Automatic Deduction — LVA 703522

Introduction to Isabelle

Clemens Ballarin

Universität Innsbruck

ISAR

Contents

ISAR

Apply scripts vs. Isar

Apply scripts

- Unreadable
- Hard to maintain
- Do not scale

No structure.

What about...

- Elegance?
- Explaining deeper insights?
- Large developments?

Isar!

A Typical Isar Proof

proof
  assume $\langle \mathit{formula}_0 \rangle$
  have $\langle \mathit{formula}_1 \rangle$ **by** simp
  $\vdots$
  have $\langle \mathit{formula}_n \rangle$ **by** blast
  show $\langle \mathit{formula}_{n+1} \rangle$ **by** ...
qed

proves $\langle \mathit{formula}_0 \rangle \Longrightarrow \langle \mathit{formula}_{n+1} \rangle$

Analogous to **assumes**/**shows** in lemma statements.

Isar Core Syntax

⟨*proof*⟩	::=	**proof** [⟨*method*⟩] ⟨*statement*⟩* **qed** [⟨*method*⟩]	
	|	**by** ⟨*method*⟩ [⟨*method*⟩]	
⟨*method*⟩	::=	(simp . . .) | (blast . . .) | (rule . . .) | . . .	
⟨*statement*⟩	::=	**fix** ⟨*variable*⟩$^+$	($\bigwedge$)
	|	**assume** ⟨*proposition*⟩	($\Longrightarrow$)
	|	[**from** ⟨*name*⟩$^+$]	
		(**have** | **show**) ⟨*proposition*⟩ ⟨*proof*⟩	
	|	**next**	(separates subgoals)
⟨*proposition*⟩	::=	[⟨*name*⟩:] ⟨*formula*⟩	

Proof and Qed

proof [⟨*method*⟩] ⟨*statement*⟩* **qed** [⟨*method*⟩]

lemma "$[\![A; B]\!] \Longrightarrow A \wedge B$"
proof (rule conjI)
  assume A: "A"
  from A **show** "A" **by** assumption
next
  assume B: "B"
  from B **show** "B" **by** assumption
qed

- **proof** ⟨*method*⟩ applies method to the stated goal
- **proof** applies method **rule**
- **proof -** does nothing to the goal

How Do I Know What to Assume and Show?

Look at the proof state!

lemma "$\llbracket A; B \rrbracket \Longrightarrow A \wedge B$"
proof (rule conjI)

- 1. $\llbracket A; B \rrbracket \Longrightarrow A$
  2. $\llbracket A; B \rrbracket \Longrightarrow B$
- So we need: **show** "A" and **show** "B"
- We are allowed to **assume** A,
  because A is in the assumptions of the proof state.

The Three Modes of Isar

- **[prove]**:
  goal has been stated, proof needs to follow.
- **[state]**:
  proof block has been openend or subgoal has been proved, new **from** statement, goal statement or assumptions can follow.
- **[chain]**:
  from statement has been made, goal statement needs to follow.

The Three Modes of Isar

- **[prove]**: goal has been stated
- **[state]**: proof block has been openend
- **[chain]**: **from** statement has been made

lemma "$\llbracket A; B \rrbracket \Longrightarrow A \wedge B$" **[prove]**
proof (rule conjI) **[state]**
  assume A: "A" **[state]**
  from A **[chain]** **show** "A" **[prove]**
    by assumption **[state]**
next **[state]**
  ...
qed **[state]**

Have

Can be used to make intermediate steps.

Example

lemma "$(x$ :: nat$) + 1 = 1 + x$"
proof -
  have A: "$x + 1 =$ Suc x" **by** simp
  have B: "$1 + x =$ Suc x" **by** simp
  show "$x + 1 = 1 + x$" **by** (simp only: A B)
qed

Demo: Isar Proofs

Backward and Forward

Method rule can do both backward and forward reasoning.

Backward reasoning

have "$A \wedge B$" **proof**

- **proof** picks an **intro** rule.
- Conclusion of rule must unify with $A \wedge B$

Forward reasoning

assume AB: "$A \wedge B$"
from AB **have** "$\ldots$" **proof**

- Now **proof** picks an **elim** rule.
- Triggered by chained facts (**from**).
- First assumption of rule must unify with AB.

Forward Reasoning

General case

from $A_1 \ldots A_n$ **have** R **proof**

- First n assumptions of rule must unify with $A_1 \ldots A_n$.
- Conclusion of rule must unify with R.

Fix and Obtain

fix $v_1 \dots v_n$

Introduces new arbitrary but fixed variables.
($\sim$ parameters, $\bigwedge$)

obtain $v_1 \dots v_n$ **where** $\langle$*prop*$\rangle$ $\langle$*proof*$\rangle$

Introduces new variables together with property.

Demo

Nested Fixed Variables

Problem

$$\textbf{fix}\ x\ \textbf{assumes}\ "A\,x"\ \ \textbf{fix}\ x\ \textbf{assumes}\ "B\,x"\ \ \langle \textit{body} \rangle$$

- Only second x is visible in $\langle \textit{body} \rangle$
- Both $A\,x$ and $B\,x$ may appear in goal!

Solution

Name variants: $x = x, x = xa$

- In $\langle \textit{body} \rangle$, x refers to xa.
- Outer x is hidden.

To see name variants in Proof General, set

Isabelle $\rightarrow$ Settings $\rightarrow$ Prems Limit $\rightarrow$ 0

Shortcuts

this	=	the previous fact (proved or assumed)
then	=	**from** this
with $A_1 \ldots A_n$	=	**from** $A_1 \ldots A_n$ this
?thesis	=	the last enclosing goal statement
thus	=	**then show**
hence	=	**then have**

Moreover and Ultimately

have X_1: P_1 ...	**have** P_1 ...
have X_2: P_2 ...	**moreover have** P_2 ...
⋮	⋮
have X_n: P_n ...	**moreover have** P_n ...
from $X_1 \ldots X_n$ **show** ...	**ultimately show** ...

Wastes lots of brain power on names $X_1 \ldots X_n$.

General Case Distinctions

show ⟨*formula*⟩
proof -
  have $P_1 \vee P_2 \vee \ldots \vee P_n$ ⟨*proof*⟩
  moreover { **assume** P_1 ... **have** ?thesis ⟨*proof*⟩ }
  moreover { **assume** P_2 ... **have** ?thesis ⟨*proof*⟩ }
    ⋮
  moreover { **assume** P_n ... **have** ?thesis ⟨*proof*⟩ }
  ultimately show ?thesis **by** blast
qed

{ ... } is a proof block similar to **proof** ... **qed**

{ **assume** P_i ... **have** P ⟨*proof*⟩ } stands for $P_i \Longrightarrow P$

Demo: moreover and ultimately

Mixing Proof Styles

from $\mathsf{A}_1 \ldots \mathsf{A}_n$
have P
  apply –
  1. $[\![A_1; \ldots; A_n]\!] \Longrightarrow P$
  apply ⟨*method*⟩
    ⋮
  apply ⟨*method*⟩
  done

apply – turns chained facts into assumptions

Calculational Reasoning

The Goal

From group axioms

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \qquad 1 \cdot x = x \qquad x^{-1} \cdot x = 1$$

show

$$\begin{aligned}
x \cdot x^{-1} &= 1 \cdot (x \cdot x^{-1}) \\
&= 1 \cdot x \cdot x^{-1} \\
&= (x^{-1})^{-1} \cdot x^{-1} \cdot x \cdot x^{-1} \\
&= (x^{-1})^{-1} \cdot (x^{-1} \cdot x) \cdot x^{-1} \\
&= (x^{-1})^{-1} \cdot 1 \cdot x^{-1} \\
&= (x^{-1})^{-1} \cdot (1 \cdot x^{-1}) \\
&= (x^{-1})^{-1} \cdot x^{-1} \\
&= 1.
\end{aligned}$$

Can We Do This in Isabelle?

- Simplifier: too eager
- Manual: difficult in apply style
- Isar: with the methods we know, too verbose

Chains of Equations

The Problem

$$a = b = c = d$$

Shows $a = d$ by transitivity of "$=$".

Each step usually nontrivial (requires subproof).

Solution in Isar

- Keywords **also** and **finally** to delimit steps.
- "$\ldots$": predefined schematic term variable, refers to right hand side of last expression
- Automatic use of transitivity rules to connect steps.

also/finally

have "$t_0 = t_1$" ⟨*proof*⟩	Calculation register
also	$t_0 = t_1$
have "$\ldots = t_2$" ⟨*proof*⟩	
also	$t_0 = t_2$
⋮	⋮
also	$t_0 = t_{n-1}$
have "$\ldots = t_n$" ⟨*proof*⟩	
finally	$t_0 = t_n$
show P	

finally chaines fact $t_0 = t_n$ into the proof.

More about also

- Works for all combinations of $=$, $\leq$ and $<$.
- Uses all transitivity rules; declared as [trans].
- To view all rules in Proof General:

  Isabelle $\rightarrow$ Show me $\rightarrow$ Transitivity rules

Designing Transitivity Rules

Anatomy of a transitivity rule

- Usual form: plain transitivity $[\![l_1 \lhd r_1; r_1 \lhd r_2]\!] \Longrightarrow l_1 \lhd r_2$
- More general form: $[\![P\ l_1\ r_1; Q\ r_1\ r_2; A]\!] \Longrightarrow C\ l_1\ r_2$

Examples

- pure transitivity: $[\![a = b; b = c]\!] \Longrightarrow a = c$
- mixed: $[\![a \le b; b < c]\!] \Longrightarrow a < c$
- substitution: $[\![P\ a; a = b]\!] \Longrightarrow P\ b$
- antisymmetry: $[\![a < b; b < a]\!] \Longrightarrow P$
- monotonicity:
  $[\![a = f\ b; b < c; \bigwedge x\ y.\ x < y \Longrightarrow f\ x < f\ y]\!] \Longrightarrow a < f\ c$

Demo

What We Have Seen so far . . .

- Three modes of Isar: prove, state, chain
- Forward and backward reasoning with the rule method
- Accumulating nameless lemmas: **moreover** / **ultimately**
- Proving chains of equations: **also** / **finally**