Overview

1. Formulas, sequents, and rules revisited
2. Application of rules
3. Fundamental methods of Isabelle/HOL
4. Logical rules and theory Main
5. Rewriting and simplification
6. Case analysis and structural induction
7. Proof automation
8. More proof methods

» slides of Sessions 2, 3.1, 3.2, and 4 & 5 by T. Nipkow
» Chapter 5 of Isabelle/HOL Tutorial til page 99
4. A Proof System for Higher-Order Logic 4.1 Methods and Rules

Formulas, sequents, and rules revisited

We need to represent:

- formulas, generalized sequents: lemmas/theorems to be proven
- rules: to be applied in a proof step
- proof (sub-)goals, i.e., open leaves in a proof tree

Examples: from Lecture.thy

- SPEC, SCHEMATIC (not allowed)
- ARULE
- GOAL

A proven lemma/theorem is automatically transformed into a rule. That is, the set of rules is not fixed in Isabelle/HOL (e.g. ARULE).

Variables

Six kinds of variables:

- (logical) variables bound by the logic-quantifiers
- (logical) variables bound by the meta-quantifier
- free (logical) variables
- schematic variables (in rules and proofs)
- type variables
- schematic type variables

Format of goals

\[ \forall x_1 \ldots x_k. \llbracket A_1; \ldots; A_m \rrbracket \Rightarrow C \]

- \( x_i \) are variables local to the subgoal (possibly none)
- \( A_i \) are called the assumptions (possibly none)
- \( C \) is called the conclusion
- usually no schematic variables

Format of rules

\[ \llbracket P_1; \ldots; P_n \rrbracket \Rightarrow Q \]

- \( P_i \) are called the premises (possibly none)
- \( P_1 \) is called the major premise
- \( Q \) is called the consequent (not standard)
- Schematic variables in \( P_i, Q \).
Proofs and methods

A proof state is characterized by the list of open subgoals:
- at the beginning: proof goal
- during the proof: not yet proven subgoals
- at the end: empty

Methods

Methods are commands working on the proof state. In particular, they allow to apply rules and to do simplification.
- Isabelle/HOL provides a fixed set of basic methods.
- New methods can only be defined based on the basic methods.
- Set of rules is not fixed, i.e., new rules can be derived.

Method “rule”

Rule application
The application of rules is based on unification:
- Unification is done w.r.t. the schematic variables.
- The unifier is applied to the complete proof state!
- Unification may involve renaming of bound variables.

Example
Applying rule \( \text{\texttt{[P_1; P_2]} \implies Q} \) with method \text{\texttt{rule}} to subgoal \( \text{\texttt{A \implies C}} \):
- If \( \sigma \) unifies \( C \) and \( Q \), then replace subgoal by two new subgoals:
  - \( \sigma(A) \implies \sigma(P_1) \)
  - \( \sigma(A) \implies \sigma(P_2) \)
Method “frule”

apply (frule <rule_name>)

- let $R \equiv \left[ P_1; \ldots; P_n \right] \implies Q$
- let $\land x_1 \ldots x_k. \left[ A_1; \ldots; A_m \right] \implies C$ be the current subgoal
- apply (frule $R$) unifies $P_1$ with some $A_j$;
  fails if no $A_j$ and unifier can be found; otherwise unifier $\sigma$
- new subgoals: For $i = 2, \ldots, n$:
  \[
  \land x_1 \ldots x_k. \sigma(\left[ A_1; \ldots; A_m \right] \setminus \left\{ A_j \right\}) \implies P_i
  \]
- helpful for applying destruction rules
- Example C1

Method “erule”

apply (erule <rule_name>)

- let $R \equiv \left[ P_1; \ldots; P_n \right] \implies Q$
- helpful for applying elimination rules
- Example GOAL

Method “drule”

apply (drule <rule_name>)

- let $R \equiv \left[ P_1; \ldots; P_n \right] \implies Q$
- apply (drule $R$) unifies $P_1$ with some $A_j$;
  fails if no $A_j$ and unifier can be found; otherwise unifier $\sigma$
- new subgoals: For $i = 2, \ldots, n$:
  \[
  \land x_1 \ldots x_k. \sigma(\left[ A_1; \ldots; A_m \right] \setminus \left\{ A_j \right\}) \implies P_i
  \]
  
  \[
  \land x_1 \ldots x_k. \sigma(\left[ A_1; \ldots; A_m \right] \setminus \left\{ A_j \right\}; \left\{ Q \right\} \implies C
  \]

  - helpful for applying destruction rules
  - Example C1

Method “rule_tac”-versions

apply ([edf]rule_tac x =<term> in <rule_name>)

similar to [edf]rule, but allow to refine unification

- Example: Isabelle/HOL Tutorial, 5.8.2, p. 79, TAC
- FIXAX2
Method “assumption”

apply (assumption)

- let $\forall x_1 \ldots x_k. \Pi A_1 \ldots A_m \Rightarrow C$ be the current subgoal
- apply (assumption) unifies $C$ with some $A_j$; fails if no $A_j$ and unifier can be found; otherwise:
  - subgoal is closed, i.e., eliminated from proof state.
  - Example GOAL

Methods “induct”, “unfold”

apply (induct \[ _tac \] ... the definition of a function
- generates the corresponding subgoals

apply (unfold \(<name_def>\))
- unfolds the definition of a constant in all subgoals
- Example SPEC

Logical rules of Isabelle/HOL

The logical rules are defined in theory Main (see IsabelleHOLMain, Sect. 2.2)

Remark
Distinguish between safe and unsafe rules:
- Safe rules preserve provability:
  e.g. conjI, impl, notI, iffI, refl, ccontr, classical, conjE, disjE
- Unsafe rules can turn a provable goal into an unprovable one:
  e.g. disjI1, disjI2, impE, iffD1, iffD2, notE
- $\neg$ Apply safe rules before unsafe ones

Applying logical rules

Example
- lemma UNSAFE: “$A \lor \neg A$”
- apply (rule disjI1)
- sorry

Remark
Working with theory Main is similar to programming with large libraries:
- The programmer cannot know the complete library
- The “verificator” cannot know all rules.
Support for finding rules is important in practice.