

Semantics and Verification of Software

Lecture 4: Operational Semantics of WHILE III

(Properties of Execution Relation)

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/svsw10/>

Summer Semester 2010

- 1 Repetition: Execution of Statements
- 2 Functional of the Operational Semantics

Execution of Statements

Remember:

$c ::= \text{skip} \mid x := a \mid c_1; c_2 \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \mid \text{while } b \text{ do } c \in \text{Cmd}$

Definition (Execution relation for statements)

For $c \in \text{Cmd}$ and $\sigma, \sigma' \in \Sigma$, the **execution relation** $\langle c, \sigma \rangle \rightarrow \sigma'$ is defined by the following rules:

$$(\text{skip}) \frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma}$$

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

$$(\text{seq}) \frac{\langle c_1, \sigma \rangle \rightarrow \sigma' \quad \langle c_2, \sigma' \rangle \rightarrow \sigma''}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma''}$$

$$(\text{if-t}) \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'}$$

$$(\text{if-f}) \frac{\langle b, \sigma \rangle \rightarrow \text{false} \quad \langle c_2, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'}$$

$$(\text{wh-f}) \frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma}$$

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

This operational semantics is well defined in the following sense:

Theorem

*The execution relation for statements is **deterministic**, i.e., whenever $c \in \text{Cmd}$ and $\sigma, \sigma', \sigma'' \in \Sigma$ such that $\langle c, \sigma \rangle \rightarrow \sigma'$ and $\langle c, \sigma \rangle \rightarrow \sigma''$, then $\sigma' = \sigma''$.*

- How to prove this theorem?
- Idea:
 - use **induction on the syntactic structure** of c
 - employ corresponding result for **expressions** (Lemma 3.5)

Determinism of Execution Relation II

- But: **proof of Theorem 3.4 fails!**
- Problematic case:

$c = \text{while } b \text{ do } c_0 \text{ where } \langle b, \sigma \rangle \rightarrow \text{true}$

- Here $\langle c, \sigma \rangle \rightarrow \sigma'$ and $\langle c, \sigma \rangle \rightarrow \sigma''$ require $\sigma_1, \sigma_2 \in \Sigma$ such that

$$(\text{wh-t}) \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c_0, \sigma \rangle \rightarrow \sigma_1 \quad \langle c, \sigma_1 \rangle \rightarrow \sigma'}{\langle c, \sigma \rangle \rightarrow \sigma'}$$

and

$$(\text{wh-t}) \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c_0, \sigma \rangle \rightarrow \sigma_2 \quad \langle c, \sigma_2 \rangle \rightarrow \sigma''}{\langle c, \sigma \rangle \rightarrow \sigma''}$$

- c_0 proper substatement of c
 $\implies$ induction hypothesis yields $\sigma_1 = \sigma_2$
- c **not** proper substatement of $c \implies$ **conclusion $\sigma' = \sigma''$ invalid!**

Determinism of Execution Relation III

Induction on derivation trees of execution relation

Induction base: $P\left(\frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma}\right)$ holds for every $\sigma \in \Sigma$, and $P(s)$ holds for every derivation tree s for an arithmetic or Boolean expression.

Induction hypothesis: $P(s_1)$, $P(s_2)$ und $P(s_3)$ holds.

Induction step: it also holds that

$$\text{(asn): } P\left(\frac{s_1}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \mapsto z]}\right)$$

$$\text{(seq): } P\left(\frac{s_1 \ s_2}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma''}\right)$$

$$\text{(if-t): } P\left(\frac{s_1 \ s_2}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'}\right)$$

(if-f): analogously

$$\text{(wh-t): } P\left(\frac{s_1 \ s_2 \ s_3}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma''}\right)$$

$$\text{(wh-f): } P\left(\frac{s_1}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma}\right)$$

Proof (Theorem 3.4).

To show:

$$\langle c, \sigma \rangle \rightarrow \sigma', \langle c, \sigma \rangle \rightarrow \sigma'' \implies \sigma' = \sigma''$$

(by structural induction on derivation trees; on the board)



- 1 Repetition: Execution of Statements
- 2 Functional of the Operational Semantics

Functional of the Operational Semantics

The determinism of the execution relation (Theorem 3.4) justifies the following definition:

Definition 4.1 (Operational functional)

The **functional of the operational semantics**,

$$\mathfrak{D}[\![\cdot]\!] : Cmd \rightarrow (\Sigma \dashrightarrow \Sigma),$$

assigns to every statement $c \in Cmd$ a partial state transformation $\mathfrak{D}[\![c]\!] : \Sigma \dashrightarrow \Sigma$, which is defined as follows:

$$\mathfrak{D}[\![c]\!]\sigma := \begin{cases} \sigma' & \text{if } \langle c, \sigma \rangle \rightarrow \sigma' \text{ for some } \sigma' \in \Sigma \\ \text{undefined} & \text{otherwise} \end{cases}$$

Functional of the Operational Semantics

The determinism of the execution relation (Theorem 3.4) justifies the following definition:

Definition 4.1 (Operational functional)

The **functional of the operational semantics**,

$$\mathfrak{D}[\![\cdot]\!] : Cmd \rightarrow (\Sigma \dashrightarrow \Sigma),$$

assigns to every statement $c \in Cmd$ a partial state transformation $\mathfrak{D}[\![c]\!] : \Sigma \dashrightarrow \Sigma$, which is defined as follows:

$$\mathfrak{D}[\![c]\!]\sigma := \begin{cases} \sigma' & \text{if } \langle c, \sigma \rangle \rightarrow \sigma' \text{ for some } \sigma' \in \Sigma \\ \text{undefined} & \text{otherwise} \end{cases}$$

Remark: $\mathfrak{D}[\![c]\!]\sigma$ can indeed be undefined
(consider e.g. $c = \text{while true do skip}$; see Corollary 3.3)

Definition 4.2 (Operational equivalence)

Two statements $c_1, c_2 \in Cmd$ are called (operationally) equivalent (notation: $c_1 \sim c_2$) if

$$\mathfrak{D}[[c_1]] = \mathfrak{D}[[c_2]].$$

Definition 4.2 (Operational equivalence)

Two statements $c_1, c_2 \in Cmd$ are called **(operationally) equivalent** (notation: $c_1 \sim c_2$) if

$$\mathcal{V}[[c_1]] = \mathcal{V}[[c_2]].$$

Thus:

- $c_1 \sim c_2$ iff $\mathcal{V}[[c_1]]\sigma = \mathcal{V}[[c_2]]\sigma$ for every $\sigma \in \Sigma$
- In particular, $\mathcal{V}[[c_1]]\sigma$ is undefined iff $\mathcal{V}[[c_2]]\sigma$ is undefined