Slide 1

NICTA Advanced Course

Theorem Proving
Principles, Techniques, Applications

$$\{P\} \ldots \{Q\}$$

Slide 2

CONTENT

- Intro & motivation, getting started with Isabelle
- Foundations & Principles
  - Lambda Calculus
  - Higher Order Logic, natural deduction
  - Term rewriting
- **Proof & Specification Techniques**
  - Inductively defined sets, rule induction
  - Datatypes, recursion, induction
  - More recursion, Calculational reasoning
  - **Hoare logic, proofs about programs**
  - Locales, Presentation

Slide 3

LAST TIME

- Recdef
- More induction
- Well founded orders
- Well founded recursion
- Calculations: also/finally
- [trans]-rules

Slide 4

A CRASH COURSE IN SEMANTICS

IMP - A SMALL IMPERATIVE LANGUAGE

Slide 5

Commands:

datatype com	=	SKIP	
	|	Assign loc aexp	$(\_ := \_)$
	|	Semi com com	$(\_; \_)$
	|	Cond bexp com com	(IF _ THEN _ ELSE _)
	|	While bexp com	(WHILE _ DO _ OD)

types loc = string
types state = loc $\Rightarrow$ nat

types aexp = state $\Rightarrow$ nat
types bexp = state $\Rightarrow$ bool

EXAMPLE PROGRAM

Slide 6

Usual syntax:

$B := 1;$
WHILE $A \neq 0$ DO
  $B := B * A;$
  $A := A - 1$
OD

Expressions are functions from state to bool or nat:

$B := (\lambda\sigma.\ 1);$
WHILE $(\lambda\sigma.\ \sigma\ A \neq 0)$ DO
  $B := (\lambda\sigma.\ \sigma\ B * \sigma\ A);$
  $A := (\lambda\sigma.\ \sigma\ A - 1)$
OD

WHAT DOES IT DO?

Slide 7

So far we have defined:

➜ **Syntax** of commands and expressions
➜ **State** of programs (function from variables to values)

Now we need: the meaning (semantics) of programs

How to define execution of a program?

➜ A wide field of its own (visit a semantics course!)
➜ Some choices:

- Operational (inductive relations, big step, small step)
- Denotational (programs as functions on states, state transformers)
- Axiomatic (pre-/post conditions, Hoare logic)

STRUCTURAL OPERATIONAL SEMANTICS

Slide 8

$$\frac{}{\langle \mathsf{SKIP}, \sigma\rangle \longrightarrow \sigma}$$

$$\frac{e\ \sigma = v}{\langle \mathsf{x := e}, \sigma\rangle \longrightarrow \sigma[x \mapsto v]}$$

$$\frac{\langle c_1, \sigma\rangle \longrightarrow \sigma' \quad \langle c_2, \sigma'\rangle \longrightarrow \sigma''}{\langle c_1; c_2, \sigma\rangle \longrightarrow \sigma''}$$

$$\frac{b\ \sigma = \mathsf{True} \quad \langle c_1, \sigma\rangle \longrightarrow \sigma'}{\langle \mathsf{IF}\ b\ \mathsf{THEN}\ c_1\ \mathsf{ELSE}\ c_2, \sigma\rangle \longrightarrow \sigma'}$$

$$\frac{b\ \sigma = \mathsf{False} \quad \langle c_2, \sigma\rangle \longrightarrow \sigma'}{\langle \mathsf{IF}\ b\ \mathsf{THEN}\ c_1\ \mathsf{ELSE}\ c_2, \sigma\rangle \longrightarrow \sigma'}$$

Slide 9

STRUCTURAL OPERATIONAL SEMANTICS

$$\frac{b\ \sigma = \mathsf{False}}{\langle \mathsf{WHILE}\ b\ \mathsf{DO}\ c\ \mathsf{OD}, \sigma\rangle \longrightarrow \sigma}$$

$$\frac{b\ \sigma = \mathsf{True} \quad \langle c, \sigma\rangle \longrightarrow \sigma' \quad \langle \mathsf{WHILE}\ b\ \mathsf{DO}\ c\ \mathsf{OD}, \sigma'\rangle \longrightarrow \sigma''}{\langle \mathsf{WHILE}\ b\ \mathsf{DO}\ c\ \mathsf{OD}, \sigma\rangle \longrightarrow \sigma''}$$

Slide 10

DEMO: THE DEFINITIONS IN ISABELLE

Slide 11

PROOFS ABOUT PROGRAMS

Now we know:

- ➜ What programs are: Syntax
- ➜ On what they work: State
- ➜ How they work: Semantics

So we can prove properties about programs

Example:
Show that example program from slide 6 implements the factorial.

$$\textbf{lemma}\ \langle \mathsf{factorial}, \sigma\rangle \longrightarrow \sigma' \Longrightarrow \sigma' B = \mathsf{fac}\ (\sigma A)$$

(where $\mathsf{fac}\ 0 = 0$, $\mathsf{fac}\ (\mathsf{Suc}\ n) = (\mathsf{Suc}\ n) * \mathsf{fac}\ n$)

Slide 12

DEMO: EXAMPLE PROOF

Too tedious

Induction needed for each loop

Slide 13

Is there something easier?

Floyd/Hoare

Idea: describe meaning of program by pre/post conditions

Examples:

$\{\mathsf{True}\} \quad x := 2 \quad \{x = 2\}$

Slide 14

$\{y = 2\} \quad x := 21 * y \quad \{x = 42\}$

$\{x = n\} \quad \mathsf{IF}\ y < 0\ \mathsf{THEN}\ x := x + y\ \mathsf{ELSE}\ x := x - y \quad \{x = n - |y|\}$

$\{A = n\} \quad \mathsf{factorial} \quad \{B = \mathsf{fac}\ n\}$

Proofs: have rules that directly work on such triples

Meaning of a Hoare-Triple

$$\{P\} \quad c \quad \{Q\}$$

What are the assertions P and Q?

- ➜ Here: again functions from state to bool (shallow embedding of assertions)
- ➜ Other choice: syntax and semantics for assertions (deep embedding)

Slide 15

What does $\{P\}\ c\ \{Q\}$ mean?

Partial Correctness:

$\models \{P\}\ c\ \{Q\} \quad \equiv \quad (\forall \sigma\ \sigma'.\ P\ \sigma \wedge \langle c, \sigma\rangle \longrightarrow \sigma' \Longrightarrow Q\ \sigma')$

Total Correctness:

$\models \{P\}\ c\ \{Q\} \quad \equiv \quad (\forall \sigma.\ P\ \sigma \Longrightarrow \exists \sigma'.\ \langle c, \sigma\rangle \longrightarrow \sigma' \wedge Q\ \sigma')$

This lecture: partial correctness only (easier)

Hoare Rules

$$\frac{}{\{P\} \quad \mathsf{SKIP} \quad \{P\}} \qquad \frac{}{\{P[x \mapsto e]\} \quad x := e \quad \{P\}}$$

$$\frac{\{P\}\ c_1\ \{R\} \quad \{R\}\ c_2\ \{Q\}}{\{P\} \quad c_1; c_2 \quad \{Q\}}$$

Slide 16

$$\frac{\{P \wedge b\}\ c_1\ \{Q\} \quad \{P \wedge \neg b\}\ c_2\ \{Q\}}{\{P\} \quad \mathsf{IF}\ b\ \mathsf{THEN}\ c_1\ \mathsf{ELSE}\ c_2 \quad \{Q\}}$$

$$\frac{\{P \wedge b\}\ c\ \{P\} \quad P \wedge \neg b \Longrightarrow Q}{\{P\} \quad \mathsf{WHILE}\ b\ \mathsf{DO}\ c\ \mathsf{OD} \quad \{Q\}}$$

$$\frac{P \Longrightarrow P' \quad \{P'\}\ c\ \{Q'\} \quad Q' \Longrightarrow Q}{\{P\} \quad c \quad \{Q\}}$$

HOARE RULES

Slide 17

$$\frac{}{\vdash \{P\}\ \ \text{SKIP}\ \ \{P\}} \qquad \frac{}{\vdash \{\lambda\sigma.\ P\ (\sigma(x := e\ \sigma))\}\ \ x := e\ \ \{P\}}$$

$$\frac{\vdash \{P\}\ c_1\ \{R\} \quad \vdash \{R\}\ c_2\ \{Q\}}{\vdash \{P\}\ \ c_1; c_2\ \ \{Q\}}$$

$$\frac{\vdash \{\lambda\sigma.\ P\ \sigma \wedge b\ \sigma\}\ c_1\ \{R\} \quad \vdash \{\lambda\sigma.\ P\ \sigma \wedge \neg b\ \sigma\}\ c_2\ \{Q\}}{\vdash \{P\}\ \ \text{IF } b \text{ THEN } c_1 \text{ ELSE } c_2\ \ \{Q\}}$$

$$\frac{\vdash \{\lambda\sigma.\ P\ \sigma \wedge b\ \sigma\}\ c\ \{P\} \quad \bigwedge\sigma.\ P\ \sigma \wedge \neg b\ \sigma \Longrightarrow Q\ \sigma}{\vdash \{P\}\ \ \text{WHILE } b \text{ DO } c \text{ OD}\ \ \{Q\}}$$

$$\frac{\bigwedge\sigma.\ P\ \sigma \Longrightarrow P'\ \sigma \quad \vdash \{P'\}\ c\ \{Q'\} \quad \bigwedge\sigma.\ Q'\ \sigma \Longrightarrow Q\sigma}{\vdash \{P\}\ \ c\ \ \{Q\}}$$

ARE THE RULES CORRECT?

Soundness: $\vdash \{P\}\ c\ \{Q\} \Longrightarrow \models \{P\}\ c\ \{Q\}$

Proof: by rule induction on $\vdash \{P\}\ c\ \{Q\}$

Slide 18

Demo: Hoare Logic in Isabelle

NICER, BUT STILL KIND OF TEDIOUS

Hoare rule application seems boring & mechanical.

Automation?

Problem: While – need creativity to fi nd right (invariant) P

Slide 19

Solution:

- ➜ annotate program with invariants
- ➜ then, Hoare rules can be applied automatically

Example:

$\{M = 0 \wedge N = 0\}$
WHILE $M \neq a$ INV $\{N = M * b\}$ DO $N := N + b; M := M + 1$ OD
$\{N = a * b\}$

WEAKEST PRECONDITIONS

pre c Q = weakest P such that $\{P\}\ c\ \{Q\}$

Slide 20

With annotated invariants, easy to get:

$$\begin{array}{lcl}
\text{pre SKIP } Q & = & Q \\
\text{pre } (x := a)\ Q & = & \lambda\sigma.\ Q(\sigma(x := a\sigma)) \\
\text{pre } (c_1; c_2)\ Q & = & \text{pre } c_1\ (\text{pre } c_2\ Q) \\
\text{pre } (\text{IF } b \text{ THEN } c_1 \text{ ELSE } c_2)\ Q & = & \lambda\sigma.\ (b \longrightarrow \text{pre } c_1\ Q\ \sigma) \wedge \\
 & & \quad (\neg b \longrightarrow \text{pre } c_2\ Q\ \sigma) \\
\text{pre } (\text{WHILE } b \text{ INV } I \text{ DO } c \text{ OD})\ Q & = & I
\end{array}$$

Slide 21

Verification Conditions

$\{\text{pre } c\ Q\}\ c\ \{Q\}$ **only true under certain conditions**

These are called **verification conditions** vc $c\ Q$:

vc SKIP Q	$=$	True
vc $(x := a)\ Q$	$=$	True
vc $(c_1; c_2)\ Q$	$=$	vc $c_2\ Q \wedge$ (vc c_1 (pre $c_2\ Q$))
vc (IF b THEN c_1 ELSE c_2) Q	$=$	vc $c_1\ Q \wedge$ vc $c_2\ Q$
vc (WHILE b INV I DO c OD) Q	$=$	$(\forall \sigma.\ I\sigma \wedge b\sigma \longrightarrow \text{pre } c\ I\ \sigma) \wedge$ $(\forall \sigma.\ I\sigma \wedge \neg b\sigma \longrightarrow Q\ \sigma) \wedge$ vc $c\ I$

$$\text{vc } c\ Q \wedge (\text{pre } c\ Q \Longrightarrow P) \Longrightarrow \{P\}\ c\ \{Q\}$$

Slide 22

Syntax Tricks

- ➜ $x := \lambda\sigma.\ 1$ instead of $x := 1$ sucks
- ➜ $\{\lambda\sigma.\ \sigma\ x = n\}$ instead of $\{x = n\}$ sucks as well

Problem: program variables are functions, not values

Solution: distinguish program variables syntactically

Choices:

- ➜ declare program variables with each Hoare triple
  - nice, usual syntax
  - works well if you state full program and only use vcg
- ➜ separate program variables from Hoare triple (use extensible records), indicate usage as function syntactically
  - more syntactic overhead
  - program pieces compose nicely

Slide 23

Records in Isabelle

Records are a tuples with named components

Example:

record A = a :: nat
b :: int

- ➜ Selectors: a :: A ⇒ nat, b :: A ⇒ int, a r = Suc 0
- ➜ Constructors: (| a = Suc 0, b = -1 |)
- ➜ Update: r(| a := Suc 0 |)

Records are extensible:

record B = A +
c :: nat list

(| a = Suc 0, b = -1, c = $[0, 0]$ |)

Slide 24

Demo

MORE

Available now in Isablle:

- procedures
- with blocks and local variables
- and (mutual) recursion
- exceptions
- arrays
- pointers

Slide 25

We're working at:

- nondeterminsm
- probability
- object orientation

WE HAVE SEEN TODAY ...

- Syntax and semantics of IMP
- Hoare logic rules
- Soundness of Hoare logic
- Verification conditions
- Example program proofs

Slide 26

EXERCISES

- Write a program in IMP that calculates quotient and reminder of $x \in \mathbb{N}$ and $y \in \mathbb{N}$
- Find the right invariant for its while loop.
- Show its correctness in Isabelle:
  $\vdash \{\mathsf{True}\}\ \mathsf{program}\ \{\,\mathcal{Q} * y + \mathcal{R} = x \wedge \mathcal{R} < y\,\}$
- Write an IMP program that sorts arrays (lists) by insertion sort.
- Formulate and show its correctness in Isabelle.

Slide 27