Semantics and Verification of Software

Winter Semester 2017/18

Lecture 12: Axiomatic Semantics of WHILE IV
(Axiomatic Equivalence & Timed Correctness Properties)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

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

Proving Total Correctness

Goal: syntactic derivation of valid total correctness properties

Definition (Hoare Logic for total correctness)

The Hoare rules for total correctness are given by (where $i \in LVar$)

$$\text{(skip)}\ \frac{}{\{A\}\ \texttt{skip}\ \{\Downarrow A\}} \qquad \text{(asgn)}\ \frac{}{\{A[x \mapsto a]\}\ x := a\ \{\Downarrow A\}}$$

$$\text{(seq)}\ \frac{\{A\}\ c_1\ \{\Downarrow C\} \quad \{C\}\ c_2\ \{\Downarrow B\}}{\{A\}\ c_1 ; c_2\ \{\Downarrow B\}} \qquad \text{(if)}\ \frac{\{A \wedge b\}\ c_1\ \{\Downarrow B\} \quad \{A \wedge \neg b\}\ c_2\ \{\Downarrow B\}}{\{A\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}\ \{\Downarrow B\}}$$

$$\text{(while)}\ \frac{\models (i \geq 0 \wedge A(i+1) \Rightarrow b) \quad \{i \geq 0 \wedge A(i+1)\}\ c\ \{\Downarrow A(i)\} \quad \models (A(0) \Rightarrow \neg b)}{\{\exists i. i \geq 0 \wedge A(i)\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{\Downarrow A(0)\}}$$

$$\text{(cons)}\ \frac{\models (A \Rightarrow A') \quad \{A'\}\ c\ \{\Downarrow B'\} \quad \models (B' \Rightarrow B)}{\{A\}\ c\ \{\Downarrow B\}}$$

A total correctness property is provable (notation: $\vdash \{A\}\ c\ \{\Downarrow B\}$) if it is derivable by the Hoare rules. In case of (while), $A(i)$ is called a (loop) invariant.

Axiomatic Equivalence

In the axiomatic semantics, two statements have to be considered equivalent if they are indistinguishable w.r.t. (partial) correctness properties:

Definition (Axiomatic equivalence)

Two statements $c_1, c_2 \in Cmd$ are called axiomatically equivalent (notation: $c_1 \approx c_2$) if, for all assertions $A, B \in Assn$,

$$\models \{A\}\, c_1\, \{B\} \quad \Longleftrightarrow \quad \models \{A\}\, c_2\, \{B\}.$$

(later: total correctness yields same notion of equivalence)

Characteristic Assertions I

The following results are based of the following encoding of states by assertions:

Definition 12.1

Given a state $\sigma \in \Sigma$ and a non-empty finite subset of program variables $X \subseteq Var$, the characteristic assertion of σ w.r.t. X is given by

$$state(\sigma, X) := \bigwedge_{x \in X} (x = \underbrace{\sigma(x)}_{\in \mathbb{Z}}) \in Assn$$

Moreover, we let $state(\sigma, \emptyset) := \text{true}$ and $state(\bot, X) := \text{false}$.

Corollary 12.2

For all finite $X \subseteq Var$ and $\sigma \in \Sigma_\bot$,

$$\sigma \models state(\sigma, X)$$

Characteristic Assertions II

Programs and characteristic state assertions are obviously related as follows:

Corollary 12.3

Let $c \in Cmd$, and let $FV(c) \subseteq Var$ denote the set of all variables occurring in c. Then, for every finite $X \supseteq FV(c)$ and $\sigma \in \Sigma$,

$$\models \{state(\sigma, X)\}\, c\, \{state(\mathfrak{C}[\![c]\!]\sigma, X)\}$$

Example 12.4 (Factorial program)

For $c := (\mathtt{y:=1;\ while\ \neg(x=1)\ do\ y:=y*x;\ x:=x-1\ end})$, $X = \{\mathtt{x}, \mathtt{y}\}$, $\sigma(\mathtt{x}) = 3$ and $\sigma(\mathtt{y}) = 0$, we obtain

$$state(\sigma, X) = (\mathtt{x=3} \wedge \mathtt{y=0}) \quad \text{and} \quad state(\mathfrak{C}[\![c]\!]\sigma, X) = (\mathtt{x=1} \wedge \mathtt{y=6})$$

and thus $\models \{state(\sigma, X)\}\, c\, \{state(\mathfrak{C}[\![c]\!]\sigma, X)\}$.

If $X \not\supseteq FV(c)$, then the claim does not hold: e.g., $\not\models \{\mathtt{y=0}\}\, c\, \{\mathtt{y=6}\}$!

Partial vs. Total Equivalence

Now we can show that considering total rather than partial correctness properties yields the same notion of equivalence:

Theorem 12.5

Let $c_1, c_2 \in Cmd$. The following propositions are equivalent:

1. $\forall A, B \in Assn: \quad \models \{A\}\, c_1\, \{B\} \iff \models \{A\}\, c_2\, \{B\}$
2. $\forall A, B \in Assn: \quad \models \{A\}\, c_1\, \{\Downarrow B\} \iff \models \{A\}\, c_2\, \{\Downarrow B\}$

Proof.

on the board $\square$

Axiomatic vs. Operational/Denotational Equiv.

Theorem 12.6

Axiomatic and operational/denotational equivalence coincide, i.e., for all $c_1, c_2 \in Cmd$,

$$c_1 \approx c_2 \iff c_1 \sim c_2.$$

Proof.

on the board $\square$

Correctness Properties for Execution Time

The Approach

- Definition 11.3: proof system for total correctness
- Can be used to show that program terminates bus does not give any information about required resources
- Goal: extend proof system to give (order of magnitude of) execution time of a statement
- Details in H.R. Nielson, F. Nielson: *Semantics with Applications: An Appetizer*, Springer, 2007, Section 10.2
- Informal guidelines (idea: each instruction of abstract machine of Lecture 4 takes one time unit):
  - `skip`: execution time $\mathcal{O}(1)$ (that is, bounded by a constant)
  - assignment: execution time $\mathcal{O}(1)$ (with maximal size of RHS as constant)
  - composition: sum of execution times of constituent statements
  - conditional: maximal execution time of branches
  - iteration: sum over all iterations of execution times of loop body
- Procedure:
  1. Extend evaluation relation for expressions to give exact evaluation times
  2. Extend execution relation for statements to give exact execution times
  3. Extend total correctness proof system to give order of magnitude of execution time of statements

Recap: Translation of Arithmetic Expressions

Definition (Translation of arithmetic expressions (Definition 5.1))

The translation function

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

is given by

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

Timed Evaluation of Arithmetic Expressions

Definition 12.7 (Timed Evaluation of arithmetic expressions (extends Definition 2.2))

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

Axioms: $\dfrac{}{\langle z, \sigma \rangle \xrightarrow{1} z}$ $\qquad \dfrac{}{\langle x, \sigma \rangle \xrightarrow{1} \sigma(x)}$

Rules:

$$\frac{\langle a_1, \sigma \rangle \xrightarrow{\tau_1} z_1 \quad \langle a_2, \sigma \rangle \xrightarrow{\tau_2} z_2}{\langle a_1 + a_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} z} \text{ where } z := z_1 + z_2$$

$$\frac{\langle a_1, \sigma \rangle \xrightarrow{\tau_1} z_1 \quad \langle a_2, \sigma \rangle \xrightarrow{\tau_2} z_2}{\langle a_1 - a_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} z} \text{ where } z := z_1 - z_2$$

$$\frac{\langle a_1, \sigma \rangle \xrightarrow{\tau_1} z_1 \quad \langle a_2, \sigma \rangle \xrightarrow{\tau_2} z_2}{\langle a_1 * a_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} z} \text{ where } z := z_1 \cdot z_2$$

Recap: Translation of Boolean Expressions

Definition (Translation of Boolean expressions (Definition 5.3))

The translation function

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

is given by

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

Timed Evaluation of Boolean Expressions

Definition 12.8 (Timed Evaluation of Boolean expressions (extends Definition 2.7))

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

$$\frac{}{\langle t, \sigma \rangle \xrightarrow{1} t}$$

$$\frac{\langle a_1, \sigma \rangle \xrightarrow{\tau_1} z \quad \langle a_2, \sigma \rangle \xrightarrow{\tau_2} z}{\langle a_1 = a_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} \mathsf{true}} \qquad \frac{\langle a_1, \sigma \rangle \xrightarrow{\tau_1} z_1 \quad \langle a_2, \sigma \rangle \xrightarrow{\tau_2} z_2}{\langle a_1 = a_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} \mathsf{false}} \text{ if } z_1 \neq z_2$$

$$\frac{\langle a_1, \sigma \rangle \xrightarrow{\tau_1} z_1 \quad \langle a_2, \sigma \rangle \xrightarrow{\tau_2} z_2}{\langle a_1 > a_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} \mathsf{true}} \text{ if } z_1 > z_2 \qquad \frac{\langle a_1, \sigma \rangle \xrightarrow{\tau_1} z_1 \quad \langle a_2, \sigma \rangle \xrightarrow{\tau_2} z_2}{\langle a_1 > a_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} \mathsf{false}} \text{ if } z_1 \leq z_2$$

$$\frac{\langle b, \sigma \rangle \xrightarrow{\tau} \mathsf{false}}{\langle \neg b, \sigma \rangle \xrightarrow{\tau+1} \mathsf{true}} \qquad \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \mathsf{true}}{\langle \neg b, \sigma \rangle \xrightarrow{\tau+1} \mathsf{false}}$$

$$\frac{\langle b_1, \sigma \rangle \xrightarrow{\tau_1} \mathsf{true} \quad \langle b_2, \sigma \rangle \xrightarrow{\tau_2} \mathsf{true}}{\langle b_1 \wedge b_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} \mathsf{true}} \qquad \frac{\langle b_1, \sigma \rangle \xrightarrow{\tau_1} \mathsf{true} \quad \langle b_2, \sigma \rangle \xrightarrow{\tau_2} \mathsf{false}}{\langle b_1 \wedge b_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} \mathsf{false}}$$

$$\frac{\langle b_1, \sigma \rangle \xrightarrow{\tau_1} \mathsf{false} \quad \langle b_2, \sigma \rangle \xrightarrow{\tau_2} \mathsf{true}}{\langle b_1 \wedge b_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} \mathsf{false}} \qquad \frac{\langle b_1, \sigma \rangle \xrightarrow{\tau_1} \mathsf{false} \quad \langle b_2, \sigma \rangle \xrightarrow{\tau_2} \mathsf{false}}{\langle b_1 \wedge b_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2+1} \mathsf{false}}$$

($\vee$ analogously)

Operational Semantics with Exact Execution Times

Recap: Translation of Statements

Definition (Translation of statements (Definition 5.4))

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

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

Operational Semantics with Exact Execution Times

Timed Execution of Statements

Definition 12.9 (Timed execution relation for statements (extends Definition 3.2))

For $c \in Cmd$, $\sigma, \sigma' \in \Sigma$, and $\tau \in \mathbb{N}$, the timed execution relation $\langle c, \sigma \rangle \xrightarrow{\tau} \sigma'$ is defined by:

$$\text{(skip)}\ \frac{}{\langle \texttt{skip}, \sigma \rangle \xrightarrow{1} \sigma} \qquad \text{(asgn)}\ \frac{\langle a, \sigma \rangle \xrightarrow{\tau} z}{\langle x := a, \sigma \rangle \xrightarrow{\tau+1} \sigma[x \mapsto z]}$$

$$\text{(seq)}\ \frac{\langle c_1, \sigma \rangle \xrightarrow{\tau_1} \sigma' \quad \langle c_2, \sigma' \rangle \xrightarrow{\tau_2} \sigma''}{\langle c_1 ; c_2, \sigma \rangle \xrightarrow{\tau_1+\tau_2} \sigma''} \qquad \text{(if-t)}\ \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \mathsf{true} \quad \langle c_1, \sigma \rangle \xrightarrow{\tau_1} \sigma'}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}, \sigma \rangle \xrightarrow{\tau+\tau_1+2} \sigma'}$$

$$\text{(wh-f)}\ \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \mathsf{false}}{\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}, \sigma \rangle \xrightarrow{\tau+1} \sigma} \qquad \text{(if-f)}\ \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \mathsf{false} \quad \langle c_2, \sigma \rangle \xrightarrow{\tau_2} \sigma'}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}, \sigma \rangle \xrightarrow{\tau+\tau_2+1} \sigma'}$$

$$\text{(wh-t)}\ \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \mathsf{true} \quad \langle c, \sigma \rangle \xrightarrow{\tau_1} \sigma' \quad \langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}, \sigma' \rangle \xrightarrow{\tau_2} \sigma''}{\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}, \sigma \rangle \xrightarrow{\tau+\tau_1+\tau_2+2} \sigma''}$$

Timed Correctness Properties

Recap: Total Correctness Properties

So far: total correctness properties of the form

$$\{A\}\ c\ \{\Downarrow B\}$$

where $c \in Cmd$ and $A, B \in Assn$

Validity of property $\{A\}\ c\ \{\Downarrow B\}$

For all states $\sigma \in \Sigma$ which satisfy A:
the execution of c in σ terminates and yields a state which satisfies B.

Timed Correctness Properties

Now: timed correctness properties of the form

$$\{A\}\, c\, \{e \Downarrow B\}$$

where $c \in Cmd$, $A, B \in Assn$, and $e \in AExp$

Validity of property $\{A\}\, c\, \{e \Downarrow B\}$

For all states $\sigma \in \Sigma$ which satisfy A: the execution of c in σ terminates in a state satisfying B, and the required execution time is in $\mathcal{O}(e)$

Example 12.10

1. $\{\mathtt{x} = 3\}$ `y:=1; while ¬(x=1) do y:=y*x; x:=x-1 end` $\{1 \Downarrow \mathsf{true}\}$ expresses that for constant input 3, the execution time of the factorial program is bounded by a constant
2. $\{\mathtt{x} > 0\}$ `y:=1; while ¬(x=1) do y:=y*x; x:=x-1 end` $\{\mathtt{x} \Downarrow \mathsf{true}\}$ expresses that for positive input values, the execution time of the factorial program is linear in that value

Semantics of Timed Correctness Properties

Definition 12.11 (Semantics of timed correctness properties (extends Definition 11.1))

Let $A, B \in Assn$, $c \in Cmd$, and $e \in AExp$. Then $\{A\}\, c\, \{e \Downarrow B\}$ is called valid (notation: $\models \{A\}\, c\, \{e \Downarrow B\}$) if there exists $k \in \mathbb{N}$ such that for each $I \in Int$ and each $\sigma \models^I A$, there exist $\sigma' \in \Sigma$ and $\tau \leq k \cdot \mathfrak{A}[\![e]\!]\sigma$ such that $\langle c, \sigma \rangle \xrightarrow{\tau} \sigma'$ and $\sigma' \models^I B$

Note: e is evaluated in initial (rather than final) state