Chapter 5

Isar: A Language for Structured Proofs

Apply scripts

- unreadable
- hard to maintain

Isar by example
Proof patterns
Streamlining Proofs
Proof by Cases and Induction
Apply scripts versus Isar proofs

Apply script = assembly language program
Isar proof = structured program with assertions

But: apply still useful for proof exploration

A typical Isar proof

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

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

proves $formula_0 \Rightarrow formula_{n+1}$
proof = proof [method] \text{ step}^* \text{ qed} \\
  | \text{ by method} \\

method = (simp ...) \mid (blast ...) \mid (induction ...) \mid ... \\

step = \text{ fix variables} \quad (\wedge) \\
  \mid \text{ assume \ prop} \quad (\implies) \\
  \mid \text{ [from fact']} \ (\text{ have} \mid \text{ show}) \ \text{ prop} \ \text{ proof} \\

prop = [\text{name:}] "\text{ formula}"
Isar core syntax

proof = proof [method] step* qed
  | by method
method = (simp ...) | (blast ...) | (induction ...) | ...

step = fix variables (∧)
  | assume prop (⇒)
  | [from fact⁺] (have | show) prop proof

prop = [name:] "formula"

fact = name | ...

Example: Cantor's theorem

lemma ¬ surj(f :: 'a ⇒ 'a set)

proof

Example: Cantor's theorem

lemma ¬ surj(f :: 'a ⇒ 'a set)
proof default proof: assume surj, show False
Example: Cantor’s theorem

lemma \( \rightarrow \) surj(f :: 'a ⇒ '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)

Example: Cantor’s theorem

lemma \( \rightarrow \) surj(f :: 'a ⇒ '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 \( c: \exists a. \{x. x \notin f \ x\} = f \ a \)
Example: Cantor's theorem

lemma \( \neg \text{surj}(f :: 'a \Rightarrow 'a \text{ set}) \)
proof default proof: assume \text{surj}, show False
  assume \( a: \text{surj } f \)
  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

Example: Cantor's theorem

lemma \( \neg \text{surj}(f :: 'a \Rightarrow 'a \text{ set}) \)
proof default proof: assume \text{surj}, show False
  assume \( a: \text{surj } f \)
  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

Isar_Demo.thy

Cantor and abbreviations
Example: Cantor’s theorem

**lemma** \( \neg \text{surj}(f :: 'a \Rightarrow 'a \ \text{set}) \)

**proof**

default proof: assume \( \text{surj} \), show \( \text{False} \)

assume \( a: \text{surj } f \)

from \( a \) have \( b: \ \forall \ A. \ \exists \ a. \ A = f \ a \)

by \((\text{simp add: surj_def})\)

from \( b \) have \( c: \ \exists \ a. \ \{ x. \ x \notin f \ x \} = f \ a \)

by \( \text{blast} \)

from \( c \) show \( \text{False} \)

by \( \text{blast} \)

\[\text{qed}\]

---

**using** and **with**

\((\text{have}|\text{show})\) prop **using** facts

\(\text{from facts (have|show) prop}\)

---

Structured lemma statement

**lemma**

\( \text{fixes } f :: 'a \Rightarrow 'a \ \text{set} \)

**assumes** \( s: \text{surj } f \)

**shows** \( \text{False} \)
Structured lemma statement

```
lemma
  fixes f :: 'a ⇒ 'a set
  assumes s: surj f
  shows False
proof —  no automatic proof step
```
Structured lemma statement

<table>
<thead>
<tr>
<th>lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixes f :: 'a ⇒ 'a set</td>
</tr>
<tr>
<td>assumes s: surj f</td>
</tr>
<tr>
<td>shows False</td>
</tr>
<tr>
<td>proof — no automatic proof step</td>
</tr>
<tr>
<td>have ⋄ a. {x. x ∉ f x} = f a using s</td>
</tr>
<tr>
<td>by(auto simp: surj-def)</td>
</tr>
</tbody>
</table>

qed

Proves surj f ⟷ False

---

Structured lemma statement

<table>
<thead>
<tr>
<th>lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixes f :: 'a ⇒ 'a set</td>
</tr>
<tr>
<td>assumes s: surj f</td>
</tr>
<tr>
<td>shows False</td>
</tr>
<tr>
<td>proof — no automatic proof step</td>
</tr>
<tr>
<td>have ⋄ a. {x. x ∉ f x} = f a using s</td>
</tr>
<tr>
<td>by(auto simp: surj-def)</td>
</tr>
<tr>
<td>thus False by blast</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>using and with</th>
</tr>
</thead>
<tbody>
<tr>
<td>(have</td>
</tr>
</tbody>
</table>

Proves surj f ⟷ 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

Case distinction

\begin{verbatim}
  show R
  proof cases
    assume P
    show R ...
  next
  assume \neg P
    show R ...
  qed
\end{verbatim}

Case distinction

\begin{verbatim}
  have P \lor Q ...
  proof cases
    assume P
    show R ...
  next
  assume \neg P
  show R ...
  qed
\end{verbatim}
Contradiction

show \neg P
proof
  assume P
  ...
  show False ...
qed

Contradiction

show \neg P
proof
  assume P
  ...
  show False ...
  Qed

Contradiction

show P
proof (rule contr)
  assume \neg P
  ...
  show False ...
  Qed

Contradiction

show P \leftrightarrow Q
proof
  assume P
  ...
  show Q ...
next
  assume Q
  ...
  show P ...
qed
∀ and ∃ introduction

\[
\begin{align*}
\text{show } & \forall x. \ P(x) \\
\text{proof} & \\
\text{fix } & \ x \text{ local fixed variable} \\
\text{show } & \ P(x) \ldots \\
\text{qed}
\end{align*}
\]

\[
\begin{align*}
\text{show } & \exists x. \ P(x) \\
\text{proof} & \\
\text{fix } & \ x \text{ local fixed variable} \\
\text{show } & \ P(x) \ldots \\
\text{qed}
\end{align*}
\]

\[
\begin{align*}
\text{show } & \exists x. \ P(x) \\
\text{proof} & \\
\vdots & \\
\text{show } & \ P(\text{witness}) \ldots \\
\text{qed}
\end{align*}
\]

Structured lemma statements

\[
\begin{align*}
\text{shows } & \ P \leftrightarrow Q \\
\text{proof} & \\
\text{assume } & \ P \\
\vdots & \\
\text{show } & \ Q \ldots \\
\text{next} & \\
\text{assume } & \ Q \\
\vdots & \\
\text{show } & \ P \ldots \\
\text{qed}
\end{align*}
\]

\[
\begin{align*}
\text{fixes } & \ x :: \tau_1 \text{ and } \ y :: \tau_2 \ldots \\
\text{assumes } & \ a: \ P \text{ and } b: \ Q \ldots \\
\text{shows } & \ R
\end{align*}
\]
∃ elimination: obtain

have \( \exists x. P(x) \)
then obtain \( x \) where \( p: P(x) \) by blast
\[ x \text{ fixed local variable} \]

obtain example

lemma \( \neg \text{surj}(f::'a \Rightarrow 'a\text{ set}) \)
proof
assume \( \text{surj} f \)
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

obtain example

lemma \( \neg \text{surj}(f::'a \Rightarrow 'a\text{ set}) \)
proof
assume \( \text{surj} f \)
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 \( a \notin f a \iff a \in f a \) by blast
lemma \rightarrow \text{surj}(f :: 'a \Rightarrow 'a \text{ set})

proof
  assume \text{surj } f
  hence \( \exists a. \{x. x \not\in f x\} = f a \) by (auto simp: surj_def)
  then obtain a where \( \{x. x \not\in f x\} = f a \) by blast
  hence \( a \not\in f a \leftrightarrow a \in f a \) by blast
  thus \textbf{False} by blast
qed

---

Set equality and subset

show \( A = B \)
proof
  show \( A \subseteq B \) ...
next
  show \( B \subseteq A \) ...
qed

---

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 \( x \in A \)
  show \( x \in B \) ...
  qed
Isar_Demo.thy

Exercise

\exists \text{ elimination: obtain}

\forall \text{ and } \exists \text{ introduction}

\text{have } \exists x. \ P(x) \\
\text{then obtain } x \text{ where } p: \ P(x) \text{ by blast}

\upiota \ x \text{ fixed local variable}

\upiota (x) \ldots

\upiota (\text{witness}) \ldots
and \( \exists \) introduction

thus "False" by blast

qed

text{* Interactive exercise

lemma assumes "\( \exists x. \forall y. \)"

proof
  fix b
  from assms obtain a with
  show "\( \exists x. P x b \)"

proof
  show ""[proof]
  qed
  qed

sorry

Textbook proof
\[
\begin{align*}
t_1 &= t_2 \quad \text{(justification)} \\
      &= t_3 \quad \text{(justification)} \\
      &\vdots \\
      &= t_n \quad \text{(justification)}
\end{align*}
\]

In Isabelle:

have "\( t_1 = t_2 \)" \((proof)\)
also have "\( \ldots = t_3 \)" \((proof)\)
\vdots
also have "\( \ldots = t_n \)" \((proof)\)
Instead of $=$ you may also use $\leq$ and $<$. 

Example

have "$t_1 < t_2" <proof>
also have "... = t_3" <proof>
also have "... $\leq t_n" <proof>
finally show "$t_1 < t_n" .
Example: pattern matching

\[
\text{show } formula_1 \leftrightarrow formula_2 \quad (\text{is } ?L \leftrightarrow ?R)
\]

\[
\text{proof}
\]
\[
\text{assume } ?L
\]
\[
\vdots
\]
\[
\text{show } ?R \ldots
\]
next
\[
\text{assume } ?R
\]
\[
\vdots
\]
\[
\text{show } ?L \ldots
\]
\[
\text{qed}
\]

\textbf{∀ and ∃ introduction}

\[
\text{show } \forall x. P(x)
\]
\[
\text{proof}
\]
\[
\text{fix } x \quad \text{local fixed variable}
\]
\[
\text{show } P(x) \ldots
\]
\[
\text{qed}
\]

?thesis

\[
\text{show } formula
\]
\[
\text{proof -}
\]
\[
\vdots
\]
\[
\text{show } ?thesis \ldots
\]
\[
\text{qed}
\]
show formula (is ?thesis)
proof -
  :  
    show ?thesis ...
qed

Quoting facts by value

By name:
  have x0: "x > 0" ... 
  :  
from x0 ...

Quoting facts by value

By name:
  have x0: "x > 0" ... 
  :  
from x0 ...

By value:
  have "x > 0" ... 
  :  
from 'x>0' ...
Quoting facts by value

By name:

```
have x0: "x > 0" ...
::
from x0 ...
```

By value:

```
have "x > 0" ...
::
from 'x>0' ...

↑ ↑
back quotes
```