COMP 4161  
NICTA Advanced Course  
Advanced Topics in Software Verification  

Gerwin Klein  
Formal Methods

HOL

Slide 1

LAST TIME ON HOL

natural deduction rules for ∧, ∨ and →  
proof by assumption  
proof by intro rule  
proof by elim rule

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

MORE PROOF RULES

Slide 4
### IFF, Negation, True and False

\[ A \Rightarrow B \quad B \Rightarrow A \] \iff

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

\[ \frac{A = B}{A ightarrow B} \] \text{id1}

\[ \frac{A ightarrow False}{\neg A} \] \text{notI}

\[ False \] \text{FalseE}

\[ \frac{True}{A} \] \text{TrueI}

\[ \frac{\neg A}{A} \] \text{notE}

\[ True \]

\[ False \]

**Slide 5**

### Equality

\[ t = t \] \text{refl}

\[ \frac{t = s}{s = t} \] \text{sym}

\[ \frac{t = s; s = t}{t = r} \] \text{trans}

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

Rarely needed explicitly — used implicitly by term rewriting

**Slide 6**

### Classical

\[ P = True \lor P = False \] \text{True-False}

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

\[ \frac{\neg A \Rightarrow False}{\neg \neg A} \] \text{ccontr}

\[ \frac{\neg A \Rightarrow A}{A} \] \text{classical}

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

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

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

**Slide 7**

### Cases

\[ P \lor P \] \text{excluded-middle}

Isabelle can do case distinctions on arbitrary terms:

\[ \text{apply (case_tac term)} \]

**Slide 8**
SAFE AND NOT SO SAFE

Safe rules preserve provability
- \( \text{conjI, impl, notI, iffI, refl, ccontr, classical, conjE, disjE} \)
- \( \frac{A}{A \land B} \text{ conjI} \)

Unsafe rules can turn a provable goal into an unprovable one
- \( \text{disjI1, disjI2, impE, iffD1, iffD2, notE} \)
- \( \frac{A}{A \lor B} \text{ disjI1} \)

Apply safe rules before unsafe ones

QUANTIFIERS

Scope
- Scope of parameters: whole subgoal
- Scope of \( \forall, \exists, \ldots \): ends with \( \land \) or \( \lor \)

Example:
- \( \forall x y. [\forall y. P y \Longrightarrow Q z y; \ Q x y] \Longrightarrow \exists x. Q x y \) means
- \( \forall x y. [\forall y_1. P y_1 \Longrightarrow Q z y_1; \ Q x y] \Longrightarrow (\exists x_1. Q x_1 y) \)
NATURAL DEDUCTION FOR QUANTIFIERS

\[ \forall x. P x \quad \exists x. P x \]

\[ \forall x. P x \quad \exists x. P x \]

\[ \text{allI} \quad \text{exI} \]

\[ \text{allE} \quad \text{exE} \]

\[ \] allI and exE introduce new parameters (\( \forall x \)).

\[ \] allE and exI introduce new unknowns (\( ?x \)).

Slide 13

INSTANTIATING RULES

apply (rule \( \lambda x. \text{term} \) in rule)

Like rule, but \( ?x \) in rule is instantiated by \text{term} before application.

Similar: erule \( \lambda x \)

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

Slide 14

TWO SUCCESSFUL PROOFS

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

apply (rule allI)

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

best practice exploration

apply (rule \( \lambda x. \text{term} \) in exI)

apply (rule exI)

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

apply (rule refl)

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

apply (rule refl)

\( \lambda y \mapsto \lambda u.u \)

simpler & clearer shorter & trickier

Slide 15

TWO UNSUCCESSFUL PROOFS

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

apply (rule \( \lambda x. \text{term} \) in exI)

apply (rule exI)

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

apply (rule refl)

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

apply (rule refl)

\( ?y \mapsto \lambda u.u \)

Principle:

\( ?f \) \( x_1 \ldots x_n \) can only be replaced by \text{term} \( t \)

If \( \text{params}(t) \subseteq x_1, \ldots, x_n \)

Slide 16
SAFE AND UNSAFE RULES

Safe allE, exI

Unsafe allI, exE

Create parameters first, unknowns later

PARAMETER NAMES

Parameter names are chosen by Isabelle

1. \( \forall x. \exists y. x = y \)
   apply (rule allI)
1. \( \exists y. x = y \)
   apply (rename_tac \( x \) = "\( x \)" in exI)

Brittle!

RENAME PARAMETERS

1. \( \forall x. \exists y. x = y \)
   apply (rule allI)
1. \( \exists y. x = y \)
   apply (rename_tac \( N \) = "\( N \)" in exI)

In general:
(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:**

\[ \sigma(B_i) = \sigma(A_i) \]

**New subgoals:**

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

...m-1. \[ \sigma([B_1; \ldots; B_n] \Rightarrow A_m) \]

m. \[ \sigma([B_1; \ldots; B_n; A] \Rightarrow C) \]

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

**EXAMPLES FOR FORWARD RULES**

\[ \frac{P \land Q}{P} \text{ conjunct1} \]

\[ \frac{P \land Q}{Q} \text{ conjunct2} \]

\[ \frac{P \Rightarrow Q}{Q} \text{ mp} \]

\[ \frac{\forall x. P \Rightarrow Q}{P_1 \Rightarrow Q} \text{ spec} \]

**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 ...

**Rule**

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

**Rule**

\[ [B_1; \ldots; B_n] \Rightarrow B \]

**Substitution**

\[ \sigma(B_i) = \sigma(A_i) \]

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

**FORWARD PROOFS: THEN**

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

Slide 21

Slide 22

Slide 23

Slide 24
**DEMO: FORWARD PROOFS**

**HILBERT’S EPSILON OPERATOR**

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

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

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

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

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

**MORE EPSILON**

\( \varepsilon \) implies Axiom of Choice:

\[ \forall x. \exists y. Q x y \implies \exists f. \forall x. Q x (f x) \]

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

Isabelle also knows the definite description operator **THE** (aka \( \iota \)):

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

**SOME AUTOMATION**

\[ \text{eq trivial} \]
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