



COMP4161: Advanced Topics in Software Verification

HOL

Gerwin Klein, June Andronick, Ramana Kumar
S2/2016

data61.csiro.au



Last time...



- natural deduction rules for \wedge , \vee , \longrightarrow , \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 [1]

- 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]

^aa1 due; ^ba2 due; ^ca3 due

Quantifiers

Scope



- Scope of parameters: whole subgoal
- Scope of \forall, \exists, \dots : ends with ; or \implies

Example:

$$\bigwedge x y. \llbracket \forall y. P y \longrightarrow Q z y; Q x y \rrbracket \implies \exists x. Q x y$$

means

$$\bigwedge x y. \llbracket (\forall y_1. P y_1 \longrightarrow Q z y_1); Q x y \rrbracket \implies (\exists x_1. Q x_1 y)$$

Natural deduction for quantifiers



$$\frac{\bigwedge x. P x}{\bigvee x. P x} \text{ allI}$$

$$\frac{\bigvee x. P x \quad P ?x \implies R}{R} \text{ allE}$$

$$\frac{P ?x}{\exists x. P x} \text{ exI}$$

$$\frac{\exists x. P x \quad \bigwedge x. P x \implies R}{R} \text{ exE}$$

- **allI** and **exE** introduce new parameters ($\bigwedge x$).
- **allE** and **exI** introduce new unknowns ($?x$).

Instantiating Rules



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

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

Similar: **erule_tac**

! x is in *rule*, not in goal **!**

Two Successful Proofs



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

apply (rule allI)

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

best practice

apply (rule_tac x = "x" in exI)

$$1. \bigwedge x. x = x$$

apply (rule refl)

simpler & clearer

exploration

apply (rule exI)

$$1. \bigwedge x. x = ?y \ x$$

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 ex1)

apply (rule ex1)

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

apply (rule all1)

1. $\bigwedge x. x = ?y$

apply (rule refl)

$?y \mapsto x$ yields $\bigwedge x'. x' = x$

Principle:

$?f\ x_1 \dots x_n$ **can only be replaced by term t**

if $params(t) \subseteq x_1, \dots, x_n$

Safe and Unsafe Rules



Safe allI, exE

Unsafe allE, exI

Create parameters first, unknowns later



DATA
61



Demo: Quantifier Proofs

Parameter names



Parameter names are chosen by Isabelle

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

apply (rule allI)

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

apply (rule_tac x = "x" in exI)

Brittle!

Renaming parameters



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

apply (rule all)

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

apply (rename_tac N)

1. $\bigwedge N. \exists y. N = y$

apply (rule_tac x = "N" in ex1)

In general:

(rename_tac $x_1 \dots x_n$) renames the rightmost (inner) n parameters to $x_1 \dots x_n$

Forward Proof: frule and drule



apply (frule $\langle rule \rangle$)

Rule: $\llbracket A_1; \dots; A_m \rrbracket \implies A$

Subgoal: 1. $\llbracket B_1; \dots; B_n \rrbracket \implies C$

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

New subgoals: 1. $\sigma(\llbracket B_1; \dots; B_n \rrbracket \implies A_2)$

\vdots

m-1. $\sigma(\llbracket B_1; \dots; B_n \rrbracket \implies A_m)$

m. $\sigma(\llbracket B_1; \dots; B_n; A \rrbracket \implies C)$

Like **frule** but also deletes B_i : **apply** (drule $\langle rule \rangle$)

Examples for Forward Rules



$$\frac{P \wedge Q}{P} \text{ conjunct1} \quad \frac{P \wedge Q}{Q} \text{ conjunct2}$$

$$\frac{P \longrightarrow Q \quad P}{Q} \text{ mp}$$

$$\frac{\forall x. P \ x}{P \ ?_x} \text{ spec}$$

Forward Proof: OF



$$r \text{ [OF } r_1 \dots r_n]$$

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

$$\text{Rule } r \quad \llbracket A_1; \dots; A_m \rrbracket \implies A$$

$$\text{Rule } r_1 \quad \llbracket B_1; \dots; B_n \rrbracket \implies B$$

$$\text{Substitution} \quad \sigma(B) \equiv \sigma(A_1)$$

$$r \text{ [OF } r_1] \quad \sigma(\llbracket B_1; \dots; B_n; A_2; \dots; A_m \rrbracket \implies A)$$

Forward proofs: THEN



r_1 [THEN r_2] means r_2 [OF r_1]



DATA
61



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)

ε also known as **description operator**.

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

$$\frac{P ?x}{P (\text{SOME } x. P x)} \text{ someI}$$

More Epsilon



ε 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 ε .

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

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

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

A decorative background pattern consisting of a grid of white hexagons on a teal background. The hexagons are arranged in a staggered, honeycomb-like structure.

DATA
61



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

Assignment



Assignment 1 is out today!

Reminder: **DO NOT COPY**

- Assignments and exams are take-home. This does NOT mean you can work in groups. Each submission is personal.
- For more info, see Plagiarism Policy