Slide 1

Theorem Proving
Principles, Techniques, Applications

HOL

Slide 2

QUASI ORDERS

\[ \leq :: \alpha \rightarrow \alpha \rightarrow \text{bool} \]

is a quasi order iff it satisfies

- \( x \leq x \) (reflexivity) and
- \( x \leq y \land y \leq z \rightarrow x \leq z \) (transitivity)

(a partial order is also antisymmetric: \( x \leq y \land y \leq x \rightarrow x = y \))

Slide 3

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

Last Time on HOL

- natural deduction rules for \( \land, \lor \) and \( \rightarrow \)
- proof by assumption

Slide 4

- proof by intro rule
- proof by elim rule
More Proof Rules

Iff, Negation, True and False

\[
\frac{A \implies B \quad B \implies A}{A = B} \text{ iff} \quad \frac{A = B \quad [A \implies B; B \implies A]}{C} \text{ iffE}
\]

\[
\frac{A = B}{A \implies B} \text{ iffD1} \quad \frac{A = B}{B \implies A} \text{ iffD2}
\]

\[
\frac{A \implies \text{False}}{\neg A} \text{ notI} \quad \frac{\neg A}{A} \text{ notE}
\]

\[
\frac{\text{True}}{P} \text{ TrueI} \quad \frac{\text{False}}{P} \text{ FalseE}
\]

Equality

\[
\frac{t = t}{s = t} \text{ refl} \quad \frac{t = s}{r = s} \text{ sym} \quad \frac{r = t}{s = t} \text{ trans}
\]

\[
\frac{s = t}{P \quad P \quad \frac{P}{s}} \text{ subst}
\]

Rarely needed explicitly — used implicitly by term rewriting

Demo
CLASSICAL

\[ P = \text{True} \lor P = \text{False} \tag{True-False} \]
\[ P \lor \neg P \tag{excluded-middle} \]

\[ \neg A \implies \text{False} \]
\[ A \]
\[ A \implies A \] classical

\rightarrow \text{excluded-middle, ccontr and classical}
not derivable from the other rules.

\rightarrow \text{if we include True-False, they are derivable}

They make the logic “classical”, “non-constructive”

CASES

\[ P \lor \neg P \tag{excluded-middle} \]

is a case distinction on type bool

\begin{align*}
\text{Apply safe rules before unsafe ones}
\end{align*}

SAFE AND NOT SO SAFE

Safe rules preserve provability

\begin{align*}
\text{conjI, impl, notI, iffi, refl, ccontr, classical, conjE, disjE} \\
\frac{A \land B}{A} \tag{conjI} \\
\end{align*}

Unsafe rules can turn a provable goal into an unprovable one

\begin{align*}
\text{disjI1, disjI2, impE, iffD1, iffD2, notE} \\
\frac{A \lor B}{A} \tag{disjI1} \\
\end{align*}

Isabelle can do case distinctions on arbitrary terms:

\begin{align*}
\text{apply (case_tac term)}
\end{align*}
QUANTIFIERS

Scope

- Scope of parameters: whole subgoal
- Scope of $\forall, \exists, \ldots$: ends with $:$ or $\implies$

Example:

$\forall x. \exists y. P x \implies Q x y$ means

$\forall x. \exists y_1. P x_1 \implies Q x_1 y_1; \ Q x y \implies (\exists x_1. \ Q x_1 y)$

Natural Deduction for Quantifiers

### Slide 15

**Natural Deduction for Quantifiers**

- **all** and **ex** introduce new parameters ($\forall x$).
- **allE** and **ex** introduce new unknowns ($?x$).

### Slide 16

**Instantiating Rules**

apply (rule_tac $x = "term"$ in rule)

Like rule, but $?x$ in rule is instantiated by $term$ before application.

Similar: erule_tac

\[ x \text{ is in rule, not in goal} \]

Two Successful Proofs
**Two Successful Proofs**

1. \( \forall x. \exists y. x = y \)
   - apply (rule all)
2. \( \forall x. \exists y. x = y \)
   - apply (rule all)

**Slide 17**

<table>
<thead>
<tr>
<th>best practice</th>
<th>exploration</th>
</tr>
</thead>
<tbody>
<tr>
<td>apply (rule tac ( x = &quot;x&quot; ) in exI)</td>
<td>apply (rule exI)</td>
</tr>
<tr>
<td>( \forall x. x = x )</td>
<td>apply (rule refl)</td>
</tr>
<tr>
<td>apply (rule refl)</td>
<td>( \forall x. x = ?y )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>simpler &amp; clearer</th>
<th>shorter &amp; trickier</th>
</tr>
</thead>
<tbody>
<tr>
<td>apply (rule refl)</td>
<td>( ?y \mapsto \lambda u. u )</td>
</tr>
</tbody>
</table>

**Two Unsuccessful Proofs**

1. \( \exists y. \forall x. x = y \)
   - apply (rule \( \forall x. x = ?y \) in exI)
2. \( \exists y. \forall x. x = y \)
   - apply (rule exI)

**Slide 18**

**Slide 19**

**Safe and Unsafe Rules**

- **Safe** all, exE
- **Unsafe** allE, exI

Create parameters first, unknowns later

**Demo: Quantifier Proofs**

**Principle:**

\( ?f \, x_1 \ldots x_n \) can only be replaced by term \( t \)
if \( \text{params}(t) \subseteq x_1, \ldots, x_n \)
Parameter names are chosen by Isabelle

1. \( \forall x. \exists y. x = y \)

apply (rule all)

1. \( \forall x. \exists y. x = y \)

apply (rename_tac \( x \) = "x" in ex1)

Brittle!

Renaming parameters

1. \( \forall x. \exists y. x = y \)

apply (rule all)

1. \( \forall x. \exists y. x = y \)

apply (rename_tac \( N \))

1. \( \forall N. \exists y. N = y \)

apply (rename_tac \( x \) = "N" in ex1)

In general:
\( \text{(rename_tac} x_1 \ldots x_n) \) renames the rightmost (inner) \( n \) parameters to \( x_1 \ldots x_n \)

FORWARD PROOF: frule and drule

apply (frule < rule >)

Rule:
\[ [A_1; \ldots; A_m] \Rightarrow A \]

Subgoal:
1. \[ [B_1; \ldots; B_n] \Rightarrow C \]

Substitution:
\[ \sigma(B_i) \equiv \sigma(A_k) \]

New subgoals:
1. \[ \sigma([B_1; \ldots; B_n] \Rightarrow A_2) \]

Like frule but also deletes \( B_i \): apply (drule < rule >)

Examples for Forward Rules

\[
\frac{P \land Q}{P} \quad \text{conject1} \quad \frac{P \land Q}{Q} \quad \text{conject2}
\]

\[
\frac{P \to Q}{P} \quad \frac{Q}{P} \quad \text{mp}
\]

\[
\frac{\forall x. P}{P \forall x} \quad \text{spec}
\]
**FORWARD PROOF: OF**

\[ r [\text{OF} r_1 \ldots r_n] \]

Prove assumption 1 of theorem \( r \) with theorem \( r_1 \), and assumption 2 with theorem \( r_2 \), and \( \ldots \)

**Slide 25**

Rule \( r \) \[ [A_1; \ldots; A_m] \Rightarrow A \]

Rule \( r_1 \) \[ [B_1; \ldots; B_n] \Rightarrow B \]

Substitution \( \sigma(B) \equiv \sigma(A_1) \)

\( r [\text{OF} r_1] \) \( \sigma([B_1; \ldots; B_n; A_2; \ldots; A_m] \Rightarrow A) \)

---

**FORWARD PROOFS: THEN**

\( r_1 \ [\text{THEN} \ r_2] \) means \( r_2 [\text{OF} \ r_1] \)

**Slide 26**

---

**DEMO: FORWARD PROOFS**

**HILBERT’S EPSILON OPERATOR**

(David Hilbert, 1862-1943)

\( \epsilon \ x. \ P \ x \) is a value that satisfies \( P \) (if such a value exists)

\( \epsilon \) also known as **description operator**.

In Isabelle the \( \epsilon \)-operator is written \( \text{SOME} \ x. \ P \ x \)

\[
\frac{P \ ?x}{P \ (\text{SOME} \ x. \ P \ x)} \text{ somel}
\]
MORE EPSILON

\[ \varepsilon \text{ implies Axiom of Choice:} \]
\[ \forall x. \exists y. Q \ x \ y \implies \exists f. \ \forall x. Q \ (f \ x) \]

Existential and universal quantification can be defined with \( \varepsilon \).

Slide 29

Isabelle also knows the definite description operator \( \text{THE} \) (also \( \iota \)):

\[ (\text{THE} \ x. \ x = a) = a \]

\[ \text{the}_\text{eq}_\text{trivial} \]

SOME AUTOMATION

More Proof Methods:

- **apply** (intro <intro-rules>) repeatedly applies intro rules
- **apply** (elim <elim-rules>) repeatedly applies elim rules
- **apply** clarify applies all safe rules that do not split the goal
- **apply** safe applies all safe rules
- **apply** blast an automatic tableaux prover (works well on predicate logic)
- **apply** fast another automatic search tactic

Slide 30

EPSILON AND AUTOMATION DEMO

WE HAVE LEARNED SO FAR...

- Proof rules for negation and contradiction
- Proof rules for predicate calculus
- Safe and unsafe rules
- Forward Proof
- The Epsilon Operator
- Some automation

Slide 31

EXERCISES
EXERCISES

➜ Download the exercise file and prove all theorems in there.

➜ Prove or disprove:

If every poor person has a rich mother, then there is a rich person with a rich grandmother.