Slide 1

NICTA Advanced Course

Theorem Proving
Principles, Techniques, Applications

HOL

Slide 2

CONTENT

➜ Intro & motivation, getting started with Isabelle

➜ Foundations & Principles
  • Lambda Calculus
  • Higher Order Logic, natural deduction
  • Term rewriting

➜ Proof & Specification Techniques
  • Datatypes, recursion, induction
  • Inductively defined sets, rule induction
  • Calculational reasoning, mathematics style proofs
  • Hoare logic, proofs about programs

Slide 3

LAST TIME ON HOL

➜ Proof rules for propositional and predicate logic
➜ Safe and unsafe rules
➜ Forward Proof
➜ The Epsilon Operator
➜ Some automation

Slide 4

DEFINING HIGHER ORDER LOGIC

LAST TIME ON HOL 1

WHAT IS HIGHER ORDER LOGIC? 2
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 with certain default types and constants

---

Defining Higher Order Logic

Default types:

| bool | _ ⇒ _ | ind |

→ bool sometimes called α

Some times called fun

Default Constants:

| → | bool ⇒ bool ⇒ bool |
| = | α ⇒ α ⇒ bool |
| ε | (α ⇒ bool) ⇒ α |

---

Higher Order Abstract Syntax

Problem: Define syntax for binders like ∀, ∃, ε

One approach: ∀ :: var ⇒ term ⇒ bool

Drawback: need to think about substitution, α conversion again.

Slide 7

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

\[ \lambda \]

So: Use λ to encode all other binders.

---

Higher Order Abstract Syntax

Example:

\[ \text{ALL} :: (α ⇒ bool) ⇒ bool \]

HOAS usual syntax

\[ \text{ALL} (\lambda x. x = 2) \quad ∀x. x = 2 \]

\[ \text{ALL} P \quad ∀x. P x \]

Isabelle can translate usual binder syntax into HOAS.
Slide 9

**Slide 9:**

**Side Track: Syntax Declarations in Isabelle**

- **mixfix:**
  - `consts drvbl :: ct ⇒ ct ⇒ fm ⇒ bool (":" ⊢ ")`
  - Legal syntax now: `Γ, Π ⊢ F`

- **priorities:**
  - Pattern can be annotated with priorities to indicate binding strength
  - Example: `drvbl :: ct ⇒ ct ⇒ fm ⇒ bool ("(" \+ "") [30,0,20] 60)`

- **infix/infixr:** short form for left/right associative binary operators
  - Example: `or :: bool ⇒ bool ⇒ bool (infixr "_30")`

- **binders:** declaration must be of the form
  - Example: `ALL :: (α ⇒ bool) ⇒ bool ("∀\cdot" 10)`

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

Slide 10

**Back to HOL**

**Base:**

- `bool, ⇒, ind` = ⊥, ⊤, ε

**And the rest is definitions:**

- `True` = `(λx :: bool. x) = (λx. x)`
- `All P` = `(λx. True)`
- `Ex P` = `(∀Q. (∀x. P x ⇒ Q) ⇒ Q)`
- `False` = `(∀P. P)`
- `¬P` = `P ⇒ False`
- `P ∨ Q` = `(∀R. (P ⇒ Q ⇒ R) ⇒ R)`
- `P ∧ Q` = `(∀R. (P ⇒ R) ⇒ (Q ⇒ R) ⇒ R)`
- `If P x y` = `(P = True ⇒ x = y) ∧ (P = False ⇒ z = y)`
- `inj f` = `(∀x y. f x = f y ⇒ x = y)`
- `surj f` = `(∀y. ∃x. y = f x)`

**THE AXIOMS OF HOL**

- `t = t` refl
- `s = t` subst
- `x = t x = g x` subst
- `(λx. f x) = (λx. g x)` ext
- `P ⇒ Q P ⇒ Q P` imp
- `P ⇒ Q Q` mp
- `(P ⇒ Q) ⇒ (Q ⇒ P) ⇒ (P = Q)` iff
- `P ⇒ True ⇒ P ⇒ False` True or False
- `P ⇒ P` somel
- `∃f :: ind ⇒ ind. inj f ∧ ¬surj f` infty

Slide 11

**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:

- `x = y` eq_reflection
- `(THE x. x = a) = a` 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"
...
shows " < prop > " < proof >

proves: [ < prop >_1; < prop >_2; ... ] \implies < prop >
```
**Universal Quantifier**

consts \( \text{ALL} \) :: \((\alpha \Rightarrow \text{bool}) \Rightarrow \text{bool}\)
\[ \text{ALL } P \equiv P = (\lambda x. \text{True}) \]

Intuition:
- \( \text{ALL } P \) is Higher Order Abstract Syntax for \( \forall x. P x \).
- \( P \) is a function that takes an \( x \) and yields a truth values.
- \( \text{ALL } P \) 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:

\[
\begin{align*}
\frac{\Delta, x: P}{} & \quad \forall x. P x \\
\frac{\forall x. P x \ P \ ?x \Longrightarrow R}{} & \quad \text{all}E
\end{align*}
\]

Proof: Isabelle Demo

---

**False**

consts \( \text{False} \) :: \text{bool}
\[ \text{False} \equiv \forall P. P \]

Intuition:
Everything can be derived from \( \text{False} \).

Proof Rules:

\[
\begin{align*}
\frac{\text{False}}{P} & \quad \text{FalseE} \\
\frac{}{\text{True} \neq \text{False}} & \quad \text{TrueE}
\end{align*}
\]

Proof: Isabelle Demo

---

**Existential Quantifier**

consts \( \text{EX} \) :: \((\alpha \Rightarrow \text{bool}) \Rightarrow \text{bool}\)
\[ \text{EX } P \equiv \forall Q. (\forall x. P x \Longrightarrow Q) \Longrightarrow Q \]

Intuition:
- \( \text{EX } P \) 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:

\[
\begin{align*}
\frac{P \ ?x}{} & \quad \exists x. P x \\
\frac{\exists x. P x}{} & \quad \text{exI}
\end{align*}
\]

Proof: Isabelle Demo

---

**Negation**

consts \( \text{Not} \) :: \text{bool} \Rightarrow \text{bool} (\sim)
\[ \sim P \equiv P \Longrightarrow \text{False} \]

Intuition:
Try \( P = \text{True} \) and \( P = \text{False} \) and the traditional truth table for \( \Longrightarrow \).

Proof Rules:

\[
\begin{align*}
\frac{A \Longrightarrow \text{False}}{\sim A} & \quad \text{notI} \\
\frac{\sim A \ A}{} & \quad \text{notE}
\end{align*}
\]

Proof: Isabelle Demo

---

**Conjunction**

consts \( \text{AND} \) :: \((\alpha \Rightarrow \text{bool}) \Rightarrow \text{bool}\)
\[ \text{AND } P \equiv P \wedge Q \]

Intuition:
- \( \text{AND } P \) 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:

\[
\begin{align*}
\frac{P \ ?x}{} & \quad \exists x. P x \\
\frac{\exists x. P x}{} & \quad \text{exI}
\end{align*}
\]

Proof: Isabelle Demo
**CONJUNCTION**

**consts** And :: bool ⇒ bool ⇒ bool (\(\land\))
\[ P \land Q \equiv \forall R. (P \rightarrow Q \rightarrow R) \rightarrow R \]

Intuition:
- Mirrors proof rules for \(\land\)
- Try truth table for \(P, Q,\) and \(R\)

**Proof Rules:**

\[
\begin{array}{c}
A \land B \equiv C \\
A \land B \vdash C
\end{array}
\]

**Proof:** Isabelle Demo

---

**DISJUNCTION**

**consts** Or :: bool ⇒ bool ⇒ bool (\(\lor\))
\[ P \lor Q \equiv \forall R. (P \rightarrow R) \rightarrow (Q \rightarrow R) \rightarrow R \]

Intuition:
- Mirrors proof rules for \(\lor\) (case distinction)
- Try truth table for \(P, Q,\) and \(R\)

**Proof Rules:**

\[
\begin{array}{c}
A \lor B \equiv C \\
A \lor B \vdash C
\end{array}
\]

**Proof:** Isabelle Demo

---

**IF-THEN-ELSE**

**consts** If :: bool ⇒ α ⇒ α (if _ then _ else _)
If \(P x y \equiv \text{SOME } z. (P = \text{True} \rightarrow z = x) \land (P = \text{False} \rightarrow z = y)\)

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:**

\[
\begin{array}{c}
\text{if True then } s \text{ else } t = s \\
\text{if False then } s \text{ else } t = t
\end{array}
\]

**Proof:** Isabelle Demo

---

**THAT WAS HOL**

---

**IF-THEN-ELSE**

---

**MORE ON AUTOMATION**
**More on Automation**

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

*This can be automated*

**Syntax:**
- \(<\text{kind}>\) for safe rules (\(<\text{kind}>\) one of intro, elim, dest)
- \(<\text{kind}>\) for unsafe rules

**Application** (roughly):
- do safe rules first, search/backtrack on unsafe rules only

**Example:**
- declare attribute globally
- remove attribute globally
- use locally
- delete locally
- declare \(\text{conj} [\text{intro}]\) allE [elim]
- apply (blast intro: somel)
- apply (blast del: conjl)

---

**Demo: Automation**

---

**We have learned today ...**

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

---

**Exercises**

- derive the classical contradiction rule \((\neg P \Rightarrow False) \Rightarrow P\) in Isabelle
- define \textit{nor} and \textit{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 \(\text{nor} \ x \ x = \text{nand} \ x \ x\)