



COMP4161: Advanced Topics in Software Verification

{P} . . . {Q}

Gerwin Klein, Johannes Åman Pohjola, Christine Rizkallah, Miki Tanaka
T3/2020

Content

→ Foundations & Principles

- Intro, Lambda calculus, natural deduction [1,2]
- Higher Order Logic, Isar (part 1) [2,3^a]
- Term rewriting [3,4]

→ Proof & Specification Techniques

- Inductively defined sets, rule induction, datatype induction, primitive recursion [4,5]
- General recursive functions, termination proofs [7^b]
- Proof automation, Hoare logic, proofs about programs, invariants [8]
- C verification [9,10]
- Practice, questions, examp prep [10^c]

^aa1 due; ^ba2 due; ^ca3 due

A Crash Course in Semantics

**(For more,
see Concrete Semantics)**

IMP - a small Imperative Language

Commands:

datatype com

=

SKIP

Assign vname aexp

{ _ := _ }

Semi com com

{ _ ; _ }

Cond bexp com com

{ IF _ THEN _ ELSE _ }

While bexp com

{ WHILE _ DO _ OD }

type_synonym vname

=

string

type_synonym state

=

vname \Rightarrow nat

type_synonym aexp

=

state \Rightarrow nat

type_synonym bexp

=

state \Rightarrow bool

Example Program

Usual syntax:

```
B := 1;  
WHILE A ≠ 0 DO  
  B := B * A;  
  A := A - 1  
OD
```

Expressions are functions from state to bool or nat:

```
B := (λσ. 1);  
WHILE (λσ. σ A ≠ 0) DO  
  B := (λσ. σ B * σ A);  
  A := (λσ. σ 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

$$\overline{\langle \text{SKIP}, \sigma \rangle \rightarrow \sigma}$$

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

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

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

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

Structural Operational Semantics

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

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

Demo: The Definitions in Isabelle

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 before implements the factorial function.

lemma $\langle \text{factorial}, \sigma \rangle \rightarrow \sigma' \implies \sigma' B = \text{fac } (\sigma A)$
(where $\text{fac } 0 = 1$, $\text{fac } (\text{Suc } n) = (\text{Suc } n) * \text{fac } n$)

Demo: Example Proof

Too tedious

Induction needed for each loop

Is there something easier?

Floyd/Hoare

Idea: describe meaning of program by pre/post conditions

Examples:

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

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

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

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

Proofs: have rules that directly work on such triples

Meaning of a Hoare-Triple

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

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:

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

Total Correctness:

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

This lecture: partial correctness only (easier)

Hoare Rules

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

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

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

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

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

Hoare Rules

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

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

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

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

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

Are the Rules Correct?

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

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

Demo: Hoare Logic in Isabelle