Topics in Automated Deduction (CS 576)

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

Basic Induction Heuristics

- Theorems about recursive functions are proved by induction
- If f defined by induction on ith argument, proof is by induction of ith argument of f

Example: Tail Recursive Reverse

```
primrec itrev ::  'a list ⇒ 'a list ⇒ 'a list where
 itrev [ ]     ys = ys |
 itrev (x#xs)  ys = itrev xs (x#ys)

lemma itrev xs [] = rev xs
```

This direction is easier to prove/use

lhs "more complex" than rhs

Demo: first attempt at itrev = rev

Generalization (first kind)

Replace constant arguments ([ ]) by variables:

```
lemma itrev xs ys = rev xs @ ys
```


Demo: second attempt at itrev = rev

Generalization (second kind)

Quantify all free variables by $\forall$,
except the induction variable

```
lemma ∀ ys.  itrev xs ys = rev xs @ ys
```


Proof Basics

- Isabelle uses *Natural Deduction* proofs
  - Uses *sequent* encoding
- Rule notation:

Rule	Sequent Encoding
$\frac{A_1 \dots A_n}{A}$	$[\![A_1, \dots, A_n]\!] \Longrightarrow A$
$\frac{\begin{array}{c}B \\ \vdots \\ A_1 \dots\ A_i\ \dots A_n\end{array}}{A}$	$[\![A_1, \dots, B \Longrightarrow A_i, \dots, A_n]\!] \Longrightarrow A$

Natural Deduction

For each logical operator $\oplus$, have two kinds of rules:

Introduction: How can I prove $A \oplus B$?

$$\frac{?}{A \oplus B}$$

Elimination: What can I prove using $A \oplus B$?

$$\frac{\dots A \oplus B \dots}{?}$$

Operational Reading

$$\frac{A_1 \dots A_n}{A}$$

Introduction rule:

To prove A it suffices to prove $A_1 \dots A_n$.

Elimination rule:

If we know A_1 and we want to prove A
it suffices to prove $A_2 \dots A_n$

Natural Deduction for Propositional Logic

$$\frac{A \quad B}{A \wedge B}\ \text{conjI} \qquad \frac{A \wedge B \quad [\![A; B]\!] \Longrightarrow C}{C}\ \text{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \text{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{c}\ \text{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \text{impI} \qquad \frac{A \longrightarrow B \quad A \quad B \Longrightarrow C}{C}\ \text{impE}$$

Natural Deduction for Propositional Logic

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \text{iffI} \qquad \frac{A = B \quad A}{B}\ \text{iffD1}$$

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

$$\frac{A \Longrightarrow \text{False}}{\neg A}\ \text{notI} \qquad \frac{\neg A \quad A}{B}\ \text{notE}$$

Equality

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

$$\frac{s = t \quad A(s)}{A(t)}\ \texttt{subst}$$

`subst` rarely needed explicitly – used implicitly by `simp`

More Rules

$$\frac{A \wedge B}{A}\ \texttt{conjunct1} \qquad \frac{A \wedge B}{B}\ \texttt{conjunct2}$$

$$\frac{A \longrightarrow B \quad A}{B}\ \texttt{mp}$$

Compare to elimination rules:

$$\frac{A \wedge B \quad [\![A; B]\!] \Longrightarrow C}{C}\ \texttt{conjE} \qquad \frac{A \longrightarrow B \quad A \; B \Longrightarrow C}{C}\ \texttt{impE}$$

"Classical" Rules

$$\frac{\not A \Longrightarrow \texttt{False}}{A}\ \texttt{ccontr} \qquad \frac{\not A \Longrightarrow A}{A}\ \texttt{classocal}$$

- `ccontr` and `classical` are not derivable from the Natural Deduction rules.
- They make the logic *"classical"*, i.e. *"non-constructive* or *"non-intuitionistic"*.

Proof by Assumption

$$\frac{A_1 \ldots A_i \ldots A_n}{A_i}$$

Rule Application: The Rough Idea

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

- Unify A and C
- Replace C with n new subgoals: $A'_1 \ \ldots \ A'_n$

Backwards reduction, like in Prolog

Example: rule: $[\![?P; ?Q]\!] \Longrightarrow ?P \wedge ?Q$
subgoal: `1. A ∧ B`

Result: `1. A`
`2. B`

Rule Application: More Complete Idea

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

- Unify A and C with (meta)-substitution σ
- Specialize goal to $\sigma(C)$
- Replace C with n new subgoals: $\sigma(A_1) \ \ldots \ \sigma(A_n)$

Note: schematic variables in C treated as existential variables

Does there exist value for $?X$ in C that makes C true?

(Still not the whole story)

`rule` Application

Rule:	$[\![A_1; \ldots; A_n]\!] \Longrightarrow A$
Subgoal:	1. $[\![B_1; \ldots; B_m]\!] \Longrightarrow C$
Substitution:	$\sigma(A) \equiv \sigma(C)$
New subgoals:	1. $[\![\sigma(B_1); \ldots; \sigma(B_m)]\!] \Longrightarrow \sigma(A_1)$ ⋮ n. $[\![\sigma(B_1); \ldots; \sigma(B_m)]\!] \Longrightarrow \sigma(A_n)$
Proves:	$[\![\sigma(B_1); \ldots; \sigma(B_m)]\!] \Longrightarrow \sigma(C)$
Command:	`apply (rule` <*rulename*>`)`

Proof by `assumption`

`apply assumption`

proves:

1. $[\![B_1; \ldots; B_m]\!] \Longrightarrow C$

by unifying C with one of the B_i

Demo: Application of Introduction Rule

Applying Elimination Rules

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

Like `rule` but also

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

Example

Rule:	$[\![$`?P`$\wedge$`?Q`$; [\![$`?P`$; $`?Q`$]\!] \Longrightarrow$`?R`$]\!] \Longrightarrow$`?R`
Subgoal:	1. $[\![$`X`$; $`A`$ \wedge $`B`$; $`Y`$]\!] \Longrightarrow$ `Z`
Unification:	$?P \wedge ?Q \equiv A \wedge B$ and $?R \equiv Z$
New subgoal:	1. $[\![$`X`$; $`Y`$]\!] \Longrightarrow [\![$`A`$; $`B`$]\!] \Longrightarrow$ `Z`
Same as:	1.$[\![$`X`$; $`Y`$; $`A`$; $`B`$]\!] \Longrightarrow$ `Z`