Automated Reasoning

Lecture 8: Isar – A Language for Structured Proofs

Jacques Fleuriot
`jdf@inf.ed.ac.uk`

Acknowledgement: Tobias Nipkow kindly provided the slides for this lecture

Apply scripts

- unreadable
- hard to maintain
- do not scale

No structure!

Apply scripts versus Isar proofs

Apply script = assembly language program

Isar proof = structured program with comments

But: **apply** still useful for proof exploration

A typical Isar proof

proof
  assume $formula_0$
  have $formula_1$ **by** *simp*
  $\vdots$
  have $formula_n$ **by** *blast*
  show $formula_{n+1}$ **by** $\ldots$
qed

proves $formula_0 \Longrightarrow formula_{n+1}$

Isar core syntax

proof	=	**proof** [method] step* **qed**
	|	**by** method

method	=	*(simp . . .)* | *(blast . . .)* | *(induction . . .)* | *(rule . . .)* | ...

step	=	**fix** variables	($\bigwedge$)
	|	**assume** prop	($\Longrightarrow$)
	|	[**from** fact$^+$] (**have** | **show**) prop proof	

prop	=	[name:] "formula"

fact	=	name | ...

Example: Cantor's theorem

Informally: The power set of a set is always larger than the set it originated from.

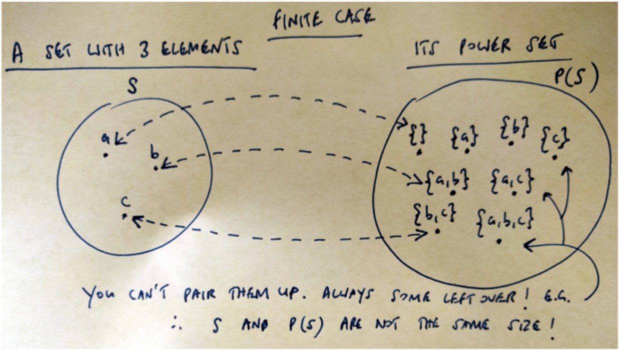

Figure 1: A finite set S and its power-set $\mathcal{P}(S)$. If you can't pair-up elements of sets, with nothing left over, then they cannot be the same size.

Figure from `https://ianwrightsite.files.wordpress.com/2018/10/infinityhandout.pdf`

Example: Cantor's theorem

Informally: The power set of a set is always larger than the set it originated from.

lemma $\neg$ *surj(f :: 'a $\Rightarrow$ 'a set)*

proof default proof: assume *surj*, show *False*

  assume *a: surj f*

  from *a* **have** *b:* $\forall$ *A.* $\exists$ *a. A = f a*

    by*(simp add: surj_def)*

  from *b* **have** *c:* $\exists$ *a.* $\{x.\ x \notin f\ x\} = f\ a$

    by *blast*

  from *c* **show** *False*

    by *blast*

qed

Abbreviations

this	=	the previous proposition proved or assumed
then	=	**from** *this*
thus	=	**then show**
hence	=	**then have**

using and with

(**have**|**show**) prop **using** facts

=

from facts (**have**|**show**) prop

with facts

=

from facts *this*

Structured lemma statement

lemma
fixes *f :: "'a ⇒ 'a set"*
assumes *s: "surj f"*
shows *"False"*
proof - no automatic proof step
have *"∃ a. {x. x ∉ f x} = f a"* **using** *s*
by*(auto simp: surj_def)*
thus *"False"* **by** *blast*
qed

Proves $surj\ f \Longrightarrow False$
but $surj\ f$ *becomes local fact* s *in proof.*

The essence of structured proofs

Assumptions and intermediate facts
can be named and referred to explicitly and selectively

Structured lemma statements

fixes $x :: \tau_1$ **and** $y :: \tau_2$ ...
assumes *a:* P **and** *b:* Q ...
shows R

- **fixes** and **assumes** sections optional
- **shows** optional if no **fixes** and **assumes**

Proof patterns: Case distinction

show *"R"*
proof *cases*
  assume *"P"*
  $\vdots$
  show *"R"* ...
next
  assume *"$\neg$ P"*
  $\vdots$
  show *"R"* ...
qed

have *"P $\vee$ Q"* ...
then show *"R"*
proof
  assume *"P"*
  $\vdots$
  show *"R"* ...
next
  assume *"Q"*
  $\vdots$
  show *"R"* ...
qed

Proof patterns: Contradiction

show *"¬ P"*
proof
  assume *"P"*
  ⋮
  show *"False"* ...
qed

show *"P"*
proof *(rule ccontr)*
  assume *"¬P"*
  ⋮
  show *"False"* ...
qed

Proof patterns: $\longleftrightarrow$

show *"$P \longleftrightarrow Q$"*
proof
  assume *"P"*
  $\vdots$
  show *"Q"* $\ldots$
next
  assume *"Q"*
  $\vdots$
  show *"P"* $\ldots$
qed

Proof patterns: $\forall$ and $\exists$ introduction

show *“$\forall$ x. P(x)”*
proof
  fix *x*   local fixed variable
  show *“P(x)”* ...
qed

show *“$\exists$ x. P(x)”*
proof
  $\vdots$
  show *“P(witness)”* ...
qed

Proof patterns: $\exists$ elimination: obtain

have $\exists x.\ P(x)$
then **obtain** x **where** *p:* $P(x)$ **by** *blast*
$\vdots$ x fixed local variable

Works for one or more x

obtain example

lemma $\neg$ *surj(f :: 'a $\Rightarrow$ 'a set)*
proof
  assume *surj f*
  hence $\exists a.\ \{x.\ x \notin f\,x\} = f\,a$ **by**(*auto simp: surj_def*)
  then obtain *a* **where** $\{x.\ x \notin f\,x\} = f\,a$ **by** *blast*
  hence $a \notin f\,a \longleftrightarrow a \in f\,a$ **by** *blast*
  thus *False* **by** *blast*
qed

Proof patterns: Set equality and subset

show "$A = B$"
proof
  show "$A \subseteq B$" ...
next
  show "$B \subseteq A$" ...
qed

show "$A \subseteq B$"
proof
  fix x
  assume "$x \in A$"
  $\vdots$
  show "$x \in B$" ...
qed

Example: pattern matching

show $\mathit{formula}_1 \longleftrightarrow \mathit{formula}_2$ (**is** $?L \longleftrightarrow ?R$)
proof
  assume $?L$
  $\vdots$
  show $?R$ ...
next
  assume $?R$
  $\vdots$
  show $?L$ ...
qed

?thesis

show *formula* (*is ?thesis*)
proof -
⋮
show *?thesis* ...
qed

Every show implicitly defines *?thesis*

let

Introducing local abbreviations in proofs:

let $?t$ $=$ "some-big-term"

$\vdots$

have "... $?t$ ..."

Quoting facts by value

By name:

have *x0: "x > 0"* ...
⋮
from *x0* ...

By value:

have *"x > 0"* ...
⋮
from *\`x>0\`* ...

↑ ↑

back quotes

Example

lemma
"$(\exists\, ys\ zs.\ xs = ys\ @\ zs \wedge length\ ys = length\ zs) \vee$
$(\exists\, ys\ zs.\ xs = ys\ @\ zs \wedge length\ ys = length\ zs + 1)$"
proof ???

When automation fails

Split proof up into smaller steps.

Or explore by **apply**:

have ... **using** ...	
apply -	to make incoming facts part of proof state. Note the "-"
apply *auto*	or whatever
apply ...	

At the end:

- **done**
- Better: convert to structured proof

moreover—ultimately

have *"P_1"* …
moreover
have *"P_2"* …
moreover
⋮
moreover
have *"P_n"* …
ultimately
have *"P"* …

$\approx$

have lab_1: *"P_1"* …
have lab_2: *"P_2"* …
⋮
have lab_n: *"P_n"* …
from lab_1 lab_2 …
have *"P"* …

With names

Raw proof blocks

$\{$ **fix** $x_1 \ldots x_n$
assume $A_1 \ldots A_m$
$\vdots$
have B
$\}$

proves $[\![A_1; \ldots ; A_m]\!] \Longrightarrow B$
where all x_i have been replaced by $?x_i$.

Proof state and Isar text

In general: **proof** *method*

Applies *method* and generates subgoal(s):

$$\bigwedge x_1 \dots x_n \; [\![\, A_1; \dots ; A_m \,]\!] \Longrightarrow B$$

How to prove each subgoal:

fix $x_1 \dots x_n$
assume $A_1 \dots A_m$
$\vdots$
show B

Separated by **next**

Summary

- Introduction to Isar and to some common proof patterns e.g. case distinction, contradiction, etc.
- Structured proofs are becoming the norm for Isabelle as they are more readable and easier to maintain.
- Mastering structured proof takes practice and it is usually better to have a clear proof plan beforehand.
- Useful resource: Isar quick reference manual (see AR web page).
- Reading: N&K (Concrete Semantics), Chapter 5.