Semantics and Verification of Software

Lecture 8: Operational/Denotational/Axiomatic Semantics of WHILE

Thomas Noll

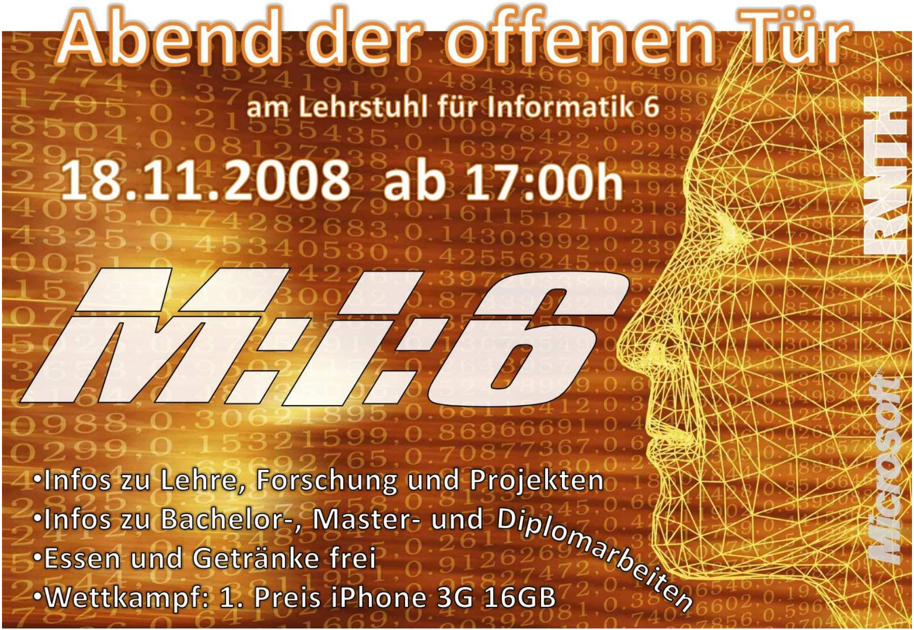
Lehrstuhl für Informatik 2
(Software Modeling and Verification)

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
M:I: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

Outline

Operational/Denotational Semantics

Definition (Operational semantics of statements)

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

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

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

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

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

Definition (Denotational semantics of statements)

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

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

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

Outline

Equivalence of Semantics I

Remember: in Def. 4.1, $\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$ was given by

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

Equivalence of Semantics I

Remember: in Def. 4.1, $\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$ was given by

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

Theorem 8.1 (Coincidence Theorem)

For every $c \in \mathit{Cmd}$,

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

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

Equivalence of Semantics II

The proof of Theorem 8.1 employs the following auxiliary propositions:

Lemma 8.2

1. *For every* $a \in AExp$, $\sigma \in \Sigma$, *and* $z \in \mathbb{Z}$*:*

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

Equivalence of Semantics II

The proof of Theorem 8.1 employs the following auxiliary propositions:

Lemma 8.2

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* $t \in \mathbb{B}$:

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

Equivalence of Semantics II

The proof of Theorem 8.1 employs the following auxiliary propositions:

Lemma 8.2

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 3.1 (structural induction on a)
2. analogously (structural induction on b)

□

Equivalence of Semantics III

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) □

Outline

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)
```

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, $\sigma(\mathtt{s}) = \sum_{i=1}^{\sigma(\mathtt{N})} i$?

The Axiomatic Approach I

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

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

Example 8.1 (continued)

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

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

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 (“$\texttt{s} = 0$”)

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

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 $\texttt{s}$?

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

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

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 $\mathtt{s} = \sum_{i=1}^{\mathtt{n}-1} i$ 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

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 $\texttt{s} = \sum_{i=1}^{\texttt{n}-1} i$ 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,

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

  is a valid property

Outline

Syntax of Assertion Language I

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

Syntax of Assertion Language I

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

Syntax of Assertion Language II

Definition 8.2 (Syntax of assertions)

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

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

Syntax of Assertion Language II

Definition 8.2 (Syntax of assertions)

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

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

Abbreviations:

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