



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/>

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)

Recap: Axiomatic Equivalence

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

$$\models \{A\} c_1 \{B\} \iff \models \{A\} c_2 \{B\}.$$

(later: total correctness yields same notion of equivalence)

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 $X \subseteq \text{Var}$, the **characteristic assertion of σ w.r.t. X** is given by

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

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

Corollary

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

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

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 $c_1, c_2 \in \text{Cmd}$. The following propositions are equivalent:

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

Proof.

on the board □

Axiomatic vs. Operational/Denotational Equivalence

Axiomatic vs. Operational/Denotational Equiv.

Theorem 13.1

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

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

Proof.

on the board □

Correctness Properties for Execution Time

The Approach

- Definition 11.13: proof system for **total correctness**
- Can be used to show that program terminates but 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

Operational Semantics with Exact Execution Times

Recap: Translation of Arithmetic Expressions

Definition (Translation of arithmetic expressions (Definition 5.1))

The translation function

$$\mathfrak{T}_a[\cdot] : AExp \rightarrow Code$$

is given by

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

Operational Semantics with Exact Execution Times

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 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:

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

Rules:

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

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

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

Operational Semantics with Exact Execution Times

Recap: Translation of Boolean Expressions

Definition (Translation of Boolean expressions (Definition 5.3))

The translation function

$$\mathfrak{T}_b[\cdot] : BExp \rightarrow Code$$

is given by

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

Operational Semantics with Exact Execution Times

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:

$$\begin{array}{c}
 \frac{\langle t, \sigma \rangle \xrightarrow{1} t}{\langle a_1, \sigma \rangle \xrightarrow{\tau_1} z \quad \langle a_2, \sigma \rangle \xrightarrow{\tau_2} z} \quad \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 \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 \quad \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}} \quad \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}} \quad \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}} \quad \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 \text{ analogously})
 \end{array}$$

Operational Semantics with Exact Execution Times

Recap: Translation of Statements

Definition (Translation of statements (Definition 5.4))

The translation function $\mathcal{T}_c[\cdot] : \mathit{Cmd} \rightarrow \mathit{Code}$ is given by

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

Operational Semantics with Exact Execution Times

Timed Execution of Statements

Definition 13.4 (Timed execution relation for statements (extends Definition 3.2))

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

$$\begin{array}{c}
 \text{(skip)} \frac{}{\langle \text{skip}, \sigma \rangle \xrightarrow{1} \sigma} \qquad \text{(asgn)} \frac{\langle a, \sigma \rangle \xrightarrow{\tau} z}{\langle x := a, \sigma \rangle \xrightarrow{\tau+1} \sigma[x \mapsto z]} \\
 \text{(seq)} \frac{\langle c_1, \sigma \rangle \xrightarrow{\tau_1} \sigma' \quad \langle c_2, \sigma' \rangle \xrightarrow{\tau_2} \sigma''}{\langle c_1 ; c_2, \sigma \rangle \xrightarrow{\tau_1 + \tau_2} \sigma''} \qquad \text{(if-t)} \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \text{true} \quad \langle c_1, \sigma \rangle \xrightarrow{\tau_1} \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end}, \sigma \rangle \xrightarrow{\tau + \tau_1 + 2} \sigma'} \\
 \text{(wh-f)} \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \text{false}}{\langle \text{while } b \text{ do } c \text{ end}, \sigma \rangle \xrightarrow{\tau+1} \sigma} \qquad \text{(if-f)} \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \text{false} \quad \langle c_2, \sigma \rangle \xrightarrow{\tau_2} \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end}, \sigma \rangle \xrightarrow{\tau + \tau_2 + 1} \sigma'} \\
 \text{(wh-t)} \frac{\langle b, \sigma \rangle \xrightarrow{\tau} \text{true} \quad \langle c, \sigma \rangle \xrightarrow{\tau_1} \sigma' \quad \langle \text{while } b \text{ do } c \text{ end}, \sigma' \rangle \xrightarrow{\tau_2} \sigma''}{\langle \text{while } b \text{ do } c \text{ end}, \sigma \rangle \xrightarrow{\tau + \tau_1 + \tau_2 + 2} \sigma''}
 \end{array}$$

Timed Correctness Properties

Recap: Total Correctness Properties

So far: **total correctness properties** of the form

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

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

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

Timed Correctness Properties

Now: **timed correctness properties** of the form

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

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

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 13.5

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

Timed Correctness Properties

Semantics of Timed Correctness Properties

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

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 $l \in Int$ and each $\sigma \in \Sigma$ with $\sigma \models^l A$, there exist $\sigma' \in \Sigma$ and $\tau \leq k \cdot \mathcal{Q}[[e]]\sigma$ such that

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

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

Proving Timed Correctness

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 $i, u \in LVar$)

$$\begin{array}{c} \text{(skip)} \frac{}{\{A\} \text{ skip } \{1 \Downarrow A\}} \qquad \text{(asgn)} \frac{}{\{A[x \mapsto a]\} x := a \{1 \Downarrow A\}} \\ \text{(seq)} \frac{\{A \wedge e'_2 = u\} c_1 \{e_1 \Downarrow C \wedge e_2 \leq u\} \quad \{C\} c_2 \{e_2 \Downarrow B\}}{\{A\} c_1 ; c_2 \{e_1 + e'_2 \Downarrow B\}} \end{array}$$

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

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

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

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

Proving Timed Correctness III

Example 13.8

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

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

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

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

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

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

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

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

- Using (asgn) for the initial assignment

Soundness and Completeness

Soundness and Completeness

Theorem 13.9 (Soundness)

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

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

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

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

Proof.

omitted (using weakest preconditions with execution time) □

Summary: Axiomatic Semantics

Summary: Axiomatic Semantics

- Formalized by **partial/total correctness properties**
- Inductively defined by **Hoare Logic** proof rules
- Technically involved (especially loop invariants)
⇒ 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>)