Program verification using Hoare Logic[1]

Automated Reasoning - Guest Lecture

Petros Papapanagiotou
pe.p@ed.ac.uk

Part 1 of 2

[1] Contains material from Mike Gordon's slides: http://www.cl.cam.ac.uk/~mjcg/HL

A simple "while" programming language

- Sequence: `a ; b`
- Skip (do nothing): `SKIP`
- Variable assignment: `X := 0`
- Conditional: `IF cond THEN a ELSE b FI`
- Loop: `WHILE cond DO c OD`

Example

Given some X

```
Y := 1 ;
Z := 0 ;
WHILE Z ≠ X DO
   Z := Z + 1 ;
   Y := Y × Z
OD
```

$\{Y = X!\}$

How do you know for sure?

Formal Methods

- *Formal Specification*:
  - Use mathematical notation to give a precise description of what a program should do
- *Formal Verification*:
  - Use logical rules to mathematically prove that a program satisfies a formal specification

- *Not* a panacea:
- Formally verified programs may still not work!
- Must be combined with testing

Modern use

- Some use cases:
  - Safety-critical systems (e.g. medical software, nuclear reactor controllers, autonomous vehicles)
  - Core system components (e.g. device drivers)
  - Security (e.g. ATM software, cryptographic algorithms)
  - Hardware verification (e.g. processors)

Formal Verification

Requires programming language *semantics*

What does it mean to execute a command C?
How does it affect the State?

(State = map of memory locations to values)

Formal Verification

- **Denotational** semantics: construct *mathematical objects* that describe the meaning
  - Programs = functions: $[\![C]\!] : \mathit{State} \to \mathit{State}$
- **Operational** semantics: describe the *steps of computation* during program execution
  - Small-step (only one transition): $\langle C,\ \sigma\rangle \to \langle C',\ \sigma'\rangle$
  - Big-step (entire transition to final value): $\langle C,\ \sigma\rangle \Downarrow \sigma'$
- **Axiomatic** semantics: define *axioms and rules of some logic* of programs
  - Hoare Logic $\{P\}\ C\ \{Q\}$

Floyd-Hoare Logic and Partial Correctness Specification

By Charles Antony ("Tony") Richard Hoare with original ideas from Robert Floyd - 1969

- **Specification**: Given a state that satisfies *preconditions* P, executing a *program* C (and assuming it terminates) results in a state that satisfies *postconditions* Q
- "Hoare triple":

$$\{P\}\ C\ \{Q\}$$

e.g.:

$$\{X = 1\}\ \mathtt{X} := \mathtt{X} + \mathtt{1}\ \{X = 2\}$$

Correctness

$$\{P\}\ C\ \{Q\}$$

Partial correctness + termination = *Total* correctness

Trivial Specifications

$$\{P\}\ C\ \{\mathbf{T}\}$$

$$\{\mathbf{F}\}\ C\ \{Q\}$$

Formal specification can be tricky!

- Specification for the maximum of two variables:

$$\{\mathbf{T}\}\ C\ \{Y = max(X, Y)\}$$

- C could be:

$$\texttt{IF X} >= \texttt{Y THEN Y} := \texttt{X ELSE SKIP FI}$$

- *But* C could also be:

$$\texttt{IF X} >= \texttt{Y THEN X} := \texttt{Y ELSE SKIP FI}$$

- Or even:

$$\texttt{Y} := \texttt{X}$$

- Better use "*auxiliary*" variables (i.e. *not* program variables) x and y:

$$\{X = x \wedge Y = y\}\ C\ \{Y = max(x, y)\}$$

Hoare Logic

- A deductive proof system for Hoare triples $\{P\}\ C\ \{Q\}$
- Can be used for *verification* with forward or backward chaining
  - Conditions P and Q are described using FOL
  - *Verification Conditions* (VCs): What needs to be proven so that $\{P\}\ C\ \{Q\}$ is *true*?
  - *Proof obligations* or simply *proof subgoals*: Working our way through proving the VCs

Hoare Logic Rules

- Similar to FOL inference rules
- One for each programming language construct:
  - Assignment
  - Sequence
  - Skip
  - Conditional
  - While
- Rules of *consequence*:
  - Precondition strengthening
  - Postcondition weakening

Assignment Axiom

$$\frac{}{\{Q[E/V]\}\ \texttt{V} := \texttt{E}\ \{Q\}}$$

- Example:

$$\{X + 1 = n + 1\}\ \texttt{X} := \texttt{X} + \texttt{1}\ \{X = n + 1\}$$

- Backwards!?
  - Why not $\{P\}\ \texttt{V} := \texttt{E}\ \{P[V/E]\}$?
    - because then: $\{X = 0\}\ \texttt{X} := \texttt{1}\ \{X = 0\}$
  - Why not $\{P\}\ \texttt{V} := \texttt{E}\ \{P[E/V]\}$?
    - because then: $\{X = 0\}\ \texttt{X} := \texttt{1}\ \{1 = 0\}$

Sequencing Rule

$$\frac{\{P\}\ C_1\ \{Q\} \quad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1\ ;\ C_2\ \{R\}}$$

- Example (`Swap X Y`): `S := X ; X := Y ; Y := S`

$$\frac{}{\{X = x \wedge Y = y\}\ \mathtt{S} := \mathtt{X}\ \{S = x \wedge Y = y\}} \quad (1)$$

$$\frac{}{\{S = x \wedge Y = y\}\ \mathtt{X} := \mathtt{Y}\ \{S = x \wedge X = y\}} \quad (2)$$

$$\frac{}{\{S = x \wedge X = y\}\ \mathtt{Y} := \mathtt{S}\ \{Y = x \wedge X = y\}} \quad (3)$$

$$\frac{\dfrac{(1) \qquad (2)}{\{X = x \wedge Y = y\}\ \mathtt{S} := \mathtt{X}\ ;\ \mathtt{X} := \mathtt{Y}\ \{S = x \wedge X = y\}} \qquad (3)}{\{X = x \wedge Y = y\}\ \mathtt{S} := \mathtt{X}\ ;\ \mathtt{X} := \mathtt{Y}\ ;\ \mathtt{Y} := \mathtt{S}\ \{Y = x \wedge X = y\}}$$

Skip Axiom

$$\frac{}{\{P\}\ \texttt{SKIP}\ \{P\}}$$

Conditional Rule

$$\frac{\{P \wedge S\}\ C_1\ \{Q\} \quad \{P \wedge \neg S\}\ C_2\ \{Q\}}{\{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2\ \texttt{FI}\ \{Q\}}$$

- Example (`Max X Y`):

$$\frac{\overline{\{X \geq X \wedge X \geq Y\}\ \texttt{MAX} := \texttt{X}\ \{MAX \geq X \wedge MAX \geq Y\}}}{\{\mathbf{T} \wedge X \geq Y\}\ \texttt{MAX} := \texttt{X}\ \{MAX \geq X \wedge MAX \geq Y\}} \tag{4}$$

$$\frac{\overline{\{Y \geq X \wedge Y \geq Y\}\ \texttt{MAX} := \texttt{Y}\ \{MAX \geq X \wedge MAX \geq Y\}}}{\{\mathbf{T} \wedge \neg(X \geq Y)\}\ \texttt{MAX} := \texttt{Y}\ \{MAX \geq X \wedge MAX \geq Y\}} \tag{5}$$

$$\frac{(4) \qquad (5)}{\{\mathbf{T}\}\ \texttt{IF X} \geq \texttt{Y THEN MAX} := \texttt{X ELSE MAX} := \texttt{Y FI}\ \{MAX \geq X \wedge MAX \geq Y\}} \tag{6}$$

Summary

- *Formal Verification*: Use logical rules to mathematically prove that a program satisfies a formal specification
- Programing language semantics
  - *denotational, operational, axiomatic*
- Specification using *Hoare triples* $\{P\}\ C\ \{Q\}$
  - Preconditions P
  - Program C
  - Postconditions Q
- *Hoare Logic*: A deductive proof system for Hoare triples
- Logical Rules:
  - One for each program construct
- *Partial* correctness + termination = *Total* correctness

Next

- Precondition strengthening
- Postcondition weakening
- WHILE loops + invariants

To be continued...

Recommended reading

Theory:

- Mike Gordon, *Background Reading on Hoare Logic*, `http://www.cl.cam.ac.uk/~mjcg/Teaching/2011/Hoare/Notes/Notes.pdf` (pp. 1-27, 37-48)
- Huth & Ryan, Sections 4.1-4.3 (pp. 256-292)
- Nipkow & Klein, Section 12.2.1 (pp. 191-199)

Practice:

- Isabelle's Hoare Logic library: `http://isabelle.in.tum.de/dist/library/HOL/HOL-Hoare`
- Tutorial exercise