COMP 4161
NICTA Advanced Course

Advanced Topics in Software Verification

Gerwin Klein
Formal Methods

HOL

Slide 1

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 2

Slide 3

QUANTIFIERS

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

Example:

$\forall x y. [ \forall y. P y \Rightarrow Q z y; Q x y ] \Rightarrow \exists x. Q x y$

means

$\forall x y. [ (\forall y. P y) \Rightarrow Q z y]; Q x y ] \Rightarrow (\exists x. Q x_1 y)$

Slide 4
NATURAL DEDUCTION FOR QUANTIFIERS

\[ \forall x. P x \hspace{1cm} \forall x. P x \]

allI

\[ \exists x. P x \hspace{1cm} \exists x. P x \]

exE

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

• allE and exI introduce new unknowns (?x).

INSTANTIATING RULES

\textbf{apply} (rule tac \( x = "\text{term}" \) in \( \text{rule} \))

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

Similar: \texttt{erule_tac}

\(! \hspace{0.5cm} \textit{x is in rule, not in goal} \)
**SAFE AND UNSAFE RULES**

Safe all, exE

Unsafe allE, exI

Create parameters first, unknowns later

---

**PARAMETER NAMES**

Parameter names are chosen by Isabelle

1. \( \forall x. \exists y. x = y \)
2. \( \forall x. \exists y. x = y \)
3. \( \forall N. \exists y. N = y \)

Brittle!

---

**DEMO: QUANTIFIER PROOFS**

**RENAME PARAMETERS**

In general:

(\text{rename}_{\text{Jac}} 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: \[ [A_1; \ldots; A_m] \implies A \]

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

Substitution: \( \sigma(B_i) = \sigma(A_i) \)

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

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

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

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

---

**FORWARD PROOF: OF**

\( r \circ OF_{r_1; \ldots; r_n} \)

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

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

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

Substitution \( \sigma(B) = \sigma(A_i) \)

\( r \circ OF_{r_1} \) \( \sigma([B_1; \ldots; B_n; A_2; \ldots; A_m] \implies A) \)

---

**EXAMPLES FOR FORWARD RULES**

\[ P \land Q \]

\( conjunct1 \)

\[ P \land Q \]

\( conjunct2 \)

\[ P \implies Q \]

\( mp \)

\[ \forall x, P \]

\( spec \)

---

**FORWARD PROOFS: THEN**

\( r_1 \circ THEN_{r_2} \) means \( r_2 \circ OF_{r_1} \)

---

**Slide 13**

**Slide 15**

**Slide 14**

**Slide 16**
HILBERT’S EPSILON OPERATOR

(David Hilbert, 1862-1943)

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

$\varepsilon$ also known as description operator.

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

$P : \exists \ v. P \ v$ somel

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 $\text{THE}$ (aka $\iota$):

$(\text{THE} \ x. x = a) = a$ \text{the_eq_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
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