

Semantics and Verification of Software

Lecture 9: Axiomatic Semantics of WHILE I (Introduction)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)



noll@cs.rwth-aachen.de

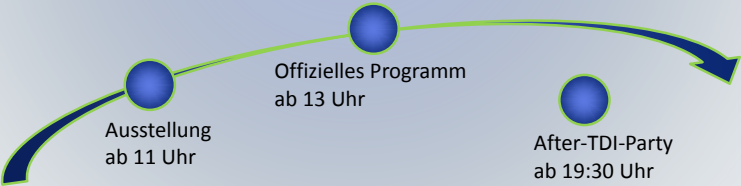
<http://www-i2.informatik.rwth-aachen.de/i2/svsw11/>

Winter Semester 2011/12

2. Dezember 2011
Informatik-Zentrum
Ahornstraße 55
Aula 2 / Foyer



Tag der Informatik 2011



Ausstellung
ab 11 Uhr

Offizielles Programm
ab 13 Uhr

After-TDI-Party
ab 19:30 Uhr

- 1 The Axiomatic Approach
- 2 The Assertion Language
- 3 Semantics of Assertions
- 4 Partial Correctness Properties
- 5 A Valid Partial Correctness Property

Example 9.1

- Let $c \in \text{Cmd}$ be given by
$$s:=0; n:=1; \text{ while } \neg(n>N) \text{ do } (s:=s+n; n:=n+1)$$
- How to show that, after termination of c , $\sigma(s) = \sum_{k=1}^{\sigma(N)} k$?
- “Running” c according to the operational semantics is insufficient: every change of $\sigma(N)$ requires a **new proof**
- Wanted: a more abstract, “**symbolic**” way of reasoning

Example 9.1 (continued)

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

```
s:=0;  
{s = 0}  
n:=1;  
{s = 0 ∧ n = 1}  
while ¬(n>N) do (s:=s+n; n:=n+1)  
{s =  $\sum_{k=1}^N k$  ∧ n > N}
```

where, e.g., “ $s = 0$ ” means “ $\sigma(s) = 0$ in the current state $\sigma \in \Sigma$ ”

The Axiomatic Approach III

How to prove the **validity** of assertions?

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

$\{\text{true}\} \text{while true do skip} \{\text{false}\}$

is a **valid** property

- 1 The Axiomatic Approach
- 2 The Assertion Language
- 3 Semantics of Assertions
- 4 Partial Correctness Properties
- 5 A Valid Partial Correctness Property

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

Syntactic categories:

Category	Domain	Meta variable(s)
Logical variables	<i>LVar</i>	<i>i</i>
Arithmetic expressions with log. var.	<i>LExp</i>	<i>a</i>
Assertions	<i>Assn</i>	<i>A, B, C</i>

Definition 9.2 (Syntax of assertions)

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

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

Abbreviations:

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

- 1 The Axiomatic Approach
- 2 The Assertion Language
- 3 Semantics of Assertions**
- 4 Partial Correctness Properties
- 5 A Valid Partial Correctness Property

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

$$Int := \{I \mid I : LVar \rightarrow \mathbb{Z}\}.$$

The **value of an arithmetic expressions with logical variables** is given by the functional

$$\mathcal{L}[\cdot] : LExp \rightarrow (Int \rightarrow (\Sigma \rightarrow \mathbb{Z}))$$

where

$$\begin{array}{ll} \mathcal{L}[z]/\sigma := z & \mathcal{L}[a_1 + a_2]/\sigma := \mathcal{L}[a_1]/\sigma + \mathcal{L}[a_2]/\sigma \\ \mathcal{L}[x]/\sigma := \sigma(x) & \mathcal{L}[a_1 - a_2]/\sigma := \mathcal{L}[a_1]/\sigma - \mathcal{L}[a_2]/\sigma \\ \mathcal{L}[i]/\sigma := I(i) & \mathcal{L}[a_1 * a_2]/\sigma := \mathcal{L}[a_1]/\sigma * \mathcal{L}[a_2]/\sigma \end{array}$$

Def. 5.1 (denotational semantics of arithmetic expressions) implies:

Corollary 9.4

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

$$\mathcal{L}[a]/\sigma = \mathcal{A}[a]\sigma.$$

- 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):

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

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

Definition 9.5 (Semantics of assertions)

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

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

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

Example 9.6

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

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

In analogy to Corollary 9.4, Def. 5.2 (denotational semantics of Boolean expressions) yields:

Corollary 9.7

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

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

Definition 9.8 (Extension)

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

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

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

Example 9.9

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

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

- 1 The Axiomatic Approach
- 2 The Assertion Language
- 3 Semantics of Assertions
- 4 Partial Correctness Properties**
- 5 A Valid Partial Correctness Property

Definition 9.10 (Partial correctness properties)

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

- 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 $I \in \text{Int}$, we let

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

if $\sigma \models^I A$ implies $\mathcal{C}[\![c]\!]\sigma \models^I B$
(or equivalently: $\sigma \in A' \implies \mathcal{C}[\![c]\!]\sigma \in B'$).

- $\{A\} c \{B\}$ 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: $\mathcal{C}[\![c]\!]A' \subseteq B'$).
- $\{A\} c \{B\}$ is called **valid** (notation: $\models \{A\} c \{B\}$) if $\models^I \{A\} c \{B\}$ for every $I \in \text{Int}$.

- 1 The Axiomatic Approach
- 2 The Assertion Language
- 3 Semantics of Assertions
- 4 Partial Correctness Properties
- 5 A Valid Partial Correctness Property

Example 9.11

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

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

- According to Def. 9.10, this is equivalent to

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

for every $\sigma \in \Sigma_{\perp}$ and $I \in \text{Int}$

- For $\sigma = \perp$ this is trivial. So let $\sigma \in \Sigma$:

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