Semantics and Verification of Software

Summer Semester 2015

Lecture 3: Operational Semantics of WHILE II (Execution of Statements)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

http://moves.rwth-aachen.de/teaching/ss-15/sv-sw/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Recap: Structural Induction & Evaluation Relations

Proof by Structural Induction

Proof principle

Given: an inductive set, i.e., a set S whose elements are either

- atomic or
- obtained from atomic elements by (finite) application of certain operations

To show: property $P(s)$ applies to every $s \in S$

Proof: we verify:

Induction base: $P(s)$ holds for every atomic element s

Induction hypothesis: assume that $P(s_1)$, $P(s_2)$ etc.

Induction step: then also $P(f(s_1, \ldots, s_n))$ holds for every operation f of arity n

Remark: structural induction is a special case of well-founded induction

Recap: Structural Induction & Evaluation Relations

Evaluation of Arithmetic Expressions

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

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

Rules:

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

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

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

Evaluation of Boolean Expressions

Definition ((Strict) evaluation relation for Boolean expressions)

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

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

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

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

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

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

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

($\vee$ analogously)

Execution of Statements

Meaning of Statements

Effect of statement = modification of program state

Example 3.1

Goal: define execution relation $\rightarrow$ such that, e.g.,

$$\langle \texttt{x := 5}, \sigma \rangle \rightarrow \sigma[\texttt{x} \mapsto 5]$$

where for every $\sigma \in \Sigma$, $x, y \in Var$, and $z \in \mathbb{Z}$:

$$\sigma[x \mapsto z](y) := \begin{cases} z & \text{if } y = x \\ \sigma(y) & \text{otherwise} \end{cases}$$

Execution of Statements

Remember:

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

Definition 3.2 (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:

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

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

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

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

An Execution Example

Example 3.3

- $c := \texttt{y := 1; while} \underbrace{\neg\texttt{(x=1)}}_{b} \texttt{do} \underbrace{\underbrace{\texttt{y := y*x}}_{c_1} \texttt{;} \underbrace{\texttt{x := x-1}}_{c_2}}_{c_0} \texttt{end}$
- 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

Non-Terminating Statements

Corollary 3.4

The execution relation for statements is not total*, i.e., there exist* $c \in Cmd$ *and* $\sigma \in \Sigma$ *such that* $\langle c, \sigma \rangle \to \sigma'$ *for no* $\sigma' \in \Sigma$.

Proof.

Example: $c = \texttt{while true do skip end}$ (proof by contradiction; on the board) $\square$

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/Execution

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* $t = t'$.

Remarks:

- Lemma 3.6(1) is not implied by Lemma 2.6
  ("$\sigma|_{FV(a)} = \sigma'|_{FV(a)} \Rightarrow (\langle a, \sigma \rangle \to z \iff \langle a, \sigma' \rangle \to z)$")!

  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(1) states that

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

- Lemma 3.6 can be shown by induction on the structure of expressions.

Excursus: Proof by Structural Induction V

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: $P(t)$, $P(a_1{=}a_2)$ and $P(a_1{>}a_2)$ 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)

$\square$

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 VI

Application: syntax of WHILE statements (Def. 1.2)

Definition: *Cmd* is the least set which

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

Induction base: $P(\texttt{skip})$ 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: $P(c_1 ; c_2)$, $P(\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end})$ and $P(\texttt{while}\ b\ \texttt{do}\ c_1\ \texttt{end})$ holds (for every $b \in \mathit{BExp}$)

Determinism of Execution Relation III

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

$$c = \texttt{while}\ b\ \texttt{do}\ c_0\ \texttt{end} \quad \text{where} \quad \langle b, \sigma \rangle \to \mathsf{true}$$

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

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

  and

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

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

Excursus: Proof by Structural Induction VII

Application: derivation trees of execution relation (Def. 3.2)

(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. 2.2), 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 $\langle b, \sigma\rangle \to \mathsf{true}$ (Def. 2.7) and, respectively, $\langle c_1, \sigma\rangle \to \sigma'$, then $\dfrac{s_1 \quad s_2}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}, \sigma\rangle \to \sigma'}$ is a derivation tree for $\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}, \sigma\rangle \to \sigma'$

(if-f): analogously

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

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

Excursus: Proof by Structural Induction VIII

Application: derivation trees of execution relation (continued)

Induction base: $P\left(\frac{}{\langle \mathtt{skip}, \sigma\rangle \to \sigma}\right)$ 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 $P(s_3)$ hold.

Induction step: it also holds that

- $P\left({\scriptstyle(\text{asgn})}\frac{s_1}{\langle x := a, \sigma\rangle \to \sigma[x \mapsto z]}\right)$
- $P\left({\scriptstyle(\text{seq})}\frac{s_1 \quad s_2}{\langle c_1 ; c_2, \sigma\rangle \to \sigma''}\right)$
- $P\left({\scriptstyle(\text{if-t}):}\frac{s_1 \quad s_2}{\langle \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\ \mathtt{end}, \sigma\rangle \to \sigma'}\right)$
- $P\left({\scriptstyle(\text{if-f}):}\frac{s_1 \quad s_2}{\langle \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\ \mathtt{end}, \sigma\rangle \to \sigma'}\right)$
- $P\left({\scriptstyle(\text{wh-t})}\frac{s_1 \quad s_2 \quad s_3}{\langle \mathtt{while}\ b\ \mathtt{do}\ c\ \mathtt{end}, \sigma\rangle \to \sigma''}\right)$
- $P\left({\scriptstyle(\text{wh-f})}\frac{s_1}{\langle \mathtt{while}\ b\ \mathtt{do}\ c\ \mathtt{end}, \sigma\rangle \to \sigma}\right)$

Determinism of Execution Relation IV

Proof (Theorem 3.5).

To show:

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

(by structural induction on derivation trees; on the board) $\square$