Semantics and Verification of Software

Winter Semester 2017/18

Lecture 13: Axiomatic Semantics of WHILE V (Proving Timed 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

Outline of Lecture 13

Recap: Correctness Properties for Execution Time

Timed Execution of Statements

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

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

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

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

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

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

Recap: Correctness Properties for Execution Time

Timed Correctness Properties

Now: timed correctness properties of the form

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

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

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

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

Recap: Correctness Properties for Execution Time

Semantics of Timed Correctness Properties

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

Let $A, B \in \mathit{Assn}$, $c \in \mathit{Cmd}$, and $e \in \mathit{AExp}$. 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 $I \in \mathit{Int}$ 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

Outline of Lecture 13

Proving Timed Correctness I

Definition 13.1 (Hoare Logic for timed correctness (extends Definition 11.3))

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

$$\text{(skip)}\ \frac{}{\{A\}\ \texttt{skip}\ \{1 \Downarrow A\}} \qquad \text{(asgn)}\ \frac{}{\{A[x \mapsto a]\}\ x := a\ \{1 \Downarrow A\}}$$

$$\text{(seq)}\ \frac{\{A \wedge e_2' = u\}\ c_1\ \{e_1 \Downarrow C \wedge e_2 \leq u\} \quad \{C\}\ c_2\ \{e_2 \Downarrow B\}}{\{A\}\ c_1 ; c_2\ \{e_1 + e_2' \Downarrow B\}}$$

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

$$\text{(while)}\ \frac{\{i \geq 0 \wedge A(i+1) \wedge e' = u\}\ c\ \{e_0 \Downarrow A(i) \wedge e \leq u\}}{\{\exists i.\, i \geq 0 \wedge A(i)\}\ \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \{e \Downarrow A(0)\}}$$

where $\models (i \geq 0 \wedge A(i+1)) \Rightarrow (b \wedge e \geq e_0 + e')$ and $\models A(0) \Rightarrow (\neg b \wedge e \geq 1)$

$$\text{(cons)}\ \frac{\models (A \Rightarrow (A' \wedge \exists k \in \mathbb{N}.\, e' \leq k \cdot e)) \quad \{A'\}\ c\ \{e' \Downarrow B'\} \quad \models (B' \Rightarrow B)}{\{A\}\ c\ \{e \Downarrow B\}}$$

Proving Timed Correctness II

(asgn) Assignment can be executed in constant time as size of expressions bounded by a constant

(seq) e_2 expresses time bound for c_2 relative to initial state of c_2 (and not $c_1 ; c_2$)
$\Rightarrow$ cannot use $e_1 + e_2$ as time bound for $c_1 ; c_2$
$\Rightarrow$ replace e_2 by e_2' such that value of e_2' in initial state of c_1 bounds value of e_2 in initial state of c_2 (= final state of c_1)

(if) e represents maximal execution time for both branches

(while) e_0/e represents execution time for body/whole loop
$\Rightarrow$ cannot use $e_0 + e$ for total time as e_0/e refers to state before/after body is executed once
$\Rightarrow$ introduce e' whose evaluation before body bounds e evaluated after body
$\Rightarrow e \geq e_0 + e'$ as e has to bound loop execution time independently of number of iterations (recurrence (in-)equation; cf. examples)

(cons) additionally allows over-approximation of execution time

Proving Timed Correctness III

Example 13.2

1. Prove that

$$\vdash \{\texttt{x} > 0\}\ \texttt{y:=1; while } \neg\texttt{(x=1) do y:=y*x; x:=x-1 end}\ \{\texttt{x} \Downarrow \text{true}\}$$

(on the board)

Proving Timed Correctness III

Example 13.2

1. Prove that

$$\vdash \{\texttt{x} > 0\}\ \texttt{y:=1; while } \neg\texttt{(x=1) do y:=y*x; x:=x-1 end}\ \{\texttt{x} \Downarrow \text{true}\}$$

   (on the board)
2. Determine expression e_{fac} such that

$$\vdash \{\texttt{x} > 0\}\ \texttt{y:=1; while } \neg\texttt{(x=1) do y:=y*x; x:=x-1 end}\ \{e_{fac} \Downarrow \text{true}\}$$

   (on the board)

Outline of Lecture 13

Soundness and Completeness

Theorem 13.3 (Soundness)

For every timed correctness property $\{A\}\, c\, \{e \Downarrow B\}$,

$$\vdash \{A\}\, c\, \{e \Downarrow B\} \quad \Rightarrow \quad \models \{A\}\, c\, \{e \Downarrow B\}.$$

Proof.

omitted (by structural induction on derivation; only (while) rule) □

Soundness and Completeness

Theorem 13.3 (Soundness)

For every timed correctness property $\{A\}\, c\, \{e \Downarrow B\}$,

$$\vdash \{A\}\, c\, \{e \Downarrow B\} \quad \Rightarrow \quad \models \{A\}\, c\, \{e \Downarrow B\}.$$

Proof.

omitted (by structural induction on derivation; only (while) rule) □

Theorem 13.4 (Relative completeness)

The Hoare Logic for timed correctness properties is relatively complete, i.e., for every $\{A\}\, c\, \{e \Downarrow B\}$*:*

$$\models \{A\}\, c\, \{e \Downarrow B\} \quad \Rightarrow \quad \vdash \{A\}\, c\, \{e \Downarrow B\}.$$

Proof.

omitted □

Outline of Lecture 13

Summary: Axiomatic Semantics

- Formalized by partial/total correctness properties

Summary: Axiomatic Semantics

- Formalized by partial/total correctness properties
- Inductively defined by Hoare Logic proof rules

Summary: Axiomatic Semantics

- Formalized by partial/total correctness properties
- Inductively defined by Hoare Logic proof rules
- Technically involved (especially loop invariants)
  $\Rightarrow$ machine support (proof assistants) indispensable for larger programs

Summary: Axiomatic Semantics

- Formalized by partial/total correctness properties
- Inductively defined by Hoare Logic proof rules
- Technically involved (especially loop invariants)
  $\Rightarrow$ machine support (proof assistants) indispensable for larger programs
- Equivalence of axiomatic and operational/denotational semantics

Summary: Axiomatic Semantics

- Formalized by partial/total correctness properties
- Inductively defined by Hoare Logic proof rules
- Technically involved (especially loop invariants)
  $\Rightarrow$ machine support (proof assistants) indispensable for larger programs
- Equivalence of axiomatic and operational/denotational semantics
- Application: estimation of worst-case execution time

Summary: Axiomatic Semantics

- Formalized by partial/total correctness properties
- Inductively defined by Hoare Logic proof rules
- Technically involved (especially loop invariants)
  $\Rightarrow$ machine support (proof assistants) indispensable for larger programs
- Equivalence of axiomatic and operational/denotational semantics
- Application: estimation of worst-case execution time
- Software engineering aspect: integrated development of program and proof (cf. assertions in Java)

Summary: Axiomatic Semantics

- Formalized by partial/total correctness properties
- Inductively defined by Hoare Logic proof rules
- Technically involved (especially loop invariants)
  $\Rightarrow$ machine support (proof assistants) indispensable for larger programs
- Equivalence of axiomatic and operational/denotational semantics
- Application: estimation of worst-case execution time
- Software engineering aspect: integrated development of program and proof (cf. assertions in Java)
- Systematic approach: mechanised program verification
  1. Start with (correctness) requirements for program
  2. Manually derive corresponding program annotations (assertions)
  3. Automatically derive corresponding verification conditions (using weakest preconditions etc.)
  4. Automatically discharge/simplify verification conditions using theorem prover
  5. Manually complete proof if required

  (cf. Mike Gordon: *Background reading on Hoare Logic*, Chapter 3, `www.cl.cam.ac.uk/~mjcg/Teaching/2011/Hoare/Notes/Notes.pdf`)