Semantics and Verification of Software

Summer Semester 2019

Lecture 10: Axiomatic Semantics of WHILE II (Soundness & Completeness)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

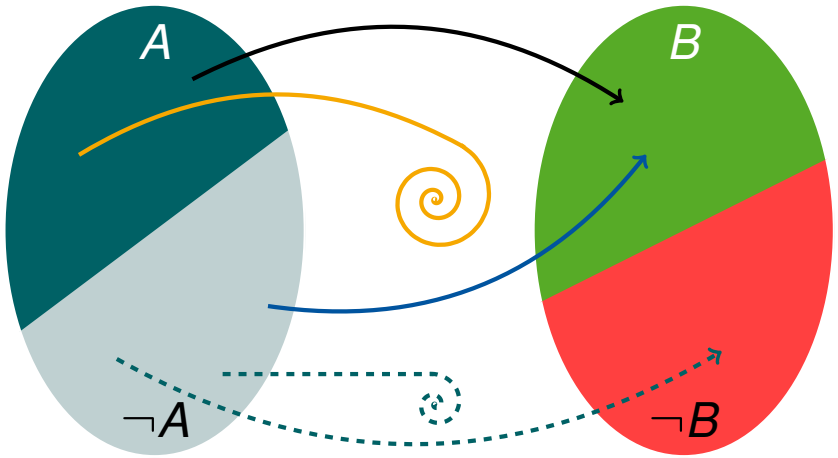
https://moves.rwth-aachen.de/teaching/ss-19/sv-sw/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Outline of Lecture 10

The Axiomatic Approach IV



(picture courtesy of C. Matheja)

Partial correctness properties

$\{A\}\ c\ \{B\}$ is valid if for all states $\sigma \in \Sigma$ which satisfy A:
if the execution of c in σ terminates in $\sigma' \in \Sigma$, then σ' satisfies B.

In particular, $\{\text{true}\}\ c\ \{\text{false}\}$ is valid for $c = \texttt{while true do skip end}$

Total correctness properties

$\{A\}\ c\ \{\Downarrow B\}$ is valid if for all states $\sigma \in \Sigma$ which satisfy A:
the execution of c in σ terminates and yields a state which satisfies B.

Recap: Axiomatic Semantics of WHILE

Syntax of the Assertion Language

Definition (Syntax of assertions)

The syntax of *Assn* is defined by the following context-free grammar:

$$a ::= z \mid x \mid i \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \in \mathit{LExp}$$
$$A ::= t \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg A \mid A_1 \wedge A_2 \mid A_1 \vee A_2 \mid \forall i.A \in \mathit{Assn}$$

- Thus: $\mathit{AExp} \subsetneq \mathit{LExp}$, $\mathit{BExp} \subsetneq \mathit{Assn}$
- The following (and other) abbreviations will be employed:

$$A_1 \Rightarrow A_2 := \neg A_1 \vee A_2$$
$$\exists i.A := \neg(\forall i.\neg A)$$
$$a_1 \geq a_2 := a_1 > a_2 \vee a_1 = a_2$$
$$\vdots$$

Recap: Axiomatic Semantics of WHILE

Semantics of Assertions I

Reminder: $A ::= t \mid a_1{=}a_2 \mid a_1{>}a_2 \mid \neg A \mid A_1 \wedge A_2 \mid A_1 \vee A_2 \mid \forall i.A \in Assn$

Definition (Semantics of assertions)

Let $A \in Assn$, $\sigma \in \Sigma$, and $I \in Int$. The relation "σ satisfies A in I" (notation: $\sigma \models^I A$) is inductively defined by:

$$
\begin{array}{rll}
\sigma \models^I & \text{true} & \\
\sigma \models^I & a_1{=}a_2 & \text{if } \mathfrak{L}[\![a_1]\!] I\sigma = \mathfrak{L}[\![a_2]\!] I\sigma \\
\sigma \models^I & a_1{>}a_2 & \text{if } \mathfrak{L}[\![a_1]\!] I\sigma > \mathfrak{L}[\![a_2]\!] I\sigma \\
\sigma \models^I & \neg A & \text{if not } \sigma \models^I A \\
\sigma \models^I & A_1 \wedge A_2 & \text{if } \sigma \models^I A_1 \text{ and } \sigma \models^I A_2 \\
\sigma \models^I & A_1 \vee A_2 & \text{if } \sigma \models^I A_1 \text{ or } \sigma \models^I A_2 \\
\sigma \models^I & \forall i.A & \text{if } \sigma \models^{I[i \mapsto z]} A \text{ for every } z \in \mathbb{Z} \\
(\textit{not } \bot \models^I & A) &
\end{array}
$$

Furthermore σ satisfies A ($\sigma \models A$) if $\sigma \models^I A$ for every interpretation $I \in Int$, and A is called valid ($\models A$) if $\sigma \models A$ for every state $\sigma \in \Sigma$.

Semantics of Assertions II

Definition (Extension)

Let $A \in \mathit{Assn}$ and $I \in \mathit{Int}$. The extension of A with respect to I is given by

$$A^I := \{\sigma \in \Sigma \mid \sigma \models^I A\}.$$

Example

For $A := (\exists i.i*i = x)$ and every $I \in \mathit{Int}$,

$$A^I = \{\sigma \in \Sigma \mid \sigma(x) \in \{0, 1, 4, 9, \ldots\}\}$$

Partial Correctness Properties

Definition (Partial correctness properties)

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

- An expression of the form $\{A\}\, c\, \{B\}$ is called a partial correctness property (PCP) with precondition A and postcondition B.
- Given $\sigma \in \Sigma$ and $I \in \mathit{Int}$, we let

$$\sigma \models^I \{A\}\, c\, \{B\}$$

  if ($\sigma \models^I A$ and $\mathfrak{C}[\![c]\!]\sigma \neq \bot$) implies $\mathfrak{C}[\![c]\!]\sigma \models^I B$
  (or equivalently: $\sigma \in A^I \Rightarrow \mathfrak{C}[\![c]\!]\sigma \in B^I \cup \{\bot\}$).
- $\{A\}\, c\, \{B\}$ is called valid in I (notation: $\models^I \{A\}\, c\, \{B\}$) if $\sigma \models^I \{A\}\, c\, \{B\}$ for every $\sigma \in \Sigma$
  (or equivalently: $\mathfrak{C}[\![c]\!]A^I \subseteq B^I \cup \{\bot\}$).
- $\{A\}\, c\, \{B\}$ is called valid (notation: $\models \{A\}\, c\, \{B\}$) if $\models^I \{A\}\, c\, \{B\}$ for every $I \in \mathit{Int}$.

Recap: Axiomatic Semantics of WHILE

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.

Outline of Lecture 10

Applying Hoare Logic I

Example 10.1 (Factorial program)

Proof of $\{A\}\,\texttt{y:=1;}\,c\,\{B\}$ where

$$
\begin{aligned}
c &:= (\texttt{while } \neg(\texttt{x=1}) \texttt{ do y := y*x; x:= x-1 end}) \\
A &:= (\texttt{x} > 0 \wedge \texttt{x} = i) \\
B &:= (\texttt{y} = i!)
\end{aligned}
$$

(on the board)

An Example

Applying Hoare Logic I

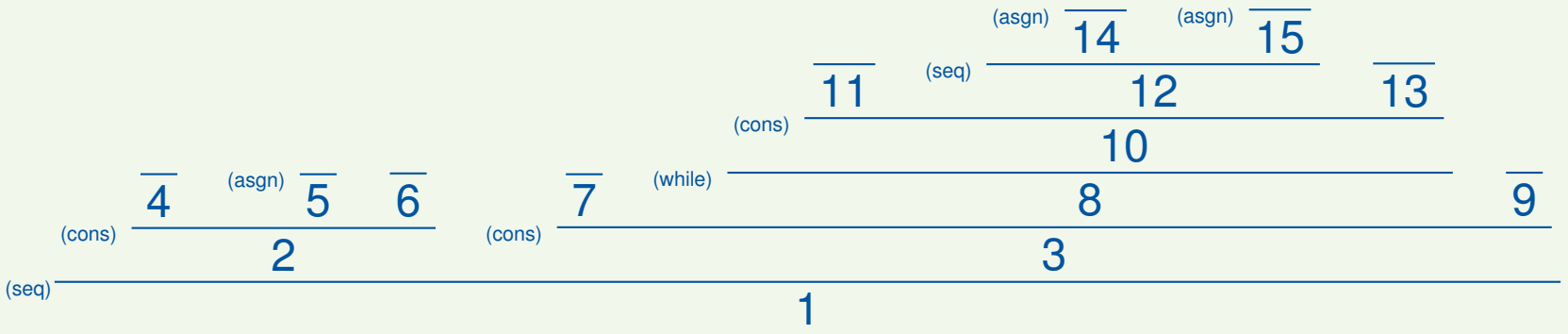
Example 10.1 (Factorial program)

Proof of $\{A\}\,\texttt{y:=1;}\,c\,\{B\}$ where

$$c := (\texttt{while } \neg(\texttt{x=1}) \texttt{ do y := y*x; x:= x-1 end})$$
$$A := (\texttt{x} > 0 \wedge \texttt{x} = i)$$
$$B := (\texttt{y} = i!)$$

(on the board)

Structure of the proof:



Applying Hoare Logic II

Example 10.1 (continued)

Here the respective propositions are given by (where $C := (\mathtt{x} > 0 \wedge \mathtt{y} * \mathtt{x}! = i!)$):

1. $\{A\}\,\mathtt{y := 1}; c\,\{B\}$
2. $\{A\}\,\mathtt{y := 1}\,\{C\}$
3. $\{C\}\,c\,\{B\}$
4. $\models (A \Rightarrow C[\mathtt{y} \mapsto 1])$
5. $\{C[\mathtt{y} \mapsto 1]\}\,\mathtt{y := 1}\,\{C\}$
6. $\models (C \Rightarrow C)$
7. $\models (C \Rightarrow C)$
8. $\{C\}\,c\,\{\neg(\neg(\mathtt{x = 1})) \wedge C\}$
9. $\models (\neg(\neg(\mathtt{x = 1})) \wedge C \Rightarrow B)$
10. $\{\neg(\mathtt{x = 1}) \wedge C\}\,\mathtt{y := y*x;\ x := x-1}\,\{C\}$
11. $\models (\neg(\mathtt{x = 1}) \wedge C \Rightarrow C[\mathtt{x} \mapsto \mathtt{x-1}, \mathtt{y} \mapsto \mathtt{y*x}])$
12. $\{C[\mathtt{x} \mapsto \mathtt{x-1}, \mathtt{y} \mapsto \mathtt{y*x}]\}\,\mathtt{y := y*x;\ x := x-1}\,\{C\}$
13. $\models (C \Rightarrow C)$
14. $\{C[\mathtt{x} \mapsto \mathtt{x-1}, \mathtt{y} \mapsto \mathtt{y*x}]\}\,\mathtt{y := y*x}\,\{C[\mathtt{x} \mapsto \mathtt{x-1}]\}$
15. $\{C[\mathtt{x} \mapsto \mathtt{x-1}]\}\,\mathtt{x := x-1}\,\{C\}$

Outline of Lecture 10

Discovering Invariants

Goal

Prove PCP $\{A\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{B\}$ by identifying invariant C:

$$\text{(while)}\ \frac{\{C \wedge b\}\, c\, \{C\}}{\{C\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{C \wedge \neg b\}}$$

Discovering Invariants

Goal

Prove PCP $\{A\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{B\}$ by identifying invariant C:

$$\overset{\text{(while)}}{\dfrac{\{C \wedge b\}\, c\, \{C\}}{\{C\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{C \wedge \neg b\}}}$$

This may require some ingenuity, but there are a few hints on how to do that:

- In general, there are several invariants but most of them are useless (for example, true is always an invariant)
- A suitable invariant has to be
  - weak enough to be implied by the precondition: $\models (A \Rightarrow C)$
  - strong enough to imply the postcondition: $\models (C \wedge \neg b \Rightarrow B)$
- In general, looking at the logical structure of the postcondition will help
- Often a suitable invariant is found by generalising the postcondition, replacing a constant by a variable that is changed in the body of the loop
- It can be helpful to "trace" the loop and inspect the values of the variables at every iteration

More on Invariants

What is the Invariant?

Example 10.2

1. $\{\mathtt{y} \geq 0 \wedge \mathtt{y} = i\}\ \mathtt{z := 1; while}\ \neg(\mathtt{y=0})\ \mathtt{do\ y := y-1; z := z*x\ end}\ \{\mathtt{z} = \mathtt{x}^i\}$
   - Invariant: $C = ?$
   - Precondition: $\mathtt{y} \geq 0 \wedge \mathtt{y} = i \wedge \mathtt{z} = 1 \Rightarrow C$
   - Postcondition: $C \wedge \mathtt{y} = 0 \Rightarrow \mathtt{z} = \mathtt{x}^i$

More on Invariants

What is the Invariant?

Example 10.2

1. $\{\mathtt{y} \geq 0 \wedge \mathtt{y} = i\}$ `z := 1; while ¬(y=0) do y := y-1; z := z*x end` $\{\mathtt{z} = \mathtt{x}^i\}$
   - Invariant: $C = (\mathtt{z} = \mathtt{x}^{i-\mathtt{y}})$
   - Precondition: $\mathtt{y} \geq 0 \wedge \mathtt{y} = i \wedge \mathtt{z} = 1 \Rightarrow C$ ✓
   - Postcondition: $C \wedge \mathtt{y} = 0 \Rightarrow \mathtt{z} = \mathtt{x}^i$ ✓

More on Invariants

What is the Invariant?

Example 10.2

1. $\{\mathtt{y} \geq 0 \wedge \mathtt{y} = i\}$ `z := 1; while ¬(y=0) do y := y-1; z := z*x end` $\{\mathtt{z} = \mathtt{x}^i\}$
   - Invariant: $C = (\mathtt{z} = \mathtt{x}^{i-\mathtt{y}})$
   - Precondition: $\mathtt{y} \geq 0 \wedge \mathtt{y} = i \wedge \mathtt{z} = 1 \Rightarrow C$ ✓
   - Postcondition: $C \wedge \mathtt{y} = 0 \Rightarrow \mathtt{z} = \mathtt{x}^i$ ✓
2. $\{\mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{x} = i\}$
   `z := 0; while y<=x do x := x-y; z := z+1 end`
   $\{i = \mathtt{z} * \mathtt{y} + \mathtt{x}\}$
   - Invariant: $C = ?$
   - Precondition: $\mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{x} = i \wedge \mathtt{z} = 0 \Rightarrow C$
   - Postcondition: $C \wedge \mathtt{y} > \mathtt{x} \Rightarrow i = \mathtt{z} * \mathtt{y} + x$

More on Invariants

What is the Invariant?

Example 10.2

1. $\{\mathtt{y} \geq 0 \wedge \mathtt{y} = i\}$ `z := 1; while ¬(y=0) do y := y-1; z := z*x end` $\{\mathtt{z} = \mathtt{x}^i\}$
   - Invariant: $C = (\mathtt{z} = \mathtt{x}^{i-\mathtt{y}})$
   - Precondition: $\mathtt{y} \geq 0 \wedge \mathtt{y} = i \wedge \mathtt{z} = 1 \Rightarrow C$ ✓
   - Postcondition: $C \wedge \mathtt{y} = 0 \Rightarrow \mathtt{z} = \mathtt{x}^i$ ✓
2. $\{\mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{x} = i\}$
   `z := 0; while y<=x do x := x-y; z := z+1 end`
   $\{i = \mathtt{z} * \mathtt{y} + \mathtt{x}\}$
   - Invariant: $C = (i = \mathtt{z} * \mathtt{y} + \mathtt{x})$
   - Precondition: $\mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{x} = i \wedge \mathtt{z} = 0 \Rightarrow C$ ✓
   - Postcondition: $C \wedge \mathtt{y} > \mathtt{x} \Rightarrow i = \mathtt{z} * \mathtt{y} + x$ ✓

Outline of Lecture 10

Soundness of Hoare Logic I

Soundness: no wrong propositions can be derived, i.e., every (syntactically) provable partial correctness property is also (semantically) valid

Soundness of Hoare Logic I

Soundness: no wrong propositions can be derived, i.e., every (syntactically) provable partial correctness property is also (semantically) valid

For the corresponding proof we use:

Lemma 10.3 (Substitution lemma)

For every $A \in \mathit{Assn}$, $x \in \mathit{Var}$, $a \in \mathit{AExp}$, $\sigma \in \Sigma$, *and* $I \in \mathit{Int}$:

$$\sigma \models^I A[x \mapsto a] \iff \sigma[x \mapsto \mathfrak{A}[\![a]\!]\sigma] \models^I A.$$

Soundness of Hoare Logic I

Soundness: no wrong propositions can be derived, i.e., every (syntactically) provable partial correctness property is also (semantically) valid

For the corresponding proof we use:

Lemma 10.3 (Substitution lemma)

For every $A \in Assn$, $x \in Var$, $a \in AExp$, $\sigma \in \Sigma$, and $I \in Int$:

$$\sigma \models^I A[x \mapsto a] \iff \sigma[x \mapsto \mathfrak{A}[\![a]\!]\sigma] \models^I A.$$

Proof.

by induction over $A \in Assn$ (omitted) □

Soundness of Hoare Logic II

Theorem 10.4 (Soundness of Hoare Logic)

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

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

Soundness of Hoare Logic II

Theorem 10.4 (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 $I \in \mathit{Int}$ such that $\sigma \models^I A$, $\mathfrak{C}[\![c]\!]\sigma = \bot$ or $\mathfrak{C}[\![c]\!]\sigma \models^I B$ (on the board). □

Outline of Lecture 10

Incompleteness of Hoare Logic I

Soundness: only valid partial correctness properties are provable ✓

Completeness: all valid partial correctness properties are systematically derivable ↯

Incompleteness of Hoare Logic I

Soundness: only valid partial correctness properties are provable ✓

Completeness: all valid partial correctness properties are systematically derivable ↯

Theorem 10.5 (Gödel's Incompleteness Theorem)

The set of all valid assertions

$$\{A \in \mathit{Assn} \mid \models A\}$$

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

Proof.

see [Winskel 1996, p. 110 ff] □

Kurt Gödel
(1906–1978)

Incompleteness of Hoare Logic II

Corollary 10.6

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

Incompleteness of Hoare Logic II

Corollary 10.6

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

Proof.

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

Incompleteness of Hoare Logic II

Corollary 10.6

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

Proof.

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

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

Outline of Lecture 10

Relative Completeness of Hoare Logic I

- We will see: actual reason of incompleteness is rule

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

since it is based on the validity of implications within *Assn*

Relative Completeness of Hoare Logic I

- We will see: actual reason of incompleteness is rule

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

  since it is based on the validity of implications within *Assn*
- The other language constructs are “enumerable”

Relative Completeness of Hoare Logic I

- We will see: actual reason of incompleteness is rule

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

  since it is based on the validity of implications within *Assn*
- The other language constructs are “enumerable”
- Therefore: separation of proof system (Hoare Logic) and assertion language (*Assn*)

Relative Completeness of Hoare Logic I

- We will see: actual reason of incompleteness is rule

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

  since it is based on the validity of implications within *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 of Hoare Logic I

- We will see: actual reason of incompleteness is rule

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

  since it is based on the validity of implications within *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

$\Rightarrow$ “Relative completeness”

Relative Completeness of Hoare Logic II

Theorem 10.7 (Cook's Completeness Theorem)

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

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

Stephen A. Cook (* 1939)

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

Relative Completeness of Hoare Logic II

Theorem 10.7 (Cook's Completeness Theorem)

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

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

Stephen A. Cook (* 1939)

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 $C \in \mathit{Assn}$ such that $\{A\}\, c_1\, \{C\}$ and $\{C\}\, c_2\, \{B\}$. How to find it?

Weakest Liberal Preconditions I

Definition 10.8 (Weakest liberal precondition)

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

$$\mathit{wlp}[\![c]\!]S := \{\sigma \in \Sigma \mid \mathfrak{C}[\![c]\!]\sigma \in S \cup \{\bot\}\}.$$

Weakest Liberal Preconditions I

Definition 10.8 (Weakest liberal precondition)

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

$$\mathit{wlp}[\![c]\!]S := \{\sigma \in \Sigma \mid \mathfrak{C}[\![c]\!]\sigma \in S \cup \{\bot\}\}.$$

Corollary 10.9

For every $c \in \mathit{Cmd}$, $A, B \in \mathit{Assn}$, and $I \in \mathit{Int}$:

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

Weakest Liberal Preconditions I

Definition 10.8 (Weakest liberal precondition)

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

$$\mathit{wlp}[\![c]\!]S := \{\sigma \in \Sigma \mid \mathfrak{C}[\![c]\!]\sigma \in S \cup \{\bot\}\}.$$

Corollary 10.9

For every $c \in \mathit{Cmd}$, $A, B \in \mathit{Assn}$, and $I \in \mathit{Int}$:

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

Remarks:

- Corollary 10.9 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 $\mathfrak{B}[\![b]\!]$).

Weakest Liberal Preconditions II

Lemma 10.10 (Weakest liberal precondition transformer)

Weakest liberal preconditions $wlp[\![.]\!] . : Cmd \times 2^\Sigma \to 2^\Sigma$ *can be computed as follows:*

$$
\begin{aligned}
wlp[\![\texttt{skip}]\!]S &= S \\
wlp[\![x := a]\!]S &= \{\sigma \in \Sigma \mid \sigma[x \mapsto \mathfrak{A}[\![a]\!]\sigma] \in S\} \\
wlp[\![c_1 ; c_2]\!]S &= wlp[\![c_1]\!](wlp[\![c_2]\!]S) \\
wlp[\![\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}]\!]S &= (\mathfrak{B}[\![b]\!] \cap wlp[\![c_1]\!]S) \cup (\mathfrak{B}[\![\neg b]\!] \cap wlp[\![c_2]\!]S) \\
wlp[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!]S &= \mathrm{FIX}(\Psi)
\end{aligned}
$$

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

$$\Psi : 2^\Sigma \to 2^\Sigma : T \mapsto (\mathfrak{B}[\![b]\!] \cap wlp[\![c]\!]T) \cup (\mathfrak{B}[\![\neg b]\!] \cap S)$$

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

Weakest Liberal Preconditions II

Lemma 10.10 (Weakest liberal precondition transformer)

Weakest liberal preconditions $wlp[\![.]\!] : Cmd \times 2^\Sigma \to 2^\Sigma$ *can be computed as follows:*

$$
\begin{aligned}
wlp[\![\texttt{skip}]\!]S &= S \\
wlp[\![x := a]\!]S &= \{\sigma \in \Sigma \mid \sigma[x \mapsto \mathfrak{A}[\![a]\!]\sigma] \in S\} \\
wlp[\![c_1 ; c_2]\!]S &= wlp[\![c_1]\!](wlp[\![c_2]\!]S) \\
wlp[\![\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}]\!]S &= (\mathfrak{B}[\![b]\!] \cap wlp[\![c_1]\!]S) \cup (\mathfrak{B}[\![\neg b]\!] \cap wlp[\![c_2]\!]S) \\
wlp[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!]S &= \mathrm{FIX}(\Psi)
\end{aligned}
$$

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

$$\Psi : 2^\Sigma \to 2^\Sigma : T \mapsto (\mathfrak{B}[\![b]\!] \cap wlp[\![c]\!]T) \cup (\mathfrak{B}[\![\neg b]\!] \cap S)$$

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

Proof.

omitted □

Weakest Liberal Preconditions III

Example 10.11

Using Lemma 10.10, we want to determine the weakest liberal precondition for

$$\{?\}\ \underbrace{\texttt{while x} \neq 0 \land \texttt{x} \neq 1 \texttt{ do } \overbrace{\texttt{x := x-2}}^{c_0} \texttt{ end}}_{c}\ \{\texttt{x} = 1\}$$

i.e., $wlp[\![c]\!]S$ for $S := \mathfrak{B}[\![\texttt{x} = 1]\!]$.

Weakest Liberal Preconditions III

Example 10.11

Using Lemma 10.10, we want to determine the weakest liberal precondition for

$$\{?\}\ \underbrace{\texttt{while x} \neq 0 \wedge \texttt{x} \neq 1 \texttt{ do } \overbrace{\texttt{x := x-2}}^{c_0} \texttt{ end}}_{c}\ \{\texttt{x} = 1\}$$

i.e., $wlp[\![c]\!]S$ for $S := \mathfrak{B}[\![\texttt{x} = 1]\!]$.

- $wlp[\![c]\!]S = \mathrm{FIX}(\Psi)$ for $\Psi(T) = (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!]T) \cup \underbrace{(\mathfrak{B}[\![\texttt{x} \in \{0,1\}]\!] \cap S)}_{=S}$

Weakest Liberal Preconditions III

Example 10.11

Using Lemma 10.10, we want to determine the weakest liberal precondition for

$$\{?\}\ \underbrace{\texttt{while x} \neq 0 \wedge \texttt{x} \neq 1 \texttt{ do } \overbrace{\texttt{x := x-2}}^{c_0} \texttt{ end}}_{c}\ \{\texttt{x} = 1\}$$

i.e., $wlp[\![c]\!]S$ for $S := \mathfrak{B}[\![\texttt{x} = 1]\!]$.

- $wlp[\![c]\!]S = \mathrm{FIX}(\Psi)$ for $\Psi(T) = (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!]T) \cup \underbrace{(\mathfrak{B}[\![\texttt{x} \in \{0,1\}]\!] \cap S)}_{=s}$
- $wlp[\![c_0]\!]T = \{\sigma \in \Sigma \mid \sigma[\texttt{x} \mapsto \sigma(\texttt{x}) - 2] \in T\}$

Weakest Liberal Preconditions III

Example 10.11

Using Lemma 10.10, we want to determine the weakest liberal precondition for

$$\{?\}\ \underbrace{\texttt{while x} \neq 0 \wedge \texttt{x} \neq 1 \texttt{ do } \overbrace{\texttt{x := x-2}}^{c_0} \texttt{ end}}_{c}\ \{\texttt{x} = 1\}$$

i.e., $wlp[\![c]\!]S$ for $S := \mathfrak{B}[\![\texttt{x} = 1]\!]$.

- $wlp[\![c]\!]S = \mathrm{FIX}(\Psi)$ for $\Psi(T) = (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!]T) \cup \underbrace{(\mathfrak{B}[\![\texttt{x} \in \{0,1\}]\!] \cap S)}_{=S}$
- $wlp[\![c_0]\!]T = \{\sigma \in \Sigma \mid \sigma[\texttt{x} \mapsto \sigma(\texttt{x}) - 2] \in T\}$
- Fixpoint iteration (with initial value $\bigcap \emptyset = \Sigma$):

$$\begin{aligned}
\Psi(\Sigma) &= (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!]\Sigma) \cup S = \mathfrak{B}[\![\texttt{x} \neq 0]\!] \\
\Psi^2(\Sigma) &= (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!](\mathfrak{B}[\![\texttt{x} \neq 0]\!])) \cup S = \mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 2]\!] \\
\Psi^3(\Sigma) &= (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!](\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 2]\!])) \cup S = \mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 2 \wedge \texttt{x} \neq 4]\!] \\
&\vdots
\end{aligned}$$

Weakest Liberal Preconditions III

Example 10.11

Using Lemma 10.10, we want to determine the weakest liberal precondition for

$$\{?\}\ \underbrace{\texttt{while x} \neq 0 \wedge \texttt{x} \neq 1 \texttt{ do } \overbrace{\texttt{x := x-2}}^{c_0} \texttt{ end}}_{c}\ \{\texttt{x} = 1\}$$

i.e., $wlp[\![c]\!]S$ for $S := \mathfrak{B}[\![\texttt{x} = 1]\!]$.

- $wlp[\![c]\!]S = \mathrm{FIX}(\Psi)$ for $\Psi(T) = (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!]T) \cup \underbrace{(\mathfrak{B}[\![\texttt{x} \in \{0,1\}]\!] \cap S)}_{=S}$
- $wlp[\![c_0]\!]T = \{\sigma \in \Sigma \mid \sigma[\texttt{x} \mapsto \sigma(\texttt{x}) - 2] \in T\}$
- Fixpoint iteration (with initial value $\bigsqcap \emptyset = \Sigma$):

$$\begin{aligned}
\Psi(\Sigma) &= (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!]\Sigma) \cup S = \mathfrak{B}[\![\texttt{x} \neq 0]\!] \\
\Psi^2(\Sigma) &= (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!](\mathfrak{B}[\![\texttt{x} \neq 0]\!])) \cup S = \mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 2]\!] \\
\Psi^3(\Sigma) &= (\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 1]\!] \cap wlp[\![c_0]\!](\mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 2]\!])) \cup S = \mathfrak{B}[\![\texttt{x} \neq 0 \wedge \texttt{x} \neq 2 \wedge \texttt{x} \neq 4]\!] \\
&\vdots
\end{aligned}$$

$$\Rightarrow \mathrm{FIX}(\Psi) = \bigcap_{n \in \mathbb{N}} \Psi^n(\Sigma) = \{\sigma \in \Sigma \mid \sigma(\texttt{x}) \in \mathbb{Z}_{<0} \cup \{1, 3, 5, \ldots\}\}$$

Weakest Liberal Preconditions IV

Definition 10.12 (Expressivity of assertion languages)

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

Weakest Liberal Preconditions IV

Definition 10.12 (Expressivity of assertion languages)

An assertion language *Assn* is called expressive if it allows to "syntactify" weakest preconditions, that is, for every $c \in Cmd$ and $B \in Assn$, there exists $A_{c,B} \in Assn$ such that $A^I_{c,B} = wlp[\![c]\!]B^I$ for every $I \in Int$.

Theorem 10.13 (Expressivity of *Assn*)

Assn is expressive.

Weakest Liberal Preconditions IV

Definition 10.12 (Expressivity of assertion languages)

An assertion language *Assn* is called expressive if it allows to "syntactify" weakest preconditions, that is, for every $c \in Cmd$ and $B \in Assn$, there exists $A_{c,B} \in Assn$ such that $A^I_{c,B} = wlp[\![c]\!]B^I$ for every $I \in Int$.

Theorem 10.13 (Expressivity of *Assn*)

Assn is expressive.

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

Given $c \in Cmd$ and $B \in Assn$, construct $A_{c,B} \in Assn$ with $\sigma \models^I A_{c,B} \iff \mathfrak{C}[\![c]\!]\sigma \models^I B$ (for every $\sigma \in \Sigma$, $I \in Int$). For example:

$$A_{\texttt{skip},B} := B \qquad A_{x:=a,B} := B[x \mapsto a]$$
$$A_{c_1;c_2,B} := A_{c_1,A_{c_2,B}} \qquad \ldots$$

(for `while`: "Gödelisation" of sequences of intermediate states) □

Relative Completeness of Hoare Logic III

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

Lemma 10.14

For every $c \in \mathit{Cmd}$ *and* $B \in \mathit{Assn}$: $\vdash \{A_{c,B}\}\, c\, \{B\}$

Relative Completeness of Hoare Logic III

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

Lemma 10.14

For every $c \in \mathit{Cmd}$ *and* $B \in \mathit{Assn}$: $\vdash \{A_{c,B}\}\, c\, \{B\}$

Proof.

by structural induction over c (omitted) □

Relative Completeness of Hoare Logic III

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

Lemma 10.14

For every $c \in \mathit{Cmd}$ *and* $B \in \mathit{Assn}$: $\vdash \{A_{c,B}\}\, c\, \{B\}$

Proof.

by structural induction over c (omitted) □

Proof (Cook’s Completeness Theorem 10.7).

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

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

Relative Completeness of Hoare Logic III

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

Lemma 10.14

For every $c \in \mathit{Cmd}$ *and* $B \in \mathit{Assn}$: $\vdash \{A_{c,B}\}\, c\, \{B\}$

Proof.

by structural induction over c (omitted) □

Proof (Cook’s Completeness Theorem 10.7).

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 10.14: $\vdash \{A_{c,B}\}\, c\, \{B\}$

Relative Completeness of Hoare Logic III

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

Lemma 10.14

For every $c \in \mathit{Cmd}$ *and* $B \in \mathit{Assn}$: $\vdash \{A_{c,B}\}\, c\, \{B\}$

Proof.

by structural induction over c (omitted) □

Proof (Cook’s Completeness Theorem 10.7).

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 10.14: $\vdash \{A_{c,B}\}\, c\, \{B\}$
- Corollary 10.9: $\models \{A\}\, c\, \{B\} \quad \Rightarrow \quad \models (A \Rightarrow A_{c,B})$

Relative Completeness of Hoare Logic III

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

Lemma 10.14

For every $c \in \mathit{Cmd}$ *and* $B \in \mathit{Assn}$: $\vdash \{A_{c,B}\}\, c\, \{B\}$

Proof.

by structural induction over c (omitted) □

Proof (Cook’s Completeness Theorem 10.7).

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 10.14: $\vdash \{A_{c,B}\}\, c\, \{B\}$
- Corollary 10.9: $\models \{A\}\, c\, \{B\} \quad \Rightarrow \quad \models (A \Rightarrow A_{c,B})$
- $$\text{(cons)}\ \frac{\models (A \Rightarrow A_{c,B}) \qquad \{A_{c,B}\}\, c\, \{B\} \qquad \models (B \Rightarrow B)}{\{A\}\, c\, \{B\}}$$

□