

Semantics and Verification of Software

Lecture 8: Operational/Denotational/Axiomatic Semantics of WHILE

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

`noll@cs.rwth-aachen.de`

`http://www-i2.informatik.rwth-aachen.de/i2/svsw08/`

Winter semester 2008/09

Abend der offenen Tür

am Lehrstuhl für Informatik 6

18.11.2008 ab 17:00h

MI:6

- Infos zu Lehre, Forschung und Projekten
- Infos zu Bachelor-, Master- und Diplomarbeiten
- Essen und Getränke frei
- Wettkampf: 1. Preis iPhone 3G 16GB

RWTH

Microsoft

- 1 Repetition: Operational/Denotational Semantics
- 2 Equivalence of Operational and Denotational Semantics
- 3 The Axiomatic Approach
- 4 The Assertion Language

Definition (Operational semantics of statements)

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

$$\begin{array}{lcl}
 \text{(skip)} \frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma} & \text{(asgn)} \frac{\langle a, \sigma \rangle \rightarrow z}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \mapsto z]} & \\
 \text{(seq)} \frac{\langle c_1, \sigma \rangle \rightarrow \sigma' \quad \langle c_2, \sigma' \rangle \rightarrow \sigma''}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma''} & \text{(if-t)} \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'} & \\
 \text{(if-f)} \frac{\langle b, \sigma \rangle \rightarrow \text{false} \quad \langle c_2, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'} & \text{(wh-f)} \frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma} & \\
 \text{(wh-t)} \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c, \sigma \rangle \rightarrow \sigma' \quad \langle \text{while } b \text{ do } c, \sigma' \rangle \rightarrow \sigma''}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma''} & &
 \end{array}$$

Definition (Denotational semantics of statements)

Denotational semantic functional for statements $\mathcal{C}[\cdot] : \text{Cmd} \rightarrow (\Sigma \dashrightarrow \Sigma)$:

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

where $\Phi : (\Sigma \dashrightarrow \Sigma) \rightarrow (\Sigma \dashrightarrow \Sigma) : f \mapsto \text{cond}(\mathcal{B}[b], f \circ \mathcal{C}[c], \text{id}_\Sigma)$

- 1 Repetition: Operational/Denotational Semantics
- 2 Equivalence of Operational and Denotational Semantics
- 3 The Axiomatic Approach
- 4 The Assertion Language

Remember: in Def. 4.1, $\mathfrak{D}[\![\cdot]\!] : Cmd \rightarrow (\Sigma \dashrightarrow \Sigma)$ was given by

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

Theorem 8.1 (Coincidence Theorem)

For every $c \in Cmd$,

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

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

Equivalence of Semantics II

The proof of Theorem 8.1 employs the following auxiliary propositions:

Lemma 8.2

- ❶ For every $a \in AExp$, $\sigma \in \Sigma$, and $z \in \mathbb{Z}$:

$$\langle a, \sigma \rangle \rightarrow z \iff \mathfrak{A}[[a]](\sigma) = z.$$

- ❷ For every $b \in BExp$, $\sigma \in \Sigma$, and $t \in \mathbb{B}$:

$$\langle b, \sigma \rangle \rightarrow t \iff \mathfrak{B}[[b]](\sigma) = t.$$

Proof.

- ❶ see Exercise 3.1 (structural induction on a)
❷ analogously (structural induction on b)



Proof (Theorem 8.1).

We have to show that

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

$\Rightarrow$ by structural induction over the derivation tree of $\langle c, \sigma \rangle \rightarrow \sigma'$

$\Leftarrow$ by structural induction over c (with a nested complete induction over fixpoint index n)

(on the board)



- 1 Repetition: Operational/Denotational Semantics
- 2 Equivalence of Operational and Denotational Semantics
- 3 The Axiomatic Approach
- 4 The Assertion Language

Example 8.1

- Let $c \in Cmd$ 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 , $\sigma(s) = \sum_{i=1}^{\sigma(N)} i$?
- “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 8.1 (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 =  $\sum_{i=1}^N i$  ∧ 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** $s = \sum_{i=1}^{n-1} i$ is satisfied
- Corresponding 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,

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

is a valid property

- 1 Repetition: Operational/Denotational Semantics
- 2 Equivalence of Operational and Denotational Semantics
- 3 The Axiomatic Approach
- 4 The Assertion Language

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

Syntactic categories:

Category	Domain	Meta variable(s)
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:

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

Abbreviations:

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