



COMP4161: Advanced Topics in Software Verification

HOL

Gerwin Klein, Johannes Åman Pohjola, Christine Rizkallah, Miki Tanaka
T3/2020

Content

→ Foundations & Principles

- Intro, Lambda calculus, natural deduction [1,2]
- Higher Order Logic, Isar (part 1) [2,3^a]
- Term rewriting [3,4]

→ Proof & Specification Techniques

- Inductively defined sets, rule induction, datatype induction, primitive recursion [4,5]
- General recursive functions, termination proofs [7^b]
- Proof automation, Hoare logic, proofs about programs, invariants [8]
- C verification [9,10]
- Practice, questions, examp prep [10^c]

^aa1 due; ^ba2 due; ^ca3 due

More on Automation

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

This can be automated

Automated methods (fast, blast, clarify etc) are not hardwired.
Safe/unsafe intro/elim rules can be declared.

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 conjl [intro!] allE [elim]
remove attribute globally	declare allE [rule del]
use locally	apply (blast intro: somel)
delete locally	apply (blast del: conil)

Demo: Automation

Exercises

- derive the classical contradiction rule $(\neg P \implies \text{False}) \implies P$ in Isabelle
- define **nor** and **nand** in Isabelle
- show $\text{nor } x \ x = \text{nand } x \ x$
- derive safe intro and elim rules for them
- use these in an automated proof of $\text{nor } x \ x = \text{nand } x \ x$

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

Defining Higher Order Logic

Default types:

bool $- \Rightarrow -$ ind

→ **bool** sometimes called *o*

→ \Rightarrow sometimes called *fun*

Default Constants:

\longrightarrow :: $\text{bool} \Rightarrow \text{bool} \Rightarrow \text{bool}$

$=$:: $\alpha \Rightarrow \alpha \Rightarrow \text{bool}$

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

Higher Order Abstract Syntax

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

One approach: $\forall :: \text{var} \Rightarrow \text{term} \Rightarrow \text{bool}$

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.

Higher Order Abstract Syntax

Example:

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

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

→ **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 :: (τ1 ⇒ τ2) ⇒ τ3 (binder "B" < p >)`

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

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

More in Isabelle/Isar Reference Manual (7.2)

Back to HOL

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

And the rest is definitions:

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

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

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

$\text{False} \equiv \forall P. P$

$\neg P \equiv P \longrightarrow \text{False}$

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

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

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

$\text{inj } f \equiv \forall x\ y. f\ x = f\ y \longrightarrow x = y$

$\text{surj } f \equiv \forall y. \exists x. y = f\ x$

The Axioms of HOL

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

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

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

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

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

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

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

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: $\llbracket \langle \textit{prop} \rangle_1; \langle \textit{prop} \rangle_2; \dots \rrbracket \implies \langle \textit{prop} \rangle$

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:

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

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

Negation

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:

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

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:

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

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

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:

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

Proof: Isabelle Demo

If-Then-Else

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

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

We have learned today ...

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