \frac{}{(THE\ x. x = a) = a} \text{ the_eq_trivial}$$

Slide 11



The Axioms of HOL

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

$$\frac{P \Longrightarrow Q}{P \longrightarrow Q} \text{ impl} \quad \frac{P \longrightarrow Q \quad P}{Q} \text{ mp}$$

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

$$\frac{}{P = True \vee P = False} \text{ True_or_False}$$

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

$$\frac{}{\exists f :: ind \Rightarrow ind. inj\ f \wedge \neg surj\ f} \text{ infTy}$$

Slide 10



DEMO: THE DEFINITIONS IN ISABELLE

Slide 12



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 [name1 :] "< prop >1"  
  assumes [name2 :] "< prop >2"  
  :  
  shows "< prop >" < proof >  
  
proves: [ [ < prop >1; < prop >2; ... ] =>< prop >
```

Slide 13

DEMO



Slide 15

True



```
consts True :: bool  
True ≡ (λx :: bool. x) = (λx. x)
```

Intuition:
right hand side is always true

Proof Rules:

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

Proof:

$$\frac{(\lambda x :: \text{bool}. x) = (\lambda x. x)}{\text{True}} \text{refl} \quad \text{unfold True_def}$$

Slide 14

Universal Quantifier



```
consts ALL :: (α ⇒ bool) ⇒ bool  
ALL P ≡ P = (λx. 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 value.
- ALL P should be true iff P yields true for all x , i.e. if it is equivalent to the function $\lambda x. \text{True}$.

Proof Rules:

$$\frac{\bigwedge x. P x}{\forall x. P x} \text{all} \quad \frac{\forall x. P x \quad P ?x \implies R}{R} \text{allE}$$

Proof: Isabelle Demo

Slide 16

False

consts False :: bool
False $\equiv \forall P.P$

Intuition:
Everything can be derived from *False*.

Proof Rules:

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

Proof: Isabelle Demo

Slide 17



Existential Quantifier

consts EX :: ($\alpha \Rightarrow \text{bool}$) \Rightarrow bool
EX $P \equiv \forall Q. (\forall x. P x \longrightarrow Q) \longrightarrow Q$

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:

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

Proof: Isabelle Demo

Slide 19



Negation

consts Not :: $\text{bool} \Rightarrow \text{bool} (\neg _)$
 $\neg P \equiv P \longrightarrow \text{False}$

Intuition:
Try $P = \text{True}$ and $P = \text{False}$ and the traditional truth table for \longrightarrow .

Proof Rules:

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

Proof: Isabelle Demo

Slide 18



Conjunction

consts And :: $\text{bool} \Rightarrow \text{bool} \Rightarrow \text{bool} (_ \wedge _)$
 $P \wedge Q \equiv \forall R. (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

Intuition:

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

Proof Rules:

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

Proof: Isabelle Demo

Slide 20



Disjunction

consts Or :: *bool* ⇒ *bool* ⇒ *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:

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

Proof: Isabelle Demo

Slide 21

If-Then-Else

consts If :: *bool* ⇒ α ⇒ α ⇒ α (if_ then _ else _)
If P x y ≡ SOME z . ($P = \text{True} \longrightarrow z = x$) ∧ ($P = \text{False} \longrightarrow z = y$)

Intuition:

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

Proof Rules:

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

Proof: Isabelle Demo

Slide 22



THAT WAS HOL

Slide 23

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 globally	declare allE [rule del]
use locally	apply (blast intro: someI)
delete locally	apply (blast del: conjI)

Slide 24





DEMO: AUTOMATION

Slide 25

We have learned today ...



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

Slide 26