Semantics and Verification of Software

Lecture 8: Axiomatic Semantics of WHILE

Thomas Noll

Lehrstuhl für Informatik 2
RWTH Aachen University
noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw/

Summer semester 2007

Outline

The Fixpoint Theorem

Theorem (Fixpoint Theorem by Tarski and Knaster)

Let $(D, \sqsubseteq)$ *be a CCPO and* $F : D \to D$ *continuous. Then*

$$\mathsf{fix}(F) := \bigsqcup \left\{ F^n \left(\bigsqcup \emptyset \right) \mid n \in \mathbb{N} \right\}$$

is the least fixpoint of F *where*

$$F^0(d) := d \text{ and } F^{n+1}(d) := F(F^n(d)).$$

Application to fix(Φ)

Altogether this completes the definition of $\mathfrak{C}[\![.]\!]$. In particular, for the `while` statement we obtain:

Corollary

Let $b \in BExp$, $c \in Cmd$, *and* $\Phi(f) := \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$. *Then*

$$\mathsf{graph}(\mathsf{fix}(\Phi)) = \bigcup_{n \in \mathbb{N}} \mathsf{graph}(\Phi^n(f_\emptyset))$$

Proof.

Using

- Lemma 5.12
  (($\Sigma \dashrightarrow \Sigma, \sqsubseteq$) CCPO with least element $f_\emptyset$; LUB = union of graphs)
- Lemma 6.7 (Φ continuous)
- Theorem 7.1 (Fixpoint Theorem)

□

Outline

1. Repetition: The Fixpoint Theorem
2. Repetition: Equivalence of Operational and Denotational Semantics
3. The Axiomatic Approach
4. The Assertion Language
5. Partial Correctness Properties

Equivalence of Semantics I

Remember: in Def. 4.3, $\mathfrak{O}[\![.]\!] : Cmd \to (\Sigma \dashrightarrow \Sigma)$ was given by

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

Theorem (Coincidence Theorem)

For every $c \in Cmd$,

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

i.e., $\mathfrak{O}[\![.]\!] = \mathfrak{C}[\![.]\!]$.

Equivalence of Semantics II

The proof of Theorem 7.4 employs the following auxiliary propositions:

Lemma

1. *For every* $a \in AExp$, $\sigma \in \Sigma$, *and* $z \in \mathbb{Z}$*:*
$$\langle a, \sigma \rangle \to z \iff \mathfrak{A}[\![a]\!](\sigma) = z.$$
2. *For every* $b \in BExp$, $\sigma \in \Sigma$, *and* $t \in \mathbb{B}$*:*
$$\langle b, \sigma \rangle \to t \iff \mathfrak{B}[\![b]\!](\sigma) = t.$$

Proof.

1. see Exercise 3.2
2. analogously

□

Equivalence of Semantics III

Proof (Theorem 7.4).

We have to show that

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

- $\Rightarrow$ by structural induction over the derivation tree of $\langle c, \sigma \rangle \to \sigma'$
- $\Leftarrow$ by structural induction over c (with a nested complete induction over fixpoint index n)

(on the board) $\square$

Reminder: Operational/Denotational Semantics

Definition (Operational semantics of statements)

Execution relation $\langle c, \sigma \rangle \to \sigma'$:

$$\frac{}{\langle \texttt{skip}, \sigma \rangle \to \sigma}\text{(skip)} \qquad \frac{\langle a, \sigma \rangle \to z}{\langle x \texttt{ := } a, \sigma \rangle \to \sigma[x \mapsto z]}\text{(asgn)}$$

$$\frac{\langle c_1, \sigma \rangle \to \sigma' \quad \langle c_2, \sigma' \rangle \to \sigma''}{\langle c_1 \texttt{;} c_2, \sigma \rangle \to \sigma''}\text{(seq)} \qquad \frac{\langle b, \sigma \rangle \to \mathsf{true} \quad \langle c_1, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'}\text{(if–t)}$$

$$\frac{\langle b, \sigma \rangle \to \mathsf{false} \quad \langle c_2, \sigma \rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \rangle \to \sigma'}\text{(if–f)} \qquad \frac{\langle b, \sigma \rangle \to \mathsf{false}}{\langle \texttt{while } b \texttt{ do } c, \sigma \rangle \to \sigma}\text{(wh–f)}$$

$$\frac{\langle b, \sigma \rangle \to \mathsf{true} \quad \langle c, \sigma \rangle \to \sigma' \quad \langle \texttt{while } b \texttt{ do } c, \sigma' \rangle \to \sigma''}{\langle \texttt{while } b \texttt{ do } c, \sigma \rangle \to \sigma''}\text{(wh–t)}$$

Definition (Denotational semantics of statements)

Denotational semantic functional for statements $\mathfrak{C}[\![.]\!] : Cmd \to (\Sigma \dashrightarrow \Sigma)$:

$$\begin{aligned}
\mathfrak{C}[\![\texttt{skip}]\!] &:= \mathsf{id}_\Sigma \\
\mathfrak{C}[\![x \texttt{ := } a]\!]\sigma &:= \sigma[x \mapsto \mathfrak{A}[\![a]\!]\sigma] \\
\mathfrak{C}[\![c_1 \texttt{;} c_2]\!] &:= \mathfrak{C}[\![c_2]\!] \circ \mathfrak{C}[\![c_1]\!] \\
\mathfrak{C}[\![\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2]\!] &:= \mathsf{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c_1]\!], \mathfrak{C}[\![c_2]\!]) \\
\mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] &:= \mathsf{fix}(\Phi)
\end{aligned}$$

where $\Phi : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$

Outline

Example 8.1

- Let $c \in Cmd$ be given by

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

- How to show that, after termination of c, $\sigma(\texttt{s}) = \sum_{i=1}^{\sigma(\texttt{N})} i$?
- "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

Example 8.1

- Let $c \in \mathit{Cmd}$ be given by

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

- How to show that, after termination of c, $\sigma(\mathtt{s}) = \sum_{i=1}^{\sigma(\mathtt{N})} i$?
- "Running" c according to the operational semantics in insufficient: every change of $\sigma(\mathtt{N})$ requires a new proof
- Wanted: a more abstract, "symbolic" way of reasoning

The Axiomatic Approach I

Example 8.1

- Let $c \in Cmd$ be given by

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

- How to show that, after termination of c, $\sigma(\mathtt{s}) = \sum_{i=1}^{\sigma(\mathtt{N})} i$?
- "Running" c according to the operational semantics in insufficient: every change of $\sigma(\mathtt{N})$ requires a new proof
- Wanted: a more abstract, "symbolic" way of reasoning

The Axiomatic Approach I

Example 8.1

- Let $c \in Cmd$ be given by

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

- How to show that, after termination of c, $\sigma(\texttt{s}) = \sum_{i=1}^{\sigma(\texttt{N})} i$?
- "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

Example 8.1 (continued)

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

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

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 (“$\mathtt{s} = 0$”)
- 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 $\mathtt{s} = \sum_{i=1}^{\mathtt{n}-1}$ is satisfied
- Proof system employs partial correctness properties of the form $\{A\}\,c\,\{B\}$ with assertions A, B and $c \in \mathit{Cmd}$
- Interpretation:

Validity of property $\{A\}\,c\,\{B\}$

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,

$$\{\mathsf{true}\}\,\mathtt{while\ true\ do\ skip}\,\{\mathsf{false}\}$$

  is a valid property

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 $\mathtt{s} = \sum_{i=1}^{\mathtt{n}-1}$ is satisfied
- Proof system employs partial correctness properties of the form $\{A\}\,c\,\{B\}$ with assertions A, B and $c \in Cmd$
- Interpretation:

Validity of property $\{A\}\,c\,\{B\}$

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,

$$\{\mathsf{true}\}\,\mathtt{while\ true\ do\ skip}\,\{\mathsf{false}\}$$

is a valid property

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 s?
- Answer: after every loop iteration, the invariant $\mathtt{s} = \sum_{i=1}^{\mathtt{n}-1}$ is satisfied
- Proof system employs partial correctness properties of the form $\{A\}\,c\,\{B\}$ with assertions A, B and $c \in Cmd$
- Interpretation:

Validity of property $\{A\}\,c\,\{B\}$

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,

$$\{\mathsf{true}\}\,\mathtt{while\ true\ do\ skip}\,\{\mathsf{false}\}$$

  is a valid property

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 s?
- Answer: after every loop iteration, the invariant $\texttt{s} = \sum_{i=1}^{\texttt{n}-1}$ is satisfied
- Proof system employs partial correctness properties of the form $\{A\}\,c\,\{B\}$ with assertions A, B and $c \in Cmd$
- Interpretation:

Validity of property $\{A\}\,c\,\{B\}$

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,

$$\{\mathsf{true}\}\,\texttt{while true do skip}\,\{\mathsf{false}\}$$

is a valid property

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 s?
- Answer: after every loop iteration, the invariant $\mathtt{s} = \sum_{i=1}^{\mathtt{n}-1}$ is satisfied
- Proof system employs partial correctness properties of the form $\{A\}\,c\,\{B\}$ with assertions A, B and $c \in Cmd$
- Interpretation:

Validity of property $\{A\}\,c\,\{B\}$

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,

$$\{\mathsf{true}\}\,\mathtt{while\ true\ do\ skip}\,\{\mathsf{false}\}$$

is a valid property

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 $\mathtt{s} = \sum_{i=1}^{\mathtt{n}-1}$ is satisfied
- Proof system employs partial correctness properties of the form $\{A\}\,c\,\{B\}$ with assertions A, B and $c \in \mathit{Cmd}$
- Interpretation:

Validity of property $\{A\}\,c\,\{B\}$

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,

$$\{\mathsf{true}\}\,\mathtt{while\ true\ do\ skip}\,\{\mathsf{false}\}$$

is a valid property

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 s?
- Answer: after every loop iteration, the invariant $\texttt{s} = \sum_{i=1}^{\texttt{n}-1}$ is satisfied
- Proof system employs partial correctness properties of the form $\{A\}\,c\,\{B\}$ with assertions A, B and $c \in \mathit{Cmd}$
- Interpretation:

Validity of property $\{A\}\,c\,\{B\}$

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,

$$\{\mathsf{true}\}\,\texttt{while true do skip}\,\{\mathsf{false}\}$$

is a valid property

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 `s`?
- Answer: after every loop iteration, the invariant $\texttt{s} = \sum_{i=1}^{\texttt{n}-1}$ is satisfied
- Proof system employs partial correctness properties of the form $\{A\}\,c\,\{B\}$ with assertions A, B and $c \in Cmd$
- Interpretation:

Validity of property $\{A\}\,c\,\{B\}$

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,

  $$\{\mathsf{true}\}\,\texttt{while true do skip}\,\{\mathsf{false}\}$$

  is a valid property

Outline

Syntax of Assertion Language I

Assertions = Boolean expressions + logical variables
(to memorize previous values of program variables)

Syntactic categories:

Category	Domain	Meta variable
Logical variables	$LVar$	i
Arithmetic expressions with log. var.	$LExp$	a
Assertions	$Assn$	A, B, C

Syntax of Assertion Language I

Assertions = Boolean expressions + logical variables
(to memorize previous values of program variables)

Syntactic categories:

Category	Domain	Meta variable
Logical variables	$LVar$	i
Arithmetic expressions with log. var.	$LExp$	a
Assertions	$Assn$	A, B, C

Definition 8.2 (Syntax of assertions)

The syntax of $Assn$ is defined by the following context–free grammar:

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

Abbreviations:

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

Syntax of Assertion Language II

Definition 8.2 (Syntax of assertions)

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

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

Abbreviations:

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

Semantics of *LExp*

Semantics now additionally depends on values of logical variables:

Definition 8.3 (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

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

Def. 4.6 immediately implies:

Corollary 8.4

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 *LExp*

Semantics now additionally depends on values of logical variables:

Definition 8.3 (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

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

Def. 4.6 immediately implies:

Corollary 8.4

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 $A \in Assn$)

- Non–terminating computations captured by undefined state $\bot$:

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

- 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 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 $\bot$:

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

- 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 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 $\bot$:

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

- 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:

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

Definition 8.5 (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:

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

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

In analogy to Corollary 8.4, Def. 4.7 yields:

Corollary 8.6

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

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

Definition 8.7 (Extension)

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

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

Semantics of Assertions III

In analogy to Corollary 8.4, Def. 4.7 yields:

Corollary 8.6

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

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

Definition 8.7 (Extension)

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

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

Outline

Partial Correctness Properties I

Definition 8.8 (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: $\sigma \in A^I \implies \mathfrak{C}[\![c]\!]\sigma \in B^I$).
- $\{A\}\,c\,\{B\}$ 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: $\mathfrak{C}[\![c]\!]A^I \subseteq B^I$).
- $\{A\}\,c\,\{B\}$ is called valid (notation: $\models \{A\}\,c\,\{B\}$) if $\models^I \{A\}\,c\,\{B\}$ for every $I \in Int$.

Partial Correctness Properties I

Definition 8.8 (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: $\sigma \in A^I \implies \mathfrak{C}[\![c]\!]\sigma \in B^I$).
- $\{A\}\,c\,\{B\}$ 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: $\mathfrak{C}[\![c]\!]A^I \subseteq B^I$).
- $\{A\}\,c\,\{B\}$ is called valid (notation: $\models \{A\}\,c\,\{B\}$) if $\models^I \{A\}\,c\,\{B\}$ for every $I \in Int$.

Partial Correctness Properties I

Definition 8.8 (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: $\sigma \in A^I \implies \mathfrak{C}[\![c]\!]\sigma \in B^I$).
- $\{A\}\, c\, \{B\}$ 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: $\mathfrak{C}[\![c]\!]A^I \subseteq B^I$).
- $\{A\}\, c\, \{B\}$ is called valid (notation: $\models \{A\}\, c\, \{B\}$) if $\models^I \{A\}\, c\, \{B\}$ for every $I \in Int$.

Partial Correctness Properties I

Definition 8.8 (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: $\sigma \in A^I \implies \mathfrak{C}[\![c]\!]\sigma \in B^I$).
- $\{A\}\,c\,\{B\}$ 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 $I \in Int$.