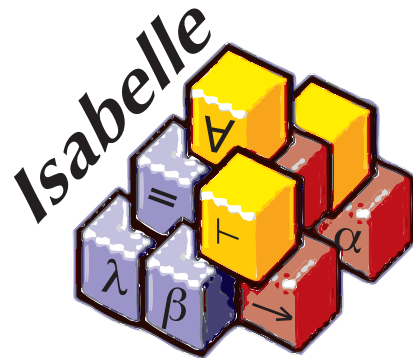


# Introduction to Isabelle

Clemens Ballarin  
Universität Innsbruck



ISAR

# Contents

- ▶ Intro & motivation, getting started with Isabelle
- ▶ Foundations & Principles
  - ▶ Lambda Calculus
  - ▶ Types & Classes
  - ▶ Natural Deduction
- ▶ **Proof & Specification Techniques**
  - ▶ **Isar: mathematics style proofs**
  - ▶ Inductively defined sets, rule induction
  - ▶ Datatypes, structural induction
  - ▶ Recursive functions & code generation

# 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  $\langle formula_0 \rangle$   
  have  $\langle formula_1 \rangle$  by simp  
   $\vdots$   
  have  $\langle formula_n \rangle$  by blast  
  show  $\langle formula_{n+1} \rangle$  by ...  
qed
```

proves  $\langle formula_0 \rangle \implies \langle formula_{n+1} \rangle$

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

# Isar Core Syntax

$\langle proof \rangle ::= \mathbf{proof} [\langle method \rangle] \langle statement \rangle^* \mathbf{qed} [\langle method \rangle]$   
|  $\mathbf{by} \langle method \rangle [\langle method \rangle]$

$\langle method \rangle ::= (\mathbf{simp} \dots) \mid (\mathbf{blast} \dots) \mid (\mathbf{rule} \dots) \mid \dots$

$\langle statement \rangle ::= \mathbf{fix} \langle variable \rangle^+ \quad (\wedge)$   
|  $\mathbf{assume} \langle proposition \rangle \quad (\implies)$   
|  $[\mathbf{from} \langle name \rangle^+]$   
|  $(\mathbf{have} \mid \mathbf{show}) \langle proposition \rangle \langle proof \rangle$   
|  $\mathbf{next} \quad (\text{separates subgoals})$

$\langle proposition \rangle ::= [\langle name \rangle:] \langle formula \rangle$

# Proof and Qed

**proof** [ $\langle method \rangle$ ]  $\langle statement \rangle^*$  **qed** [ $\langle method \rangle$ ]

**lemma** "  $\llbracket A; B \rrbracket \implies A \wedge B$  "

**proof** (rule conjI)

**assume** A: "  $A$  "

**from** A **show** "  $A$  " **by** assumption

**next**

**assume** B: "  $B$  "

**from** B **show** "  $B$  " **by** assumption

**qed**

- ▶ **proof**  $\langle method \rangle$  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** " $\llbracket A; B \rrbracket \implies A \wedge B$ "

**proof** (rule conjI)

- ▶ 1.  $\llbracket A; B \rrbracket \implies A$
- ▶ 2.  $\llbracket A; B \rrbracket \implies B$
- ▶ 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 opened 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 opened
- ▶ **[chain]**: **from** statement has been made

```
lemma "[[A; B]]  $\implies$  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 "( $x :: \text{nat}$ ) + 1 = 1 +  $x$ "  
proof -  
  have A: " $x + 1 = \text{Suc } x$ " by simp  
  have B: " $1 + x = \text{Suc } x$ " 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** "... " **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**  $A_1 \dots A_n$  **have**  $R$  **proof**

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

# Fix and Obtain

**fix**  $v_1 \dots v_n$

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

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

Introduces new variables together with property.

# Demo

# Nested Fixed Variables

## Problem

**fix**  $x$  **assumes** " $A x$ " **fix**  $x$  **assumes** " $B x$ "  $\langle body \rangle$

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

## Solution

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

- ▶ In  $\langle body \rangle$ ,  $x$  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**  $A_1 \dots A_n$  = **from**  $A_1 \dots A_n$  this

?thesis = the last enclosing goal statement

**thus** = **then show**

**hence** = **then have**

# Moreover and Ultimately

have  $X_1$ :  $P_1 \dots$

have  $X_2$ :  $P_2 \dots$

$\vdots$

have  $X_n$ :  $P_n \dots$

from  $X_1 \dots X_n$  show  $\dots$

have  $P_1 \dots$

**moreover** have  $P_2 \dots$

$\vdots$

**moreover** have  $P_n \dots$

**ultimately** show  $\dots$

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

# General Case Distinctions

**show**  $\langle formula \rangle$

**proof** -

**have**  $P_1 \vee P_2 \vee \dots \vee P_n$   $\langle proof \rangle$

**moreover** { **assume**  $P_1$  ... **have** ?thesis  $\langle proof \rangle$  }

**moreover** { **assume**  $P_2$  ... **have** ?thesis  $\langle proof \rangle$  }

⋮

**moreover** { **assume**  $P_n$  ... **have** ?thesis  $\langle proof \rangle$  }

**ultimately show** ?thesis **by** blast

**qed**

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

{ **assume**  $P_i$  ... **have**  $P$   $\langle proof \rangle$  } stands for  $P_i \implies P$

Demo: moreover and  
ultimately

# Mixing Proof Styles

**from**  $A_1 \dots A_n$

**have**  $P$

**apply** –

1.  $\llbracket A_1; \dots; A_n \rrbracket \implies P$

**apply**  $\langle method \rangle$

$\vdots$

**apply**  $\langle method \rangle$

**done**

**apply** – turns chained facts into assumptions

# Computational Reasoning

# The Goal

From group axioms

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

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.
- ▶ “...”: predefined schematic term variable, refers to right hand side of last expression
- ▶ Automatic use of transitivity rules to connect steps.

## also/finally

<b>have</b> " $t_0 = t_1$ " $\langle proof \rangle$	Calculation register
<b>also</b>	$t_0 = t_1$
<b>have</b> " $\dots = t_2$ " $\langle proof \rangle$	
<b>also</b>	$t_0 = t_2$
$\vdots$	$\vdots$
<b>also</b>	$t_0 = t_{n-1}$
<b>have</b> " $\dots = t_n$ " $\langle proof \rangle$	
<b>finally</b>	$t_0 = t_n$
<b>show</b> P	

**finally** chains 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  $\llbracket l_1 \triangleleft r_1; r_1 \triangleleft r_2 \rrbracket \Longrightarrow l_1 \triangleleft r_2$
- ▶ More general form:  $\llbracket P \ l_1 \ r_1; Q \ r_1 \ r_2; A \rrbracket \Longrightarrow C \ l_1 \ r_2$

## Examples

- ▶ pure transitivity:  $\llbracket a = b; b = c \rrbracket \Longrightarrow a = c$
- ▶ mixed:  $\llbracket a \leq b; b < c \rrbracket \Longrightarrow a < c$
- ▶ substitution:  $\llbracket P \ a; a = b \rrbracket \Longrightarrow P \ b$
- ▶ antisymmetry:  $\llbracket a < b; b < a \rrbracket \Longrightarrow P$
- ▶ monotonicity:  
 $\llbracket a = f \ b; b < c; \bigwedge x \ y. x < y \Longrightarrow f \ x < f \ y \rrbracket \Longrightarrow a < f \ c$

# 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**