Programming Languages and Compiler Design

Axiomatic Semantics - Hoare Logic

Yliès Falcone, Jean-Claude Fernandez

Master of Sciences in Informatics at Grenoble (MoSIG)
Master 1 info

Univ. Grenoble Alpes
(Université Joseph Fourier, Grenoble INP)

Academic Year 2015 - 2016

Outline - Axiomatic Semantics - Hoare Logic

Outline - Axiomatic Semantics - Hoare Logic

Outline - Axiomatic Semantics - Hoare Logic

Partial correctness and total correctness

Goal: specify an " **input** / **output** " relationship that the *program must satisfy*.

Example (Program Fact)

$y := 1;$
while $\neg(x = 1)$ do $(y := y * x; x := x - 1)$ od

- **Partial Correctness:**
  If the initial value of x is $n > 0$ *and if* the program terminates **then** the final value of y is $n!$
- **Total Correctness:**
  If the initial value of x is $n > 0$ **then** the program terminates *and* the final value of y is $n!$

Outline - Axiomatic Semantics - Hoare Logic

Verifying semantic properties - a motivating example

How can we prove the (partial) correctness of Fact using NOS?

Fact:
$y := 1;$
while $\neg(x = 1)$ do $(y := y * x; x := x - 1)$ od

Formalization:

$$(\text{Fact}, \sigma) \rightarrow \sigma' \text{ implies } \sigma'(y) = \sigma(x)! \text{ and } \sigma(x) > 0$$

Stage 1 Correctness of the loop body.
Stage 2 Correctness of the loop.
Stage 3 Correction of the program.
$\hookrightarrow$ Study of the derivation tree.

Verifying semantic properties - a motivating example (ctd)

Correctness of Fact

Stage 1 The loop body satisfies:

$$\begin{array}{rll} \text{if} & (y := y * x; x := x - 1, \sigma) \rightarrow \sigma'' & \text{and } \sigma(x)'' > 0 \\ \text{then} & \sigma(y) * \sigma(x)! = \sigma''(y) * \sigma''(x)! & \text{and } \sigma(x) > 0. \end{array}$$

Stage 2 The loop satisfies:

$$\begin{array}{rl} \text{if} & (\text{while } \neg(x = 1) \text{ do } y := y * x; x := x - 1 \text{ od }, \sigma) \rightarrow \sigma' \\ \text{then} & \sigma(y) * \sigma(x)! = \sigma'(y) \text{ and } \sigma'(x) = 1 \text{ and } \sigma(x) > 0. \end{array}$$

Stage 3 Partial correctness of the program:

$$\begin{array}{rll} \text{if} & (\text{Fact}, \sigma) \rightarrow \sigma' & \\ \text{then} & \sigma'(y) = \sigma(x)! & \text{and } \sigma(x) > 0. \end{array}$$

Lessons learned from the motivating example

Proving semantic properties about programs *could* be done using OS.

But: This does not scale:

- tedious,
- long,
- not practical,
- too closely connected to the semantics.

We want to focus on the essential properties we want to prove.

We will exhibit the essential properties of the language constructs.

Outline - Axiomatic Semantics - Hoare Logic

Outline - Axiomatic Semantics - Hoare Logic

Hoare Triple on a example

Example (Program Fact)

$$\{x = n \land n > 0\}$$
$$y := 1;$$
$$\text{while } \neg(x = 1) \text{ do } (y := y * x; x := x - 1) \text{ od}$$
$$\{y = n! \land n > 0\}$$

- Precondition: $\{x = n \land n > 0\}$
- Post-condition: $\{y = n! \land n > 0\}$.

We will use an assertion language to specify pre and post conditions.

Hoare Triple

Idea: specify properties of programs as assertions (i.e., "claims").

Definition (Hoare Triple - Assertion)

$$\{P\}\ S\ \{Q\}$$

- S a statement
- P a pre-condition
- Q a post-condition

Meaning:

if P holds in the initial state (before executing S),
if the execution of S on that state *terminates*,
then Q will holds in the state in which S terminates.

We say that $\{P\}\ \ S\ \ \{Q\}$ holds.

Remark It is not necessary that S terminates if P is satisfied. □

Outline - Axiomatic Semantics - Hoare Logic

The assertion language

Example (Program Fact)

$\{x = n \wedge n > 0\}$
$y := 1;$
while $\neg(x = 1)$ do $(y := y * x; x := x - 1)$ od
$\{y = n! \wedge n > 0\}$

Remark Specifying an "input/output" relation, we could not replace $\{y = n! \wedge n > 0\}$ by $\{y = x! \wedge x > 0\}$ □

n is a logical variable:

- must not appear in the program,
- used to "remember the initial values of program variables".

Two kinds of variables:

- program variables (**Var**),
- logical variables.

The assertion language: predicates

Intuition: a Boolean expression b defines a predicate $\mathcal{B}[b] : \textbf{State} \to \{\textbf{tt}, \textbf{ff}\}$

Definition (Predicate)

A predicate is a function from **State** to $\{\textbf{tt}, \textbf{ff}\}$ described using the syntactic category **Bexp** extended with logical variables.

For a predicate P, we note $P(\sigma) \in \{\textbf{tt}, \textbf{ff}\}$ the *evaluation* of P on σ.

Example (Predicate)

- $P_1 \equiv x = n$
- $P_2 \equiv n > 0 \wedge x = n!$
- $P_2' \equiv x = n \wedge y = n!$

- $\sigma_1 = [x \mapsto n]$
- $\sigma_2 = [x \mapsto n + 1]$
- $\sigma_3 = [x \mapsto 3, y \mapsto 6]$

- $P_1(\sigma_1) = \textbf{tt}$
- $P_2(\sigma_2) = \textbf{ff}$

$$P_2'(\sigma_3) = \begin{cases} \textbf{tt} & \text{if } n = 3 \\ \textbf{ff} & \text{otherwise} \end{cases}$$

The assertion language: predicates (ctd)

Notations:

- $\mathbf{P_1} \wedge \mathbf{P_2}$: $(P_1 \wedge P_2)(\sigma) = P_1(\sigma)$ and $P_2(\sigma)$
- $\mathbf{P_1} \vee \mathbf{P_2}$: $(P_1 \vee P_2)(\sigma) = P_1(\sigma)$ or $P_2(\sigma)$
- $\neg\mathbf{P}$: $(\neg P)(\sigma) = \neg(P(\sigma))$
- $\mathbf{P[a/x]}$: P where each occurrence of x is replaced by a
- $\mathbf{P_1} \Rightarrow \mathbf{P_2}$: $\forall \sigma \in \mathbf{State} : P_1(\sigma)$ implies $P_2(\sigma)$

Example (Predicate)

Recall that $P_1 \equiv x = n$ and $P_2 \equiv x = n!$:

- $(P_1 \wedge P_2)([x \mapsto 2]) = \mathbf{tt}$
- $(P_1 \wedge P_2)([x \mapsto 4]) = \mathbf{ff}$
- $(P_1 \wedge P_2')([x \mapsto 3, y \mapsto 6] = \mathbf{tt}$
- $(P_1 \wedge P_2')([x \mapsto 3, y \mapsto 8]) = \mathbf{ff}$

Outline - Axiomatic Semantics - Hoare Logic

The inference system - Hoare Calculus

Partial correctness assertions will be specified by an inference system (axioms and rules).

(Similarly to inference trees in Natural Operational Semantics).

Formulae are of the form:

$$\{P\}\ S\ \{Q\}$$

- $S \in \mathbf{Stm}$: a statement in language **While**.
- $\{P\}$ and $\{Q\}$ are predicates.

The inference system: axioms (schemes)

Definition (Axioms)

$$\{P\}\ \text{skip}\ \{P\}$$

$$\{P[a/x]\}\ x := a\ \{P\}$$

"Schemes" that need to be instantiated for a particular choice of P.

The inference system: inference rules

Deducing assertion about compound from assertions about constituents.

Definition (Inference Rules)

Compositional statements:

$$\frac{\{P\}\ S_1\ \{Q\} \qquad \{Q\}\ S_2\ \{R\}}{\{P\}\ S_1; S_2\ \{R\}}$$

Conditional statements:

$$\frac{\{b \wedge P\}\ S_1\ \{Q\} \quad \{\neg b \wedge P\}\ S_2\ \{Q\}}{\{P\}\ \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}\ \{Q\}}$$

Iterative statements:

$$\frac{\{b \wedge P\}\ S\ \{P\}}{\{P\}\ \text{while } b\ \text{ do } S \text{ od}\ \{\neg b \wedge P\}}$$

Consequence: If $P \Rightarrow P'$ and $Q' \Rightarrow Q$, then:

$$\frac{\{P'\}\ S\ \{Q'\}}{\{P\}\ S\ \{Q\}}$$

Comparison with Natural Operational Semantics

We have defined a set of rules and axioms.

Natural Operational Semantics	Axiomatic Semantics
Axioms	Axioms
Inference rules	Inference rules
Derivation trees = description/proof of a computation expressed at the root	Inference trees = proof of a property expressed at the root
Leaves = Instance of axioms	Leaves = Instance of axioms
Internal nodes = instances of rules	Internal nodes = instances of rules

Proving properties using the inference system

An inference tree gives a proof of the property expressed at its root.

Notation

When inferring $\{P\}\ S\ \{Q\}$ (with rules and axioms) we note:

$$\vdash \{P\}\ S\ \{Q\}$$

Example (Proving properties)

- $\vdash \{x = 0\}\ x := x + 1; x := x + 1\ \{x = 2\}$
- $\vdash \{x > 0\}\ y := 1\ \{x = x * y\}$

Exercise: a proof

Prove that

$$\vdash \{\texttt{True}\}\ \text{while } \mathit{true}\ \text{ do skip od}\ \{\texttt{True}\}$$

where $\forall \sigma \in \textbf{State} : \texttt{True}(\sigma) = \textbf{tt}$

Outline - Axiomatic Semantics - Hoare Logic

Properties of the semantics

Definition (Semantic equivalence between programs)

S_1 and S_2 are provably equivalent according to the axiomatic semantics if

- for all pre-conditions P,
- for all post-conditions Q:

$$\vdash \{P\}\ S_1\ \{Q\} \text{ iff } \vdash \{P\}\ S_2\ \{Q\}$$

Proving a property of the axiomatic semantics:

Induction on the shape of Inference trees

In order to prove a given property Prop for all inference trees:

- *Prove* Prop holds for all simple trees, i.e., axioms
- *Prove* Prop holds for all composite inference trees.
  For each rule:
  - *Assume* Prop holds for its premises
    $\hookrightarrow$ **Induction Hypothesis**
  - *Assume* the conditions of the rule are satisfied
  - *Prove* Prop holds for the conclusion

Soundness and completeness of Hoare logic

Definition (Validity of a Hoare triple)

The triple $\{P\}\ S\ \{Q\}$ is valid, noted

$$\models \{P\}\ S\ \{Q\}$$

iff for all states σ, σ':

- If $P(\sigma)$ and $(S, \sigma) \to \sigma'$
- then $Q(\sigma')$.

We say that S is partially correct wrt. P and Q.

Correctness (We can infer *only* valid triples)

$$\text{If } \vdash \{P\}\ S\ \{Q\} \text{ then } \models \{P\}\ S\ \{Q\}$$

Completeness (We can infer *all* valid triples)

$$\text{If } \models \{P\}\ S\ \{Q\} \text{ then } \vdash \{P\}\ S\ \{Q\}$$

Soundness of Hoare logic

Proof by induction on the shape of the inference tree to infer $\{P\}\ S\ \{Q\}$.

[ass] Suppose $(x := a, \sigma) \to \sigma'$ and $P[a/x](\sigma) = \mathbf{tt}$.
$[\mathrm{ass}^{\mathrm{ns}}]$ gives $\sigma' = \sigma[x \mapsto \mathcal{A}[a]\sigma]$.
$P(\sigma') = \mathbf{tt}$ (from correctness of substitution).

[skip] Straightforward.

[comp] Suppose $(S_1; S_2, \sigma) \to \sigma''$, $\models \{P\}\ S_1\ \{Q\}, \{Q\}\ S_2\ \{R\}$, and $P(\sigma) = \mathbf{tt}$.
$[\mathrm{comp}^{\mathrm{ns}}]$ gives $(S_1, \sigma) \to \sigma'$ and $(S_2, \sigma') \to \sigma''$.
From $(S_1, \sigma) \to \sigma'$ and $\models \{P\}\ S_1\ \{Q\}$, we get $Q(\sigma') = \mathbf{tt}$.
From $(S_2, \sigma') \to \sigma''$ and $\models \{Q\}\ S_2\ \{R\}$, we get $R(\sigma'') = \mathbf{tt}$.

[if] Suppose (if b then S_1 else S_2 fi$, \sigma) \to \sigma'$, $\models \{b \wedge P\}\ S_1\ \{Q\}$ and $\models \{\neg b \wedge P\}\ S_2\ \{Q\}$.
Two cases:

- $\mathcal{B}[b]\sigma = \mathbf{tt}$ then $(P \wedge b)(\sigma) = \mathbf{tt}$.
  $[\mathrm{if}_{\mathrm{ns}}]$ gives $(S_1, \sigma) \to \sigma'$.
  $\models \{b \wedge P\}\ S_1\ \{Q\}$ gives $Q(\sigma') = \mathbf{tt}$.
- $\mathcal{B}[b]\sigma = \mathbf{ff}$. Similar.

Soundness of Hoare logic

Proof by induction on the shape of the inference tree to infer $\{P\}\ S\ \{Q\}$

[while] Suppose (while b do S od $, \sigma) \to \sigma''$ and $\models \{b \wedge P\}\ S\ \{P\}$
(We want to prove $\models \{P\}$ while b do S od $\{\neg b \wedge P\}$)
Two cases:

- $\mathcal{B}[b]\sigma = \mathbf{tt}$ then $(S, \sigma) \to \sigma'$ and (while b do S od $, \sigma') \to \sigma''$
  $(b \wedge P)(\sigma) = \mathbf{tt}$ and $\models \{b \wedge P\}\ S\ \{P\}$ gives $P(\sigma') = \mathbf{tt}$.
  IH on (while b do S od $, \sigma') \to \sigma''$ gives $(\neg b \wedge P)(\sigma'') = \mathbf{tt}$.
- $\mathcal{B}[b]\sigma = \mathbf{ff}$ then $\sigma' = \sigma''$ and $(\neg b \wedge P)(\sigma'') = \mathbf{tt}$.

[cons] Suppose $\models \{P'\}\ S\ \{Q'\}$, $P \Rightarrow P'$ and $Q' \Rightarrow Q$.
Suppose $(S, \sigma) \to \sigma'$ and $P(\sigma) = \mathbf{tt}$.
From $P(\sigma) = \mathbf{tt}$ and $P \Rightarrow P'$, we get $P'(\sigma)$.
From $P'(\sigma) = \mathbf{tt}$ and $\models \{P'\}\ S\ \{Q'\}$ we get $Q'(\sigma')$.
From $Q'(\sigma') = \mathbf{tt}$ and $Q' \Rightarrow Q$, we get $Q(\sigma')$.

Outline - Axiomatic Semantics - Hoare Logic

Total correctness assertions

Partial vs Total correctness.

Triples of the form:

$$\{P\}\ S\ \{\Downarrow Q\}$$

if the precondition P is fulfilled
then (S is guaranteed to terminate ($\Downarrow$)
and the final state will satisfy the post-condition Q)

Inference of a triple

$$\vdash \{P\}\ S\ \{\Downarrow Q\}$$

Validity of Hoare triples

$$\models \{P\}\ S\ \{\Downarrow Q\}$$

iff $\forall \sigma \in \textbf{State} : P(\sigma)$ implies $\exists \sigma' \in \textbf{State} : \begin{cases} Q(\sigma') = \textbf{tt} \\ (S, \sigma) \to \sigma' \end{cases}$

The inference system: axioms (schemes)

Definition (Axioms)

$$\{P\}\ \text{skip}\ \{\Downarrow P\}$$

$$\{P[a/x]\}\ x := a\ \{\Downarrow P\}$$

"Schemes" that need to be instantiated for a particular choice of P.

The inference system: inference rules

Definition (Inference Rules)

Compositional statements:

$$\frac{\{P\}\ S_1\ \{\Downarrow Q\} \qquad \{Q\}\ S_2\ \{\Downarrow R\}}{\{P\}\ S_1; S_2\ \{\Downarrow R\}}$$

Conditional statements:

$$\frac{\{b \wedge P\}\ S_1\ \{\Downarrow Q\} \quad \{\neg b \wedge P\}\ S_2\ \{\Downarrow Q\}}{\{P\}\ \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi}\ \{\Downarrow Q\}}$$

Iterative statements:

$$\frac{\{P(z+1)\}\ S\ \{\Downarrow P(z)\}}{\{\exists z \in \mathbb{N}. P(z)\}\ \text{while } b\ \text{ do } S \text{ od }\ \{\Downarrow P(0)\}}$$

where

- $P(z+1) \Rightarrow \mathcal{B}[b]$
- $P(0) \Rightarrow \neg\mathcal{B}[b]$

Consequence: If $P \Rightarrow P'$ and $Q' \Rightarrow Q$, then:

$$\frac{\{P'\}\ S\ \{\Downarrow Q'\}}{\{P\}\ S\ \{\Downarrow Q\}}$$

Outline - Axiomatic Semantics - Hoare Logic

Summary

Axiomatic Semantics

- Focus on the essential properties.
- Hoare triple.
- Hoare calculus - inference system.
- Soundness and completeness of Hoare logic.
- Partial vs total correctness of programs.