

**COMP 4161**  
NICTA Advanced Course

**Advanced Topics in Software Verification**

Gerwin Klein, June Andronick, Toby Murray, Christine Rizkallah

# HOL

- Intro & motivation, getting started [1]
  
- Foundations & Principles
  - ▶ Lambda Calculus, natural deduction [1,2]
  - ▶ Higher Order Logic [3<sup>a</sup>]
  - ▶ Term rewriting [4]
  
- Proof & Specification Techniques
  - ▶ Inductively defined sets, rule induction [5]
  - ▶ Datatypes, recursion, induction [6, 7]
  - ▶ Hoare logic, proofs about programs, C verification [8<sup>b</sup>,9]
  - ▶ (mid-semester break)
  - ▶ Writing Automated Proof Methods [10]
  - ▶ Isar, codegen, typeclasses, locales [11<sup>c</sup>,12]

---

<sup>a</sup>a1 due; <sup>b</sup>a2 due; <sup>c</sup>a3 due

# DEFINING HIGHER ORDER LOGIC

# 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  $\lambda^{\rightarrow}$  with certain default types and constants

## Default types:

`bool`                      `_ ⇒ _`                      `ind`

→ **bool** sometimes called *o*

→ `⇒` sometimes called *fun*

## Default Constants:

`→`     `::`      $bool \Rightarrow bool \Rightarrow bool$   
`=`     `::`      $\alpha \Rightarrow \alpha \Rightarrow bool$   
`ε`     `::`      $(\alpha \Rightarrow bool) \Rightarrow \alpha$

# Higher Order Abstract Syntax

---

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

**One approach:**  $\forall :: var \Rightarrow term \Rightarrow bool$

**Drawback:** need to think about substitution,  $\alpha$  conversion again.

**But:** Already have binder, substitution,  $\alpha$  conversion in meta logic

$\lambda$

**So:** Use  $\lambda$  to encode all other binders.

## 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.

→ **mixfix:**

**consts** `drvbl :: ct ⇒ ct ⇒ fm ⇒ bool ("_,_ ⊢ _")`

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

→ **priorities:**

pattern can be annotated with priorities to indicate binding strength

**Example:** `drvbl :: ct ⇒ ct ⇒ fm ⇒ bool ("_,_ ⊢ _" [30, 0, 20] 60)`

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

**Example:** `or :: bool ⇒ bool ⇒ bool (infixr " ∨ " 30)`

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

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

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

**Example** `ALL :: (α ⇒ bool) ⇒ bool (binder "∀" 10)`

More in Isabelle/Isar Reference Manual (7.2)



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

## And the rest is definitions:

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

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

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

$False \equiv \forall P. P$

$\neg P \equiv P \longrightarrow False$

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

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

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

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

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

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

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

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

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

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

$$\frac{}{\exists f :: \text{ind} \implies \text{ind. inj } f \wedge \neg \text{surj } f} \text{ infnty}$$

## 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**

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:**  $\llbracket \langle prop \rangle_1; \langle prop \rangle_2; \dots \rrbracket \implies \langle prop \rangle$

# True

---

**consts** True :: *bool*

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

## Intuition:

right hand side is always true

## Proof Rules:

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

## Proof:

$$\frac{\frac{(\lambda x :: \textit{bool}. x) = (\lambda x. x)}{\text{True}}}{\text{True}} \begin{array}{l} \text{refl} \\ \text{unfold True\_def} \end{array}$$

# DEMO

# Universal Quantifier

---

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

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

## 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 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:

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

## Proof: Isabelle Demo



# False

---

**consts** False :: *bool*

False  $\equiv \forall P.P$

## Intuition:

Everything can be derived from *False*.

## Proof Rules:

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

**Proof:** Isabelle Demo

**consts** Not :: *bool*  $\Rightarrow$  *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:

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

**Proof:** Isabelle Demo

# Existential Quantifier

**consts** EX ::  $(\alpha \Rightarrow \text{bool}) \Rightarrow \text{bool}$

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:

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

**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:

$$\frac{A \quad B}{A \wedge B} \text{ conjI} \qquad \frac{A \wedge B \quad \llbracket A; B \rrbracket \Longrightarrow C}{C} \text{ conjE}$$

## Proof: Isabelle Demo

**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:

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

## Proof: Isabelle Demo

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

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

## Intuition:

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

## Proof Rules:

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

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

## Proof: Isabelle Demo

# THAT WAS HOL

**Last time:** safe and unsafe, 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	<b>declare</b> conjl [intro!] allE [elim]
remove attribute globally	<b>declare</b> allE [rule del]
use locally	<b>apply</b> (blast intro: someI)
delete locally	<b>apply</b> (blast del: conjl)



## DEMO: AUTOMATION

## We have learned today ...

---



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