# Preuves Interactives et Applications

Burkhart Wolff

https://www.lri.fr/~wolff/teach-material/2017-18/M2-CSMR/index.html

Université Paris-Saclay

#### Advanced Structured Proof Techniques in Isar

#### Revisions

- •The Isar language so far
- The declarative "proof" construct
- support for induct and cases

• Core: the proof environment:

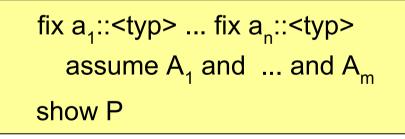
```
proof (<method>)
  [case | fix - assume - let - have -]
  show "<goal>" <proof>
  next
  ...
  next
  [case | fix - assume - let- have-]
  show "<goal>" <proof>
  qed
```

• ... a switch from procedural to declarative style can be done by rephrasing the goals

• Instead of the goal format:

$$\bigwedge a_1 \dots a_n \therefore A_1 \Longrightarrow \dots A_m \Longrightarrow \mathsf{P}$$

#### the "ISAR"-format:



is preferable

• Reason: instead of the required goal:

$$\bigwedge a_1 \dots a_4 \dots A_1 \Longrightarrow \dots A_9 \Longrightarrow \mathsf{P}$$

the "ISAR"-subproof can abstract from irrelevant parameters and assumptions, e.g.:

fix a<sub>3</sub>::<typ> assume A<sub>5</sub> and A<sub>6</sub> show P

thus facilitating automation and readability <sup>10/10/17</sup> in the sub-proof. <sup>B. Wolff - M2 - PIA</sup>

• By the way:

The order of the "offered" sub-lemmas is independent from the order of the "demanded goals".

 The methods induct and cases produce a list of local contexts (shown by the diagnostic command print\_cases) with the appropriate fix'es and assume's

• Example:

```
lemma "reflect(refect t) = t"
proof(induct t) print_cases
  case (leaf x) then show ?case sorry
next
  case (node x1a t1 t2) then show ?case sorry
ged
```

10/10/17

- Structured Proofs are often criticised to be unnecessarily verbose. However:
  - ... you are not forced to use it
  - ... there are many ways to overcome unnecessary verbosity, most notably:
    - abbreviations (via "let")
    - pattern matching (via "is …"

and "where ...")

• in the fix - assume - let - have part (pp. 3) you may write the statement:

let ?<var> = "<big term>"

and abbreviate later on an assumption or conclusion by

#### Sample Proof:

• ... from the HOL/Isar\_Examples – library:

```
theorem sum_of_odds:
 "(\sum i:: nat=0..< n. 2 * i + 1) = n^Suc (Suc 0)"
  (is "?P n" is "?S n = ")
proof (induct n)
 show "?P O" by simp
next
  fix n
 let ?two="Suc(Suc(0))"
  have "?S (n + 1) = ?S n + 2 * n + 1"
   by simp
  also assume "?S n = n^?two"
  also have "... + 2 * n + 1 = (n + 1)^?two"
    by simp
 finally show "?P (Suc n)"
    by simp
qed
```

• in the fix - assume - let - have part (pp. 3) you may write the have statement:

have [<label>:] "<prop>" <proot>

which allows to prove a local conclusion from already stated assumptions (and, thus, a another forward proof element).

• In have statements, the "..." notation may be used to refer to the right-hand-side of the last calculation:

have "<term> = <big term>" <proof>

have "... = <another term>" <proof>

... such that the chaining of calculational proofs can be represented nicely . . . This works also for chains of in-equalities <, <=, =, <, ... . See also: HOL/Isar\_Examples/Group.thy

- The Isar-interpreter possesses over a "one-time-buffer" into which facts and results of calculation can be stored.
  - -then
  - -also
  - -using
  - -... take these intermediate results and chain them into the next command (proof or method).
     Details are method-specific and often obscure.

See Isabelle/Isar Reference, Appendix A "Quick Reference" for more details.

#### A Structured "Classical" Proof

- Nesting Proof-constructs also results in a structured stack of facts that is managed in the "local context".
- The '<prop>' (or <<prop>) notation allows to retrieved such facts from the local context, even if they have not been labelled.

#### A Structured "Classical" Proof

• Example:

```
theorem "((A \longrightarrow B) \longrightarrow A) \longrightarrow A"
proof
assume "(A \longrightarrow B) \longrightarrow A"
show A
proof (rule classical)
assume "¬ A"
have "A \longrightarrow B"
proof
assume A
with <¬ A> show B by contradiction
qed
with <(A \longrightarrow B) \longrightarrow A> show A ...
qed
qed
```

#### Conclusion

- Isabelle offers a structured proof language called Isar
- Besides support for inductions and casedistinctions, it offers:
  - -abbreviations and pattern matching
  - labelling of facts and calculations in the local context as well as a mechanism of explicit retrieval
  - -support for (in-)equational reasoning