COMP 4161
NICTA Advanced Course

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Toby Murray

HOL
Rough timeline

Intro & motivation, getting started

Foundations & Principles
- Lambda Calculus, natural deduction
- Higher Order Logic
- Term rewriting

Proof & Specification Techniques
- Isar
- Inductively defined sets, rule induction
- Datatypes, recursion, induction
- Calculational reasoning, mathematics style proofs
- Hoare logic, proofs about programs

\[\text{References:}\]

- [1]
- [2,3,4\text{a}]
- [5,6\text{b},7]
- [8,9,10\text{c}]
- [11,12\text{d}]
- [13\text{e},15]
- [16,17\text{f},18,19]
- [20]
- [21\text{g},22,23]

\text{Notes:}\ a1 \text{ out};\ a1 \text{ due};\ a2 \text{ out};\ a2 \text{ due};\ \text{session break}; f a3 \text{ out}; g a3 \text{ due}
QUANTIFIERS
Scope

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

Example:

$$\forall x \, y. \left[ \forall y. \, P \, y \rightarrow Q \, z \, y; \; Q \, x \, y \right] \Rightarrow \exists x. \, Q \, x \, y$$

means

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

\[ \begin{align*}
\forall x. P x & \quad \text{allI} \\
\forall x. P x & \quad \text{allE} \\
\exists x. P x & \quad \text{exI} \\
\exists x. P x & \quad \text{exE}
\end{align*} \]

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

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

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

Similar: \textbf{erule_tac}

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

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

apply (rule allI)

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

best practice

apply (rule_tac x = "x" in exI)

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

apply (rule refl)

exploration

apply (rule exI)

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

apply (rule refl)

?y \mapsto \lambda u.u

simpler & clearer

shorter & trickier
Two Unsuccessful Proofs

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

**apply** (rule_tac \( x = ??? \) in exI)  
**apply** (rule exI)

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

**apply** (rule allI)

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

**apply** (rule refl)

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

**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, exI

Create parameters first, unknowns later
DEMO: QUANTIFIER PROOFS
Parameter names are chosen by Isabelle

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

**apply** (rule allI)

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

**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:

(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:**

\[
\sigma(B_i) \equiv \sigma(A_1)
\]

**New subgoals:**

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

\[
\vdots
\]

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

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

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

\[
\frac{\forall x. P \, x}{P \, ?x} \quad \text{spec}
\]
Prove assumption 1 of theorem $r$ with theorem $r_1$, and assumption 2 with theorem $r_2$, and ...
Forward proofs: THEN

\[ r_1 \ [\text{THEN} \ r_2] \quad \text{means} \quad r_2 \ [\text{OF} \ r_1] \]
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 $\iota$):

$$\text{THE} \ x. \ x = a = a$$
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
We said that $\varepsilon$ implies the Axiom of Choice:

$$\forall x. \exists y. Q x y \implies \exists f. \forall x. Q x (f x)$$

Prove the axiom of choice as a lemma, using only the introduction and elimination rules for $\forall$ and $\exists$, namely allI, exI, allE, exE, and the introduction rule for $\varepsilon$, someI, using only the proof methods rule, rule_tac, erule, erule_tac and assumption.