Semantics and Verification of Software

Summer Semester 2015

Lecture 12: Axiomatic Semantics of WHILE IV (Axiomatic Equivalence)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

`http://moves.rwth-aachen.de/teaching/ss-15/sv-sw/`

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Hoare Logic

Goal: syntactic derivation of valid partial correctness properties.
Here $A[x \mapsto a]$ denotes the syntactic replacement of every occurrence of x by a in A.

Tony Hoare (* 1934)

Definition (Hoare Logic)

The Hoare rules are given by

$$\text{(skip)}\ \frac{}{\{A\}\ \texttt{skip}\ \{A\}} \qquad \text{(asgn)}\ \frac{}{\{A[x \mapsto a]\}\ x := a\ \{A\}}$$

$$\text{(seq)}\ \frac{\{A\}\ c_1\ \{C\} \quad \{C\}\ c_2\ \{B\}}{\{A\}\ c_1 ; c_2\ \{B\}} \qquad \text{(if)}\ \frac{\{A \wedge b\}\ c_1\ \{B\} \quad \{A \wedge \neg b\}\ c_2\ \{B\}}{\{A\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}\ \{B\}}$$

$$\text{(while)}\ \frac{\{A \wedge b\}\ c\ \{A\}}{\{A\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{A \wedge \neg b\}} \qquad \text{(cons)}\ \frac{\models (A \Rightarrow A') \quad \{A'\}\ c\ \{B'\} \quad \models (B' \Rightarrow B)}{\{A\}\ c\ \{B\}}$$

A partial correctness property is provable (notation: $\vdash \{A\}\ c\ \{B\}$) if it is derivable by the Hoare rules. In (while), A is called a (loop) invariant.

Recap: Partial & Total Correctness Properties

Proving Total Correctness

Goal: syntactic derivation of valid total correctness properties

Definition (Hoare Logic for total correctness)

The Hoare rules for total correctness are given by (where $i \in LVar$)

$$\text{(skip)}\ \frac{}{\{A\}\ \mathtt{skip}\ \{\Downarrow A\}} \qquad \text{(asgn)}\ \frac{}{\{A[x \mapsto a]\}\ x := a\ \{\Downarrow A\}}$$

$$\text{(seq)}\ \frac{\{A\}\ c_1\ \{\Downarrow C\} \quad \{C\}\ c_2\ \{\Downarrow B\}}{\{A\}\ c_1 ; c_2\ \{\Downarrow B\}} \qquad \text{(if)}\ \frac{\{A \wedge b\}\ c_1\ \{\Downarrow B\} \quad \{A \wedge \neg b\}\ c_2\ \{\Downarrow B\}}{\{A\}\ \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\ \mathtt{end}\ \{\Downarrow B\}}$$

$$\text{(while)}\ \frac{\models (i \geq 0 \wedge A(i+1) \Rightarrow b) \quad \{i \geq 0 \wedge A(i+1)\}\ c\ \{\Downarrow A(i)\} \quad \models (A(0) \Rightarrow \neg b)}{\{\exists i. i \geq 0 \wedge A(i)\}\ \mathtt{while}\ b\ \mathtt{do}\ c\ \mathtt{end}\ \{\Downarrow A(0)\}}$$

$$\text{(cons)}\ \frac{\models (A \Rightarrow A') \quad \{A'\}\ c\ \{\Downarrow B'\} \quad \models (B' \Rightarrow B)}{\{A\}\ c\ \{\Downarrow B\}}$$

A total correctness property is provable (notation: $\vdash \{A\}\ c\ \{\Downarrow B\}$) if it is derivable by the Hoare rules. In case of (while), $A(i)$ is called a (loop) invariant.

Operational and Denotational Equivalence

Definition 4.1: $\mathfrak{O}[\![.]\!] : Cmd \to (\Sigma \dashrightarrow \Sigma)$ given by

$$\mathfrak{O}[\![c]\!]\sigma = \sigma' \iff \langle c, \sigma \rangle \to \sigma'$$

Definition 4.2: Two statements $c_1, c_2 \in Cmd$ are operationally equivalent (notation: $c_1 \sim c_2$) if

$$\mathfrak{O}[\![c_1]\!] = \mathfrak{O}[\![c_2]\!]$$

Theorem 8.5: For every $c \in Cmd$,

$$\mathfrak{O}[\![c]\!] = \mathfrak{C}[\![c]\!]$$

Axiomatic Equivalence I

In the axiomatic semantics, two statements have to be considered equivalent if they are indistinguishable w.r.t. partial correctness properties:

Definition 12.1 (Axiomatic equivalence)

Two statements $c_1, c_2 \in Cmd$ are called axiomatically equivalent (notation: $c_1 \approx c_2$) if, for all assertions $A, B \in Assn$,

$$\models \{A\}\, c_1\, \{B\} \iff \models \{A\}\, c_2\, \{B\}.$$

Axiomatic Equivalence II

Example 12.2

We show that $\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end} \approx \texttt{if}\ b\ \texttt{then}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}$ (cf. Lemma 4.3). Let $A, B \in \mathit{Assn}$:

$$
\begin{array}{ll}
 & \models \{A\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{B\} \\
\iff & \vdash \{A\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{B\} \quad \text{(Theorem 10.2, 10.5)} \\
\iff & \text{ex. } C \in \mathit{Assn} \text{ such that } \models (A \Rightarrow C), \models (C \wedge \neg b \Rightarrow B), \\
 & \vdash \{C\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{C \wedge \neg b\} \quad \text{(rule (cons))} \\
\iff & \text{ex. } C \in \mathit{Assn} \text{ such that } \models (A \Rightarrow C), \models (C \wedge \neg b \Rightarrow B), \\
 & \vdash \{C \wedge b\}\ c\ \{C\} \quad \text{(rule (while))} \\
\iff & \text{ex. } C \in \mathit{Assn} \text{ such that } \models (A \Rightarrow C), \models (C \wedge \neg b \Rightarrow B), \\
 & \vdash \{C \wedge b\}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{C \wedge \neg b\} \quad \text{(rule (seq))}, \\
 & \vdash \{C \wedge \neg b\}\ \texttt{skip}\ \{C \wedge \neg b\} \quad \text{(rule (skip))} \\
\iff & \text{ex. } C \in \mathit{Assn} \text{ such that } \models (A \Rightarrow C), \models (C \wedge \neg b \Rightarrow B), \\
 & \vdash \{C\}\ \texttt{if}\ b\ \texttt{then}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}\ \{C \wedge \neg b\} \text{ (rule (if))} \\
\iff & \vdash \{A\}\ \texttt{if}\ b\ \texttt{then}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}\ \{B\} \quad \text{(rule (cons))} \\
\iff & \models \{A\}\ \texttt{if}\ b\ \texttt{then}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}\ \{B\} \text{ (Thm. 10.2, 10.5)}
\end{array}
$$

Characteristic Assertions I

The following results are based of the following encoding of states by assertions:

Definition 12.3

Given a finite subset of program variables $X \subseteq \mathit{Var}$ and a state $\sigma \in \Sigma$, the characteristic assertion of σ w.r.t. X is given by

$$\mathit{State}(\sigma, X) := \bigwedge_{x \in X} (x = \underbrace{\sigma(x)}_{\in \mathbb{Z}}) \in \mathit{Assn}$$

Moreover, we let $\mathit{State}(\sigma, \emptyset) := \mathrm{true}$ and $\mathit{State}(\bot, X) := \mathrm{false}$.

Characteristic Assertions II

Programs and characteristic state assertions are obviously related in the following way:

Corollary 12.4

Let $c \in Cmd$, and let $FV(c) \subseteq Var$ denote the set of all variables occurring in c. Then, for every finite $X \supseteq FV(c)$ and $\sigma \in \Sigma$,

$$\{State(\sigma, X)\}\ c\ \{State(\mathfrak{C}[\![c]\!]\sigma, X)\}$$

Example 12.5 (Factorial program)

For $c := (\texttt{y:=1; while } \neg(\texttt{x=1}) \texttt{ do y:=y*x; x:=x-1 end})$, $X = \{\texttt{x}, \texttt{y}\}$, $\sigma(\texttt{x}) = 3$, and $\sigma(\texttt{y}) = 0$, we obtain

$$State(\sigma, X) = (\texttt{x=3} \ \wedge \ \texttt{y=0})$$
$$State(\mathfrak{C}[\![c]\!]\sigma, X) = (\texttt{x=1} \ \wedge \ \texttt{y=6})$$

Partial vs. Total Equivalence

Now we can show that considering total rather than partial correctness properties yields the same notion of equivalence:

Theorem 12.6

Let $c_1, c_2 \in Cmd$. The following propositions are equivalent:

1. $\forall A, B \in Assn: \quad \models \{A\}\, c_1\, \{B\} \iff \models \{A\}\, c_2\, \{B\}$
2. $\forall A, B \in Assn: \quad \models \{A\}\, c_1\, \{\Downarrow B\} \iff \models \{A\}\, c_2\, \{\Downarrow B\}$

Proof.

on the board □

Axiomatic vs. Operational/Denotational Equivalence

Axiomatic vs. Operational/Denotational Equiv.

Theorem 12.7

Axiomatic and operational/denotational equivalence coincide, i.e., for all $c_1, c_2 \in Cmd$,

$$c_1 \approx c_2 \iff c_1 \sim c_2.$$

Proof.

on the board □

Summary: Axiomatic Semantics

- Formalized by partial/total correctness properties
- Inductively defined by Hoare Logic proof rules
- Technically involved (especially loop invariants)
  $\Rightarrow$ machine support (proof assistants) indispensable for larger programs
- Equivalence of axiomatic and operational/denotational semantics
- Software engineering aspect: integrated development of program and proof (cf. assertions in Java)
- Systematic approach: mechanised program verification
  1. Start with (correctness) requirements for program
  2. Manually derive corresponding program annotations (assertions)
  3. Automatically derive corresponding verification conditions (using weakest preconditions etc.)
  4. Automatically discharge/simplify verification conditions using theorem prover
  5. Manually complete proof if required

  (cf. Mike Gordon: *Background reading on Hoare Logic*, Chapter 3, `www.cl.cam.ac.uk/~mjcg/Teaching/2011/Hoare/Notes/Notes.pdf`)