



DATA  
61

COMP4161: Advanced Topics in Software Verification

$\lambda \rightarrow$  and HOL

Gerwin Klein, June Andronick, Ramana Kumar  
S2/2016

[data61.csiro.au](http://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 [1]
  
- Foundations & Principles
  - Lambda Calculus, natural deduction [1,2]
  - Higher Order Logic [3<sup>a</sup>]
  - 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<sup>b</sup>,9]
  - (mid-semester break)
  - Writing Automated Proof Methods [10]
  - Isar, codegen, typeclasses, locales [11<sup>c</sup>,12]

---

<sup>a</sup>a1 due; <sup>b</sup>a2 due; <sup>c</sup>a3 due



DATA  
61



# Preview: Proofs in Isabelle

# Proofs in Isabelle



## General schema:

```
lemma name: " <goal> "  
apply <method>  
apply <method>  
...  
done
```

- Sequential application of methods until all **subgoals** are solved.

# The Proof State



1.  $\bigwedge x_1 \dots x_p. \llbracket A_1; \dots; A_n \rrbracket \implies B$
2.  $\bigwedge y_1 \dots y_q. \llbracket C_1; \dots; C_m \rrbracket \implies D$

$x_1 \dots x_p$	Parameters
$A_1 \dots A_n$	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*
- *ImpTh*<sub>*i*</sub>: name of *imported* theories. Import transitive.

Unless you need something special:

```
theory MyTh imports Main begin ... end
```

# Natural Deduction Rules



$$\frac{A \quad B}{A \wedge B} \text{ conjI}$$

$$\frac{A \wedge B \quad [[A; B]] \Rightarrow C}{C} \text{ conjE}$$

$$\frac{A}{A \vee B} \quad \frac{B}{A \vee B} \text{ disjI1/2}$$

$$\frac{A \vee B \quad A \Rightarrow C \quad B \Rightarrow C}{C} \text{ disjE}$$

$$\frac{A \Rightarrow B}{A \rightarrow B} \text{ implI}$$

$$\frac{A \rightarrow B \quad A \quad B \Rightarrow C}{C} \text{ impE}$$

For each connective ( $\wedge$ ,  $\vee$ , etc):  
**introduction** and **elimination** rules



# Proof by assumption



**apply** assumption

proves

$$1. \llbracket B_1; \dots; B_m \rrbracket \implies C$$

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



**Intro** rules decompose formulae to the right of  $\implies$ .

**apply** (rule  $\langle$ intro-rule $\rangle$ )

Intro rule  $\llbracket A_1; \dots; A_n \rrbracket \implies A$  means

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

Applying rule  $\llbracket A_1; \dots; A_n \rrbracket \implies A$  to subgoal  $C$ :

→ unify  $A$  and  $C$

→ replace  $C$  with  $n$  new subgoals  $A_1 \dots A_n$

# Elim rules



**Elim** rules decompose formulae on the left of  $\implies$ .

**apply** (erule <elim-rule>)

Elim rule  $\llbracket A_1; \dots; A_n \rrbracket \implies A$  means

→ If I know  $A_1$  and want to prove  $A$  it suffices to show  $A_2 \dots A_n$

Applying rule  $\llbracket A_1; \dots; A_n \rrbracket \implies A$  to subgoal  $C$ :

Like **rule** but also

- unifies first premise of rule with an assumption
- eliminates that assumption

A background pattern of white hexagons on a dark teal background, arranged in a staggered grid.

DATA  
61



# Demo

# More Proof Rules

# Iff, Negation, True and False



$$\frac{A \implies B \quad B \implies A}{A = B} \text{ iffI} \qquad \frac{A = B \quad \llbracket A \longrightarrow B; B \longrightarrow A \rrbracket \implies C}{C} \text{ iffE}$$

$$\frac{A = B}{A \implies B} \text{ iffD1}$$

$$\frac{A = B}{B \implies A} \text{ iffD2}$$

$$\frac{A \implies \text{False}}{\neg A} \text{ notI}$$

$$\frac{\neg A \quad A}{P} \text{ notE}$$

$$\frac{}{\text{True}} \text{ TrueI}$$

$$\frac{\text{False}}{P} \text{ FalseE}$$

# Equality



$$\frac{}{t = t} \text{ refl} \quad \frac{s = t}{t = s} \text{ sym} \quad \frac{r = s \quad s = t}{r = t} \text{ trans}$$

$$\frac{s = t \quad P \ s}{P \ t} \text{ subst}$$

Rarely needed explicitly — used implicitly by term rewriting

# Classical



$$\overline{P = True \vee P = False} \text{ True-or-False}$$

$$\overline{P \vee \neg P} \text{ excluded-middle}$$

$$\frac{\neg A \implies False}{A} \text{ ccontr} \quad \frac{\neg A \implies A}{A} \text{ classical}$$

- **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”**



$\overline{P \vee \neg P}$  excluded-middle

is a case distinction on type *bool*

Isabelle can do case distinctions on arbitrary terms:

**apply** (case\_tac *term*)

# Safe and not so safe



Safe rules preserve provability

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

$$\frac{A \quad B}{A \wedge B} \text{ conjI}$$

Unsafe rules can turn a provable goal into an unprovable one

disjI1, disjI2, impE, iffD1, iffD2, notE

$$\frac{A}{A \vee B} \text{ disjI1}$$

**Apply safe rules before unsafe ones**

A background pattern of white hexagons on a dark teal background, arranged in a staggered grid.

DATA  
61



# Demo

# What we have learned so far...



- 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*