Script generated by TTT

Title: FDS (26.05.2017)

Date: Fri May 26 08:30:13 CEST 2017

Duration: 91:58 min

Pages: 77

Chapter 5

Isar: A Language for Structured Proofs

Apply scripts

- unreadable
- hard to maintain

Apply scripts versus Isar proofs

Apply script = assembly language program

Isar proof = structured program with assertions

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

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**
$\quad\quad\quad\;|$ **by** method

Isar core syntax

proof $=$ **proof** [method] step* **qed**
$\quad\quad\quad\;|$ **by** method

Isar core syntax

proof $=$ **proof** [method] step* **qed**
$\quad\quad\quad\;|$ **by** method

method $=$ $(simp\ \ldots) \mid (blast\ \ldots) \mid (induction\ \ldots) \mid \ldots$

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

Isar core syntax

proof $=$ **proof** [method] step* **qed**
$\quad\quad\quad\;|$ **by** method

method $=$ $(simp\ \ldots) \mid (blast\ \ldots) \mid (induction\ \ldots) \mid \ldots$

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

prop $=$ [name:] "formula"

Isar core syntax

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

method = $(simp \ldots) \mid (blast \ldots) \mid (induction \ldots) \mid \ldots$

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

prop = [name:] "formula"

fact = name | . . .

Example: Cantor's theorem

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

Example: Cantor's theorem

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

Example: Cantor's theorem

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

Example: Cantor's theorem

lemma $\neg$ *surj*(f :: $'a \Rightarrow 'a$ *set*)
proof default proof: assume *surj*, show *False*
  assume a: *surj f*

Example: Cantor's theorem

lemma $\neg$ *surj*(f :: $'a \Rightarrow 'a$ *set*)
proof default proof: assume *surj*, show *False*
  assume a: *surj f*
  from a **have** b: $\forall\ A.\ \exists\ a.\ A = f\ a$

Example: Cantor's theorem

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*)

Example: Cantor's theorem

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: $\exists\ a.\ \{x.\ x \notin f\ x\} = f\ a$

Example: Cantor's theorem

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$(simp\ add\colon surj\_def)$
  from b **have** c: $\exists\ a.\ \{x.\ x \notin f\ x\} = f\ a$
    by *blast*

Example: Cantor's theorem

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$(simp\ add\colon surj\_def)$
  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 $\neg$ $surj(f :: 'a \Rightarrow 'a\ set)$
proof default proof: assume *surj*, show *False*
  assume a: $surj\ f$
  from a **have** b: $\forall\ A.\ \exists\ a.\ A = f\ a$
    by$(simp\ add\colon surj\_def)$
  from b **have** c: $\exists\ a.\ \{x.\ x \notin f\ x\} = f\ a$
    by *blast*
  from c **show** *False*
    by *blast*
qed

`Isar_Demo.thy`

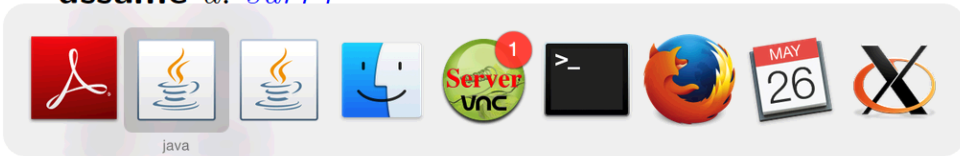
Cantor and abbreviations

Example: Cantor's theorem

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

Example: Cantor's theorem

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



    by *blast*
  from c **show** *False*
    by *blast*
qed

using and **with**

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

Structured lemma statement

lemma
  fixes $f :: {}'a \Rightarrow {}'a\ set$
  assumes s: *surj f*
  shows *False*

Structured lemma statement

lemma
  fixes $f :: 'a \Rightarrow 'a\ set$
  assumes s: $surj\ f$
  shows $False$
proof − no automatic proof step

Structured lemma statement

lemma
  fixes $f :: 'a \Rightarrow 'a\ set$
  assumes s: $surj\ f$
  shows $False$

Structured lemma statement

lemma
  fixes $f :: 'a \Rightarrow 'a\ set$
  assumes s: $surj\ f$
  shows $False$
proof −

Structured lemma statement

lemma
  fixes $f :: 'a \Rightarrow 'a\ set$
  assumes s: $surj\ f$
  shows $False$
proof − no automatic proof step

Structured lemma statement

lemma
fixes $f :: 'a \Rightarrow 'a\ set$
assumes s: $surj\ f$
shows $False$
proof – no automatic proof step
have $\exists\ a.\ \{x.\ x \notin f\ x\} = f\ a$ **using** s
by$(auto\ simp\text{:}\ surj\_def)$

Structured lemma statement

lemma
fixes $f :: 'a \Rightarrow 'a\ set$
assumes s: $surj\ f$
shows $False$
proof – no automatic proof step
have $\exists\ a.\ \{x.\ x \notin f\ x\} = f\ a$ **using** s
by$(auto\ simp\text{:}\ surj\_def)$
thus $False$ **by** $blast$
qed

Proves $surj\ f \Longrightarrow False$

Structured lemma statement

lemma
fixes $f :: 'a \Rightarrow 'a\ set$
assumes s: $surj\ f$
shows $False$
proof – no automatic proof step
have $\exists\ a.\ \{x.\ x \notin f\ x\} = f\ a$ **using** s
by$(auto\ simp\text{:}\ surj\_def)$
thus $False$ **by** $blast$
qed

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

using and **with**

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

The essence of structured proofs

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

Case distinction

```
show R
proof cases
  assume P
  ⋮
  show R  ...
next
  assume ¬ P
  ⋮
  show R  ...
qed
```


Case distinction

```
show R
proof cases
  assume P
  ⋮
  show R  ...
next
  assume ¬ P
  ⋮
  show R  ...
qed
```

```
have P ∨ Q  ...
then show R
proof
  assume P
  ⋮
  show R  ...
next
  assume Q
  ⋮
  show R  ...
qed
```

Contradiction

show $\neg\ P$
proof
  assume P
  $\vdots$
  show $False \quad \ldots$
qed

Contradiction

show $\neg\ P$
proof
  assume P
  $\vdots$
  show $False \quad \ldots$
qed

show P
proof $(rule\ ccontr)$
  assume $\neg P$
  $\vdots$
  show $False \quad \ldots$
qed

$\longleftrightarrow$

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

$\longleftrightarrow$

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

$\forall$ and $\exists$ introduction

show $\forall\, x.\ P(x)$
proof
  fix x local fixed variable
  show $P(x)$ $\ldots$
qed

$\forall$ and $\exists$ introduction

show $\forall\, x.\ P(x)$
proof
  fix x local fixed variable
  show $P(x)$ $\ldots$
qed

show $\exists\, x.\ P(x)$
proof
  $\vdots$
  show $P(witness)$ $\ldots$
qed

$\longleftrightarrow$

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

Structured lemma statements

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

$\exists$ elimination: **obtain**

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

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**$(auto\ simp\colon surj\_def)$

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**$(auto\ simp\colon surj\_def)$
then obtain a **where** $\{x.\ x \notin f\ x\} = f\ a$ **by** $blast$

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**$(auto\ simp\colon surj\_def)$
then obtain a **where** $\{x.\ x \notin f\ x\} = f\ a$ **by** $blast$
hence $a \notin f\ a \longleftrightarrow a \in f\ a$ **by** $blast$

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

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

148

Set equality and subset

show $A = B$
proof
  show $A \subseteq B$ $\ldots$
next
  show $B \subseteq A$ $\ldots$
qed

Set equality and subset

show $A = B$
proof
  show $A \subseteq B$ $\ldots$
next
  show $B \subseteq A$ $\ldots$
qed

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

`Isar_Demo.thy`

Exercise

$\exists$ elimination: **obtain**

$\exists$ elimination: **obtain**

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

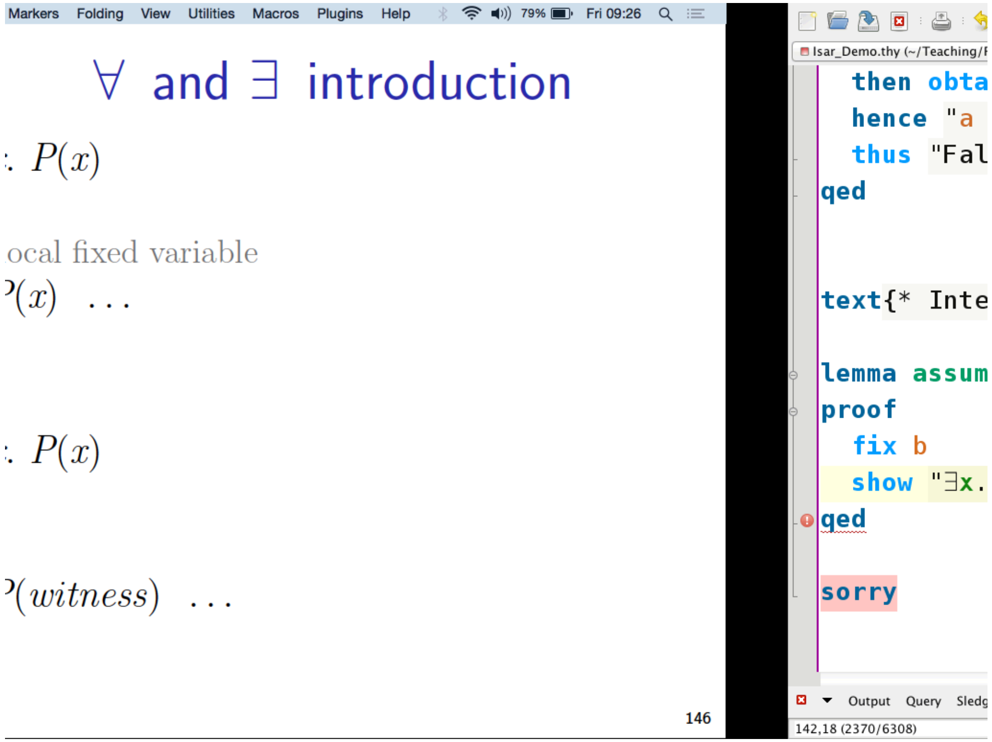
$\forall$ and $\exists$ introduction

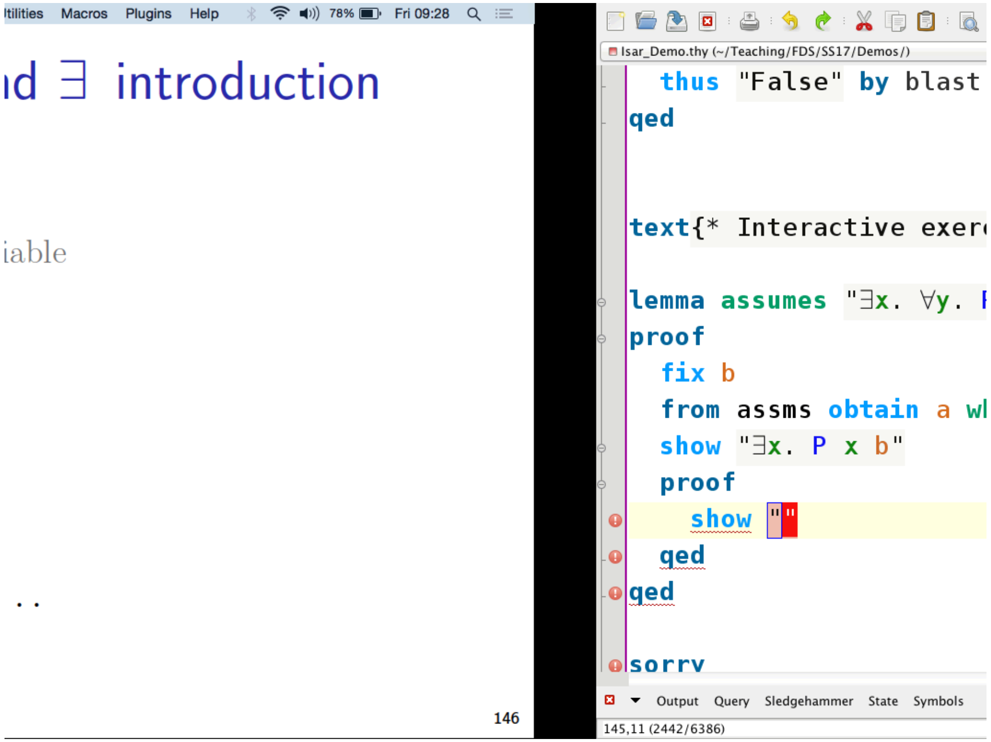
: $P(x)$

ocal fixed variable
$P(x)$ ...

: $P(x)$

$P(witness)$ ...





Chains of equations

Textbook proof

$$\begin{aligned} t_1 &= t_2 && \langle\text{justification}\rangle \\ &= t_3 && \langle\text{justification}\rangle \\ &\vdots \\ &= t_n && \langle\text{justification}\rangle \end{aligned}$$

Chains of equations

Textbook proof

$$\begin{aligned} t_1 &= t_2 && \langle\text{justification}\rangle \\ &= t_3 && \langle\text{justification}\rangle \\ &\vdots \\ &= t_n && \langle\text{justification}\rangle \end{aligned}$$

In Isabelle:

have "$t_1 = t_2$" $\langle proof\rangle$
also have "$\ldots = t_3$" $\langle proof\rangle$
$\vdots$
also have "$\ldots = t_n$" $\langle proof\rangle$

Chains of equations

Textbook proof

$$\begin{aligned} t_1 &= t_2 && \langle\text{justification}\rangle \\ &= t_3 && \langle\text{justification}\rangle \\ &\vdots \\ &= t_n && \langle\text{justification}\rangle \end{aligned}$$

In Isabelle:

have "$t_1 = t_2$" $\langle proof\rangle$
also have "$\ldots = t_3$" $\langle proof\rangle$
$\vdots$
also have "$\ldots = t_n$" $\langle proof\rangle$
finally show "$t_1 = t_n$" .

Chains of equations and inequations

Instead of $=$ you may also use $\leq$ and $<$.

Chains of equations and inequations

Instead of $=$ you may also use $\leq$ and $<$.

Example

have "$t_1 < t_2$" $\langle proof \rangle$
also have "... $= t_3$" $\langle proof \rangle$
$\vdots$
also have "... $\leq t_n$" $\langle proof \rangle$
finally show "$t_1 < t_n$" .

Isar_Demo.thy

Example & Exercise

Isabelle File Edit Search Markers Folding View Utilities Macros Plugins Help 74% Fri 09:35

Isar_Demo.thy

Example & Exercise

Example: pattern matching

show $formula_1 \longleftrightarrow formula_2$ (**is** $?L \longleftrightarrow ?R$)

∀ and ∃ introduction

show $\forall\, x.\ P(x)$
proof
  fix x local fixed variable
  show $P(x)$ ...
qed

Example: pattern matching

show $formula_1 \longleftrightarrow formula_2$ (**is** $?L \longleftrightarrow ?R$)
proof
  assume $?L$
  ⋮
  show $?R$ ...
next
  assume $?R$
  ⋮
  show $?L$ ...
qed

$?thesis$

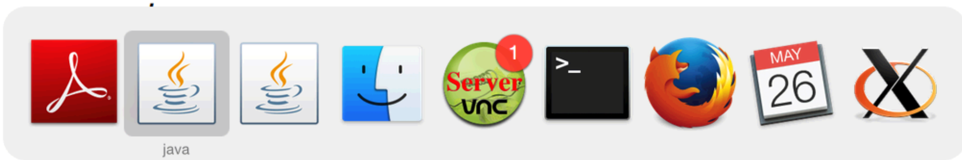
show $formula$
proof -
  ⋮
  show $?thesis$ ...
qed

?thesis

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

?thesis

show *formula* (is *?thesis*)
proof -



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 $`x{>}0`$ ...

$\uparrow \quad \uparrow$

back quotes

Quoting facts by value

By name:

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

By value:

have $"x > 0"$ ...
⋮
from $`x{>}0`$ ...

$\uparrow \quad \uparrow$

back quotes

161