NICTA

COMP 4161

NICTA Advanced Course

Advanced Topics in Software Verification

Gerwin Klein
Formal Methods

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

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

LAST TIME

- Calculations: also/finally
- [trans]-rules
- Code generation

FINDING THEOREMS

Command **find_theorems** (C-c C-f) finds combinations of:

- ➜ pattern: **"_ + _ + _"**
- ➜ lhs of simp rules: **simp: "_ * (_ + _)"**
- ➜ intro/elim/dest on current goal
- ➜ lemma name: **name: assoc**
- ➜ exclusions thereof: **-name: "HOL."**

Example:

find_theorems dest -"hd" name: "List."

finds all theorems in the current context that

- ➜ match the goal as dest rule,
- ➜ do not contain the constant "hd"
- ➜ are in the List theory (name starts with "List.")

ISAR: DEFINE AND DEFINES

Can define local constant in Isar proof context:

proof

. . .

define "f $\equiv$ big term"

have "g = f x" . . .

like definition, not automatically unfolded (f_def)

different to **let** ?f = "big term"

Also available in lemma statement:

lemma . . .:

fixes . . .

assumes . . .

defines . . .

shows . . .

A Crash Course in Semantics

NICTA

Commands:

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

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

NICTA

Usual syntax:

$B := 1;$

WHILE $A \neq 0$ DO

$\quad B := B * A;$

$\quad A := A - 1$

OD

Expressions are functions from state to bool or nat:

$B := (\lambda\sigma.\ 1);$

WHILE $(\lambda\sigma.\ \sigma\ A \neq 0)$ DO

$\quad B := (\lambda\sigma.\ \sigma\ B * \sigma\ A);$

$\quad A := (\lambda\sigma.\ \sigma\ A - 1)$

OD

What does it do?

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
- ➜ 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

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

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

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

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

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

Structural Operational Semantics

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

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

DEMO: THE DEFINITIONS IN ISABELLE

NICTA

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 8 implements the factorial.

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

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

Demo: Example Proof

Too tedious

Induction needed for each loop

Is there something easier?

Idea: describe meaning of program by pre/post conditions

Examples:

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

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

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

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

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)

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

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

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

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

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

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

Hoare Rules

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

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

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

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

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

Are the Rules Correct?

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

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

Demo: Hoare Logic in Isabelle