Semantics and Verification of Software

Summer Semester 2019

Lecture 11: Axiomatic Semantics of WHILE III
(Completeness & Total Correctness)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ss-19/sv-sw/
Recap: Hoare Logic

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

- **(skip)**
  
  $\{A\} \text{ skip } \{A\}$

- **(seq)**
  
  $\{A\} c_1 \{C\} \{C\} c_2 \{B\}$
  
  $\{A\} c_1 ; c_2 \{B\}$

- **(asgn)**
  
  $\{A[x \mapsto a]\} x := a \{A\}$

- **(if)**
  
  $\{A \land b\} c_1 \{B\} \{A \land \neg b\} c_2 \{B\}$
  
  $\{A\} \text{ if } b \text{ then } c_1 \text{ else } c_2 \text{ end } \{B\}$

  $\models (A \Rightarrow A') \{A'\} c \{B'\} \models (B' \Rightarrow B)$

- **(while)**
  
  $\{A\} \text{ while } b \text{ do } c \text{ end } \{A \land \neg b\}$

- **(cons)**
  
  $\{A\} c \{B\}$

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

Soundness of Hoare Logic

Theorem (Soundness of Hoare Logic)

For every partial correctness property \( \{ A \} \) \( c \) \( \{ B \} \),

\[
\vdash \{ A \} c \{ B \} \quad \Rightarrow \quad \models \{ A \} c \{ B \}.
\]

Proof.

Let \( \vdash \{ A \} c \{ B \} \). By induction over the structure of the corresponding proof tree we show that, for every \( \sigma \in \Sigma \) and \( I \in \text{Int} \) such that \( \sigma \models \lnot A, \mathcal{C}[c]\sigma = \bot \) or \( \mathcal{C}[c]\sigma \models \lnot B \) (on the board).
Incompleteness of Hoare Logic

Soundness: only valid partial correctness properties are provable ✓
Completeness: all valid partial correctness properties are systematically derivable ✗

Theorem 11.1 (Gödel’s Incompleteness Theorem)

The set of all valid assertions

\[ \{ A \in \text{Assn} \mid \models A \} \]

is not recursively enumerable, i.e., there exists no proof system for \text{Assn} in which all valid assertions are systematically derivable.

Proof.

see [Winskel 1996, p. 110 ff]
Incompleteness of Hoare Logic

Incompleteness of Hoare Logic II

Corollary 11.2

There is no proof system in which all valid partial correctness properties can be enumerated.

Proof.

Given $A \in Assn$, $\models A$ is obviously equivalent to $\{\text{true}\} \triangleright \text{skip} \{A\}$. Thus the enumerability of all valid partial correctness properties would imply the enumerability of all valid assertions.

Remark: alternative proof (using computability theory):
$\{\text{true}\} c \{\text{false}\}$ is valid iff $c$ does not terminate on any input state. But the set of all non-terminating WHILE statements is not enumerable.
Relative Completeness of Hoare Logic

Relative Completeness of Hoare Logic I

- We will see: actual reason of incompleteness is rule

\[
\frac{\models (A \Rightarrow A') \quad \{A\} \ c \ \{B'\} \quad \models (B' \Rightarrow B)}{(\text{cons})} \quad \{A\} \ c \ \{B\}
\]

since it is based on the validity of implications within \(\text{Assn}\)
- The other language constructs are “enumerable”
- Therefore: separation of proof system (Hoare Logic) and assertion language (Assn)
- One can show: if an “oracle” is available which decides whether a given assertion is valid, then all valid partial correctness properties can be systematically derived

⇒ “Relative completeness”
Theorem 11.3 (Cook’s Completeness Theorem)

Hoare Logic is relatively complete, i.e., for every partial correctness property $\{A\} c \{B\}$:

$$\models \{A\} c \{B\} \Rightarrow \vdash \{A\} c \{B\}.$$  

Thus: if we know that a partial correctness property is valid, then we know that there is a corresponding proof.

The proof uses the following concept: assume that, e.g., $\{A\} c_1 ; c_2 \{B\}$ has to be derived. This requires an intermediate assertion $C \in Assn$ such that $\{A\} c_1 \{C\}$ and $\{C\} c_2 \{B\}$. How to find it?
Relative Completeness of Hoare Logic

Weakest Liberal Preconditions I

Definition 11.4 (Weakest liberal precondition)

Given \( c \in \text{Cmd} \) and \( S \subseteq \Sigma \), the weakest (liberal) precondition of \( S \) with respect to \( c \) collects all states \( \sigma \) such that running \( c \) in \( \sigma \) does not terminate or yields a state in \( S \):

\[
\text{wlp}[c] S := \{ \sigma \in \Sigma \mid C[c] \sigma \in S \cup \{ \bot \} \}.
\]

Corollary 11.5

For every \( c \in \text{Cmd}, A, B \in \text{Assn}, \) and \( I \in \text{Int} \):

1. \( \models^I \{A\} c \{B\} \iff A^I \subseteq \text{wlp}[c] B^I \)
2. If \( A_0 \in \text{Assn} \) such that \( A_0^I = \text{wlp}[c] B^I \) for every \( I \in \text{Int} \), then \( \models \{A\} c \{B\} \iff \models (A \Rightarrow A_0) \)

Remarks:

- Corollary 11.5 justifies the notion of weakest precondition: it is entailed by every precondition \( A \) that makes \( \{A\} c \{B\} \) valid.
- In the following, we do not distinguish between sets of program states (such as \( S \) or \( A^I \)) and predicates on program states (such as \( B[b] \)).
Relative Completeness of Hoare Logic

Weakest Liberal Preconditions II

Lemma 11.6 (Weakest liberal precondition transformer)

Weakest liberal preconditions $\text{wlp}[] : \text{Cmd} \times 2^\Sigma \rightarrow 2^\Sigma$ can be computed as follows:

- $\text{wlp}[\text{skip}]S = S$
- $\text{wlp}[x:=a]S = \{\sigma \in \Sigma \mid \sigma[x \mapsto \mathcal{A}[a]\sigma] \in S\}$
- $\text{wlp}[c_1;c_2]S = \text{wlp}[c_1]\left(\text{wlp}[c_2]S\right)$
- $\text{wlp}[\text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end}]S = (\mathcal{B}[b] \cap \text{wlp}[c_1]S) \cup (\mathcal{B}[\neg b] \cap \text{wlp}[c_2]S)$
- $\text{wlp}[\text{while } b \text{ do } c \text{ end}]S = \text{FIX}(\Psi)$

where $\text{FIX}(\Psi)$ denotes the greatest fixpoint (w.r.t. $(2^\Sigma, \subseteq)$) of

$$\Psi : 2^\Sigma \rightarrow 2^\Sigma : T \mapsto (\mathcal{B}[b] \cap \text{wlp}[c]T) \cup (\mathcal{B}[\neg b] \cap S)$$

Remark: $\text{FIX}(\Psi)$ of a continuous function $\Psi$ on lattice $(2^\Sigma, \subseteq)$ can be computed by fixpoint iteration (see following slide)

Proof.

omitted
Relative Completeness of Hoare Logic

Weakest Liberal Preconditions III

Example 11.7

Using Lemma 11.6, we want to determine the weakest liberal precondition for

\[
\{ ? \} \text{while } x \neq 0 \land x \neq 1 \text{ do } x := x - 2 \text{ end } \{ x = 1 \}
\]

i.e., \( wlpc S \) for \( S := B x = 1 \).

- \( wlpc S = \text{FIX}(\Psi) \) for \( \Psi(T) = (B x \neq 0 \land x \neq 1) \cap wlpc_0 T \cup (B x \in \{0, 1\} \cap S) \)
- \( wlpc_0 T = \{ \sigma \in \Sigma \mid \sigma[x \mapsto x] \in T \} \)
- Fixpoint iteration (with initial value \( \prod \emptyset = \Sigma \)):

\[
\begin{align*}
\Psi(\Sigma) &= (B x \neq 0 \land x \neq 1) \cap wlpc_0 \Sigma \cup S = B x \neq 0 \\
\Psi^2(\Sigma) &= (B x \neq 0 \land x \neq 1) \cap wlpc_0 (B x \neq 0) \cup S = B x \neq 0 \land x \neq 2 \\
\Psi^3(\Sigma) &= (B x \neq 0 \land x \neq 1) \cap wlpc_0 (B x \neq 0 \land x \neq 2) \cup S = B x \neq 0 \land x \neq 2 \land x \neq 4 \\
&\vdots \\
\Rightarrow \text{FIX}(\Psi) = \bigcap_{n \in \mathbb{N}} \Psi^n(\Sigma) = \{ \sigma \in \Sigma \mid \sigma(x) \in \mathbb{Z}_{<0} \cup \{1, 3, 5, \ldots\} \}
\end{align*}
\]
Relative Completeness of Hoare Logic

Weakest Liberal Preconditions IV

Definition 11.8 (Expressivity of assertion languages)

An assertion language $\textbf{Assn}$ is called expressive if it allows to “syntactify” weakest preconditions, that is, for every $c \in \textbf{Cmd}$ and $B \in \textbf{Assn}$, there exists $A_{c,B} \in \textbf{Assn}$ such that $A_{c,B}^l = wlp[c]B^l$ for every $l \in \textbf{Int}$.

Theorem 11.9 (Expressivity of $\textbf{Assn}$)

$\textbf{Assn}$ is expressive.

Proof (idea; see (Winskel 1996, p. 103 ff) for details).

Given $c \in \textbf{Cmd}$ and $B \in \textbf{Assn}$, construct $A_{c,B} \in \textbf{Assn}$ with $\sigma \models^l A_{c,B} \iff \sigma \models^l \ell c \sigma \mid^l B$ (for every $\sigma \in \Sigma$, $l \in \textbf{Int}$). For example:

$A_{\text{skip},B} := B \quad A_{x:=a,B} := B[x \mapsto a]$

$A_{c_1; c_2, B} := A_{c_1,A_{c_2,B}} \ldots$

(for while: “Gödelisation” of sequences of intermediate states)
Relative Completeness of Hoare Logic

Relative Completeness of Hoare Logic III

The following lemma shows that syntactic weakest preconditions are “provable”:

Lemma 11.10

\[ \text{For every } c \in \text{Cmd and } B \in \text{Assn: } \vdash \{A_{c,B}\} c \{B\} \]

Proof.

by structural induction over \( c \) (omitted)

Proof (Cook’s Completeness Theorem 11.3).

We have to show that Hoare Logic is relatively complete, i.e., that

\[ \models \{A\} c \{B\} \quad \Rightarrow \quad \vdash \{A\} c \{B\}. \]

- Lemma 11.10: \( \vdash \{A_{c,B}\} c \{B\} \)
- Corollary 11.5: \( \models \{A\} c \{B\} \quad \Rightarrow \quad \models (A \Rightarrow A_{c,B}) \)
  \[ \models (A \Rightarrow A_{c,B}) \cdot \{A_{c,B}\} c \{B\} \quad \Rightarrow \quad \models (B \Rightarrow B) \]
- (cons) \[ \models \{A\} c \{B\} \]
Total Correctness

Observation: partial correctness properties only speak about terminating computations of a given program.

Total correctness additionally requires the proof that the program indeed stops (on the input states admitted by the precondition).

Consider total correctness properties of the form\[ \{A\} c \{\Downarrow B\} \]

where \( c \in Cmd \) and \( A, B \in Assn \).

Interpretation:

Validity of property \( \{A\} c \{\Downarrow B\} \):

For all states \( \sigma \in \Sigma \) which satisfy \( A \):
the execution of \( c \) in \( \sigma \) terminates and yields a state which satisfies \( B \).
Total Correctness

Semantics of Total Correctness Properties

Definition 11.11 (Semantics of total correctness properties)

Let $A, B \in \text{Assn}$ and $c \in \text{Cmd}$.

1. $\{A\} c \{\downarrow B\}$ is called valid in $\sigma \in \Sigma$ and $I \in \text{Int}$ (notation: $\sigma \models I \{A\} c \{\downarrow B\}$) if $\sigma \models I A$ and $c[\sigma] \models I B$.
2. $\{A\} c \{\downarrow B\}$ is called valid in $I \in \text{Int}$ (notation: $\models I \{A\} c \{\downarrow B\}$) if $\sigma \models I \{A\} c \{\downarrow B\}$ for every $\sigma \in \Sigma$.
3. $\{A\} c \{\downarrow B\}$ is called valid (notation: $\models \{A\} c \{\downarrow B\}$) if $\models I \{A\} c \{\downarrow B\}$ for every $I \in \text{Int}$.

Obviously, total implies partial correctness (but not vice versa):

Corollary 11.12

For all $A, B \in \text{Assn}$ and $c \in \text{Cmd}$,

$$\models \{A\} c \{\downarrow B\} \Rightarrow \models \{A\} c \{B\}.$$
Total Correctness

Proving Total Correctness I

Definition 11.13 (Hoare Logic for total correctness)

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

\[
\begin{align*}
\text{(skip)} & \quad \{A\} \text{skip} \{\downarrow A\} \\
\text{(asgn)} & \quad \{A[x \mapsto a]\} x := a \{\downarrow A\} \\
\text{(seq)} & \quad \{A\} c_1 \{\downarrow C\} \quad \{C\} c_2 \{\downarrow B\} \\
& \quad \{A\} c_1 ; c_2 \{\downarrow B\} \\
\text{(if)} & \quad \{A \land b\} c_1 \{\downarrow B\} \quad \{A \land \neg b\} c_2 \{\downarrow B\} \\
& \quad \{A\} \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end} \{\downarrow B\} \\
\text{(while)} & \quad \{\exists i. i \geq 0 \land A(i + 1) \Rightarrow b\} \\
& \quad \{i \geq 0 \land A(i + 1)\} c \{\downarrow A(i)\} \\
& \quad \{A\} \text{while } b \text{ do } c \text{ end} \{\downarrow A(0)\} \\
\text{(cons)} & \quad \{A \Rightarrow A'\} \quad \{A'\} c \{\downarrow B'\} \\
& \quad \{A\} c \{\downarrow B\} \\
& \quad \{A(0) \Rightarrow \neg b\}
\end{align*}
\]

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

Total Correctness

Proving Total Correctness II

• In rule

\[
\vdash (i \geq 0 \land A(i + 1) \Rightarrow b) \quad \{i \geq 0 \land A(i + 1)\} \ c \ \{\downarrow A(i)\} \quad \vdash (A(0) \Rightarrow \neg b)
\]

\[
\{\exists i. i \geq 0 \land A(i)\} \ \text{while} \ b \ \text{do} \ c \ \text{end} \ \{\downarrow A(0)\}
\]

the notation \(A(i)\) indicates that assertion \(A\) parametrically depends on the value of the logical variable \(i \in LVar\).

• Idea: \(i\) represents the remaining number of loop iterations

• Loop to be traversed \(i + 1\) times \((i \geq 0)\)
  \(\Rightarrow A(i + 1)\) holds
  \(\Rightarrow\) execution condition \(b\) satisfied

Thus: \(\vdash (i \geq 0 \land A(i + 1) \Rightarrow b)\), and \(i + 1\) decreased to \(i\) after execution of \(c\)

• Execution terminated
  \(\Rightarrow A(0)\) holds
  \(\Rightarrow\) execution condition \(b\) violated

Thus: \(\vdash (A(0) \Rightarrow \neg b)\)
Total Correctness

Total Correctness of Factorial Program I

Example 11.14

Proof of \{A\} y := 1; c \{\Downarrow B\} where
\[
A := (x > 0 \land x = i)
\]
\[
c := \text{while } \neg(x=1) \text{ do } y := y \times x; \ x := x - 1 \text{ end}
\]
\[
B := (y = i!)
\]

First we show that the assertion \(C(j) = (x > 0 \land y \times x! = i! \land x = j + 1)\) is an invariant of \(c\). Applying (asgn) twice yields
\[
\vdash \{ j \geq 0 \land C(j)[x \mapsto x-1]\} x := x - 1 \{ \Downarrow j \geq 0 \land C(j) \}
\]
and
\[
\vdash \{ j \geq 0 \land C(j)[x \mapsto x-1][y \mapsto y \times x]\} y := y \times x \{ \Downarrow j \geq 0 \land C(j)[x \mapsto x-1] \}
\]
such that (seq) implies
\[
\vdash \{ j \geq 0 \land C(j)[x \mapsto x-1][y \mapsto y \times x]\} y := y \times x; \ x := x - 1 \{ \Downarrow j \geq 0 \land C(j) \}.
\]

Now \(C(j + 1) = (x > 0 \land y \times x! = i! \land x = j + 2)\) and
\(C(j)[x \mapsto x-1][y \mapsto y \times x] = (x - 1 > 0 \land y \times x \times (x - 1)! = i! \land x - 1 = j + 1)\)
such that
\[
\models ((j \geq 0 \land C(j + 1))) \Rightarrow (j \geq 0 \land C(j)[x \mapsto x-1][y \mapsto y \times x]) \text{ and}
\]
\[
\models ((j \geq 0 \land C(j)) \Rightarrow C(j)).
\]
Total Correctness

Total Correctness of Factorial Program II

Example 11.14 (continued; $C(j) = (x > 0 \land y \times x! = i! \land x = j + 1)$)

Hence (cons) implies
$$\vdash \{j \geq 0 \land C(j + 1)\} \ y := y \times x \ ; \ x := x - 1 \ \{\downarrow C(j)\}.$$  

Moreover we have
$$\models ((j \geq 0 \land C(j + 1)) \Rightarrow \neg(x = 1)) \ \text{and} \ \models (C(0) \Rightarrow \neg(\neg(x = 1)))$$

such that (while) yields
$$\vdash \{\exists j. j \geq 0 \land C(j)\} \ c \ \{\downarrow C(0)\}.$$  

For the initializing assignment, (asgn) implies
$$\vdash \{\exists j. j \geq 0 \land C(j)[y \mapsto 1]\} \ y := 1 \ \{\downarrow \exists j. j \geq 0 \land C(j)\},$$

such that (seq) allows to conclude
$$\vdash \{\exists j. j \geq 0 \land C(j)[y \mapsto 1]\} \ y := 1 \ ; \ c \ \{\downarrow C(0)\}.$$  

On the other hand we have (choose $j := i - 1$):
$$\models ((x > 0 \land x = i) \Rightarrow (\exists j. j \geq 0 \land C(j)[y \mapsto 1])) \ \text{and} \ \models (C(0) \Rightarrow y = i!)$$

such that (cons) yields the desired result:
$$\vdash \{x > 0 \land x = i\} \ y := 1 \ ; \ c \ \{\downarrow y = i!\}.$$  

22 of 22  
Semantics and Verification of Software  
Summer Semester 2019  
Lecture 11: Axiomatic Semantics of WHILE III  
(Completeness & Total Correctness)