COMP4161: Advanced Topics in Software Verification

λ and HOL

Gerwin Klein, June Andronick, Christine Rizkallah, Miki Tanaka
S2/2018

data61.csiro.au
Last time...

- Simply typed lambda calculus: $\lambda \rightarrow$
- Typing rules for $\lambda \rightarrow$, type variables, type contexts
- $\beta$-reduction in $\lambda \rightarrow$ satisfies subject reduction
- $\beta$-reduction in $\lambda \rightarrow$ always terminates
- Types and terms in Isabelle
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, invariants [8^b, 9]
  • (mid-semester break)
  • C verification [10]
  • CakeML, Isar [11^c]
  • Concurrency [12]

^a1 due; ^b2 due; ^c3 due
Preview: Proofs in Isabelle
Proofs in Isabelle

General schema:

\textbf{lemma} name: "<goal>"
\textbf{apply} <method>
\textbf{apply} <method>
\ldots
\textbf{done}

→ Sequential application of methods until all \textbf{subgoals} are solved.
The Proof State

1. \( \bigwedge x_1 \ldots x_p \cdot [A_1; \ldots; A_n] \implies B \)
2. \( \bigwedge y_1 \ldots y_q \cdot [C_1; \ldots; C_m] \implies D \)

\( x_1 \ldots x_p \) Parameters
\( A_1 \ldots A_n \) Local assumptions
\( B \) Actual (sub)goal
Isabelle Theories

Syntax:
theory *MyTh*
imports *ImpTh₁ ... ImpThₙ*
begin
(declarations, definitions, theorems, proofs, ...)*
end

➔ *MyTh*: name of theory. Must live in file *MyTh.thy*
➔ *ImpThᵢ*: name of imported theories. Import transitive.

Unless you need something special:
theory *MyTh* imports Main begin ... end
Natural Deduction Rules

For each connective (\(\land\), \(\lor\), etc):

- **introduction** and **elimination** rules
Proof by assumption

apply assumption

proves

1. $[B_1; \ldots; B_m] \Rightarrow C$

by unifying $C$ with one of the $B_i$

There may be more than one matching $B_i$ and multiple unifiers.

Backtracking!

Explicit backtracking command: back
Intro rules decompose formulae to the right of $\implies$.

**apply** (rule `<intro-rule>`)  

Intro rule $[A_1; \ldots ; A_n] \implies A$ means

→ To prove $A$ it suffices to show $A_1 \ldots A_n$

Applying rule $[A_1; \ldots ; A_n] \implies A$ to subgoal $C$:

→ unify $A$ and $C$
→ replace $C$ with $n$ new subgoals $A_1 \ldots A_n$
**Elim rules**

**Elim** rules decompose formulae on the left of \( \Rightarrow \).

**apply** \((\text{erule } <\text{elim-rule}>\))

Elim rule \([A_1; \ldots; A_n] \Rightarrow A\) means

\( \rightarrow \) If I know \( A_1 \) and want to prove \( A \) it suffices to show \( A_2 \ldots A_n \)

Applying rule \([A_1; \ldots; A_n] \Rightarrow A\) to subgoal \( C \):

Like **rule** but also

\( \rightarrow \) unifies first premise of rule with an assumption
\( \rightarrow \) eliminates that assumption
Demo
More Proof Rules
Iff, Negation, True and False

\[
\frac{A \implies B \quad B \implies A}{A = B} \quad \text{iffI}
\]

\[
\frac{A = B}{A \implies B} \quad \text{iffD1}
\]

\[
\frac{A \implies \text{False}}{\neg A} \quad \text{notI}
\]

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

\[
\frac{A = B}{B \implies A} \quad \text{iffD2}
\]

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

\[
\frac{False}{P} \quad \text{FalseE}
\]

\[
\frac{True}{P} \quad \text{Truel}
\]
Equality

\[
\begin{align*}
    t \equiv t & \quad \text{refl} \\
    t \equiv s & \quad \text{sym} \\
    r \equiv s \quad s \equiv t & \quad \text{trans} \\
    s \equiv t & \quad \text{subst}
\end{align*}
\]

Rarely needed explicitly — used implicitly by term rewriting
Classical

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

\[ P \lor \neg P \]  
excluded-middle

\[ \neg A \implies \text{False} \]  
\[ A \]  
ccontr

\[ \neg A \implies A \]  
classical

- excluded-middle, ccontr and classical
  not derivable from the other rules.
- if we include True-or-False, they are derivable

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

\[ \overline{P \lor \neg P} \quad \text{excluded-middle} \]

is a case distinction on type \textit{bool}

Isabelle can do case distinctions on arbitrary terms:

\textbf{apply (case_tac term)}
Safe and not so safe

**Safe rules**  preserve provability

- `conjI`, `impl`, `notI`, `iffI`, `refl`, `ccontr`, `classical`, `conjE`, `disjE`

\[
\frac{A \quad B}{A \land B} \quad \text{conjI}
\]

**Unsafe rules**  can turn a provable goal into an unprovable one

- `disjI1`, `disjI2`, `impE`, `iffD1`, `iffD2`, `notE`

\[
\frac{A}{A \lor B} \quad \text{disjI1}
\]

---

Apply safe rules before unsafe ones
Demo
What we have learned so far...

- 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
Assignment

Assignment 1 will be out on Monday, the 3rd of August!

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