Topics in Automated Deduction (CS 576)

Elsa L. Gunter

2112 Siebel Center

egunter@illinois.edu

http://www.cs.illinois.edu/class/sp10/cs576/

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

$\Longrightarrow$ vs $\longrightarrow$

- Theorems usually more useful written as

  $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$

  instead of $A_1 \wedge \ldots \wedge A_n \longrightarrow A$ (easier to apply)

- **Exception:** (in `apply`-style): induction variable must not occur in premises

- Example: For induction on x, transform:

  $[\![A; B(x)]\!] \Longrightarrow C(x) \rightsquigarrow A \Longrightarrow B(x) \longrightarrow C(x)$

  Reverse transformation (after proof):

  `lemma abc [rule_format]:` $A \Longrightarrow B(x) \longrightarrow C(x)$

Parameters

Subgoal:

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

The x_i are called *parameters* of the subgoal

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

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

Scope

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

$$\bigwedge xy.\ [\![\forall y.\ P\ y\ \longrightarrow Q\ z\ y;\ Q\ x\ y]\!] \Longrightarrow \exists x.\ Q\ x\ y$$

means

$$\bigwedge xy.\ [\![(\forall y_1.\ P\ y_1\ \longrightarrow Q\ z\ y_1);\ Q\ x\ y]\!] \Longrightarrow \exists x_1.\ Q\ x_1\ y$$

α-Conversion and Scope of Variables

- $\forall \mathtt{x}.\ \mathtt{P}\ \mathtt{x}$: $\mathtt{x}$ can appear in $\mathtt{P}\ \mathtt{x}$.

  Example: $\forall \mathtt{x}.\ \mathtt{x} = \mathtt{x}$ is obtained by $\mathtt{P} \mapsto \lambda \mathtt{u}.\ \mathtt{u} = \mathtt{u}$

- $\forall \mathtt{x}.\ \mathtt{P}$: $\mathtt{x}$ cannot appear in $\mathtt{P}$

  Example: $\mathtt{P} \mapsto \mathtt{x} = \mathtt{x}$ yields $\forall \mathtt{x}'.\ \mathtt{x} = \mathtt{x}$

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

Natural Deduction Rules for Quantifiers

$$\frac{\bigwedge x.\ P\ x}{\forall x.P\ x}\ \texttt{allI} \qquad \frac{\forall x.\ P\ x \qquad P\ ?x \Longrightarrow R}{R}\ \texttt{allE}$$

$$\frac{P\ ?x}{\exists x.\ P\ x}\ \texttt{exI} \qquad \frac{\exists x.\ P\ x \qquad \bigwedge x.\ P\ x \Longrightarrow R}{R}\ \texttt{exE}$$

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

Safe and Unsafe Rules

Safe: `allI`, `exE`

Unsafe: `allE`, `exI`

Create parameters first, unknowns later

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 !

Two Successful Proofs

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

```
apply (rule allI)
```

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

Better practice:

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

1. $\bigwedge$x. x = x

```
apply (rule refl)
```

simpler & cleaner

Exploration:

```
apply (rule exI)
```

1. $\bigwedge$x. x =?y x

```
apply (rule refl)
```

?y $\mapsto$ λu. u

shorter & trickier

Two Unsuccessful Proof Attempts

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

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

Parameter Names

Parameter names are chosen by Isabelle

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

```
apply(rule allI)
```

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

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

Works, but is brittle!!

Renaming Parameters

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

```
apply(rule allI)
```

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

```
apply(rename_tac xxx)
```

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

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

In general: `(rename_tac` $x_1 \ldots x_n$`)` renames the rightmost (inner) n parameters to $x_1 \ldots x_n$

Forward Proofs: `frule` and `drule`

"Forward" rule:	$\mathtt{A_1} \Longrightarrow \mathtt{A}$
Subgoal:	$1.\ [\![\mathtt{B_1}; \ldots; \mathtt{B_n}]\!] \Longrightarrow \mathtt{C}$
Substitution:	$\sigma(\mathtt{B}_i) \equiv \sigma(\mathtt{A_1})$
New subgoal:	$1.\ \sigma([\![\mathtt{B_1}; \ldots; \mathtt{B_n}; \mathtt{A}]\!] \Longrightarrow \mathtt{C})$

Command:

`apply(frule` $< rulename >$`)`

Like `frule` but also deletes $\mathtt{B}_i$:

`apply(drule` $< rulename >$`)`

`frule` and `drule`: The General Case

Rule: $[\![\mathtt{A}_1; \ldots; \mathtt{A}_m]\!] \Longrightarrow \mathtt{A}$

Creates additional subgoals:

$$
\begin{array}{l}
1.\ \sigma([\![\mathtt{B}_1; \ldots; \mathtt{B}_n]\!] \Longrightarrow \mathtt{A}_2) \\
\quad \vdots \\
m-1.\ \sigma([\![\mathtt{B}_1; \ldots; \mathtt{B}_n]\!] \Longrightarrow \mathtt{A}_m) \\
m.\ \sigma([\![\mathtt{B}_1; \ldots; \mathtt{B}_n; \mathtt{A}]\!] \Longrightarrow \mathtt{C})
\end{array}
$$

Forward Proofs: OF

$$\texttt{r [OF r}_1 \ldots \texttt{r}_n\texttt{]}$$

Prove assumption 1 of theorem r with theorem $\texttt{r}_1$, and assumption 2 with theorem $\texttt{r}_2$, etc.

Rule r	$[\![\texttt{A}_1; \ldots; \texttt{A}_\texttt{m}]\!] \Longrightarrow \texttt{A}$
Rule $\texttt{r}_1$	$[\![\texttt{B}_1; \ldots; \texttt{B}_\texttt{n}]\!] \Longrightarrow \texttt{B}$
Substitution	$\sigma(\texttt{B}) \equiv \sigma(\texttt{A}_1)$
r [OF $\texttt{r}_1$]	$\sigma([\![\texttt{B}_1; \ldots; \texttt{B}_\texttt{n}; \texttt{A}_2; \ldots; \texttt{A}_\texttt{m}]\!] \Longrightarrow \texttt{A})$