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/svsw11/

Winter Semester 2011/12

Outline

Execution of Statements

Remember:

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

Definition (Execution relation for statements)

For $c \in Cmd$ and $\sigma, \sigma' \in \Sigma$, the execution relation $\langle c, \sigma \rangle \to \sigma'$ is defined by the following rules:

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

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

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

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

Determinism of Execution Relation I

This operational semantics is well defined in the following sense:

Theorem

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

- How to prove this theorem?
- Idea:
  - employ corresponding result for expressions (Lemma 3.6)
  - use induction on the syntactic structure of c ↯
- Instead: structural induction on derivation trees

Determinism of Execution Relation II

Proof (Theorem 3.5).

To show:

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

Proof by structural induction on derivation tree for $\langle c, \sigma \rangle \to \sigma'$.
Already considered:

- $(\text{skip})\dfrac{}{\langle \texttt{skip}, \sigma \rangle \to \sigma}$ (i.e., $c = \texttt{skip}$, $\sigma' = \sigma$):
  since this axiom is the only applicable derivation rule, it follows that also $\sigma'' = \sigma = \sigma'$.
- $(\text{asgn})\dfrac{\langle a, \sigma \rangle \to z}{\langle x := a, \sigma \rangle \to \sigma[x \mapsto z]}$ (i.e., $c = (x := a)$, $\sigma' = \sigma[x \mapsto z]$):
  here the second derivation must be of the form
  $$(\text{asgn})\dfrac{\langle a, \sigma \rangle \to z'}{\langle x := a, \sigma \rangle \to \sigma[x \mapsto z']}$$
  such that Lemma 3.6(1) implies $z = z'$, and hence
  $\sigma'' = \sigma[x \mapsto z'] = \sigma[x \mapsto z] = \sigma'$.
- ... (on the board)

Outline

Functional of the Operational Semantics

The determinism of the execution relation (Theorem 3.5) justifies the following definition:

Definition 4.1 (Operational functional)

The functional of the operational semantics,

$$\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma),$$

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

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

Functional of the Operational Semantics

The determinism of the execution relation (Theorem 3.5) justifies the following definition:

Definition 4.1 (Operational functional)

The functional of the operational semantics,

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

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

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

Remark: $\mathfrak{O}[\![c]\!]\sigma$ can indeed be undefined (consider e.g. $c = \texttt{while true do skip}$; see Corollary 3.4)

Equivalence of Statements

Underlying principle: two (syntactic) objects are considered (semantically) equivalent if they have the same "meaning"

- finite automata: $A_1 \sim A_2$ iff $L(A_1) = L(A_2)$
- context-free grammars: $G_1 \sim G_2$ iff $L(G_1) = L(G_2)$
- Turing machines: $T_1 \sim T_2$ iff both compute same function

Equivalence of Statements

Underlying principle: two (syntactic) objects are considered (semantically) equivalent if they have the same "meaning"

- finite automata: $A_1 \sim A_2$ iff $L(A_1) = L(A_2)$
- context-free grammars: $G_1 \sim G_2$ iff $L(G_1) = L(G_2)$
- Turing machines: $T_1 \sim T_2$ iff both compute same function

Definition 4.2 (Operational equivalence)

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

$$\mathfrak{O}[\![c_1]\!] = \mathfrak{O}[\![c_2]\!].$$

Equivalence of Statements

Underlying principle: two (syntactic) objects are considered (semantically) equivalent if they have the same "meaning"

- finite automata: $A_1 \sim A_2$ iff $L(A_1) = L(A_2)$
- context-free grammars: $G_1 \sim G_2$ iff $L(G_1) = L(G_2)$
- Turing machines: $T_1 \sim T_2$ iff both compute same function

Definition 4.2 (Operational equivalence)

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

$$\mathfrak{O}[\![c_1]\!] = \mathfrak{O}[\![c_2]\!].$$

Thus:

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

"Unwinding" of Loops

Simple application of statement equivalence: test of execution condition in a `while` loop can be represented by an `if` statement

Lemma 4.3

For every $b \in BExp$ *and* $c \in Cmd$,

$$\texttt{while}\ b\ \texttt{do}\ c \sim \texttt{if}\ b\ \texttt{then}\ (c;\texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else}\ \texttt{skip}.$$

“Unwinding” of Loops

Simple application of statement equivalence: test of execution condition in a `while` loop can be represented by an `if` statement

Lemma 4.3

For every $b \in BExp$ *and* $c \in Cmd$,

$$\texttt{while}\ b\ \texttt{do}\ c \sim \texttt{if}\ b\ \texttt{then}\ (c\,\texttt{;}\,\texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else}\ \texttt{skip}.$$

Proof.

on the board □