Semantics and Verification of Software

Winter Semester 2017/18

Lecture 9: Axiomatic Semantics of WHILE I (Hoare Logic)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

http://moves.rwth-aachen.de/teaching/ws-1718/sv-sw/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Outline of Lecture 9

The Axiomatic Approach I

Example 9.1

- Let $c \in Cmd$ be given by

$$\texttt{s := 0; n := 1; while } \neg\texttt{(n>N) do s := s+n; n := n+1 end}$$

The Axiomatic Approach I

Example 9.1

- Let $c \in Cmd$ be given by

```
s := 0; n := 1; while ¬(n>N) do s := s+n; n := n+1 end
```

- How to show that, after termination of c,

$$\sigma(\mathtt{s}) = \sum_{k=1}^{\sigma(\mathtt{N})} k \quad ?$$

The Axiomatic Approach I

Example 9.1

- Let $c \in Cmd$ be given by

```
s := 0; n := 1; while ¬(n>N) do s := s+n; n := n+1 end
```

- How to show that, after termination of c,

$$\sigma(\mathtt{s}) = \sum_{k=1}^{\sigma(\mathtt{N})} k \quad ?$$

- "Running" c according to the operational semantics in insufficient: every change of $\sigma(\mathtt{N})$ requires a new proof

The Axiomatic Approach I

Example 9.1

- Let $c \in Cmd$ be given by

  $$\texttt{s := 0; n := 1; while } \neg\texttt{(n>N) do s := s+n; n := n+1 end}$$

- How to show that, after termination of c,

  $$\sigma(\texttt{s}) = \sum_{k=1}^{\sigma(\texttt{N})} k \quad ?$$

- "Running" c according to the operational semantics in insufficient: every change of $\sigma(\texttt{N})$ requires a new proof
- Wanted: a more abstract, "symbolic" way of reasoning

The Axiomatic Approach II

Example 9.1 (continued)

Obviously c satisfies the following assertions (after execution of the respective statement):

$$
\begin{array}{l}
\texttt{s := 0;} \\
\{\texttt{s} = 0\} \\
\texttt{n := 1;} \\
\{\texttt{s} = 0 \land \texttt{n} = 1\} \\
\texttt{while } \neg\texttt{(n>N) do s := s+n; n := n+1 end} \\
\{\texttt{s} = \sum_{k=1}^{\texttt{N}} k \land \texttt{n} > \texttt{N}\}
\end{array}
$$

where, e.g., "$\texttt{s} = 0$" means "$\sigma(\texttt{s}) = 0$ in the current state $\sigma \in \Sigma$"

The Axiomatic Approach III

How to prove the validity of assertions?

- Assertions following assignments are evident (“$s = 0$”)

The Axiomatic Approach III

How to prove the validity of assertions?

- Assertions following assignments are evident (“$\mathtt{s} = 0$”)
- Also, “$\mathtt{n} > \mathtt{N}$” follows directly from the loop’s execution condition

The Axiomatic Approach III

How to prove the validity of assertions?

- Assertions following assignments are evident (“$\mathtt{s} = \mathtt{0}$”)
- Also, “$\mathtt{n} > \mathtt{N}$” follows directly from the loop’s execution condition
- But how to obtain the final value of $\mathtt{s}$?

The Axiomatic Approach III

How to prove the validity of assertions?

- Assertions following assignments are evident (“$\mathtt{s} = 0$”)
- Also, “$\mathtt{n} > \mathtt{N}$” follows directly from the loop’s execution condition
- But how to obtain the final value of $\mathtt{s}$?
- Answer: after every loop iteration, the invariant $\mathtt{s} = \sum_{k=1}^{\mathtt{n}-1} k$ is satisfied

The Axiomatic Approach III

How to prove the validity of assertions?

- Assertions following assignments are evident ("$\mathtt{s} = 0$")
- Also, "$\mathtt{n} > \mathtt{N}$" follows directly from the loop's execution condition
- But how to obtain the final value of $\mathtt{s}$?
- Answer: after every loop iteration, the invariant $\mathtt{s} = \sum_{k=1}^{\mathtt{n}-1} k$ is satisfied
- Our proof system employs partial correctness properties of the form $\{A\}\, c\, \{B\}$ with assertions A, B and $c \in Cmd$

The Axiomatic Approach III

How to prove the validity of assertions?

- Assertions following assignments are evident (“$\mathtt{s} = 0$”)
- Also, “$\mathtt{n} > \mathtt{N}$” follows directly from the loop’s execution condition
- But how to obtain the final value of $\mathtt{s}$?
- Answer: after every loop iteration, the invariant $\mathtt{s} = \sum_{k=1}^{\mathtt{n}-1} k$ is satisfied
- Our proof system employs partial correctness properties of the form $\{A\}\, c\, \{B\}$ with assertions A, B and $c \in Cmd$
- Interpretation:

Validity of partial correctness property

$\{A\}\, c\, \{B\}$ is valid iff 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.

The Axiomatic Approach III

How to prove the validity of assertions?

- Assertions following assignments are evident (“$\mathtt{s} = 0$”)
- Also, “$\mathtt{n} > \mathtt{N}$” follows directly from the loop’s execution condition
- But how to obtain the final value of $\mathtt{s}$?
- Answer: after every loop iteration, the invariant $\mathtt{s} = \sum_{k=1}^{\mathtt{n}-1} k$ is satisfied
- Our proof system employs partial correctness properties of the form $\{A\}\, c\, \{B\}$ with assertions A, B and $c \in Cmd$
- Interpretation:

Validity of partial correctness property

$\{A\}\, c\, \{B\}$ is valid iff 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.

- “Partial” means that nothing is said about c if it fails to terminate

The Axiomatic Approach III

How to prove the validity of assertions?

- Assertions following assignments are evident ("$\texttt{s} = 0$")
- Also, "$\texttt{n} > \texttt{N}$" follows directly from the loop's execution condition
- But how to obtain the final value of $\texttt{s}$?
- Answer: after every loop iteration, the invariant $\texttt{s} = \sum_{k=1}^{\texttt{n}-1} k$ is satisfied
- Our proof system employs partial correctness properties of the form $\{A\}\, c\, \{B\}$ with assertions A, B and $c \in Cmd$
- Interpretation:

Validity of partial correctness property

$\{A\}\, c\, \{B\}$ is valid iff 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.

- "Partial" means that nothing is said about c if it fails to terminate
- In particular, $\{\mathsf{true}\}$ `while true do skip end` $\{\mathsf{false}\}$ is a valid property

Outline of Lecture 9

Syntax of Assertion Language I

Assertions = Boolean expressions + logical variables

- to memorize previous values of program variables
- to formulate more involved state properties

Syntax of Assertion Language I

Assertions = Boolean expressions + logical variables

- to memorize previous values of program variables
- to formulate more involved state properties

Syntactic categories:

Category	Domain	Meta variable(s)
Logical variables	*LVar*	i
Arithmetic expressions with logical variables	*LExp*	a
Assertions	*Assn*	A, B, C

Syntax of Assertion Language II

Definition 9.2 (Syntax of assertions)

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

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

Syntax of Assertion Language II

Definition 9.2 (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:

$$\begin{aligned} A_1 \Rightarrow A_2 &:= \neg A_1 \vee A_2 \\ \exists i.A &:= \neg(\forall i.\neg A) \\ a_1 \geq a_2 &:= a_1{>}a_2 \vee a_1{=}a_2 \\ &\vdots \end{aligned}$$

Outline of Lecture 9

Semantics of *LExp*

The semantics now additionally depends on values of logical variables:

Definition 9.3 (Semantics of *LExp*)

An interpretation is an element of the set $Int := \{I \mid I : LVar \to \mathbb{Z}\}$. The value of an arithmetic expressions with logical variables is given by the functional

$$\mathfrak{L}[\![.]\!] : LExp \to (Int \to (\Sigma \to \mathbb{Z}))$$

where

$$\begin{aligned} \mathfrak{L}[\![z]\!]I\sigma &:= z & \mathfrak{L}[\![a_1+a_2]\!]I\sigma &:= \mathfrak{L}[\![a_1]\!]I\sigma + \mathfrak{L}[\![a_2]\!]I\sigma \\ \mathfrak{L}[\![x]\!]I\sigma &:= \sigma(x) & \mathfrak{L}[\![a_1-a_2]\!]I\sigma &:= \mathfrak{L}[\![a_1]\!]I\sigma - \mathfrak{L}[\![a_2]\!]I\sigma \\ \mathfrak{L}[\![i]\!]I\sigma &:= I(i) & \mathfrak{L}[\![a_1*a_2]\!]I\sigma &:= \mathfrak{L}[\![a_1]\!]I\sigma \cdot \mathfrak{L}[\![a_2]\!]I\sigma \end{aligned}$$

Semantics of *LExp*

The semantics now additionally depends on values of logical variables:

Definition 9.3 (Semantics of *LExp*)

An interpretation is an element of the set $Int := \{I \mid I : LVar \to \mathbb{Z}\}$. The value of an arithmetic expressions with logical variables is given by the functional

$$\mathfrak{L}[\![.]\!] : LExp \to (Int \to (\Sigma \to \mathbb{Z}))$$

where

$$\begin{aligned} \mathfrak{L}[\![z]\!]I\sigma &:= z & \mathfrak{L}[\![a_1+a_2]\!]I\sigma &:= \mathfrak{L}[\![a_1]\!]I\sigma + \mathfrak{L}[\![a_2]\!]I\sigma \\ \mathfrak{L}[\![x]\!]I\sigma &:= \sigma(x) & \mathfrak{L}[\![a_1-a_2]\!]I\sigma &:= \mathfrak{L}[\![a_1]\!]I\sigma - \mathfrak{L}[\![a_2]\!]I\sigma \\ \mathfrak{L}[\![i]\!]I\sigma &:= I(i) & \mathfrak{L}[\![a_1*a_2]\!]I\sigma &:= \mathfrak{L}[\![a_1]\!]I\sigma \cdot \mathfrak{L}[\![a_2]\!]I\sigma \end{aligned}$$

Definition 6.1 (denotational semantics of arithmetic expressions) implies:

Corollary 9.4

For every $a \in AExp$ (without logical variables), $I \in Int$, and $\sigma \in \Sigma$:

$$\mathfrak{L}[\![a]\!]I\sigma = \mathfrak{A}[\![a]\!]\sigma.$$

Semantics of Assertions I

- Formalized by a satisfaction relation of the form

$$\sigma \models A$$

(where $\sigma \in \Sigma$ and $A \in Assn$)

Semantics of Assertions I

- Formalized by a satisfaction relation of the form

$$\sigma \models A$$

(where $\sigma \in \Sigma$ and $A \in Assn$)
- Non-terminating computations captured by undefined state $\bot$:

$$\Sigma_\bot := \Sigma \cup \{\bot\}$$

Semantics of Assertions I

- Formalized by a satisfaction relation of the form

$$\sigma \models A$$

(where $\sigma \in \Sigma$ and $A \in Assn$)

- Non-terminating computations captured by undefined state $\perp$:

$$\Sigma_\perp := \Sigma \cup \{\perp\}$$

- Modification of interpretations (in analogy to program states):

$$I[i \mapsto z](j) := \begin{cases} z & \text{if } j = i \\ I(j) & \text{otherwise} \end{cases}$$

Semantics of Assertions II

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 9.5 (Semantics of assertions)

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

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

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 III

Example 9.6

The following assertion expresses that, in the current state $\sigma \in \Sigma$, $\sigma(y)$ is the greatest divisor of $\sigma(x)$:

$$(\exists i. i > 1 \wedge i * y = x) \wedge \forall j. \forall k. (j > 1 \wedge j * k = x \Rightarrow k \leq y)$$

Semantics of Assertions III

Example 9.6

The following assertion expresses that, in the current state $\sigma \in \Sigma$, $\sigma(y)$ is the greatest divisor of $\sigma(x)$:

$$(\exists i. i > 1 \wedge i * y = x) \wedge \forall j. \forall k. (j > 1 \wedge j * k = x \Rightarrow k \leq y)$$

In analogy to Corollary 9.4, Definition 6.2 (denotational semantics of Boolean expressions) yields:

Corollary 9.7

For every $b \in BExp$ (without logical variables), $I \in Int$, and $\sigma \in \Sigma$:

$$\sigma \models^I b \iff \mathfrak{B}[\![b]\!]\sigma = \text{true}.$$

Semantics of Assertions IV

Definition 9.8 (Extension)

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

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

Note that, for every $A \in \mathit{Assn}$ and $I \in \mathit{Int}$, $\perp \in A^I$.

Semantics of Assertions IV

Definition 9.8 (Extension)

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

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

Note that, for every $A \in Assn$ and $I \in Int$, $\perp \in A^I$.

Example 9.9

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

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

Outline of Lecture 9

Partial Correctness Properties

Definition 9.10 (Partial correctness properties)

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

- An expression of the form $\{A\}\, c\, \{B\}$ is called a partial correctness property with precondition A and postcondition B.

Partial Correctness Properties

Definition 9.10 (Partial correctness properties)

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

- An expression of the form $\{A\}\, c\, \{B\}$ is called a partial correctness property with precondition A and postcondition B.
- Given $\sigma \in \Sigma_\perp$ and $I \in Int$, we let
$$\sigma \models^I \{A\}\, c\, \{B\}$$
if $\sigma \models^I A$ implies $\mathfrak{C}[\![c]\!]\sigma \models^I B$ (or equivalently: $\sigma \in A^I \Rightarrow \mathfrak{C}[\![c]\!]\sigma \in B^I$).

Partial Correctness Properties

Definition 9.10 (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 with precondition A and postcondition B.
- Given $\sigma \in \Sigma_\perp$ and $I \in \mathit{Int}$, we let
$$\sigma \models^I \{A\}\, c\, \{B\}$$
if $\sigma \models^I A$ implies $\mathfrak{C}[\![c]\!]\sigma \models^I B$ (or equivalently: $\sigma \in A^I \Rightarrow \mathfrak{C}[\![c]\!]\sigma \in B^I$).
- $\{A\}\, c\, \{B\}$ is called valid in I (notation: $\models^I \{A\}\, c\, \{B\}$) if $\sigma \models^I \{A\}\, c\, \{B\}$ for every $\sigma \in \Sigma_\perp$ (or equivalently: $\mathfrak{C}[\![c]\!]A^I \subseteq B^I$).

Partial Correctness Properties

Definition 9.10 (Partial correctness properties)

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

- An expression of the form $\{A\}\, c\, \{B\}$ is called a partial correctness property with precondition A and postcondition B.
- Given $\sigma \in \Sigma_\bot$ and $I \in Int$, we let
$$\sigma \models^I \{A\}\, c\, \{B\}$$
if $\sigma \models^I A$ implies $\mathfrak{C}[\![c]\!]\sigma \models^I B$ (or equivalently: $\sigma \in A^I \Rightarrow \mathfrak{C}[\![c]\!]\sigma \in B^I$).
- $\{A\}\, c\, \{B\}$ is called valid in I (notation: $\models^I \{A\}\, c\, \{B\}$) if $\sigma \models^I \{A\}\, c\, \{B\}$ for every $\sigma \in \Sigma_\bot$ (or equivalently: $\mathfrak{C}[\![c]\!]A^I \subseteq B^I$).
- $\{A\}\, c\, \{B\}$ is called valid (notation: $\models \{A\}\, c\, \{B\}$) if $\models^I \{A\}\, c\, \{B\}$ for every $I \in Int$.

Outline of Lecture 9

A Valid Partial Correctness Property

Example 9.11

- Let $\texttt{x} \in \mathit{Var}$ and $i \in \mathit{LVar}$. We have to show:

$$\models \{i \leq \texttt{x}\}\ \texttt{x := x+1}\ \{i < \texttt{x}\}$$

A Valid Partial Correctness Property

Example 9.11

- Let $\mathtt{x} \in \mathit{Var}$ and $i \in \mathit{LVar}$. We have to show:
$$\models \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$
- According to Definition 9.10, this is equivalent to
$$\sigma \models^I \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$
for every $\sigma \in \Sigma_\perp$ and $I \in \mathit{Int}$

A Valid Partial Correctness Property

Example 9.11

- Let $\texttt{x} \in \mathit{Var}$ and $i \in \mathit{LVar}$. We have to show:

$$\models \{i \leq \texttt{x}\}\, \texttt{x := x+1}\, \{i < \texttt{x}\}$$

- According to Definition 9.10, this is equivalent to

$$\sigma \models^I \{i \leq \texttt{x}\}\, \texttt{x := x+1}\, \{i < \texttt{x}\}$$

  for every $\sigma \in \Sigma_\perp$ and $I \in \mathit{Int}$
- For $\sigma = \perp$ this is trivial. So let $\sigma \in \Sigma$:

$$\sigma \models^I (i \leq \texttt{x})$$

A Valid Partial Correctness Property

Example 9.11

- Let $\mathtt{x} \in \mathit{Var}$ and $i \in \mathit{LVar}$. We have to show:

$$\models \{i \leq \mathtt{x}\}\,\mathtt{x}\ \ \mathtt{:=}\ \ \mathtt{x+1}\,\{i < \mathtt{x}\}$$

- According to Definition 9.10, this is equivalent to

$$\sigma \models^I \{i \leq \mathtt{x}\}\,\mathtt{x}\ \ \mathtt{:=}\ \ \mathtt{x+1}\,\{i < \mathtt{x}\}$$

  for every $\sigma \in \Sigma_\perp$ and $I \in \mathit{Int}$
- For $\sigma = \perp$ this is trivial. So let $\sigma \in \Sigma$:

$$\begin{aligned} & \sigma \models^I (i \leq \mathtt{x}) \\ \Rightarrow\ & \mathfrak{L}[\![i]\!] I\sigma \leq \mathfrak{L}[\![\mathtt{x}]\!] I\sigma && \text{(Definition 9.5)} \end{aligned}$$

A Valid Partial Correctness Property

Example 9.11

- Let $\mathtt{x} \in \mathit{Var}$ and $i \in \mathit{LVar}$. We have to show:
$$\models \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$
- According to Definition 9.10, this is equivalent to
$$\sigma \models^I \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$
for every $\sigma \in \Sigma_\bot$ and $I \in \mathit{Int}$
- For $\sigma = \bot$ this is trivial. So let $\sigma \in \Sigma$:
$$\begin{aligned} & \sigma \models^I (i \leq \mathtt{x}) & \\ \Rightarrow\ & \mathfrak{L}[\![i]\!]I\sigma \leq \mathfrak{L}[\![\mathtt{x}]\!]I\sigma & \text{(Definition 9.5)} \\ \Rightarrow\ & I(i) \leq \sigma(\mathtt{x}) & \text{(Definition 9.3)} \end{aligned}$$

A Valid Partial Correctness Property

Example 9.11

- Let $\mathtt{x} \in \mathit{Var}$ and $i \in \mathit{LVar}$. We have to show:

$$\models \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$

- According to Definition 9.10, this is equivalent to

$$\sigma \models^I \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$

  for every $\sigma \in \Sigma_\perp$ and $I \in \mathit{Int}$
- For $\sigma = \perp$ this is trivial. So let $\sigma \in \Sigma$:

$$\begin{aligned}
& \sigma \models^I (i \leq \mathtt{x}) & \\
\Rightarrow\ & \mathfrak{L}[\![i]\!] I\sigma \leq \mathfrak{L}[\![\mathtt{x}]\!] I\sigma & \text{(Definition 9.5)} \\
\Rightarrow\ & I(i) \leq \sigma(\mathtt{x}) & \text{(Definition 9.3)} \\
\Rightarrow\ & I(i) < \sigma(\mathtt{x}) + 1 & \\
& \quad = (\mathfrak{C}[\![\mathtt{x\ :=\ x+1}]\!]\sigma)(\mathtt{x}) &
\end{aligned}$$

A Valid Partial Correctness Property

Example 9.11

- Let $\mathtt{x} \in \mathit{Var}$ and $i \in \mathit{LVar}$. We have to show:

$$\models \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$

- According to Definition 9.10, this is equivalent to

$$\sigma \models^I \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$

  for every $\sigma \in \Sigma_\perp$ and $I \in \mathit{Int}$
- For $\sigma = \perp$ this is trivial. So let $\sigma \in \Sigma$:

$$\begin{array}{lr} \quad \sigma \models^I (i \leq \mathtt{x}) & \\ \Rightarrow \mathfrak{L}[\![i]\!]I\sigma \leq \mathfrak{L}[\![\mathtt{x}]\!]I\sigma & \text{(Definition 9.5)} \\ \Rightarrow I(i) \leq \sigma(\mathtt{x}) & \text{(Definition 9.3)} \\ \Rightarrow I(i) < \sigma(\mathtt{x}) + 1 & \\ \qquad\quad = (\mathfrak{C}[\![\mathtt{x\ :=\ x+1}]\!]\sigma)(\mathtt{x}) & \\ \Rightarrow \mathfrak{C}[\![\mathtt{x\ :=\ x+1}]\!]\sigma \models^I (i < \mathtt{x}) & \end{array}$$

A Valid Partial Correctness Property

Example 9.11

- Let $\mathtt{x} \in \mathit{Var}$ and $i \in \mathit{LVar}$. We have to show:

$$\models \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$

- According to Definition 9.10, this is equivalent to

$$\sigma \models^{I} \{i \leq \mathtt{x}\}\, \mathtt{x\ :=\ x+1}\, \{i < \mathtt{x}\}$$

  for every $\sigma \in \Sigma_\perp$ and $I \in \mathit{Int}$
- For $\sigma = \perp$ this is trivial. So let $\sigma \in \Sigma$:

$$\begin{aligned}
& \sigma \models^{I} (i \leq \mathtt{x}) \\
\Rightarrow\ & \mathfrak{L}[\![i]\!]I\sigma \leq \mathfrak{L}[\![\mathtt{x}]\!]I\sigma && \text{(Definition 9.5)} \\
\Rightarrow\ & I(i) \leq \sigma(\mathtt{x}) && \text{(Definition 9.3)} \\
\Rightarrow\ & I(i) < \sigma(\mathtt{x}) + 1 \\
& \quad = (\mathfrak{C}[\![\mathtt{x\ :=\ x+1}]\!]\sigma)(\mathtt{x}) \\
\Rightarrow\ & \mathfrak{C}[\![\mathtt{x\ :=\ x+1}]\!]\sigma \models^{I} (i < \mathtt{x}) \\
\Rightarrow\ & \text{claim}
\end{aligned}$$

Outline of Lecture 9

Hoare Logic I

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 9.12 (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.

Hoare Logic II

Example 9.13 (Factorial program)

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

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

(on the board)

Hoare Logic II

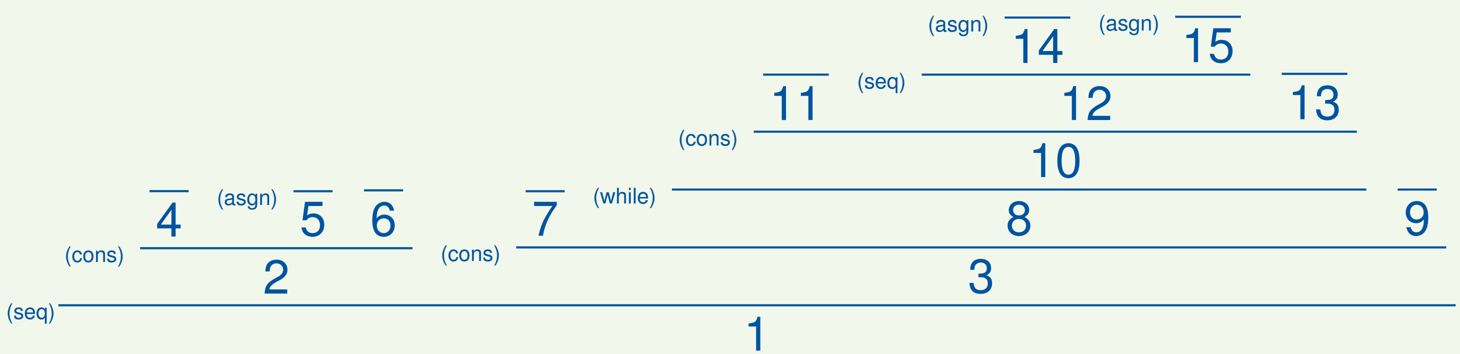
Example 9.13 (Factorial program)

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

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

(on the board)

Structure of the proof:



Hoare Logic III

Example 9.13 (continued)

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

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