COMP 4161

NICTA Advanced Course

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Toby Murray

Isar

Content

Rough timeline

- ➜ Intro & motivation, getting started [1]
- ➜ Foundations & Principles
  - Lambda Calculus, natural deduction [2,3,4[a]]
  - Higher Order Logic [5,6[b],7]
  - Term rewriting [8,9,10[c]]
- ➜ Proof & Specification Techniques
  - Isar [11,12[d]]
  - Inductively defined sets, rule induction [13[e],15]
  - Datatypes, recursion, induction [16,17[f],18,19]
  - Calculational reasoning, mathematics style proofs [20]
  - Hoare logic, proofs about programs [21[g],22,23]

[a] a1 out; [b] a1 due; [c] a2 out; [d] a2 due; [e] session break; [f] a3 out; [g] a3 due

ISAR

A LANGUAGE FOR STRUCTURED PROOFS

apply scripts	**What about..**
➜ unreadable	➜ Elegance?
➜ hard to maintain	➜ Explaining deeper insights?
➜ do not scale	➜ Large developments?
No structure.	**Isar!**

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 $formula_0 \Longrightarrow formula_{n+1}$

(analogous to **assumes**/**shows** in lemma statements)

Isar core syntax

proof = **proof** [method] statement* **qed**

$\mid$ **by** method

method = (simp . . .) $\mid$ (blast . . .) $\mid$ (rule . . .) $\mid$ . . .

statement = **fix** variables $(\bigwedge)$

$\mid$ **assume** proposition $(\Longrightarrow)$

$\mid$ [**from** name$^+$] (**have** $\mid$ **show**) proposition proof

$\mid$ **next** (separates subgoals)

proposition = [name:] formula

proof and qed

proof [method] statement* **qed**

lemma "$\llbracket A; B \rrbracket \Longrightarrow A \wedge B$"
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 a single rule that fits
➜ **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)

- **proof** (rule conjI) changes proof state to
  1. $\llbracket A; B \rrbracket \Longrightarrow A$
  2. $\llbracket A; B \rrbracket \Longrightarrow B$
- so we need 2 shows: **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 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.

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]** **. . .**

Have

Can be used to make intermediate steps.

Example:

lemma "$(x :: \mathsf{nat}) + 1 = 1 + x$"

proof -

  have A: "$x + 1 = \mathsf{Suc}\ x$" **by** simp

  have B: "$1 + x = \mathsf{Suc}\ x$" **by** simp

  show "$x + 1 = 1 + x$" **by** (simp only: A B)

qed

DEMO

Backward and Forward

NICTA

Backward reasoning: ... **have** "$A \wedge B$" **proof**

- ➜ **proof** picks an **intro** rule automatically
- ➜ 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 automatically
- ➜ triggered by **from**
- ➜ first assumption of rule must unify with AB

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 $v_1 \ldots v_n$

Introduces new arbitrary but fixed variables

($\sim$ parameters, $\bigwedge$)

obtain $v_1 \ldots v_n$ **where** $<$prop$>$ $<$proof$>$

Introduces new variables together with property

DEMO

Fancy Abbreviations

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

Moreover and Ultimately

have X_1: P_1 ...

have X_2: P_2 ...

$\vdots$

have X_n: P_n ...

from $X_1 \ldots X_n$ **show** ...

have P_1 ...

moreover **have** P_2 ...

$\vdots$

moreover **have** P_n ...

ultimately **show** ...

wastes lots of brain power

on names $X_1 \ldots X_n$

General Case Distinctions

show $formula$

proof -

  have $P_1 \vee P_2 \vee P_3$ <proof>

  moreover { **assume** P_1 . . . **have** ?thesis <proof> }

  moreover { **assume** P_2 . . . **have** ?thesis <proof> }

  moreover { **assume** P_3 . . . **have** ?thesis <proof> }

  ultimately show ?thesis **by** blast

qed

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

{ **assume** P_1 . . . **have** P <proof> }
stands for $P_1 \Longrightarrow P$

Mixing proof styles

from . . .

have . . .

  apply -   make incoming facts assumptions

  apply (. . .)

  ⋮

  apply (. . .)

  done