Getting Started With Isabelle

Lecture III: Interactive Proof

Lawrence C. Paulson
Computer Laboratory
Lecture Outline

- Syntax of rules
- Proof states; subgoals
- Specialist tactics
- Primitive tactics
- Automatic tactics
- Simplification tactics
- The tableau prover (classical reasoner)
Expressing Inference Rules in Isabelle

\[ P \land Q \implies P \]

premises               conclusion

\[ [\mid P \rightarrow Q; P \mid ] \implies Q \]

general premise       conclusion with HOL quantifier

\[ (\forall x. P x) \implies \forall x. P x \]

\[ \implies \text{ and } \forall \text{ belong to the logical framework} \]
An Isabelle Proof State

Goal "(i * j) * k = i * ((j * k)::nat)";
by (induct_tac "i" 1);

Level 1 (2 subgoals)
i * j * k = i * (j * k)
1. 0 * j * k = 0 * (j * k)
2. !!n. n * j * k = n * (j * k)
    ==> Suc n * j * k = Suc n * (j * k)

Subgoal 1 is the base case.
Subgoal 2 is the inductive step.

- The !!n names a natural number
- The ==> separates the hypothesis and conclusion
The Form of a Subgoal

Each subgoal of a proof state looks like this:

$$!!x_1 \ldots x_k. [ | \phi_1 ; \ldots ; \phi_n | ] \implies \phi$$

- **Parameters** stand for arbitrary values
- **Assumptions** are typical of Natural Deduction

$$[ | \phi_1 ; \phi_2 | ] \implies \psi$$ is the same as

$$\phi_1 \implies (\phi_2 \implies \psi)$$
**Specialist Tactics**

- `induct_tac "x" i` induction over a datatype value $x$
- `case_tac "P" i` case analysis on property $P$
- `subgoal_tac "P" i` introduce $P$ as a lemma
- `Clarify_tac i` perform all obvious steps

Replace subgoal $i$ by new subgoals
May add new assumptions & parameters
Primitive Tactics: Single-Step Proof

Apply to subgoal $i$ the rule

$$
\begin{array}{c}
\phi_1 & \ldots & \phi_n \\
\hline
\psi
\end{array}
$$

$\texttt{rtac rule i}$ replace goal $\psi$ by subgoals $\phi_1, \ldots, \phi_n$
—backward proof

$\texttt{dtac rule i}$ replace assumption $\phi_1$ by $\psi$
—new subgoals $\phi_2, \ldots, \phi_n$
—forward proof

$\texttt{etac rule i}$ apply an elimination rule —
new subgoals $\phi_2, \ldots, \phi_n$
"Try Everything" Tactics

**Auto_tac**  
break up & try to prove all subgoals  
— may leave many subgoals

**Force_tac i**  
prove subgoal $i$ using everything —  
or give up

These call the simplifier and the classical reasoner.
**Simplification Tactics**

\[
\text{Simp\_tac } i \quad \text{simplify conclusion}
\]

\[
\text{Asm\_simp\_tac } i \quad \ldots \text{ using assumptions as extra rewrite rules}
\]

\[
\text{Full\_simp\_tac } i \quad \text{simplify assumptions and conclusion}
\]

\[
\text{Asm\_full\_simp\_tac } i \quad \ldots \text{ using assumptions as extra rewrite rules}
\]

These apply rewrite rules and specialized proof procedures to subgoal \(i\).
Using Your Own Simplification Rules

Add them **globally**:

\[ \text{Addsimp} \text{ [my\_thm];} \]

Or add them **locally**:

\[ \text{by (simp_tac (simpset() addsimps \{my\_thm\}) 2);} \]

\[ ! \]

\[ ! \text{ note lower case!} \]

- Try **conditional** rules like \( m < n \implies m \mod n = m \).
- To sort, use **permutative** rules like \( m \times n = n \times m \).
Using the Tableau Prover

\texttt{Blast_tac} \ i \ \ \text{search for a proof of subgoal} \ i

Some rules that work with \texttt{Blast_tac}:

\[
| x \leq y; \ y \leq x | \implies x = y
\]

\text{Introduction rule: backward proof}

\[
\{x\} = \{y\} \implies x = y
\]

\text{Destruction rule: forward proof}

\[
| P \mid Q; \ P \implies R; \ Q \implies R | \implies R
\]

\text{Elimination rule}
Using Your Own Tableau Rules

Easy way: prove an equivalence like \texttt{finite_Un}:

\[
\text{finite} \ (A \cup B) = (\text{finite} \ A \land \text{finite} \ B)
\]

Then install it—to simplifier also—by

\texttt{AddIffs [finite_Un];}

Or add them locally:

\texttt{by (blast_tac (claset() addIs intro_rules addDs destruction_rules addEs elim_rules) 2);}

Rules are used to break down formulas
Finding Theorems in the Library

thms_containing ["map", "rev"];

[("List.rev_concat",
"rev (concat ?xs) = concat (map rev (rev ?xs))"),
("List.rev_map",
"rev (map ?f ?xs) = map ?f (rev ?xs)")]
: (string * thm) list

Result is a list of names and theorems — as ML values.
An infix has a declared name or the default op-form. See theory file!