Isabelle Tutorial:
Basic Proof Techniques

Burkhart Wolff

Université Paris-Sud

What we will talk about Isabelle with:

- Elementary Forward Proofs
- Tactic Proofs ("apply style")
- Proof Contexts and Structured Proof

Introduction to proving with Isabelle/HOL

Foundations of Proofs: „thm"'s revisited

- Instead of:

$$\Gamma \vdash_{\Theta} \phi$$

- we write:

$$\Theta,\Pi,\Gamma \vdash \phi$$

  - where Θ global context
  - and Γ the local context
  - and Π the impending promisses (not discussed here)

Wenzels Kernel Model at a Glance [Damp 09]:

A) The synchronous part I ("LCF Kernel")

$$\frac{\Theta, \Pi, \Gamma \vdash q : B}{\Theta, \Pi, \Gamma - \{p : A\} \vdash (\lambda p : A.\ q) : (A \Longrightarrow B)} \ (\textit{imp-intro})$$

$$\frac{\Theta_1, \Pi_1, \Gamma_1 \vdash p : (A \Longrightarrow B) \quad \Theta_2, \Pi_2, \Gamma_2 \vdash q : A}{\Theta_1 \cup \Theta_2, \Pi_1 \cup \Pi_2, \Gamma_1 \cup \Gamma_2 \vdash p\ q : B} \ (\textit{imp-elim})$$

$$\frac{\Theta, \Pi, \Gamma \vdash p : (\bigwedge x.\ B[x])}{\Theta, \Pi, \Gamma \vdash p\ a : B[a]} \ (\textit{all-elim})$$

$$\frac{\Theta, \Pi, \Gamma \vdash p[x] : B[x] \quad x \notin \mathrm{FV}\,\Gamma}{\Theta, \Pi, \Gamma \vdash (\lambda x.\ p[x]) : (\bigwedge x.\ B[x])} \ (\textit{all-intro})$$

Wenzels Kernel Model at a Glance [Damp 09]:

A) The synchronous part II ("LCF Kernel")

$$\frac{}{\Theta, \Pi, \{p : A\} \vdash p : A} \ (assm)$$

$$\frac{(c : A[?\overline{\alpha}]) \in \Theta}{\Theta, \emptyset, \emptyset \vdash c : A[\overline{\tau}]} \ (axiom)$$

Tactic „Rule Composition Layer“ for Pure [Paulson]

... based on Kernel Rules, an infra-structure for forward reasoning is added

A) ... adds Schema-Variable Instantiation

B) ... adds Higher-order Unification in rule-composition (wrt. Schema Variables)

C) ... adds lifting over assumptions in rule compositions

D) ... adds lifing over parameters

Tactic „Rule Composition Layer" for Pure [Paulson]

A) rule - instantiation:

$$\frac{\Gamma \vdash_\Theta \overline{A} \Longrightarrow B}{\theta\Gamma \vdash_\Theta \theta\overline{A} \Longrightarrow \theta B}$$

ML{* (* based on: *) Thm.instantiate
for both schema variables and
type schema variables ... *}

Tactic „Rule Composition Layer“ for Pure [Paulson]

B) rule- composition:

$$\frac{\Gamma \vdash_\Theta \overline{A} \Longrightarrow B \qquad \Gamma \vdash_\Theta B' \Longrightarrow C}{\theta\Gamma \vdash_\Theta \theta\overline{A} \Longrightarrow \theta C}$$

where (unification condition): $\theta B = \theta B'$

```
ML{* op Drule.COMP;
      (* based on: *) Thm.biresolution  *}
```

Tactic „Rule Composition Layer“ for Pure [Paulson]

C) lifting over assumptions ($\Longrightarrow$-lift):

$$\frac{\Gamma \vdash_\Theta \overline{A} \Longrightarrow B}{\Gamma \vdash_\Theta (\overline{H} \Longrightarrow \overline{A}) \Longrightarrow (\overline{H} \Longrightarrow B)}$$

ML{* (* based on: *) Thm.lift_rule *}

Tactic „Rule Composition Layer" for Pure [Paulson]

D) lifting over parameters (Λ-lift)::

$$\frac{\Gamma \vdash_\Theta \overline{A}\,\overline{a} \Longrightarrow B\,\overline{a}}{\bigwedge \overline{x}.\ \Gamma(\overline{a}\,\overline{x}) \vdash_\Theta \overline{A}(\overline{a}\,\overline{x}) \Longrightarrow C(\overline{a}\,\overline{x})}$$

ML{* (* based on: *) Thm.lift_rule *}

Attributing Theorems (or: small forward-proofs)

- The syntax for <name> of theorems is actually structured: it is possible to instantiate, combine and even simplify on the fly theorems by Isar-"attributes":
  - <thmname>[of _ "<s>" _] (* instantiate *)
  - <thmname>[THEN <thmname>] (* lift COMP *)
  - <thmname>[THEN[n] <thmname>] (* RSN *)
  - <thmname>[OF <thmname>] (* lift rev COMP *)
  - <thmname>[simplified <thmname>]

Simple Proof Commands

- Renaming thm's:

```
lemmas <thmname> = <thmname>
```

  - Recall that <thmname> may have attributes (and thus forward proofs) …
    - … with these four proof-attributes one can in principle do anything you can do in Isabelle proofs …

Demo I / Exo 1

- Renaming thm's:
  - lemma "¬x ≤ (y::int) ⟹ y ≤ x" oops
  - lemma "¬x ≤ (y::int) ⟹ y ≤ x" oops
  - lemma "¬y = (x::int) ⟹ ¬(x≤y)∨¬(y≤x)" oops
  - lemma "y ≤ x ⟹ x=y ∨ (¬x≤y)" oops

  - derive: 3 + 2 = 1 + 6 from 5 = 1 + 6
    and 3 + 2 = 5
  - reduce: Suc x - 1 ≤ 2 * x to x ≤ x + x and
    suitable equalities.

Demo I / Exo 1

- Renaming thm's:
  - find theorems concerning implication in HOL
  - $P1 \Longrightarrow Q1 \Longrightarrow Q \Longrightarrow (P1 \wedge Q1) \wedge Q$
  - $(P1 \Longrightarrow Q1) \Longrightarrow (P1 \longrightarrow Q1) \vee Q$
  - $(P \Longrightarrow True) \Longrightarrow P \longrightarrow P \vee Q \wedge R$
  - derive „$(P \vee \neg P) = True$“ (find it !)
    the fact: $P1 \vee \neg P1$
  - $(A \Longrightarrow B \Longrightarrow A) \Longrightarrow A \longrightarrow B \longrightarrow A$
  - $(A \Longrightarrow B \Longrightarrow True) \Longrightarrow A \longrightarrow B \longrightarrow A$
    (what does „$A \Longrightarrow B \Longrightarrow True$“ mean ?)
  - ... $\Longrightarrow (A \longrightarrow B \longrightarrow C) \longrightarrow (A \longrightarrow B) \longrightarrow A \longrightarrow C$

Simple Proof Commands

- Simple (Backward) Proofs:

```
lemma  <thmname> :
  [<contextelem>+ shows] "<ϕ>"
   <proof>
```

  - where <contextelem> declare elements of a proof context Γ (to be discussed further)
  - where <proof> is just a call of a high-level proof method by(simp), by(auto), by(metis), by(arith) or the discharger sorry (for the moment).

Simple Proof Commands

- Simple (Backward) Proofs:

```
lemma  <thmname> :
  [<contextelem>+ shows] “<phi>”
   <proof>
```

example:

```
lemma m : "conc (Seq a (Seq b Empty)) (Seq c Empty) =
                              Seq a (Seq b (Seq c Empty))"
          by(simp)

schematic_lemma
        m' : "conc (Seq a (Seq b Empty))(Seq c Empty) =
                              ?X"
          by(simp)
```

Demo I / Exo 1

- Renaming thm's:
  - Prove :

    "conc (Seq a (Seq b Empty)) (Seq c Empty) =

    Seq a (Seq b (Seq c Empty))"

    "(reverse (Seq a (Seq b Empty))) = (Seq b (Seq a Empty))"
  - Evaluate:

    "conc (Seq a (Seq b Empty)) (Seq c Empty)"