Program verification using Hoare Logic[1]

Automated Reasoning - Guest Lecture

Petros Papapanagiotou

pe.p@ed.ac.uk

25 Oct 2016

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

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)

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

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`

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:

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

- *But* C could also be:

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

- Or even:

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

- 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 to extract *verification conditions* (VCs)
  - Conditions P and Q are described using FOL
  - VCs = What needs to be proven so that $\{P\}\ C\ \{Q\}$ is *true*?
  - *Proof obligations* or simply *proof subgoals*

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

$$\overline{\{Q[E/V]\}\ \mathtt{V} := \mathtt{E}\ \{Q\}}$$

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

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

- How can we get the following?

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

Precondition Strengthening

$$\frac{P \longrightarrow P' \quad \{P'\}\ C\ \{Q\}}{\{P\}\ C\ \{Q\}}$$

- Replace a *precondition* with a stronger condition
- Example:

$$\frac{X = n \longrightarrow X + 1 = n + 1 \quad \overline{\{X + 1 = n + 1\}\ \texttt{X := X + 1}\ \{X = n + 1\}}}{\{X = n\}\ \texttt{X := X + 1}\ \{X = n + 1\}}$$

Postcondition Weakening

$$\frac{\{P\}\ C\ \{Q'\} \quad Q' \longrightarrow Q}{\{P\}\ C\ \{Q\}}$$

- Replace a *postcondition* with a weaker condition
- Example:

$$\frac{\{X = n\}\ \texttt{X} := \texttt{X} + \texttt{1}\ \{X = n + 1\} \quad X = n + 1 \longrightarrow X > n}{\{X = n\}\ \texttt{X} := \texttt{X} + \texttt{1}\ \{X > n\}}$$

Sequencing Rule

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

- Example (`Swap X Y`):

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

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

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

$$\frac{\dfrac{(1) \qquad (2)}{\{X = x \land Y = y\}\ \mathtt{S := X\ ;\ X := Y}\ \{S = x \land X = y\}} \qquad (3)}{\{X = x \land Y = y\}\ \mathtt{S := X\ ;\ X := Y\ ;\ Y := S}\ \{Y = x \land X = y\}} \quad (4)$$

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`):

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

$$\frac{\mathbf{T} \wedge \neg(X \geq Y) \longrightarrow Y = max(X,Y) \quad \overline{\{Y := max(X,Y)\}\ \texttt{MAX} := \texttt{Y}\ \{MAX = max(X,Y)\}}}{\{\mathbf{T} \wedge \neg(X \geq Y)\}\ \texttt{MAX} := \texttt{Y}\ \{MAX = max(X,Y)\}} \tag{6}$$

$$\frac{(5) \qquad\qquad (6)}{\{\mathbf{T}\}\ \texttt{IF X} \geq \texttt{Y THEN MAX} := \texttt{X ELSE MAX} := \texttt{Y FI}\ \{MAX = max(X,Y)\}} \tag{7}$$

Conditional Rule - VCs

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

- Example (`Max X Y`):

  $\{\mathbf{T}\}\ \texttt{IF X} \geq \texttt{Y THEN MAX} := \texttt{X ELSE MAX} := \texttt{Y FI}\ \{MAX = max(X, Y)\}$

- FOL VCs:

  $$\mathbf{T} \land X \geq Y \longrightarrow X = max(X, Y)$$
  $$\mathbf{T} \land \neg(X \geq Y) \longrightarrow Y = max(X, Y)$$

- Hoare Logic rules can be applied *automatically* to generate VCs
  - e.g. Isabelle's `vcg` tactic
- We need to provide proofs for the VCs / *proof obligations*

WHILE Rule

$$\frac{\{P \wedge S\}\ C\ \{P\}}{\{P\}\ \texttt{WHILE}\ S\ \texttt{DO}\ C\ \texttt{OD}\ \{P \wedge \neg S\}}$$

- P is an *invariant* for C whenever S holds
- *WHILE rule*: If executing C *once* preserves the truth of P, then executing C *any number of times* also preserves the truth of P
- If P is an invariant for C when S holds then P is an invariant of the *whole* WHILE loop, i.e. a *loop invariant*

WHILE Rule

$$\frac{\{P \wedge S\}\ C\ \{P\}}{\{P\}\ \texttt{WHILE}\ S\ \texttt{DO}\ C\ \texttt{OD}\ \{P \wedge \neg S\}}$$

- Example (factorial) - Original specification:

$$\begin{array}{lcl}
\{Y = 1 \wedge Z = 0\} & & \{P\} \\
\texttt{WHILE Z} \neq \texttt{X DO} & & \texttt{WHILE Z} \neq \texttt{X DO} \\
\quad \texttt{Z := Z + 1 ;} & \overset{?}{\leadsto} & \quad \texttt{Z := Z + 1 ;} \\
\quad \texttt{Y := Y} \times \texttt{Z} & & \quad \texttt{Y := Y} \times \texttt{Z} \\
\texttt{OD} & & \texttt{OD} \\
\{Y = X!\} & & \{P \wedge \neg Z \neq X\}
\end{array}$$

- What is P?

WHILE Rule - How to find an invariant

$$\frac{\{P \wedge S\}\ C\ \{P\}}{\{P\}\ \texttt{WHILE}\ S\ \texttt{DO}\ C\ \texttt{OD}\ \{P \wedge \neg S\}}$$

- The invariant P should:
  - Say what *has been done so far* together with what *remains to be done*
  - Hold *at each iteration* of the loop.
  - Give the *desired result* when the loop terminates

WHILE Rule - Invariant VCs

$$\frac{\{P \wedge S\}\ C\ \{P\}}{\{P\}\ \texttt{WHILE}\ S\ \texttt{DO}\ C\ \texttt{OD}\ \{P \wedge \neg S\}}$$

$$\{\mathbf{Y} = \mathbf{1} \wedge \mathbf{Z} = \mathbf{0}\}\ \texttt{WHILE Z} \neq \texttt{X DO Z} := \texttt{Z} + \texttt{1 ; Y} := \texttt{Y} \times \texttt{Z OD}\ \{\mathbf{Y} = \mathbf{X}!\}$$

$$\{P\}\ \texttt{WHILE Z} \neq \texttt{X DO Z} := \texttt{Z} + \texttt{1 ; Y} := \texttt{Y} \times \texttt{Z OD}\ \{P \wedge \neg Z \neq X\}$$

- We need to find an invariant P such that:
  - $\{P \wedge Z \neq X\}$ $\texttt{Z} := \texttt{Z} + \texttt{1 ; Y} := \texttt{Y} \times \texttt{Z}$ $\{P\}$ (WHILE rule)
  - $Y = 1 \wedge Z = 0 \longrightarrow P$ (precondition strengthening)
  - $P \wedge \neg(Z \neq X) \longrightarrow Y = X!$ (postcondition weakening)

WHILE Rule - Loop invariant for factorial

$$\{\mathbf{Y} = \mathbf{1} \land \mathbf{Z} = \mathbf{0}\}\ \texttt{WHILE Z} \neq \texttt{X DO Z} := \texttt{Z} + \texttt{1}\ ;\ \texttt{Y} := \texttt{Y} \times \texttt{Z OD}\ \{\mathbf{Y} = \mathbf{X}!\}$$

- We need to find an invariant P such that:
  - $\{P \land Z \neq X\}\ \texttt{Z} := \texttt{Z} + \texttt{1}\ ;\ \texttt{Y} := \texttt{Y} \times \texttt{Z}\ \{P\}$
  - $Y = 1 \land Z = 0 \longrightarrow P$
  - $P \land \neg(Z \neq X) \longrightarrow Y = X!$
- $\mathbf{Y} = \mathbf{Z}!$
- VCs:
  - $\{Y = Z! \land Z \neq X\}\ \texttt{Z} := \texttt{Z} + \texttt{1}\ ;\ \texttt{Y} := \texttt{Y} \times \texttt{Z}\ \{Y = Z!\}$ because: $Y = Z! \land Z \neq X \longrightarrow \mathbf{Y} \times (\mathbf{Z} + \mathbf{1}) = (\mathbf{Z} + \mathbf{1})!$ $(*)$
  - $Y = 1 \land Z = 0 \longrightarrow Y = Z!$ because: $0! = 1$
  - $Y = Z! \land \neg(Z \neq X) \longrightarrow Y = X!$ because: $\neg(Z \neq X) \leftrightarrow Z = X$

$$\frac{\overline{\{\mathbf{Y} \times (\mathbf{Z} + \mathbf{1}) = (\mathbf{Z} + \mathbf{1})!\}\ \texttt{Z} := \texttt{Z} + \texttt{1}\ \{Y \times Z = Z!\}} \qquad \overline{\{Y \times Z = Z!\}\ \texttt{Y} := \texttt{Y} \times \texttt{Z}\ \{Y = Z!\}}}{\{\mathbf{Y} \times (\mathbf{Z} + \mathbf{1}) = (\mathbf{Z} + \mathbf{1})!\}\ \texttt{Z} := \texttt{Z} + \texttt{1}\ ;\ \texttt{Y} := \texttt{Y} \times \texttt{Z}\ \{Y = Z!\}}\ (*)$$

WHILE Rule - Complete factorial example

$$\{\mathbf{Y = 1 \wedge Z = 0}\}$$
$$\{Y = Z!\}$$

```
WHILE Z ≠ X DO
```

$$\{Y = Z! \wedge Z \neq X\}$$
$$\{Y \times (Z+1) = (Z+1)!\}$$

```
   Z := Z + 1 ;
```

$$\{Y \times Z = Z!\}$$

```
   Y := Y × Z
```

$$\{Y = Z!\}$$

```
OD
```

$$\{Y = Z! \wedge \neg(Z \neq X)\}$$
$$\{\mathbf{Y = X}!\}$$

Hoare Logic Rules (it does!)

$$\frac{P \longrightarrow P' \quad \{P'\}\ C\ \{Q\}}{\{P\}\ C\ \{Q\}} \qquad \frac{\{P\}\ C\ \{Q'\} \quad Q' \longrightarrow Q}{\{P\}\ C\ \{Q\}}$$

$$\frac{}{\{Q[E/V]\}\ \mathtt{V} := \mathtt{E}\ \{Q\}} \qquad \frac{}{\{P\}\ \mathtt{SKIP}\ \{P\}}$$

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

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

$$\frac{\{P \wedge S\}\ C\ \{P\}}{\{P\}\ \mathtt{WHILE}\ S\ \mathtt{DO}\ C\ \mathtt{OD}\ \{P \wedge \neg S\}}$$

Other topics

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

- Weakest preconditions, strongest postconditions

Other topics

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

- Meta-theory: Is Hoare logic...
  - ... *sound*? - Yes! Based on programming language semantics (but what about more complex languages?)
  - ... *decidable*? - No! $\{\mathbf{T}\}\ C\ \{\mathbf{F}\}$ is the halting problem!
  - ... *complete*? - *Relatively* / only for simple languages

Other topics

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

- Automatic Verification Condition Generation (VCG)
- Automatic generation/inference of loop invariants!
- More complex languages - e.g. Pointers = Separation logic
- Functional programming (recursion = induction)

Summary

- *Formal Verification*: Use logical rules to mathematically prove that a program satisfies a formal specification
- 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
  - Precondition strengthening
  - Postcondition weakening
- Automated generation of *Verification Conditions* (VCs)
- Only one problem: *Loop invariants*!
  - Properties that hold during *while* loops
  - Loop invariant generation is generally *undecidable*
- *Partial* correctness + termination = *Total* correctness

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