

Topics in Automated Deduction (CS 576)

Elsa L. Gunter
2112 Siebel Center
egunter@illinois.edu
<http://www.cs.illinois.edu/class/sp10/cs576/>

1

Safe and Unsafe Rules

Safe rules preserve provability:

conjI, impI, notI, iffI, refl, ccontr, classical, conje, disjE

Unsafe rules can reduce a provable goal to one that is not:

disjI1, disjI2, impE, iffD1, iffD2, notE

Try safe rules before unsafe ones

2

\Longrightarrow vs \longrightarrow

- Theorems usually more useful written as $\llbracket A_1; \dots; A_n \rrbracket \Longrightarrow A$ instead of $A_1 \wedge \dots \wedge A_n \longrightarrow A$ (easier to apply)
- Exception:** (in `apply`-style): induction variable must not occur in premises
- Example: For induction on x , transform:
 $\llbracket A; B(x) \rrbracket \Longrightarrow C(x) \rightsquigarrow A \Longrightarrow B(x) \longrightarrow C(x)$
Reverse transformation (after proof):
`lemma abc [rule_format]: A \Longrightarrow B(x) \longrightarrow C(x)`

3

Parameters

Subgoal:

1. $\Lambda x_1 \dots x_n. Formula$

The x_i are called *parameters* of the subgoal

Intuition: local constants, i.e. arbitrary fixed values

Rules are automatically lifted passed $\Lambda x_1 \dots x_n$ and applied directly to *Formula*

4

Scope

- Scope of parameters: whole subgoal
- Scope of \forall , \exists , \dots : ends with $;$ or \Longrightarrow , or enclosing $)$

$\Lambda xy. \llbracket \forall y. P y \longrightarrow Q z y; Q x y \rrbracket \Longrightarrow \exists x. Q x y$
means

$\Lambda xy. \llbracket (\forall y_1. P y_1 \longrightarrow Q z y_1); Q x y \rrbracket \Longrightarrow \exists x_1. Q x_1 y$

5

α -Conversion and Scope of Variables

- $\forall x. P x$: x can appear in P
Example: $\forall x. x = x$ is obtained by $P \mapsto \lambda u. u = u$
- $\forall x. P$: x cannot appear in P
Example: $P \mapsto x = x$ yields $\forall x'. x = x$

Bound variables are renamed automatically to avoid name clashes with other variables.

6

Natural Deduction Rules for Quantifiers

$$\frac{\Lambda x. P x}{\forall x. P x} \text{allI} \quad \frac{\forall x. P x \quad P ?x \implies R}{R} \text{allE}$$

$$\frac{P ?x}{\exists x. P x} \text{exI} \quad \frac{\exists x. P x \quad \Lambda x. P x \implies R}{R} \text{exE}$$

- `allI` and `exE` introduce new parameters (Λx)
- `allE` and `exI` introduce new unknowns ($?x$)

7

Safe and Unsafe Rules

Safe: `allI`, `exE`

Unsafe: `allE`, `exI`

Create parameters first, unknowns later

8

Instantiating Variables in Rules

`apply (rule_tac x = "term" in rule)`

Like `rule`, but $?x$ in `rule` is instantiated with `term` before application.

$?x$ must be schematic variable occurring in statement of `rule`.

Similar: `erule_tac`

! x is in rule, not in goal !

9

Two Successful Proofs

1. $\forall x. \exists y. x = y$
`apply (rule allI)`
 1. $\Lambda x. \exists y. x = y$

Better practice:

`apply(rule_tac x = "x" in exI)`
 1. $\Lambda x. x = x$
`apply (rule refl)`

Exploration:

`apply (rule exI)`
 1. $\Lambda x. x = ?y x$
`apply (rule refl)`
 $?y \mapsto \lambda u. u$

simpler & cleaner

shorter & trickier

10

Two Unsuccessful Proof Attempts

```

1.  $\exists y. \forall x. x = y$ 
apply(rule_tac
  x = ??? in exI)
  1.  $\forall x. x = ?y$ 
apply(rule allI)
  1.  $\Lambda x. x = ?y$ 
apply(rule refl)
 $?y \mapsto x$  yields  $\Lambda x'. x' = x$ 
  
```

Principles: $?f x_1 \dots x_n$ can only be replaced by term t if $params(t) \subseteq \{x_1, \dots, x_n\}$

11

Parameter Names

Parameter names are chosen by Isabelle

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

`apply(rule allI)`

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

`apply(rule_tac x = "x" in exI)`

Works, but is brittle!!

12

Renaming Parameters

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

`apply(rule allI)`

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

`apply(rename_tac xxx)`

1. $\lambda xxx. \exists y. xxx = y$

`apply(rule_tac x = "xxx" in exI)`

In general: `(rename_tac $x_1 \dots x_n$)` renames the right-most (inner) n parameters to $x_1 \dots x_n$

13

Forward Proofs: `frule` and `drule`

"Forward" rule: $A_1 \implies A$

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

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

New subgoal: 1. $\sigma(\llbracket B_1; \dots; B_n; A \rrbracket) \implies C$

Command:

`apply(frule < rulename >)`

Like `frule` but also deletes B_i :

`apply(drule < rulename >)`

14

`frule` and `drule`: The General Case

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

Creates additional subgoals:

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

⋮

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

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

15

Forward Proofs: `OF`

`r [OF $r_1 \dots r_n$]`

Prove assumption 1 of theorem `r` with theorem `r1`, and assumption 2 with theorem `r2`, etc.

Rule `r` $\llbracket A_1; \dots; A_m \rrbracket \implies A$

Rule `r1` $\llbracket B_1; \dots; B_n \rrbracket \implies B$

Substitution $\sigma(B) \equiv \sigma(A_1)$

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

16