# Semantics and Verification of Software

## Summer Semester 2019

## Lecture 13: Axiomatic Semantics of WHILE V (Correctness Properties for Execution Time)

**Thomas Noll**
**Software Modeling and Verification Group**
**RWTH Aachen University**

https://moves.rwth-aachen.de/teaching/ss-19/sv-sw/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

## Exam in Semantics and Verification of Software

- Registration via RWTHonline until 14 June
- Two exam periods:
  - Week 31 (29 July – 2 August)
  - Week 38 (16 September – 20 September)
- Appointment via Foodle (later)

## Outline of Lecture 13

Recap: Axiomatic Equivalence

Axiomatic vs. Operational/Denotational Equivalence

Correctness Properties for Execution Time

Operational Semantics with Exact Execution Times

Timed Correctness Properties

Proving Timed Correctness

Soundness and Completeness

Summary: Axiomatic Semantics

## 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 $c_1, c_2 \in \mathit{Cmd}$ are called axiomatically equivalent (notation: $c_1 \approx c_2$) if, for all assertions $A, B \in \mathit{Assn}$,

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

# Recap: Axiomatic Equivalence

## Characteristic Assertions

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

### Definition

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

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

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

### Corollary

*For all finite $X \subseteq \mathit{Var}$ and $\sigma \in \Sigma$,*

$$\sigma \models \mathit{state}(\sigma, X)$$

# Recap: Axiomatic Equivalence

## Partial vs. Total Equivalence

Using characteristic state assertions, we can show that considering total rather than partial correctness properties yields the same notion of equivalence:

**Theorem**

*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 □

## Outline of Lecture 13

Recap: Axiomatic Equivalence

Axiomatic vs. Operational/Denotational Equivalence

Correctness Properties for Execution Time

Operational Semantics with Exact Execution Times

Timed Correctness Properties

Proving Timed Correctness

Soundness and Completeness

Summary: Axiomatic Semantics

## Axiomatic vs. Operational/Denotational Equiv.

### Theorem 13.1

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

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

## Axiomatic vs. Operational/Denotational Equiv.

### Theorem 13.1

*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 □

## Outline of Lecture 13

Recap: Axiomatic Equivalence

Axiomatic vs. Operational/Denotational Equivalence

Correctness Properties for Execution Time

Operational Semantics with Exact Execution Times

Timed Correctness Properties

Proving Timed Correctness

Soundness and Completeness

Summary: Axiomatic Semantics

# Correctness Properties for Execution Time

## The Approach

- Definition 11.13: proof system for total correctness
- Can be used to show that program terminates bus does not give any information about required resources

# Correctness Properties for Execution Time

## The Approach

- Definition 11.13: 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

# Correctness Properties for Execution Time

## The Approach

- Definition 11.13: 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

# Correctness Properties for Execution Time

## The Approach

- Definition 11.13: 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 proof system for total correctness to give order of magnitude of execution time of statements

## Outline of Lecture 13

Recap: Axiomatic Equivalence

Axiomatic vs. Operational/Denotational Equivalence

Correctness Properties for Execution Time

Operational Semantics with Exact Execution Times

Timed Correctness Properties

Proving Timed Correctness

Soundness and Completeness

Summary: Axiomatic Semantics

## 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

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

## Timed Evaluation of Arithmetic Expressions

### Definition 13.2 (Timed Evaluation of arithmetic expressions (extends Definition 2.2))

Expression $a$ evaluates to $z \in \mathbb{Z}$ in state $\sigma$ 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:

$$\frac{}{\langle z, \sigma \rangle \xrightarrow{1} z} \qquad \frac{}{\langle x, \sigma \rangle \xrightarrow{1} \sigma(x)}$$

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

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

is given by

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

## Timed Evaluation of Boolean Expressions

### Definition 13.3 (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:

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

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

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

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

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

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

($\vee$ analogously)

## Recap: Translation of Statements

### Definition (Translation of statements (Definition 5.4))

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

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

## Timed Execution of Statements

### Definition 13.4 (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:

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

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

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

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

## Outline of Lecture 13

Recap: Axiomatic Equivalence

Axiomatic vs. Operational/Denotational Equivalence

Correctness Properties for Execution Time

Operational Semantics with Exact Execution Times

Timed Correctness Properties

Proving Timed Correctness

Soundness and Completeness

Summary: Axiomatic Semantics

## Recap: Total Correctness Properties

So far: total correctness properties of the form

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

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

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

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

## Timed Correctness Properties

Now: timed correctness properties of the form

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

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

## 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 $\sigma$ terminates in a state satisfying $B$, and the required execution time is in $\mathcal{O}(e)$

## 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 $\sigma$ terminates in a state satisfying $B$, and the required execution time is in $\mathcal{O}(e)$

### Example 13.5

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

## 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 $\sigma$ terminates in a state satisfying $B$, and the required execution time is in $\mathcal{O}(e)$

### Example 13.5

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

## Semantics of Timed Correctness Properties

**Definition 13.6 (Semantics of timed correctness properties (extends Definition 11.11))**

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 $\sigma \in \Sigma$ with $\sigma \models' A$, there exist $\sigma' \in \Sigma$ and $\tau \leq k \cdot \mathfrak{A}[\![e]\!]\sigma$ such that

$$\langle c, \sigma \rangle \xrightarrow{\tau} \sigma' \quad \text{and} \quad \sigma' \models' B$$

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

## Outline of Lecture 13

Recap: Axiomatic Equivalence

Axiomatic vs. Operational/Denotational Equivalence

Correctness Properties for Execution Time

Operational Semantics with Exact Execution Times

Timed Correctness Properties

Proving Timed Correctness

Soundness and Completeness

Summary: Axiomatic Semantics

## Proving Timed Correctness I

### Definition 13.7 (Hoare Logic for timed correctness (extends Definition 11.13))

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

(cons) additionally allows over-approximation of execution time

## Proving Timed Correctness III

### Example 13.8

Prove that $\vdash \{x > 0\}$ `y:=1; while ¬(x=1) do y:=y*x; x:=x-1 end` $\{x \Downarrow \text{true}\}$:

## Proving Timed Correctness III

### Example 13.8

Prove that $\vdash \{\mathtt{x} > 0\}\ \mathtt{y:=1;\ while\ \neg(x=1)\ do\ y:=y*x;\ x:=x-1\ end}\ \{\mathtt{x} \Downarrow \mathsf{true}\}$:

- We use invariant $C(i) = (\mathtt{x} > 0 \wedge \mathtt{x} = i + 1)$ and $u_1, u_2 \in \mathit{LVar}$ for the loop (while) and its body (seq), resp.

## Proving Timed Correctness III

### Example 13.8

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

- We use invariant $C(i) = (\mathtt{x} > 0 \wedge \mathtt{x} = i + 1)$ and $u_1, u_2 \in \mathit{LVar}$ for the loop (while) and its body (seq), resp.
- Applying $\text{(asgn)}\dfrac{}{\{A[x \mapsto a]\}\ x := a\ \{1 \Downarrow A\}}$ twice yields

$$\vdash \{(C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}]\}\ \mathtt{x:=x-1}\ \{1 \Downarrow C(i) \wedge \mathtt{x} \leq u_1\}$$
$$\vdash \{((C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2)[\mathtt{y} \mapsto \mathtt{y*x}]\}\ \mathtt{y:=y*x}\ \{1 \Downarrow (C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2\}$$

## Proving Timed Correctness III

### Example 13.8

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

- We use invariant $C(i) = (\mathtt{x} > 0 \wedge \mathtt{x} = i + 1)$ and $u_1, u_2 \in \mathit{LVar}$ for the loop (while) and its body (seq), resp.
- Applying $\text{(asgn)}\dfrac{}{\{A[x \mapsto a]\}\ x := a\ \{1 \Downarrow A\}}$ twice yields

$$\vdash \{(C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}]\}\ \mathtt{x:=x-1}\ \{1 \Downarrow C(i) \wedge \mathtt{x} \leq u_1\}$$

$$\vdash \{((C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2)[\mathtt{y} \mapsto \mathtt{y*x}]\}\ \mathtt{y:=y*x}\ \{1 \Downarrow (C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2\}$$

- Now $(C(i+1) \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2) \equiv (\mathtt{x} > 0 \wedge \mathtt{x} - 1 = i + 1 \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2)$ implies $((C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2)[\mathtt{y} \mapsto \mathtt{y*x}] \equiv (\mathtt{x - 1} > 0 \wedge \mathtt{x} - 1 = i + 1 \wedge \mathtt{x} - 1 \leq u_1) \wedge 1 \leq u_2)$, such that $\text{(cons)}\dfrac{\models (A \Rightarrow (A' \wedge \exists k \in \mathbb{N}. e' \leq k \cdot e)) \quad \{A'\}\ c\ \{e' \Downarrow B'\} \quad \models (B' \Rightarrow B)}{\{A\}\ c\ \{e \Downarrow B\}}$ with $e = e' = k = 1$ yields

$$\vdash \{C(i+1) \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2\}\ y := y * x\ \{1 \Downarrow (C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2\}$$

# Proving Timed Correctness III

## Example 13.8

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

- Applying $\text{(asgn)}\ \frac{}{\{A[x \mapsto a]\}\ x := a\ \{1 \Downarrow A\}}$ twice yields

$$\vdash \{(C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}]\}\ \mathtt{x:=x-1}\ \{1 \Downarrow C(i) \wedge \mathtt{x} \leq u_1\}$$
$$\vdash \{((C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2)[\mathtt{y} \mapsto \mathtt{y*x}]\}\ \mathtt{y:=y*x}\ \{1 \Downarrow (C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2\}$$

- Now $(C(i+1) \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2) \equiv (\mathtt{x} > 0 \wedge \mathtt{x} - 1 = i + 1 \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2)$ implies $((C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2)[\mathtt{y} \mapsto \mathtt{y*x}] \equiv (\mathtt{x\ -\ 1} > 0 \wedge \mathtt{x} - 1 = i + 1 \wedge \mathtt{x} - 1 \leq u_1) \wedge 1 \leq u_2)$, such that $\text{(cons)}\ \frac{\models (A \Rightarrow (A' \wedge \exists k \in \mathbb{N}.e' \leq k \cdot e))\quad \{A'\}\,c\,\{e' \Downarrow B'\}\quad \models (B' \Rightarrow B)}{\{A\}\,c\,\{e \Downarrow B\}}$ with $e = e' = k = 1$ yields

$$\vdash \{C(i+1) \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2\}\ y := y * x\ \{1 \Downarrow (C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2\}$$

- Thus by $\text{(seq)}\ \frac{\{A \wedge e_2' = u\}\,c_1\,\{e_1 \Downarrow C \wedge e_2 \leq u\}\quad \{C\}\,c_2\,\{e_2 \Downarrow B\}}{\{A\}\,c_1\,;c_2\,\{e_1 + e_2' \Downarrow B\}}$ with $u = u_2$ and $e_1 = e_2 = e_2' = 1$, and with (cons):

$$\vdash \{C(i+1) \wedge \mathtt{x} - 1 = u_1\}\ \mathtt{y\ :=\ y*x;\ x\ :=\ x-1}\ \{1 \Downarrow C(i) \wedge \mathtt{x} \leq u_1\}$$

## Proving Timed Correctness III

### Example 13.8

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

- Now $(C(i+1) \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2) \equiv (\mathtt{x} > 0 \wedge \mathtt{x} - 1 = i + 1 \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2)$ implies $((C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2)[\mathtt{y} \mapsto \mathtt{y*x}] \equiv (\mathtt{x\ -\ 1} > 0 \wedge \mathtt{x} - 1 = i + 1 \wedge \mathtt{x} - 1 \leq u_1) \wedge 1 \leq u_2)$, such that $\text{(cons)}\dfrac{\models (A \Rightarrow (A' \wedge \exists k \in \mathbb{N}.e' \leq k \cdot e))\quad \{A'\}\,c\,\{e' \Downarrow B'\}\quad \models (B' \Rightarrow B)}{\{A\}\,c\,\{e \Downarrow B\}}$ with $e = e' = k = 1$ yields
$$\vdash \{C(i+1) \wedge \mathtt{x} - 1 = u_1 \wedge 1 = u_2\}\ y := y * x\ \{1 \Downarrow (C(i) \wedge \mathtt{x} \leq u_1)[\mathtt{x} \mapsto \mathtt{x-1}] \wedge 1 \leq u_2\}$$
- Thus by $\text{(seq)}\dfrac{\{A \wedge e_2' = u\}\,c_1\,\{e_1 \Downarrow C \wedge e_2 \leq u\}\quad \{C\}\,c_2\,\{e_2 \Downarrow B\}}{\{A\}\,c_1\,;c_2\,\{e_1 + e_2' \Downarrow B\}}$ with $u = u_2$ and $e_1 = e_2 = e_2' = 1$, and with (cons):
$$\vdash \{C(i+1) \wedge \mathtt{x} - 1 = u_1\}\ \mathtt{y\ :=\ y*x;\ x\ :=\ x-1}\ \{1 \Downarrow C(i) \wedge \mathtt{x} \leq u_1\}$$
- Moreover $\models (i \geq 0 \wedge C(i+1)) \Rightarrow (\neg(\mathtt{x} = 1) \wedge \mathtt{x} \geq 1 + (\mathtt{x} - 1))$ and $\models C(0) \Rightarrow (\neg(\neg(\mathtt{x} = 1)) \wedge \mathtt{x} \geq 1)$ such that we can apply $\text{(while)}\dfrac{\models (i \geq 0 \wedge A(i+1)) \Rightarrow (b \wedge e \geq e_0 + e')\quad \{i \geq 0 \wedge A(i+1) \wedge e' = u\}\,c\,\{e_0 \Downarrow A(i) \wedge e \leq u\}\quad \models A(0) \Rightarrow (\neg b \wedge e \geq 1)}{\{\exists i.i \geq 0 \wedge A(i)\}\ \mathtt{while}\ b\ \mathtt{do}\ c\ \mathtt{end}\ \{e \Downarrow A(0)\}}$ with $e_0 = 1$, $e' = \mathtt{x} - 1$ and $e = x$ and get
$$\vdash \{\exists i.i \geq 0 \wedge C(i)\}\ \mathtt{while\ \neg(x=1)\ do\ y:=y*x;\ x:=x-1\ end}\ \{\mathtt{x} \Downarrow C(0)\}$$

## Proving Timed Correctness III

### Example 13.8

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

- Thus by (seq) $\dfrac{\{A \wedge e_2' = u\}\, c_1\, \{e_1 \Downarrow C \wedge e_2 \leq u\} \quad \{C\}\, c_2\, \{e_2 \Downarrow B\}}{\{A\}\, c_1 ; c_2\, \{e_1 + e_2' \Downarrow B\}}$ with $u = u_2$ and $e_1 = e_2 = e_2' = 1$, and with (cons):

$$\vdash \{C(i+1) \wedge \mathtt{x} - 1 = u_1\}\ \mathtt{y := y*x;\ x := x-1}\ \{1 \Downarrow C(i) \wedge \mathtt{x} \leq u_1\}$$

- Moreover $\models (i \geq 0 \wedge C(i+1)) \Rightarrow (\neg(\mathtt{x} = 1) \wedge \mathtt{x} \geq 1 + (\mathtt{x} - 1))$ and $\models C(0) \Rightarrow (\neg(\neg(\mathtt{x} = 1)) \wedge \mathtt{x} \geq 1)$ such that we can apply (while) $\dfrac{\models (i \geq 0 \wedge A(i+1)) \Rightarrow (b \wedge e \geq e_0 + e') \quad \{i \geq 0 \wedge A(i+1) \wedge e' = u\}\, c\, \{e_0 \Downarrow A(i) \wedge e \leq u\} \quad \models A(0) \Rightarrow (\neg b \wedge e \geq 1)}{\{\exists i.\, i \geq 0 \wedge A(i)\}\ \mathtt{while}\ b\ \mathtt{do}\ c\ \mathtt{end}\ \{e \Downarrow A(0)\}}$
  with $e_0 = 1$, $e' = \mathtt{x} - 1$ and $e = x$ and get

$$\vdash \{\exists i.\, i \geq 0 \wedge C(i)\}\ \mathtt{while\ \neg(x=1)\ do\ y:=y*x;\ x:=x-1\ end}\ \{\mathtt{x} \Downarrow C(0)\}$$

- Using (asgn) for the initial assignment

$$\vdash \{\exists i.\, i \geq 0 \wedge C(i) \wedge x \leq u_3\}\ \mathtt{y:=1}\ \{1 \Downarrow \exists i.\, i \geq 0 \wedge C(i) \wedge x \leq u_3\}$$

  and the implication $\models (\mathtt{x} > 0 \wedge \mathtt{x} = u_3) \Rightarrow (\exists i.\, i \geq 0 \wedge C(i) \wedge x \leq u_3)$, (cons) yields

$$\vdash \{\mathtt{x} > 0 \wedge \mathtt{x} = u_3\}\ \mathtt{y:=1}\ \{1 \Downarrow \exists i.\, i \geq 0 \wedge C(i) \wedge x \leq u_3\}$$

## Proving Timed Correctness III

### Example 13.8

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

- Moreover $\models (i \geq 0 \wedge C(i+1)) \Rightarrow (\neg(\mathtt{x} = 1) \wedge \mathtt{x} \geq 1 + (\mathtt{x} - 1))$ and $\models C(0) \Rightarrow (\neg(\neg(\mathtt{x} = 1)) \wedge \mathtt{x} \geq 1)$ such that we can apply (while) $\dfrac{\models (i \geq 0 \wedge A(i+1)) \Rightarrow (b \wedge e \geq e_0 + e') \quad \{i \geq 0 \wedge A(i+1) \wedge e' = u\}\ c\ \{e_0 \Downarrow A(i) \wedge e \leq u\} \quad \models A(0) \Rightarrow (\neg b \wedge e \geq 1)}{\{\exists i.\, i \geq 0 \wedge A(i)\}\ \mathtt{while}\ b\ \mathtt{do}\ c\ \mathtt{end}\ \{e \Downarrow A(0)\}}$
  with $e_0 = 1$, $e' = \mathtt{x} - 1$ and $e = x$ and get
  $$\vdash \{\exists i.\, i \geq 0 \wedge C(i)\}\ \mathtt{while\ \neg(x=1)\ do\ y:=y*x;\ x:=x-1\ end}\ \{\mathtt{x} \Downarrow C(0)\}$$
- Using (asgn) for the initial assignment
  $$\vdash \{\exists i.\, i \geq 0 \wedge C(i) \wedge x \leq u_3\}\ \mathtt{y:=1}\ \{1 \Downarrow \exists i.\, i \geq 0 \wedge C(i) \wedge x \leq u_3\}$$
  and the implication $\models (\mathtt{x} > 0 \wedge \mathtt{x} = u_3) \Rightarrow (\exists i.\, i \geq 0 \wedge C(i) \wedge x \leq u_3)$, (cons) yields
  $$\vdash \{\mathtt{x} > 0 \wedge \mathtt{x} = u_3\}\ \mathtt{y:=1}\ \{1 \Downarrow \exists i.\, i \geq 0 \wedge C(i) \wedge x \leq u_3\}$$
- (seq) now gives
  $$\vdash \{\mathtt{x} > 0\}\ \mathtt{y:=1;\ while\ \neg(x=1)\ do\ y:=y*x;\ x:=x-1\ end}\ \{1 + \mathtt{x} \Downarrow C(0)\}$$
  and with $\models (\mathtt{x} > 0) \Rightarrow (1 + \mathtt{x} \leq 2 \cdot \mathtt{x})$ and $\models C(0) \Rightarrow \mathtt{true}$, another application of (cons) yields
  $$\vdash \{\mathtt{x} > 0\}\ \mathtt{y:=1;\ while\ \neg(x=1)\ do\ y:=y*x;\ x:=x-1\ end}\ \{\mathtt{x} \Downarrow \mathtt{true}\}$$

## Outline of Lecture 13

Recap: Axiomatic Equivalence

Axiomatic vs. Operational/Denotational Equivalence

Correctness Properties for Execution Time

Operational Semantics with Exact Execution Times

Timed Correctness Properties

Proving Timed Correctness

Soundness and Completeness

Summary: Axiomatic Semantics

## Soundness and Completeness

**Theorem 13.9 (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 tree) □

## Soundness and Completeness

**Theorem 13.9 (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 tree) □

**Theorem 13.10 (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 (using weakest preconditions with execution time) □

# Summary: Axiomatic Semantics

## Outline of Lecture 13

Recap: Axiomatic Equivalence

Axiomatic vs. Operational/Denotational Equivalence

Correctness Properties for Execution Time

Operational Semantics with Exact Execution Times

Timed Correctness Properties

Proving Timed Correctness

Soundness and Completeness

Summary: Axiomatic Semantics

## 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 informal (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, `https://www.cl.cam.ac.uk/archive/mjcg/HL/Notes/Notes.pdf`)