

COMP 4161 NICTA Advanced Course

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Toby Murray, Christine Rizkallah



Last time...



- → Simply typed lambda calculus: λ^{\rightarrow}
- → Typing rules for λ^{\rightarrow} , type variables, type contexts
- → β -reduction in λ^{\rightarrow} satisfies subject reduction
- → β -reduction in λ^{\rightarrow} always terminates
- ➔ Types and terms in Isabelle

Content

	NICTA
Intro & motivation, getting started	[1]
 Foundations & Principles Lambda Calculus, natural deduction Higher Order Logic Term rewriting 	[1,2] [3ª] [4]
 Proof & Specification Techniques Inductively defined sets, rule induction Datatypes, recursion, induction Hoare logic, proofs about programs, C verification (mid-semester break) Writing Automated Proof Methods Isar, codegen, typeclasses, locales 	[5] [6, 7] [8 ^b ,9] [10] [11 ^c ,12]
	 Higher Order Logic Term rewriting Proof & Specification Techniques Inductively defined sets, rule induction Datatypes, recursion, induction Hoare logic, proofs about programs, C verification (mid-semester break) Writing Automated Proof Methods

^aa1 due; ^ba2 due; ^ca3 due

0



PREVIEW: 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.
$$\bigwedge x_1 \dots x_p \cdot \llbracket A_1; \dots; A_n \rrbracket \Longrightarrow B$$

2. $\bigwedge y_1 \dots y_q \cdot \llbracket C_1; \dots; C_m \rrbracket \Longrightarrow D$

$x_1 \dots x_p$	Parameters
$A_1 \dots A_n$	Local assumptions
В	Actual (sub)goal

Isabelle Theories



Syntax:

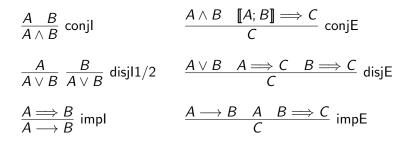
```
theory MyTh
imports ImpTh_1 \dots ImpTh_n
begin
(declarations, definitions, theorems, proofs, ...)*
end
```

- → *MyTh*: name of theory. Must live in file *MyTh*.thy
- → $ImpTh_i$: 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



apply assumption

proves

1. $\llbracket B_1; \ldots; B_m \rrbracket \Longrightarrow C$

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 \Longrightarrow .

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

Intro rule $\llbracket A_1; \ldots; A_n \rrbracket \Longrightarrow A$ means

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

Applying rule $\llbracket A_1; \ldots; A_n \rrbracket \Longrightarrow A$ to subgoal C:

- \rightarrow 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 \Longrightarrow .

apply (erule <elim-rule>)

Elim rule $\llbracket A_1; \ldots; A_n \rrbracket \Longrightarrow A$ means

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

Applying rule $\llbracket A_1; \ldots; A_n \rrbracket \Longrightarrow A$ to subgoal *C*: Like **rule** but also

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



Dемо



MORE PROOF RULES

Iff, Negation, True and False



$$\begin{array}{ccc} \underline{A \Longrightarrow B & \underline{B \Longrightarrow A}} & \mathrm{iffl} & & \underline{A = B & \llbracket A \longrightarrow B; B \longrightarrow A \rrbracket \Longrightarrow C} \\ \underline{A = B} & \mathrm{iffD1} & & \underline{A = B} \\ \underline{A \Longrightarrow B} & \mathrm{iffD1} & & \underline{B \Longrightarrow A} & \mathrm{iffD2} \\ \\ \underline{A \Longrightarrow False} \\ \neg A & \mathrm{notl} & & \underline{\neg A & A} \\ \hline \hline True & \mathrm{Truel} & & \frac{False}{P} & \mathrm{FalseE} \end{array}$$

Equality



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

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

Rarely needed explicitly — used implicitly by term rewriting

Classical



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

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

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

- → 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"



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

is a case distinction on type bool

Isabelle can do case distinctions on arbitrary terms:

apply (case_tac term)



Safe rules preserve provability

conjl, impl, notl, iffi, refl, ccontr, classical, conjE, disjE

$$\frac{A \quad B}{A \land B}$$
 conjl

Unsafe rules can turn a provable goal into an unprovable one

disjl1, disjl2, impE, iffD1, iffD2, notE

$$\frac{A}{A \lor B} \text{ disjl}1$$

Apply safe rules before unsafe ones



Dемо



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