

Semantics and Verification of Software

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

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)



noll@cs.rwth-aachen.de

<http://www-i2.informatik.rwth-aachen.de/i2/svsw13/>

Summer Semester 2013

- 1 Recapitulation: Structural Induction & Evaluation Relations
- 2 Execution of Statements
- 3 Determinism of Evaluation/Execution

- **Inductive set**: elements are either
 - atomic or
 - obtained from atomic elements by (finite) application of certain operations
- Structural induction on $\mathbb{N}$: **mathematical induction**
 - induction step: $P(n) \Rightarrow P(n+1)$
- **Complete induction** = well-founded induction on $\mathbb{N}$
 - induction step: $P(0), P(1), \dots, P(n) \Rightarrow P(n+1)$
 - also known as **strong** or **course-of-values** induction

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: $\langle a, \sigma \rangle \rightarrow z$) if this relationship is derivable by means of the following rules:

Axioms: $\frac{}{\langle z, \sigma \rangle \rightarrow z} \quad \frac{}{\langle x, \sigma \rangle \rightarrow \sigma(x)}$

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

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

Evaluation of Boolean Expressions

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

Definition (Evaluation relation for Boolean expressions)

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

$$\begin{array}{c} \frac{}{\langle t, \sigma \rangle \rightarrow t} \\[10pt] \frac{\langle a_1, \sigma \rangle \rightarrow z \quad \langle a_2, \sigma \rangle \rightarrow z}{\langle a_1 = a_2, \sigma \rangle \rightarrow \text{true}} \qquad \frac{\langle a_1, \sigma \rangle \rightarrow z_1 \quad \langle a_2, \sigma \rangle \rightarrow z_2}{\langle a_1 = a_2, \sigma \rangle \rightarrow \text{false}} \text{ if } z_1 \neq z_2 \\[10pt] \frac{\langle a_1, \sigma \rangle \rightarrow z_1 \quad \langle a_2, \sigma \rangle \rightarrow z_2}{\langle a_1 > a_2, \sigma \rangle \rightarrow \text{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1, \sigma \rangle \rightarrow z_1 \quad \langle a_2, \sigma \rangle \rightarrow z_2}{\langle a_1 > a_2, \sigma \rangle \rightarrow \text{false}} \text{ if } z_1 \leq z_2 \\[10pt] \frac{}{\langle b, \sigma \rangle \rightarrow \text{false}} \qquad \frac{}{\langle b, \sigma \rangle \rightarrow \text{true}} \\[10pt] \frac{}{\langle \neg b, \sigma \rangle \rightarrow \text{true}} \qquad \frac{}{\langle \neg b, \sigma \rangle \rightarrow \text{false}} \\[10pt] \frac{\langle b_1, \sigma \rangle \rightarrow \text{true} \quad \langle b_2, \sigma \rangle \rightarrow \text{true}}{\langle b_1 \wedge b_2, \sigma \rangle \rightarrow \text{true}} \qquad \frac{\langle b_1, \sigma \rangle \rightarrow \text{true} \quad \langle b_2, \sigma \rangle \rightarrow \text{false}}{\langle b_1 \wedge b_2, \sigma \rangle \rightarrow \text{false}} \\[10pt] \frac{\langle b_1, \sigma \rangle \rightarrow \text{false} \quad \langle b_2, \sigma \rangle \rightarrow \text{true}}{\langle b_1 \wedge b_2, \sigma \rangle \rightarrow \text{false}} \qquad \frac{\langle b_1, \sigma \rangle \rightarrow \text{false} \quad \langle b_2, \sigma \rangle \rightarrow \text{false}}{\langle b_1 \wedge b_2, \sigma \rangle \rightarrow \text{false}} \end{array}$$

($\vee$ analogously)

- 1 Recapitulation: Structural Induction & Evaluation Relations
- 2 Execution of Statements
- 3 Determinism of Evaluation/Execution

Effect of statement = **modification of program state**

Example 3.1

Goal: define **execution relation** $\rightarrow$ such that

$$\langle x := 2+3, \sigma \rangle \rightarrow \sigma[x \mapsto 5]$$

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:

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

Definition 3.2 (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:

$$\begin{array}{lcl} \text{(skip)} \frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma} & \text{(asgn)} \frac{\langle a, \sigma \rangle \rightarrow z}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \mapsto z]} \\ \text{(seq)} \frac{\langle c_1, \sigma \rangle \rightarrow \sigma' \quad \langle c_2, \sigma' \rangle \rightarrow \sigma''}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma''} & \text{(if-t)} \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'} \\ \text{(if-f)} \frac{\langle b, \sigma \rangle \rightarrow \text{false} \quad \langle c_2, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'} & \text{(wh-f)} \frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma} \\ \text{(wh-t)} \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c, \sigma \rangle \rightarrow \sigma' \quad \langle \text{while } b \text{ do } c, \sigma' \rangle \rightarrow \sigma''}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma''} \end{array}$$

Example 3.3

- $c := y := 1; \text{ while } \underbrace{\neg(x=1)}_b \text{ do } \underbrace{y := y*x}_{c_1}; \underbrace{x := x-1}_{c_2}$
 $\underbrace{\hspace{10em}}_{c_0}$
- Claim: $\langle c, \sigma \rangle \rightarrow \sigma_{1,6}$ for every $\sigma \in \Sigma$ with $\sigma(x) = 3$
- Notation: $\sigma_{i,j}$ means $\sigma(x) = i, \sigma(y) = j$
- Derivation tree: on the board

Corollary 3.4

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

Proof.

Counterexample: $c = \text{while true do skip}$
(by contradiction; on the board)



- 1 Recapitulation: Structural Induction & Evaluation Relations
- 2 Execution of Statements
- 3 Determinism of Evaluation/Execution

This operational semantics is well defined in the following sense:

Theorem 3.5

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

The proof is based on the corresponding result for expressions.

Lemma 3.6

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

Remarks:

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

The latter just implies

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

while Lemma 3.6(1) states that

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

- 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)



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

Application: syntax of WHILE statements (Def. 1.2)

Definition: Cmd is the least set which

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

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

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
(for every $b \in BExp$)

Determinism of Execution Relation III

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

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

- Here $\langle c, \sigma \rangle \rightarrow \sigma'$ and $\langle c, \sigma \rangle \rightarrow \sigma''$ require existence of $\sigma_1, \sigma_2 \in \Sigma$ 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
 $\Rightarrow$ induction hypothesis yields $\sigma_1 = \sigma_2$
- c **not** proper substatement of $c \Rightarrow$ **conclusion $\sigma' = \sigma''$ invalid!**

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

- (skip): for every $\sigma \in \Sigma$, $\frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma}$ is a derivation tree for $\langle \text{skip}, \sigma \rangle \rightarrow \sigma$
- (asgn): if s is a derivation tree for $\langle a, \sigma \rangle \rightarrow z$ (Def. 2.2), then $\frac{s}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \mapsto z]}$ is a derivation tree for $\langle x := a, \sigma \rangle \rightarrow \sigma[x \mapsto z]$
- (seq): if s_1 and s_2 are derivation trees for $\langle c_1, \sigma \rangle \rightarrow \sigma'$ and, respectively, $\langle c_2, \sigma' \rangle \rightarrow \sigma''$, then $\frac{s_1 \ s_2}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma''}$ is a derivation tree for $\langle c_1; c_2, \sigma \rangle \rightarrow \sigma''$
- (if-t): if s_1 and s_2 are derivation trees for $\langle b, \sigma \rangle \rightarrow \text{true}$ (Def. 2.7) and, respectively, $\langle c_1, \sigma \rangle \rightarrow \sigma'$, then $\frac{s_1 \ s_2}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'}$ is a derivation tree for $\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'$
- (if-f): analogously
- (wh-t): if s_1 , s_2 and s_3 are derivation trees for $\langle b, \sigma \rangle \rightarrow \text{true}$ (Def. 2.7), $\langle c, \sigma \rangle \rightarrow \sigma'$ and $\langle \text{while } b \text{ do } c, \sigma' \rangle \rightarrow \sigma''$, respectively, then $\frac{s_1 \ s_2 \ s_3}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma''}$ is a derivation tree for $\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma''$
- (wh-f): if s is a derivation tree for $\langle b, \sigma \rangle \rightarrow \text{false}$ (Def. 2.7), then $\frac{s}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma}$ is a derivation tree for $\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma$

Application: derivation trees of execution relation (continued)

Induction base: $P\left(\frac{\quad}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma}\right)$ 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 3.5).

To show:

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

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

