Semantics and Verification of Software

Winter Semester 2017/18

Lecture 5: Operational Semantics of WHILE IV (The Compiler & Its Correctness)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

http://moves.rwth-aachen.de/teaching/ws-1718/sv-sw/

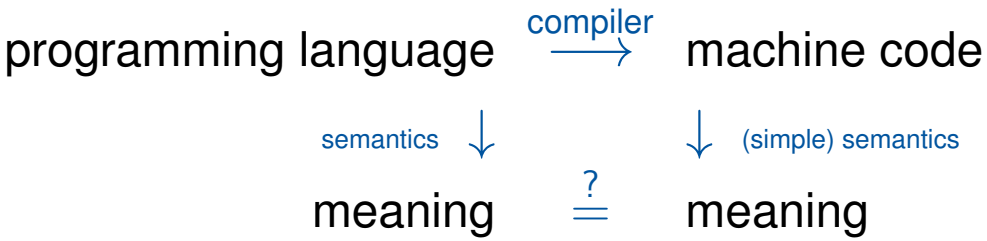
Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Exam in Semantics and Verification of Software

- In oral form
- Dates to be announced later
- Registration via Campus enabled
- Admission requires 50% of exercise points

Recap: Compiler Correctness

Compiler Correctness



To do:

1. Definition of abstract machine
2. Definition of (operational) semantics of machine instructions
3. Definition of translation WHILE $\rightarrow$ machine code (“compiler”)
4. Proof: semantics of generated machine code = semantics of original source code

Recap: Compiler Correctness

The Abstract Machine

Definition (Abstract machine)

The abstract machine (AM) is given by

- programs $P \in \mathit{Code}$ and instructions p:

$$
\begin{aligned}
P &::= p^* \\
p &::= \texttt{PUSH}(z) \mid \texttt{PUSH}(t) \mid \texttt{ADD} \mid \texttt{SUB} \mid \texttt{MULT} \mid \texttt{EQ} \mid \texttt{GT} \mid \texttt{NOT} \mid \texttt{AND} \mid \texttt{OR} \mid \\
&\quad\ \texttt{LOAD}(x) \mid \texttt{STO}(x) \mid \texttt{JMP}(k) \mid \texttt{JMPF}(k)
\end{aligned}
$$

  (where $z, k \in \mathbb{Z}$, $t \in \mathbb{B}$, and $x \in \mathit{Var}$)
- configurations of the form $\langle \mathit{pc}, e, \sigma \rangle \in \mathit{Cnf}$ where
  - $\mathit{pc} \in \mathbb{Z}$ is the program counter (i.e., address of next instruction to be executed)
  - $e \in \mathit{Stk} := (\mathbb{Z} \cup \mathbb{B})^*$ is the evaluation stack (top right)
  - $\sigma \in \Sigma := (\mathit{Var} \to \mathbb{Z})$ is the (storage) state

  (thus $\mathit{Cnf} = \mathbb{Z} \times \mathit{Stk} \times \Sigma$)
- initial configurations of the form $\langle 0, \varepsilon, \sigma \rangle$
- final configurations of the form $\langle |P|, e, \sigma \rangle$

Application: Extension of Code and Stack

Lemma

If $P \vdash \langle pc, e, \sigma \rangle \rhd^* \langle pc', e', \sigma' \rangle$, *then*

$$P_1; P; P_2 \vdash \langle |P_1| + pc, e_0 : e, \sigma \rangle \rhd^* \langle |P_1| + pc', e_0 : e', \sigma' \rangle$$

for all $P_1, P_2 \in \mathit{Code}$ *and* $e_0 \in \mathit{Stk}$.

Interpretation: both the code and the stack component can be extended without actually changing the behaviour of the machine

Proof.

by induction on the length of the computation (on the board) □

Another Property: Determinism

Lemma

The semantics of AM is deterministic: for all $\gamma, \gamma', \gamma'' \in \mathit{Cnf}$,

$$\gamma \rhd \gamma' \text{ and } \gamma \rhd \gamma'' \text{ imply } \gamma' = \gamma''.$$

Proof (Idea).

- Instruction to be executed is unambiguously given by program counter
- Topmost stack entries and storage state then yield unique successor configuration □

Thus the following function is well defined:

Definition (Semantics of AM)

The semantics of an AM program is given by $\mathfrak{M}[\![.]\!] : \mathit{Code} \to (\Sigma \dashrightarrow \Sigma)$ as follows:

$$\mathfrak{M}[\![P]\!]\sigma := \begin{cases} \sigma' & \text{if } P \vdash \langle 0, \varepsilon, \sigma \rangle \rhd^* \langle |P|, e, \sigma' \rangle \text{ for some } e \in \mathit{Stk} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Recap: Syntax of WHILE Programs

Definition (Syntax of WHILE (Definition 1.2))

The syntax of WHILE programs is defined by the following context-free grammar:

$$a ::= z \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \in \mathit{AExp}$$
$$b ::= t \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in \mathit{BExp}$$
$$c ::= \texttt{skip} \mid x := a \mid c_1 \,;\, c_2 \mid \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end} \mid \texttt{while } b \texttt{ do } c \texttt{ end} \in \mathit{Cmd}$$

Translation of Arithmetic Expressions

Definition 5.1 (Translation of arithmetic expressions)

The translation function

$$\mathfrak{T}_a[\![.]\!] : \mathit{AExp} \to \mathit{Code}$$

is given by

$$\begin{aligned}
\mathfrak{T}_a[\![z]\!] &:= \mathtt{PUSH}(z) \\
\mathfrak{T}_a[\![x]\!] &:= \mathtt{LOAD}(x) \\
\mathfrak{T}_a[\![a_1 + a_2]\!] &:= \mathfrak{T}_a[\![a_1]\!]; \mathfrak{T}_a[\![a_2]\!]; \mathtt{ADD} \\
\mathfrak{T}_a[\![a_1 - a_2]\!] &:= \mathfrak{T}_a[\![a_1]\!]; \mathfrak{T}_a[\![a_2]\!]; \mathtt{SUB} \\
\mathfrak{T}_a[\![a_1 * a_2]\!] &:= \mathfrak{T}_a[\![a_1]\!]; \mathfrak{T}_a[\![a_2]\!]; \mathtt{MULT}
\end{aligned}$$

Example 5.2

$$\begin{aligned}
\mathfrak{T}_a[\![\mathtt{x + 1}]\!] &= \mathfrak{T}_a[\![\mathtt{x}]\!]; \mathfrak{T}_a[\![\mathtt{1}]\!]; \mathtt{ADD} \\
&= \mathtt{LOAD(x)}; \mathtt{PUSH(1)}; \mathtt{ADD}
\end{aligned}$$

Translation of Boolean Expressions

Definition 5.3 (Translation of Boolean expressions)

The translation function

$$\mathfrak{T}_b[\![.]\!] : \mathit{BExp} \to \mathit{Code}$$

is given by

$$\begin{aligned}
\mathfrak{T}_b[\![\mathsf{true}]\!] &:= \mathtt{PUSH}(\mathsf{true}) \\
\mathfrak{T}_b[\![\mathsf{false}]\!] &:= \mathtt{PUSH}(\mathsf{false}) \\
\mathfrak{T}_b[\![a_1{=}a_2]\!] &:= \mathfrak{T}_a[\![a_1]\!]\,;\mathfrak{T}_a[\![a_2]\!]\,;\mathtt{EQ} \\
\mathfrak{T}_b[\![a_1{>}a_2]\!] &:= \mathfrak{T}_a[\![a_1]\!]\,;\mathfrak{T}_a[\![a_2]\!]\,;\mathtt{GT} \\
\mathfrak{T}_b[\![\neg b]\!] &:= \mathfrak{T}_b[\![b]\!]\,;\mathtt{NOT} \\
\mathfrak{T}_b[\![b_1 \wedge b_2]\!] &:= \mathfrak{T}_b[\![b_1]\!]\,;\mathfrak{T}_b[\![b_2]\!]\,;\mathtt{AND} \\
\mathfrak{T}_b[\![b_1 \vee b_2]\!] &:= \mathfrak{T}_b[\![b_1]\!]\,;\mathfrak{T}_b[\![b_2]\!]\,;\mathtt{OR}
\end{aligned}$$

Translation of Statements

Definition 5.4 (Translation of statements)

The translation function $\mathfrak{T}_c[\![.]\!] : \mathit{Cmd} \to \mathit{Code}$ is given by

$$
\begin{aligned}
\mathfrak{T}_c[\![\texttt{skip}]\!] &:= \varepsilon \\
\mathfrak{T}_c[\![x := a]\!] &:= \mathfrak{T}_a[\![a]\!]\,;\texttt{STO}(x) \\
\mathfrak{T}_c[\![c_1\,;c_2]\!] &:= \mathfrak{T}_c[\![c_1]\!]\,;\mathfrak{T}_c[\![c_2]\!] \\
\mathfrak{T}_c[\![\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}]\!] &:= \mathfrak{T}_b[\![b]\!]\,;\texttt{JMPF}(|\mathfrak{T}_c[\![c_1]\!]| + 2)\,; \\
&\quad\ \mathfrak{T}_c[\![c_1]\!]\,;\texttt{JMP}(|\mathfrak{T}_c[\![c_2]\!]| + 1)\,; \\
&\quad\ \mathfrak{T}_c[\![c_2]\!] \\
\mathfrak{T}_c[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!] &:= \mathfrak{T}_b[\![b]\!]\,;\texttt{JMPF}(|\mathfrak{T}_c[\![c]\!]| + 2)\,; \\
&\quad\ \mathfrak{T}_c[\![c]\!]\,;\texttt{JMP}(-(|\mathfrak{T}_b[\![b]\!]| + |\mathfrak{T}_c[\![c]\!]| + 1))
\end{aligned}
$$

Translation of Statements

Example 5.5 (Factorial program)

$$
\begin{aligned}
&\mathfrak{T}_c[\![\texttt{y:=1; while } \neg\texttt{(x=1) do y:=y*x; x:=x-1 end}]\!] \\
&= \mathfrak{T}_c[\![\texttt{y:=1}]\!];\mathfrak{T}_c[\![\texttt{while } \underbrace{\neg\texttt{(x=1)}}_{b} \texttt{ do } \underbrace{\texttt{y:=y*x; x:=x-1}}_{c} \texttt{ end}]\!] \\
&= \texttt{PUSH(1);STO(y);} \\
&\quad \mathfrak{T}_b[\![b]\!];\texttt{JMPF}(|\mathfrak{T}_c[\![c]\!]| + 2); \\
&\quad \mathfrak{T}_c[\![c]\!];\texttt{JMP}(-(|\mathfrak{T}_b[\![b]\!]| + |\mathfrak{T}_c[\![c]\!]| + 1)) \\
&= \texttt{PUSH(1);STO(y);} \\
&\quad \texttt{LOAD(x);PUSH(1);EQ;NOT;JMPF}(8+2); \\
&\quad \texttt{LOAD(y);LOAD(x);MULT;STO(y);} \\
&\quad \texttt{LOAD(x);PUSH(1);SUB;STO(x);JMP}(-(4+8+1)) \\
&= \texttt{PUSH(1);STO(y);} \\
&\quad \texttt{LOAD(x);PUSH(1);EQ;NOT;JMPF(10);} \\
&\quad \texttt{LOAD(y);LOAD(x);MULT;STO(y);} \\
&\quad \texttt{LOAD(x);PUSH(1);SUB;STO(x);JMP(-13)}
\end{aligned}
$$

Execution of Factorial Program

Example 5.6 (Factorial program)

Let $P :=$ `0:PUSH(1); 1:STO(y); 2:LOAD(x); 3:PUSH(1); 4:EQ; 5:NOT; 6:JMPF(10);`
`7:LOAD(y); 8:LOAD(x); 9:MULT; 10:STO(y);`
`11:LOAD(x); 12:PUSH(1); 13:SUB; 14:STO(x); 15:JMP(-13)`
and $\sigma \in \Sigma$ with $\sigma(x) = 2$.

$$
\begin{aligned}
&\langle\ 0, \varepsilon\ , \sigma\ \rangle \\
\rhd\ &\langle\ 1, 1\ , \sigma\ \rangle \\
\rhd\ &\langle\ 2, \varepsilon\ , \sigma[y \mapsto 1]\rangle \\
\rhd\ &\langle\ 3, 2\ , \sigma[y \mapsto 1]\rangle \\
\rhd\ &\langle\ 4, 2:1, \sigma[y \mapsto 1]\rangle \\
\rhd\ &\langle\ 5, \mathsf{false}, \sigma[y \mapsto 1]\rangle \\
\rhd\ &\langle\ 6, \mathsf{true}\ , \sigma[y \mapsto 1]\rangle \\
\rhd\ &\langle\ 7, \varepsilon\ , \sigma[y \mapsto 1]\rangle \\
\rhd\ &\langle\ 8, 1\ , \sigma[y \mapsto 1]\rangle \\
\rhd\ &\langle\ 9, 1:2, \sigma[y \mapsto 1]\rangle \\
\rhd\ &\langle 10, 2\ , \sigma[y \mapsto 1]\rangle
\end{aligned}
\qquad
\begin{aligned}
\rhd\ &\langle 11, \varepsilon\ , \sigma[y \mapsto 2]\ \rangle \\
\rhd\ &\langle 12, 2\ , \sigma[y \mapsto 2]\ \rangle \\
\rhd\ &\langle 13, 2:1, \sigma[y \mapsto 2]\ \rangle \\
\rhd\ &\langle 14, 1\ , \sigma[y \mapsto 2]\ \rangle \\
\rhd\ &\langle 15, \varepsilon\ , \sigma[x \mapsto 1, y \mapsto 2]\rangle \\
\rhd\ &\langle\ 2, \varepsilon\ , \sigma[x \mapsto 1, y \mapsto 2]\rangle \\
\rhd\ &\langle\ 3, 1\ , \sigma[x \mapsto 1, y \mapsto 2]\rangle \\
\rhd\ &\langle\ 4, 1:1, \sigma[x \mapsto 1, y \mapsto 2]\rangle \\
\rhd\ &\langle\ 5, \mathsf{true}\ , \sigma[x \mapsto 1, y \mapsto 2]\rangle \\
\rhd\ &\langle\ 6, \mathsf{false}, \sigma[x \mapsto 1, y \mapsto 2]\rangle \\
\rhd\ &\langle 16, \varepsilon\ , \sigma[x \mapsto 1, y \mapsto 2]\rangle
\end{aligned}
$$

Correctness of $\mathfrak{T}_a[\![.]\!]$ I

Definition (Recap: Evaluation of arithmetic expressions (Definition 2.2))

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:

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

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

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

Correctness of $\mathfrak{T}_a[\![.]\!]$ II

Lemma 5.7 (Correctness of $\mathfrak{T}_a[\![.]\!]$)

For every $a \in AExp$, $\sigma \in \Sigma$ and $z \in \mathbb{Z}$,

$$\langle a, \sigma \rangle \to z \quad \textit{implies} \quad \mathfrak{T}_a[\![a]\!] \vdash \langle 0, \varepsilon, \sigma \rangle \rhd^* \langle |\mathfrak{T}_a[\![a]\!]|, z, \sigma \rangle.$$

Note: implication sufficient to ensure soundness and completeness as expression evaluation total and semantics of machine code deterministic (Lemma 4.10)

Proof.

by induction on the syntactic structure of a (on the board) □

Proof of Compiler Correctness

Correctness of $\mathfrak{T}_b[\![.]\!]$ I

Definition (Recap: Semantics of Boolean expressions (Definition 2.7))

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

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

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

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

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

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

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

($\vee$ analogously)

Correctness of $\mathfrak{T}_b[\![.]\!]$ II

Lemma 5.8 (Correctness of $\mathfrak{T}_b[\![.]\!]$)

For every $b \in BExp$, $\sigma \in \Sigma$ and $t \in \mathbb{B}$,

$$\langle b, \sigma \rangle \to t \quad \textit{implies} \quad \mathfrak{T}_b[\![b]\!] \vdash \langle 0, \varepsilon, \sigma \rangle \rhd^* \langle |\mathfrak{T}_b[\![b]\!]|, t, \sigma \rangle.$$

Note: implication sufficient to ensure soundness and completeness as expression evaluation total and semantics of machine code deterministic (Lemma 4.10)

Proof.

by induction on the syntactic structure of b (omitted) □

Correctness of $\mathfrak{T}_c[\![.]\!]$ I

Definition (Recap: Operational functional (Definition 4.1))

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:

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

Definition (Recap: Semantics of machine code (Definition 4.11))

The semantics of an AM program is given by $\mathfrak{M}[\![.]\!] : \mathit{Code} \to (\Sigma \dashrightarrow \Sigma)$ as follows:

$$\mathfrak{M}[\![P]\!]\sigma := \begin{cases} \sigma' & \text{if } P \vdash \langle 0, \varepsilon, \sigma\rangle \rhd^* \langle |P|, e, \sigma'\rangle \text{ for some } e \in \mathit{Stk} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Correctness of $\mathfrak{T}_c[\![.]\!]$ II

Theorem 5.9 (Correctness of $\mathfrak{T}_c[\![.]\!]$)

For every $c \in Cmd$,

$$\mathfrak{O}[\![c]\!] = \mathfrak{M}[\![\mathfrak{T}_c[\![c]\!]]\!].$$

Proof carried out in two parts. First step: from source to machine code

Lemma 5.10 (Completeness of $\mathfrak{T}_c[\![.]\!]$)

For every $c \in Cmd$ and $\sigma, \sigma' \in \Sigma$,

$$\langle c, \sigma\rangle \to \sigma' \quad \textit{implies} \quad \mathfrak{T}_c[\![c]\!] \vdash \langle 0, \varepsilon, \sigma\rangle \rhd^* \langle |\mathfrak{T}_c[\![c]\!]|, \varepsilon, \sigma'\rangle.$$

Proof.

by induction on the derivation tree of $\langle c, \sigma\rangle \to \sigma'$ (on the board) □

Proof of Compiler Correctness

Correctness of $\mathfrak{T}_c[\![.]\!]$ III

Second step: from machine to source code

Lemma 5.11 (Soundness of $\mathfrak{T}_c[\![.]\!]$)

For every $c \in Cmd$, $\sigma, \sigma' \in \Sigma$, and $e \in Stk$,

$$\mathfrak{T}_c[\![c]\!] \vdash \langle 0, \varepsilon, \sigma \rangle \rhd^* \langle |\mathfrak{T}_c[\![c]\!]|, e, \sigma' \rangle \quad \textit{implies} \quad \langle c, \sigma \rangle \to \sigma' \textit{ and } e = \varepsilon.$$

Proof.

by induction on the length of the computation sequence

$$\langle 0, \varepsilon, \sigma \rangle \rhd^* \langle |\mathfrak{T}_c[\![c]\!]|, e, \sigma' \rangle$$

(see exercises) □