Semantics and Verification of Software

Lecture 3: Operational Semantics of WHILE

Thomas Noll

Lehrstuhl für Informatik 2
RWTH Aachen University
noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw/

Summer semester 2007

Outline

Evaluation of Arithmetic Expressions

Remember: $a ::= z \mid x \mid a_1\texttt{+}a_2 \mid a_1\texttt{-}a_2 \mid a_1\texttt{*}a_2 \in AExp$

Definition (Evaluation relation for arithmetic expressions)

If $a \in AExp$ and $\sigma \in \Sigma$, then $\langle a, \sigma \rangle$ is called a configuration.

Expression a evaluates to $z \in \mathbb{Z}$ in state σ (notation: $\langle a, \sigma \rangle \to z$) if this relationship is derivable by means of the following rules:

Axioms:
$$\frac{}{\langle z, \sigma \rangle \to z} \qquad \frac{}{\langle x, \sigma \rangle \to \sigma(x)}$$

Rules:
$$\frac{\langle a_1, \sigma \rangle \to z_1 \quad \langle a_2, \sigma \rangle \to z_2}{\langle a_1\texttt{+}a_2, \sigma \rangle \to z} \quad \text{where } z := z_1 + z_2$$

$$\frac{\langle a_1, \sigma \rangle \to z_1 \quad \langle a_2, \sigma \rangle \to z_2}{\langle a_1\texttt{-}a_2, \sigma \rangle \to z} \quad \text{where } z := z_1 - z_2$$

$$\frac{\langle a_1, \sigma \rangle \to z_1 \quad \langle a_2, \sigma \rangle \to z_2}{\langle a_1\texttt{*}a_2, \sigma \rangle \to z} \quad \text{where } z := z_1 * z_2$$

Evaluation of Boolean Expressions

Remember: $b ::= t \mid a_1\texttt{=}a_2 \mid a_1\texttt{>}a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in \mathit{BExp}$

Definition (Evaluation relation for Boolean expressions)

For $b \in \mathit{BExp}$ and $\sigma \in \Sigma$, and $t \in \mathbb{B}$, the evaluation relation $\langle b, \sigma \rangle \to t$ is defined by the following rules:

$$\frac{}{\langle t, \sigma \rangle \to t}$$

$$\frac{\langle a_1, \sigma \rangle \to z \quad \langle a_2, \sigma \rangle \to z}{\langle a_1\texttt{=}a_2, \sigma \rangle \to \mathsf{true}} \qquad \frac{\langle a_1, \sigma \rangle \to z_1 \quad \langle a_2, \sigma \rangle \to z_2}{\langle a_1\texttt{=}a_2, \sigma \rangle \to \mathsf{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1, \sigma \rangle \to z_1 \quad \langle a_2, \sigma \rangle \to z_2}{\langle a_1\texttt{>}a_2, \sigma \rangle \to \mathsf{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1, \sigma \rangle \to z_1 \quad \langle a_2, \sigma \rangle \to z_2}{\langle a_1\texttt{>}a_2, \sigma \rangle \to \mathsf{false}} \text{ if } z_1 \leq z_2$$

$$\frac{\langle b, \sigma \rangle \to \mathsf{false}}{\langle \neg b, \sigma \rangle \to \mathsf{true}} \qquad \frac{\langle b, \sigma \rangle \to \mathsf{true}}{\langle \neg b, \sigma \rangle \to \mathsf{false}}$$

$$\frac{\langle b_1, \sigma \rangle \to \mathsf{true} \quad \langle b_2, \sigma \rangle \to \mathsf{true}}{\langle b_1 \wedge b_2, \sigma \rangle \to \mathsf{true}} \qquad \frac{\langle b_1, \sigma \rangle \to \mathsf{true} \quad \langle b_2, \sigma \rangle \to \mathsf{false}}{\langle b_1 \wedge b_2, \sigma \rangle \to \mathsf{false}}$$

$$\frac{\langle b_1, \sigma \rangle \to \mathsf{false} \quad \langle b_2, \sigma \rangle \to \mathsf{true}}{\langle b_1 \wedge b_2, \sigma \rangle \to \mathsf{false}} \qquad \frac{\langle b_1, \sigma \rangle \to \mathsf{false} \quad \langle b_2, \sigma \rangle \to \mathsf{false}}{\langle b_1 \wedge b_2, \sigma \rangle \to \mathsf{false}}$$

($\vee$ analogously)

Execution of Statements

Remember:

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

Definition (Execution relation for statements)

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

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

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

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

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

Outline

An Execution Example

Example 3.4

- $c := \underbrace{}$ `y := 1; while` $\underbrace{\neg\texttt{(x=1)}}_{b}$ `do` $\underbrace{\underbrace{\texttt{y := y*x}}_{c_1}\texttt{;}\underbrace{\texttt{x := x-1}}_{c_2}}_{c_0}$
- Claim: $\langle c, \sigma\rangle \to \sigma_{1,6}$ for every $\sigma \in \Sigma$ with $\sigma(\texttt{x}) = 3$
- Notation: $\sigma_{i,j}$ means $\sigma(\texttt{x}) = i$, $\sigma(\texttt{y}) = j$
- Derivation tree: on the board

Outline

Determinism of Execution Relation I

This operational semantics is well defined in the following sense:

Theorem 3.5

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* $\sigma' = \sigma''$.

The proof is based on the corresponding result for expressions.

Determinism of Evaluation Relations

Lemma 3.6

1. *For every* $a \in AExp$, $\sigma \in \Sigma$, *and* $z, z' \in \mathbb{Z}$: $\langle a, \sigma \rangle \to z$ *and* $\langle a, \sigma \rangle \to z'$ *implies* $z = z'$.
2. *For every* $b \in BExp$, $\sigma \in \Sigma$, *and* $t, t' \in \mathbb{B}$: $\langle b, \sigma \rangle \to t$ *and* $\langle b, \sigma \rangle \to t'$ *implies* $t = t'$.

Remark: Lemma 3.6 is not implied by Lemma 2.6 ("$\sigma|_{FV(a)} = \sigma'|_{FV(a)} \implies (\langle a, \sigma \rangle \to z \iff \langle a, \sigma' \rangle \to z)$")!

The latter just implies

$$\{z \in \mathbb{Z} \mid \langle a, \sigma \rangle \to z\} = \{z \in \mathbb{Z} \mid \langle a, \sigma' \rangle \to z\}$$

while Lemma 3.6 states that

$$|\{z \in \mathbb{Z} \mid \langle a, \sigma \rangle \to z\}| \leq 1.$$

Excursus: Proof by Structural Induction IV

Application: Boolean expressions (Def. 1.2)

Definition: $BExp$ is the least set which

- contains the truth values $t \in \mathbb{B}$ and, for every $a_1, a_2 \in AExp$, a_1=a_2 and a_1>a_2, and
- contains $\neg b_1$, $b_1 \wedge b_2$ and $b_1 \vee b_2$ whenever $b_1, b_2 \in BExp$

Induction base: $P(t)$, $P(a_1\texttt{=}a_2)$ and $P(a_1\texttt{>}a_2)$ holds (for every $t \in \mathbb{B}$, $a_1, a_2 \in AExp$)

Induction hypothesis: $P(b_1)$ and $P(b_2)$ holds

Induction step: $P(\neg b_1)$, $P(b_1 \wedge b_2)$ and $P(b_1 \vee b_2)$ holds

Proof (Lemma 3.6).

1. by structural induction on a (omitted)
2. by structural induction on b (omitted)

□

Determinism of Execution Relation II

- How to prove that $\langle c, \sigma \rangle \to \sigma'$ is deterministic (Theorem 3.5)?
- Idea: use induction on the syntactic structure of c

Excursus: Proof by Structural Induction V

Application: syntax of WHILE statements (Def. 1.2)

Definition: Cmd is the least set which

- contains `skip` and, for every $x \in \mathit{Var}$ and $a \in \mathit{AExp}$, x `:=` a, and
- contains c_1`;`c_2, `if` b `then` c_1 `else` c_2 and `while` b `do` c_1 whenever $b \in \mathit{BExp}$ and $c_1, c_2 \in \mathit{Cmd}$

Induction base: $P($`skip`$)$ and $P(x$ `:=` $a)$ holds (for every $x \in \mathit{Var}$ and $a \in \mathit{AExp}$)

Induction hypothesis: $P(c_1)$ and $P(c_2)$ holds

Induction step: $P(c_1$`;`$c_2)$, $P($`if` b `then` c_1 `else` $c_2)$ and $P($`while` b `do` $c_1)$ holds

Determinism of Execution Relation III

Remark:

- But: proof of Theorem 3.5 fails!
- Problematic case:

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

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

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

and

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

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

Application: derivation trees of execution relation (Def. 3.3)

(skip): for every $\sigma \in \Sigma$, $\dfrac{}{\langle \texttt{skip}, \sigma\rangle \to \sigma}$ is a derivation tree for $\langle \texttt{skip}, \sigma\rangle \to \sigma$

(asgn): if s is a derivation tree for $\langle a, \sigma\rangle \to z$ (Def. 2.3), then $\dfrac{s}{\langle x := a, \sigma\rangle \to \sigma[x \mapsto z]}$ is a derivation tree for $\langle x := a, \sigma\rangle \to \sigma[x \mapsto z]$

(seq): if s_1 and s_2 are derivation trees for $\langle c_1, \sigma\rangle \to \sigma'$ and, respectively, $\langle c_2, \sigma'\rangle \to \sigma''$, then $\dfrac{s_1 \quad s_2}{\langle c_1 ; c_2, \sigma\rangle \to \sigma''}$ is a derivation tree for $\langle c_1 ; c_2, \sigma\rangle \to \sigma''$

(if–t): if s_1 and s_2 are derivation trees for $\langle b, \sigma\rangle \to \textsf{true}$ (Def. 2.7) and, respectively, $\langle c_1, \sigma\rangle \to \sigma'$, then $\dfrac{s_1 \quad s_2}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \to \sigma'}$ is a derivation tree for $\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \to \sigma'$

(if–f): analogously

(wh–t): if s_1, s_2 and s_3 are derivation trees for $\langle b, \sigma\rangle \to \textsf{true}$ (Def. 2.7), $\langle c, \sigma\rangle \to \sigma'$ and $\langle \texttt{while } b \texttt{ do } c, \sigma'\rangle \to \sigma''$, respectively, then $\dfrac{s_1 \quad s_2 \quad s_3}{\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma''}$ is a derivation tree for $\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma''$

(wh–f): if s is a derivation tree for $\langle b, \sigma\rangle \to \textsf{false}$ (Def. 2.7), then $\dfrac{s}{\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma}$ is a derivation tree for $\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma$

Application: derivation trees of execution relation (continued)

Induction base: $P\left(\frac{}{\langle \texttt{skip}, \sigma\rangle \to \sigma}\right)$ 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

(asgn): $P\left(\frac{s_1}{\langle x\,\texttt{:=}\,a, \sigma\rangle \to \sigma[x \mapsto z]}\right)$

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

(if–t): $P\left(\frac{s_1 \quad s_2}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \sigma\rangle \to \sigma'}\right)$

(if–f): analogously

(wh–t): $P\left(\frac{s_1 \quad s_2 \quad s_3}{\langle \texttt{while}\ b\ \texttt{do}\ c, \sigma\rangle \to \sigma''}\right)$

(wh–f): $P\left(\frac{s_1}{\langle \texttt{while}\ b\ \texttt{do}\ c, \sigma\rangle \to \sigma}\right)$

Determinism of Execution Relation IV

Proof.

To show:

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

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