



Interactive Theorem Proving using Isabelle/HOL

Session 2

René Thiemann

Department of Computer Science

Outline

- The Pure Framework
- Structured Proofs

The Pure Framework

The Minimal Logic Isabelle/Pure

Pure = Generic Natural Deduction Framework

Pure Terms

- inference rules
- logical propositions

Deduction

higher-order resolution (that is, resolution using higher-order unification)

The type prop

- Isabelle/Pure contains a type of propositions: `prop`
- let $\varphi :: \text{prop}$ and $\psi :: \text{prop}$, then
 - $\varphi \implies \psi :: \text{prop}$ (meta-)implication
 - $\bigwedge x. \varphi :: \text{prop}$ (meta-)quantification
- in Isabelle/HOL, every HOL-formula ($t :: \text{bool}$) is also of type `prop`

Isabelle Symbols

symbol	internal	auto completion	abbreviation
\implies	<code>\<Longrightarrow></code>	<code>= = ></code>	<code>. ></code>
\bigwedge	<code>\<And></code>	<code>! !</code>	

Remarks

- \implies is right-associative
- propositions with multiple assumptions are encoded by **currying**

Natural Deduction via Pure Connectives

- every Pure proposition can be read as **natural deduction rule**
- proposition $P_1 \implies \dots \implies P_n \implies C$ corresponds to rule

$$\frac{P_1 \quad \dots \quad P_n}{C}$$

with **premises** P_1, \dots, P_n and **conclusion** C

- scope of variables (like **eigenvariable condition**) enforced by \bigwedge Demo02.thy
- there is no distinction between inference rules and theorems!

Examples

- $A \implies B \implies A \wedge B$ (conjunction introduction)
- $(A \implies B) \implies A \longrightarrow B$ (implication introduction)
in order to prove $A \longrightarrow B$ it suffices to prove B under the assumption A
- $(\bigwedge y. P y) \implies \forall x. P x$ (all introduction)
in order to prove $\forall x. P x$, fix some variable y and prove $P y$

Schematic Variables

- besides free and bound variables, there are **schematic** variables (dark blue; these have leading ?)
- schematic variables can be **instantiated arbitrarily**
- proven inference rules such as $A \implies B \implies A \wedge B$ in Isabelle are written via schematic variables:

$$?A \implies ?B \implies ?A \wedge ?B \quad (\text{thm conjI})$$

- whenever a proof of a statement is finished, all free variables and outermost \bigwedge -variables in that statement are turned into schematic ones;
example: each of the following two lines result in $?A \implies ?B \implies ?A \wedge ?B$
 - lemma " $A \implies B \implies A \wedge B$ " *<proof>*
 - lemma " $\bigwedge A B. A \implies B \implies A \wedge B$ " *<proof>*
- schematic variables may occur in proof goals, then the user can choose how to instantiate

Apply Single Inference Rule – The rule Method

- remember: each theorem can be seen as inference rule
- assume we have to prove goal with conclusion G
- assume *thm* has shape $P_1 \implies \dots \implies P_n \implies C$
- `proof (rule thm)` tries to unify C with G via unifier σ and replaces G by new subgoals coming from instantiated premises $P_1\sigma, \dots, P_n\sigma$

Example

- consider goal $x < 5 \implies x < 3 \wedge x < 2$
- the command `proof (rule conjI)` (conjI: $?A \implies ?B \implies ?A \wedge ?B$)
 - successfully unifies conclusion $x < 3 \wedge x < 2$ with $?A \wedge ?B$
 - only schematic variables can be instantiated in unification, i.e., here $?A$ and $?B$, but not x
 - unifier: replace $?A$ by $x < 3$ and $?B$ by $x < 2$
 - and replaces the previous goal by two new subgoals
 - $x < 5 \implies x < 3$
 - $x < 5 \implies x < 2$

Another Example

- consider goal $\exists y. 5 < y$
- the command `proof` (rule `exI`) delivers one new subgoal: $5 < ?y$

$(\text{exI: } ?P \ ?x \implies \exists x. ?P \ x)$
 Demo02.thy
- details
 - try higher-order unification of $\exists x. ?P \ x$ and $\exists y. 5 < y$
 - solution: replace `?P` by $\lambda z. 5 < z$
 - reason: after instantiation we get two terms
 - $\exists x. (\lambda z. 5 < z) \ x$
 - $\exists y. 5 < y$
 - these two terms are equivalent modulo $\alpha\beta\eta$
 - the unused schematic variable `?x` is renamed to `?y` since the goal used the name `y` in the existential quantor
 - the new subgoal is $(\lambda z. 5 < z) \ ?y$ which is equal to $5 < ?y$ modulo $\alpha\beta\eta$
- **higher-order unification** of terms s and t : find σ such that $s\sigma$ and $t\sigma$ are equivalent modulo $\alpha\beta\eta$

Equality in Isabelle

- all terms are normalized w.r.t. $\alpha\beta\eta$
- α -conversion: the names of bound variables are ignored:

example: $\exists x. P\ x$ is the same as $\exists y. P\ y$

- β -reduction

$(\lambda x. t)\ u$ is the same as $t[x/u]$

(here, $t[x/u]$ denotes the term t where x gets replaced by u)

- η -expansion

$t :: ty \Rightarrow ty'$ is the same as $\lambda x. t\ x$

- Demo02.thy

Structured Proofs

Proofs – Outer Syntax

<i>proof</i>	::= sorry by <i>method</i> <i>method</i> ? proof <i>method</i> ? <i>statement</i> * qed <i>method</i> ?	fake proof atomic proof structured proof
<i>statement</i>	::= fix <i>variables</i> (<i>:: type</i>)? assume <i>proposition</i> ⁺ (from <i>fact</i> ⁺)? (have show) <i>proposition</i> <i>proof</i> { <i>statement</i> * }	arbitrary but fixed values local assumptions (intermediate) result raw proof block
<i>proposition</i>	::= (<i>label</i> :)? <i>term</i>	
<i>fact</i>	::= <i>label</i> <i><term></i>	literal fact
<i>method</i>	::= auto fact rule <i>fact</i> - ...	
<i>command</i>	::= lemma <i>proposition</i> <i>proof</i> ...	

Remarks

- *symbol*? denotes **optional** *symbol*; *symbol** denotes **arbitrarily many** occurrences of *symbol*

Demo – Drinker's Paradox

- statement: there is a person p , that if p drinks then everyone drinks
- formal proof is contained in `Demo02.thy` and it will illustrate various elements and variations of a proof w.r.t. the previous slide
- the upcoming slides mainly serve as a written down explanation, if something was not mentioned in the theory file or during the live demonstration

Remarks (cont'd)

- without *method* argument `proof` applies method standard
- idiom for starting structured proof without initial method “`proof -`”
- special label `this` refers to latest fact
- `show` used for statement that shows conclusion of surrounding `proof ... qed`

Some Proof Methods

- `rule fact` – apply single inference rule, namely *fact*
- `standard` – perform a single standard (with respect to current context) inference step
- `- -` do nothing
- `auto` – combines classical reasoning with simplification

Isabelle Symbols – Cartouches

symbol	internal	auto completion	abbreviations
<	<code>\<open></code>		<code>`</code> and <code><</code> <code><</code>
>	<code>\<close></code>		<code>`</code> and <code>></code> <code>></code>

Proving Propositions

- prove “ $\bigwedge x. P\ x$ ” by
`fix x`
`have "P x" <proof>`
- prove “ $A \implies B$ ” by
`assume "A"`
`have "B" <proof>`

Raw Proof Blocks

- the block

```
{
  fix x y
  assume "P x y" "Q x"
  have "R y" <proof>           (* intermediate statement *)
  have "S x" <proof>          (* last statement *)
}
```
- is exported as $P\ ?x\ ?y \implies Q\ ?x \implies S\ ?x$

Further Remarks and Statements

- introduce **arbitrary but fixed** value x by `fix x`
- introduce **assumption** by `assume " ... "`
- indicate proposition to be proved by `have " ... " <proof>`
- local definition of c by `define c where "c = term"`
(definition becomes available as theorem `c_def`)
- local abbreviation of $?c$ by `let ?c = term`
- abbreviation `?thesis` refers to proposition before current `proof-qed`-block
- obtain witness satisfying P by `obtain x where "P x" <proof>`

The rule Method using Current Facts

- on slide 8 it was explained what the rule method does without current facts
 - example Isabelle statement: `have P proof (rule thm)`
 - *thm* should have form of an introduction rule
 - conclusion in *thm* introduces some specific connective, e.g. $\dots \implies ?A \wedge ?B$
- if there are current facts, the behavior is different and it is tried to apply an elimination rule
 - example Isabelle statement: `from Q have P proof (rule thm)`
 - *thm* should have form of an elimination rule
 - major premise in *thm* contains specific connective, e.g., $?A \wedge ?B \implies \dots$, which is then unified with *Q*
 - in detail: given theorem $P_1 \implies \dots \implies P_n \implies C$, unify **major premise** P_1 of rule with first of current facts; unify remaining current facts with remaining premises; add rest of premises correspondingly instantiated as new subgoals

Example

```

...
have "x > 5 ∨ x = 2" ⟨proof⟩
from this have "A x"
proof (rule disjE)
  - ⟨disjE: ?P ∨ ?Q ⇒ (?P ⇒ ?R) ⇒ (?Q ⇒ ?R) ⇒ ?R⟩
  show "x > 5 ⇒ A x" ⟨proof⟩
  show "x = 2 ⇒ A x" ⟨proof⟩
qed

```

The Difference Between `have` and `show`

- `have` is used to state arbitrary intermediate propositions
- `show` is used to discharge a current proof obligation
- `show` might reject a statement if it does not match a proof obligation
 - if assumptions have been used that are not present in proof obligation
 - if the types of variables are too specific or differ

Examples

- `lemma "P x"`
`proof -`
 `assume "Q x"`
 `from this show "P x" (* rejected, because of assumption Q x *)`
- `lemma "∃ x. x < 5"`
`proof (rule exI)`
 `show "(3 :: nat) < 5" (* rejected, since type is too specific *)`

The Difference Between HOL- and Meta-Implication/Quantification

- there are meta-connectives \wedge and \implies
- there are HOL-connectives \forall and \longrightarrow
- usually the meta-connectives are preferable; example:
 - in $A \implies B \implies C \implies D$ we can just assume B
 - in $A \longrightarrow B \longrightarrow C \longrightarrow D$ we first have to apply implication introduction to access B
- the meta-connectives can only be used on the outside, so certain statements require HOL-connectives; example:
 - $\exists x. x > 5 \longrightarrow (\forall y. P\ x\ y)$
(implication and universal quantor appear below existential quantor)
- consequence: most theorems in Isabelle are written using meta-connectives
 - lemma " $P\ x \implies Q \implies R\ x$ " is preferred over
lemma " $\forall x. P\ x \longrightarrow Q \longrightarrow R\ x$ "

Proofs – Outer Syntax, Extended Grammar

<i>proof</i>	::= sorry by <i>method</i> <i>method</i> [?] proof <i>method</i> [?] <i>statement</i> [*] qed <i>method</i> [?]	fake proof atomic proof structured proof
<i>statement</i>	::= fix <i>variables</i> (:: <i>type</i>) [?] assume <i>proposition</i> ⁺ (from <i>fact</i> ⁺) [?] (have show) <i>proposition</i> <i>proof</i> { <i>statement</i> [*] } let ? <i>x</i> = <i>term</i> (from <i>fact</i> ⁺) [?] obtain <i>vars</i> where <i>prop.</i> <i>proof</i>	arbitrary but fixed values local assumptions (intermediate) result raw proof block local abbreviation get witness
<i>proposition</i>	::= (<i>label</i> :) [?] <i>term</i>	
<i>fact</i>	::= <i>label</i> this <i><term></i>	previous proposition literal fact
<i>method</i>	::= auto fact rule <i>fact</i> - ...	
<i>command</i>	::= lemma <i>proposition</i> <i>proof</i> ...	