COMP4161: Advanced Topics in Software Verification

HOL

Gerwin Klein, June Andronick, Ramana Kumar, Miki Tanaka
S2/2017

data61.csiro.au
Last time...

- natural deduction rules for $\land$, $\lor$, $\rightarrow$, $\neg$, iff...
- proof by assumption, by intro rule, elim rule
- safe and unsafe rules

- indent your proofs! (one space per subgoal)
- prefer implicit backtracking (chaining) or `rule_tac`, instead of `back`
- prefer and defer
- *oops* and *sorry*
Content

→ Intro & motivation, getting started

→ Foundations & Principles
  • Lambda Calculus, natural deduction [1,2]
  • Higher Order Logic [3^a]
  • Term rewriting [4]

→ Proof & Specification Techniques
  • Inductively defined sets, rule induction [5]
  • Datatypes, recursion, induction [6, 7]
  • Hoare logic, proofs about programs, C verification [8^b,9]
  • (mid-semester break)
  • Writing Automated Proof Methods [10]
  • Isar, codegen, typeclasses, locales [11^c,12]

^a1 due; ^b2 due; ^c3 due
Quantifiers
Scope

- 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

$$\forall x \ y. [ (\forall y_1. P \ y_1 \rightarrow Q \ z \ y_1); \ Q \ x \ y ] \Rightarrow (\exists x_1. Q \ x_1 \ y)$$
Natural deduction for quantifiers

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

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

- **allI** and **exE** introduce new parameters (\( \forall x \)).
- **allE** and **exI** introduce new unknowns (\( ?x \)).
Instantiating Rules

apply \( (\text{rule_tac} \; x = "\text{term}" \; \text{in} \; \text{rule}) \)

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

Similar: \texttt{erule_tac}

\(! \; x \; \text{is in} \; \text{rule}, \; \text{not in} \; \text{goal} \; !\)
Two Successful Proofs

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

apply (rule allI)

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

best practice

apply (rule_tac x = "x" in exI)

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

apply (rule refl)

simpler & clearer

exploration

apply (rule exI)

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

apply (rule refl)

?y \mapsto \lambda u. u

shorter & trickier
Two Unsuccessful Proofs

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

   **apply** (rule_tac \( x = ??? \) in exl)

   **apply** (rule exl)

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

   **apply** (rule alll)

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

   **apply** (rule refl)

   \( ?y \mapsto x \) yields \( \land x'. x' = x \)

**Principle:**

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

if \( \text{params}(t) \subseteq x_1, \ldots, x_n \)
Safe and Unsafe Rules

Safe  allI, exE
Unsafe allE, exl

Create parameters first, unknowns later
Demo: Quantifier Proofs
Parameter names

Parameter names are chosen by Isabelle

1. $\forall x. \exists y. x = y$

apply (rule allI)

1. $\land x. \exists y. x = y$

apply (rule_tac x = "x" in exI)

Brittle!
Renaming parameters

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

**apply** (rule allI)

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

**apply** (rename_tac \( N \))

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

**apply** (rule_tac \( x = "N" \) in exI)

---

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: \([A_1; \ldots; A_m] \implies A\)

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

Substitution: \(\sigma(B_i) \equiv \sigma(A_1)\)

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

\[\vdots\]

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 >)
Examples for Forward Rules

\[
\frac{P \land Q}{P} \quad \text{conjunct1} \quad \frac{P \land Q}{Q} \quad \text{conjunct2}
\]

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

\[
\forall x. P \ x \quad \frac{P ?x}{P} \quad \text{spec}
\]
Forward Proof: OF

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

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

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

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

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

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

\[ \sigma([B_1; \ldots; B_n; A_2; \ldots; A_m] \implies A) \]
Forward proofs: THEN

\[ r_1 \text{ [THEN } r_2 \text{]} \quad \text{means} \quad r_2 \text{ [OF } r_1 \text{]} \]
Demo: Forward Proofs
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 SOME $x. \ P x$

\[
\frac{P \ ? x}{P \ (\text{SOME } x. \ P x)} \ \text{somel}
\]
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 \( \iota \)):

\[
\frac{(\text{THE } x. \ x = a) = a}{\text{the_eq_trivial}}
\]
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 predicate calculus
- Safe and unsafe rules
- Forward Proof
- The Epsilon Operator
- Some automation