Formal Methods and Functional Programming

Axiomatic Semantics

Peter Müller

Chair of Programming Methodology
ETH Zurich

Program Correctness

- Semantics can be used to prove correctness of a program
- Partial correctness expresses that if a program terminates then there will be a certain relationship between the initial and the final state
- Total correctness expresses that a program will terminate and there will be a certain relationship between the initial and the final state
  - The relationship is expressed by a formal specification

$$\boxed{\text{total correctness} = \text{partial correctness} + \text{termination}}$$

3. Axiomatic Semantics

3.1 Hoare Logic

3.1.1 Proofs of Program Correctness
3.1.2 Assertion Language
3.1.3 Inference System
3.1.4 Properties of the Semantics
3.1.5 Extensions

3.2 Soundness and Completeness

Program Correctness: Example

- Consider the factorial statement

```
y := 1;
while not x = 1 do
  y := y * x;
  x := x - 1
end
```

- Specification:
  The final value of `y` is the factorial of the initial value of `x`
- The statement is partially correct
  - It does not terminate for $\mathtt{x} < 1$

Formal Specification

- Specification:
  The final value of $\mathtt{y}$ is the factorial of the initial value of $\mathtt{x}$
- We can express the specification formally based on a formal semantics

$$\langle \mathtt{y{:=}1;while\ not\ x} = 1\ \mathtt{do\ y{:=}y} * \mathtt{x;x{:=}x} - 1\ \mathtt{end}, \sigma\rangle \to \sigma'$$
$$\Rightarrow \sigma'(\mathtt{y}) = \sigma(\mathtt{x})! \wedge \sigma(\mathtt{x}) > 0$$

- This specification expresses partial correctness in natural semantics

Correctness Proof

- We prove partial correctness in three steps
- Step 1: The body of the loop satisfies

$$\langle \texttt{y:=y} * \texttt{x;x:=x} - 1, \sigma \rangle \to \sigma'' \wedge \sigma''(\texttt{x}) > 0 \Rightarrow$$
$$\sigma(\texttt{y}) \times \sigma(\texttt{x})! = \sigma''(\texttt{y}) \times \sigma''(\texttt{x})! \wedge \sigma(\texttt{x}) > 0$$

- Step 2: The loop satisfies

$$\langle \texttt{while not x} = 1 \texttt{ do y:=y} * \texttt{x;x:=x} - 1 \texttt{ end}, \sigma \rangle \to \sigma'' \Rightarrow$$
$$\sigma(\texttt{y}) \times \sigma(\texttt{x})! = \sigma''(\texttt{y}) \wedge \sigma''(\texttt{x}) = 1 \wedge \sigma(\texttt{x}) > 0$$

- Step 3: The whole statement is partially correct

$$\langle \texttt{y:=1;while not x} = 1 \texttt{ do y:=y} * \texttt{x;x:=x} - 1 \texttt{ end}, \sigma \rangle \to \sigma' \Rightarrow$$
$$\sigma'(\texttt{y}) = \sigma(\texttt{x})! \wedge \sigma(\texttt{x}) > 0$$

Proof: Step 1—Loop Body

- Since we have the transition $\langle \texttt{y:=y * x;x:=x} - 1, \sigma\rangle \to \sigma''$, we can assume that there are transitions $\langle \texttt{y:=y * x}, \sigma\rangle \to \sigma'$ and $\langle \texttt{x:=x} - 1, \sigma'\rangle \to \sigma''$
- We get $\sigma' = \sigma[\texttt{y} \mapsto \mathcal{A}[\![\texttt{y * x}]\!]\sigma]$ and $\sigma'' = \sigma'[\texttt{x} \mapsto \mathcal{A}[\![\texttt{x} - 1]\!]\sigma']$, which imply $\sigma'' = \sigma[\texttt{y} \mapsto \sigma(\texttt{y}) \times \sigma(\texttt{x})][\texttt{x} \mapsto \sigma(\texttt{x}) - 1]$
- By $\sigma''(\texttt{x}) > 0$, we calculate

$$
\begin{array}{l}
\sigma''(\texttt{y}) \times \sigma''(\texttt{x})! = \\
\sigma(\texttt{y}) \times \sigma(\texttt{x}) \times (\sigma(\texttt{x}) - 1)! = \sigma(\texttt{y}) \times \sigma(\texttt{x})!
\end{array}
$$

- By $\sigma''(\texttt{x}) = \sigma(\texttt{x}) - 1$, we get $\sigma(\texttt{x}) > 0$

Proof: Step 2—Loop

- Step 2: The loop satisfies

$$\langle \texttt{while not x = 1 do y:=y * x;x:=x - 1 end}, \sigma \rangle \to \sigma'' \Rightarrow$$
$$\sigma(\texttt{y}) \times \sigma(\texttt{x})! = \sigma''(\texttt{y}) \wedge \sigma''(\texttt{x}) = 1 \wedge \sigma(\texttt{x}) > 0$$

- We prove this property by induction on the shape of the derivation tree
- Relevant base case: while-rule for $\mathcal{B}[\![\texttt{not x = 1}]\!]\sigma = \mathit{ff}$
  - We have $\sigma(\texttt{x}) = 1$ and $\sigma = \sigma''$
  - Since $1 = 1!$, we get $\sigma(\texttt{y}) \times \sigma(\texttt{x})! = \sigma(\texttt{y}) = \sigma''(\texttt{y})$
  - We trivially get $\sigma''(\texttt{x}) = 1$ and $\sigma(\texttt{x}) > 0$

Proof: Step 2—Loop (Case 2)

- Relevant inductive case: while-rule for $\mathcal{B}[\![\texttt{not x} = 1]\!]\sigma = tt$
- From the rule of the natural semantics we get for some σ'''

  (1) $\langle \texttt{y:=y} * \texttt{x;x:=x} - 1, \sigma\rangle \to \sigma'''$

  (2) $\langle \texttt{while not x} = 1 \texttt{ do y:=y} * \texttt{x;x:=x} - 1 \texttt{ end}, \sigma'''\rangle \to \sigma''$

- Applying the induction hypothesis to (2) yields
  $\sigma'''(\texttt{y}) \times \sigma'''(\texttt{x})! = \sigma''(\texttt{y}) \wedge \sigma''(\texttt{x}) = 1 \wedge \sigma'''(\texttt{x}) > 0$
- By (1), $\sigma'''(\texttt{x}) > 0$, and Proof Step 1, we get
  $\sigma(\texttt{y}) \times \sigma(\texttt{x})! = \sigma'''(\texttt{y}) \times \sigma'''(\texttt{x})! \wedge \sigma(\texttt{x}) > 0$
- Combining these results yields
  $\sigma(\texttt{y}) \times \sigma(\texttt{x})! = \sigma''(\texttt{y}) \wedge \sigma''(\texttt{x}) = 1 \wedge \sigma(\texttt{x}) > 0$

Proof: Step 3—Factorial Statement

- Step 3: The whole statement is partially correct

$$\langle \texttt{y:=1;while not x = 1 do y:=y * x;x:=x - 1 end}, \sigma\rangle \to \sigma' \Rightarrow$$
$$\sigma'(\texttt{y}) = \sigma(\texttt{x})! \wedge \sigma(\texttt{x}) > 0$$

- From the natural semantics we get for some σ''

  (1) $\langle \texttt{y:=1}, \sigma\rangle \to \sigma''$

  (2) $\langle \texttt{while not x = 1 do y:=y * x;x:=x - 1 end}, \sigma''\rangle \to \sigma'$

- By (1), we get $\sigma'' = \sigma[\texttt{y} \mapsto 1]$ and, thus, $\sigma''(\texttt{x}) = \sigma(\texttt{x})$
- By (2), and Proof Step 2, we get
$\sigma''(\texttt{y}) \times \sigma''(\texttt{x})! = \sigma'(\texttt{y}) \wedge \sigma'(\texttt{x}) = 1 \wedge \sigma''(\texttt{x}) > 0$
- We conclude $1 \times \sigma(\texttt{x})! = \sigma'(\texttt{y}) \wedge \sigma(\texttt{x}) > 0$

Verification Example: Observations

- We can prove correctness of a program based on a formal semantics
  - The proof would also be possible with SOS and denotational semantics, but even more complicated
- Proofs are too detailed to be practical
  - We have to consider how whole states are modified
  - We would like to focus on certain properties of states
- Axiomatic Semantics describes essential properties of syntactic constructs
  - The choice of essential properties depends on what we want to prove

3. Axiomatic Semantics

3.1 Hoare Logic

3.1.1 Proofs of Program Correctness
3.1.2 Assertion Language
3.1.3 Inference System
3.1.4 Properties of the Semantics
3.1.5 Extensions

3.2 Soundness and Completeness

Assertions

- Properties of programs are specified as assertions

$$\boxed{\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}}$$

  where s is a statement and **P** and **Q** are predicates

- Terminology
  - Assertions are also called (Hoare) triples
  - **P** is called precondition
  - **Q** is called postcondition

Meaning of Assertions

- The meaning of $\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$ is

  If **P** holds in the initial state σ, and
  if the execution of s from σ terminates in a state σ'
  then **Q** will hold in σ'

- This meaning describes partial correctness, that is, termination is not an essential property

- It is also possible to assign different meanings to assertions

Assertions: Example

- Specification of the factorial statement by an assertion

$$\{\ \mathit{true}\ \}\quad \texttt{y:=1;while not x} = 1\ \texttt{do y:=y} * \texttt{x;x:=x} - 1\ \texttt{end}\quad \{\ \texttt{y} = \texttt{x}! \land \texttt{x} > 0\ \}$$

- In general, this assertion does not hold
  - Consider an initial state $\{\ \texttt{x} \mapsto 2, \texttt{y} \mapsto 0\ \}$
  - The final state will be $\{\ \texttt{x} \mapsto 1, \texttt{y} \mapsto 2\ \}$
- We have to express that `y` in the final state is the factorial of `x` in the initial state

Logical Variables

- Assertions can contain logical variables
  - Logical variables may occur only in pre- and postconditions
  - Logical variables are not program variables and may, thus, not be accessed in programs

- Logical variables can be used to save values of the initial state for the final state

$$\{\ \mathtt{x} = N\ \}$$
$$\mathtt{y{:=}1;while\ not\ x} = 1\ \mathtt{do\ y{:=}y} * \mathtt{x;x{:=}x} - 1\ \mathtt{end}$$
$$\{\ \mathtt{y} = N! \wedge N > 0\ \}$$

- We assume states to map logical variables to their values

Assertion Language

- Pre- and postconditions are predicates, that is functions State $\rightarrow$ Bool
- Each boolean expression b defines a predicate $\mathcal{B}[\![b]\!]$
- If P, P_1, and P_2 are predicates, then we use the following notation for predicates

$$
\begin{array}{ll}
P_1 \wedge P_2 & \text{where } (P_1 \wedge P_2)(\sigma) \;\Leftrightarrow\; P_1(\sigma) \wedge P_2(\sigma) \\
P_1 \vee P_2 & \text{where } (P_1 \vee P_2)(\sigma) \;\Leftrightarrow\; P_1(\sigma) \vee P_2(\sigma) \\
\neg P & \text{where } (\neg P)(\sigma) \;\Leftrightarrow\; \neg P(\sigma) \\
P[x \mapsto e] & \text{where } (P[x \mapsto e])(\sigma) \;\Leftrightarrow\; P(\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]) \\
P_1 \Rightarrow P_2 & \text{where } (P_1 \Rightarrow P_2)(\sigma) \;\Leftrightarrow\; P_1(\sigma) \Rightarrow P_2(\sigma)
\end{array}
$$

3. Axiomatic Semantics

3.1 Hoare Logic

3.1.1 Proofs of Program Correctness
3.1.2 Assertion Language
3.1.3 Inference System
3.1.4 Properties of the Semantics
3.1.5 Extensions

3.2 Soundness and Completeness

Inference System

- We formalize the semantics of a programming language by describing the valid assertions
- This is done by an inference system
  - An inference system consists of a set of axioms and rules
  - The formulas of the inference system are assertions

$$\boxed{\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}}$$

- The inference system specifies an axiomatic semantics of the programming language

Axiomatic Semantics of IMP

- `skip` does not modify the state

$$\{\ \mathbf{P}\ \}\ \texttt{skip}\ \{\ \mathbf{P}\ \}$$

- x:=e assigns the value of e to variable x

$$\{\ \mathbf{P}[x \mapsto e]\ \}\ x\texttt{:=}e\ \{\ \mathbf{P}\ \}$$

  - Let σ be the initial state
  - Precondition: $(\mathbf{P}[x \mapsto e])(\sigma)$, i.e., $\mathbf{P}(\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma])$
  - Final state: $\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$
  - Consequently, $\mathbf{P}$ holds in the final state

- The rules are axiom schemes

Axiomatic Semantics of IMP (cont'd)

- Sequential composition $s_1 ; s_2$

$$\frac{\{\ \mathbf{P}\ \}\ s_1\ \{\ \mathbf{Q}\ \} \qquad \{\ \mathbf{Q}\ \}\ s_2\ \{\ \mathbf{R}\ \}}{\{\ \mathbf{P}\ \}\ s_1 ; s_2\ \{\ \mathbf{R}\ \}}$$

- Conditional statement `if` b `then` s_1 `else` s_2 `end`

$$\frac{\{\ \mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}\ s_1\ \{\ \mathbf{Q}\ \} \qquad \{\ \neg\mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}\ s_2\ \{\ \mathbf{Q}\ \}}{\{\ \mathbf{P}\ \}\ \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{end}\ \{\ \mathbf{Q}\ \}}$$

Axiomatic Semantics of IMP (cont'd)

- Loop statement `while` b `do` s `end`

$$\frac{\{\ \mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}\ s\ \{\ \mathbf{P}\ \}}{\{\ \mathbf{P}\ \}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\ \neg\mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}}$$

  - $\mathbf{P}$ is the loop invariant

- Rule of consequence

$$\frac{\{\ \mathbf{P}'\ \}\ s\ \{\ \mathbf{Q}'\ \}}{\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}}\ \text{if } \mathbf{P} \Rightarrow \mathbf{P}' \text{ and } \mathbf{Q}' \Rightarrow \mathbf{Q}$$

  - We can strengthen preconditions
  - We can weaken postconditions
  - Here, $\mathbf{P} \Rightarrow \mathbf{Q}$ should be read as:
    "We can prove for all states σ, that $\mathbf{P}(\sigma)$ implies $\mathbf{Q}(\sigma)$"

Inference Trees

- Axioms and rules are used like in natural semantics or natural deduction
- Derivation trees are called inference trees since they show how to infer an assertion
  - The leaves are instances of axiom schemes
  - The internal nodes correspond to instances of rules
- A finite inference tree gives a proof of the assertion at its root
- To express that an assertion $\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$ can be inferred, we write

$$\boxed{\vdash \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}}$$

Inference Trees: Example 1

- Prove that the following statement swaps the values in the variables `x` and `y`

$$\boxed{\texttt{z:=x; x:=y; y:=z}}$$

- We can build the following inference tree

$$\dfrac{\dfrac{\dfrac{\{\ \mathbf{P}\ \}\ \texttt{z:=x}\ \{\ \texttt{z} = X_0 \land \texttt{y} = Y_0\ \}}{\{\ \mathbf{P}\ \}\ \texttt{z:=x}\ \{\ \texttt{y} = Y_0 \land \texttt{z} = X_0\ \}} \qquad \{\ \texttt{y} = Y_0 \land \texttt{z} = X_0\ \}\ \texttt{x:=y}\ \{\ \texttt{x} = Y_0 \land \texttt{z} = X_0\ \}}{\{\ \mathbf{P}\ \}\ \texttt{z:=x; x:=y}\ \{\ \texttt{x} = Y_0 \land \texttt{z} = X_0\ \}} \qquad \{\ \texttt{x} = Y_0 \land \texttt{z} = X_0\ \}\ \texttt{y:=z}\ \{\ \mathbf{Q}\ \}}{\{\ \mathbf{P}\ \}\ \texttt{z:=x; x:=y; y:=z}\ \{\ \mathbf{Q}\ \}}$$

  where we write **P** for $\texttt{x} = X_0 \land \texttt{y} = Y_0$ and **Q** for $\texttt{x} = Y_0 \land \texttt{y} = X_0$

- We often omit the application of $\mathcal{B}$
  - We write $\texttt{x} = X_0$ to abbreviate $\mathcal{B}[\![\texttt{x} = X_0]\!]$

Inference Trees: Example 2

- Consider the non-terminating loop

$$\boxed{\texttt{while true do skip end}}$$

- We can build the following inference tree

$$\cfrac{\cfrac{\cfrac{\{\,\mathit{true}\,\}\ \texttt{skip}\ \{\,\mathit{true}\,\}}{\{\,\mathit{true} \wedge \mathit{true}\,\}\ \texttt{skip}\ \{\,\mathit{true}\,\}}}{\{\,\mathit{true}\,\}\ \texttt{while true do skip end}\ \{\,\neg\mathit{true} \wedge \mathit{true}\,\}}}{\{\,\mathit{true}\,\}\ \texttt{while true do skip end}\ \{\,\mathit{false}\,\}}$$

where we write *true* for $\mathcal{B}[\![\texttt{true}]\!]$ and *false* for $\mathcal{B}[\![\texttt{not true}]\!]$

- This proof illustrates that we have partial correctness

Proof Outlines

- Inference trees tend to get very large and are, thus, inconvenient to write
  - Most statements are written many times
  - Many assertions are written many times
- An alternative is to group the assertions around the program text
- We write assertions before and after each statement to indicate which properties hold in the states before and after the execution of this statement

Proof Outlines: Notation

- We write instances of axioms as:

$$\{\ \mathbf{P}\ \}\quad \texttt{skip}\quad \{\ \mathbf{P}\ \}$$

$$\{\ \mathbf{P}[x \mapsto e]\ \}\quad x\texttt{:=}e\quad \{\ \mathbf{P}\ \}$$

- We write an instance of the rule for sequential composition as:

$$\{\ \mathbf{P}\ \}\quad s_1\,;\quad \{\ \mathbf{Q}\ \}\quad s_2\quad \{\ \mathbf{R}\ \}$$

  - This expresses $\vdash \{\ \mathbf{P}\ \}\ s_1\ \{\ \mathbf{Q}\ \}$, $\vdash \{\ \mathbf{Q}\ \}\ s_2\ \{\ \mathbf{R}\ \}$, and $\vdash \{\ \mathbf{P}\ \}\ s_1\,;s_2\ \{\ \mathbf{R}\ \}$
  - We write each statement and the intermediate assertion **Q** only once

Proof Outlines: Notation (cont'd)

- We write an instance of the rule for conditional statements as:

$$
\begin{array}{l}
\{\ \mathbf{P}\ \} \\
\quad \texttt{if}\ b\ \texttt{then} \\
\quad\quad \{\ \mathcal{B}[\![b]\!] \land \mathbf{P}\ \} \\
\quad\quad\quad s_1 \\
\quad\quad \{\ \mathbf{Q}\ \} \\
\quad \texttt{else} \\
\quad\quad \{\ \neg\mathcal{B}[\![b]\!] \land \mathbf{P}\ \} \\
\quad\quad\quad s_2 \\
\quad\quad \{\ \mathbf{Q}\ \} \\
\quad \texttt{end} \\
\{\ \mathbf{Q}\ \}
\end{array}
$$

- We write an instance of the rule for loops as:

$$
\begin{array}{l}
\{\ \mathbf{P}\ \} \\
\quad \texttt{while}\ b\ \texttt{do} \\
\quad\quad \{\ \mathcal{B}[\![b]\!] \land \mathbf{P}\ \} \\
\quad\quad\quad s \\
\quad\quad \{\ \mathbf{P}\ \} \\
\quad \texttt{end} \\
\{\ \neg\mathcal{B}[\![b]\!] \land \mathbf{P}\ \}
\end{array}
$$

Proof Outlines: Notation (cont'd)

- We write an instance of the rule of consequence as:

$$\begin{array}{l}\{\ \mathbf{P}\ \}\\ \Rightarrow\\ \{\ \mathbf{P'}\ \}\\ \quad s\\ \{\ \mathbf{Q'}\ \}\\ \Rightarrow\\ \{\ \mathbf{Q}\ \}\end{array}$$

  - We omit the implication when $\mathbf{P}$ and $\mathbf{P'}$ or $\mathbf{Q}$ and $\mathbf{Q'}$ are syntactically identical

$$\begin{array}{l}\{\ \mathbf{P}\ \}\\ \quad s\\ \{\ \mathbf{Q'}\ \}\\ \Rightarrow\\ \{\ \mathbf{Q}\ \}\end{array} \qquad \begin{array}{l}\{\ \mathbf{P}\ \}\\ \Rightarrow\\ \{\ \mathbf{P'}\ \}\\ \quad s\\ \{\ \mathbf{Q}\ \}\end{array}$$

Proof Outlines: Example

- Back to our swap-example:

```
z:=x; x:=y; y:=z
```

- Proof outline:

$$\{ \mathtt{x} = X_0 \wedge \mathtt{y} = Y_0 \}$$
$$\Rightarrow$$
$$\{ \mathtt{y} = Y_0 \wedge \mathtt{x} = X_0 \}$$
$$\boxed{\mathtt{z := x;}}$$
$$\{ \mathtt{y} = Y_0 \wedge \mathtt{z} = X_0 \}$$
$$\boxed{\mathtt{x := y;}}$$
$$\{ \mathtt{x} = Y_0 \wedge \mathtt{z} = X_0 \}$$
$$\boxed{\mathtt{y := z}}$$
$$\{ \mathtt{x} = Y_0 \wedge \mathtt{y} = X_0 \}$$

- Proof outlines are typically developed bottom-up

Verification of Factorial Statement

$\{ \mathtt{x} = N \}$
$\quad \mathtt{y\!:=\!1;while\ not\ x} = 1\ \mathtt{do\ y\!:=\!y} * \mathtt{x;x\!:=\!x} - 1\ \mathtt{end}$
$\{ \mathtt{y} = N! \wedge N > 0 \}$

Verification of Factorial Statement

$\{ \mathtt{x} = N \}$
  `y:=1;while not x = 1 do y:=y * x;x:=x - 1 end`
$\{ \mathtt{y} = N! \wedge N > 0 \}$

- Determining the loop invariant

Iteration	0	1	2	i	$N-1$
x	N	$N-1$	$N-2$	$N-i$	1
y	1	N	$N \times (N-1)$	$N \times (N-1) \times \ldots \times (N-i+1)$	$N!$

Verification of Factorial Statement

$$\{\ \mathtt{x} = N\ \}$$
$$\mathtt{y{:=}1;while\ not\ x} = 1\ \mathtt{do\ y{:=}y} * \mathtt{x;x{:=}x} - 1\ \mathtt{end}$$
$$\{\ \mathtt{y} = N! \land N > 0\ \}$$

- Determining the loop invariant

Iteration	0	1	2	i	$N-1$
$\mathtt{x}$	N	$N-1$	$N-2$	$N-i$	1
$\mathtt{y}$	1	N	$N\times(N-1)$	$N\times(N-1)\times\ldots\times(N-i+1)$	$N!$

- Invariant: $\mathtt{x} > 0 \Rightarrow \mathtt{y} \times \mathtt{x}! = N! \land N \geq \mathtt{x}$

Proof Outline for Factorial Statement

$\{ \mathtt{x} = N \}$

$\Rightarrow$

$\{ \mathtt{x} > 0 \Rightarrow 1 \times \mathtt{x}! = N! \land N \geq \mathtt{x} \}$

$\boxed{\mathtt{y\ :=\ 1;}}$

$\{ \mathtt{x} > 0 \Rightarrow \mathtt{y} \times \mathtt{x}! = N! \land N \geq \mathtt{x} \}$

$\boxed{\mathtt{while\ not\ x\ =\ 1\ do}}$

$\{ \mathtt{x} \neq 1 \land (\mathtt{x} > 0 \Rightarrow \mathtt{y} \times \mathtt{x}! = N! \land N \geq \mathtt{x}) \}$

$\Rightarrow$

$\{ \mathtt{x-1} > 0 \Rightarrow \mathtt{y*x} \times \mathtt{x-1}! = N! \land N \geq \mathtt{x-1} \}$

$\boxed{\mathtt{y\ :=\ y*x;}}$

$\{ \mathtt{x-1} > 0 \Rightarrow \mathtt{y} \times \mathtt{x-1}! = N! \land N \geq \mathtt{x-1} \}$

$\boxed{\mathtt{x\ :=\ x-1}}$

$\{ \mathtt{x} > 0 \Rightarrow \mathtt{y} \times \mathtt{x}! = N! \land N \geq \mathtt{x} \}$

$\boxed{\mathtt{end}}$

$\{ \mathtt{x} = 1 \land (\mathtt{x} > 0 \Rightarrow \mathtt{y} \times \mathtt{x}! = N! \land N \geq \mathtt{x}) \}$

$\Rightarrow$

$\{ \mathtt{y} = N! \land N > 0 \}$

Proof Outline for Zune Example

$\{\ 0 < D\ \}$
$\Rightarrow$
$\{\ 1980 - 1980 \leq (D - D)/365 \land 0 < D\ \}$
`year := 1980;`
$\{\ \mathtt{year} - 1980 \leq (D - D)/365 \land 0 < D\ \}$
`days := D;`
$\{\ \mathtt{year} - 1980 \leq (D - \mathtt{days})/365 \land 0 < \mathtt{days}\ \}$
`while L(year) and 366 < days or not L(year) and 365 < days do`
$\{\ (L(\mathtt{year}) \land 366 < \mathtt{days} \lor \neg L(\mathtt{year}) \land 365 < \mathtt{days}) \land \mathtt{year} - 1980 \leq (D - \mathtt{days})/365 \land 0 < \mathtt{days}\ \}$
$\Rightarrow$
$\{\ \mathtt{year} - 1980 \leq (D - \mathtt{days})/365 \land (L(\mathtt{year}) \Rightarrow 366 < \mathtt{days}) \land 365 < \mathtt{days}\ \}$
`if L(year) then`
$\{\ L(\mathtt{year}) \land \mathtt{year} - 1980 \leq (D - \mathtt{days})/365 \land (L(\mathtt{year}) \Rightarrow 366 < \mathtt{days}) \land 365 < \mathtt{days}\ \}$
$\Rightarrow$
$\{\ \mathtt{year} + 1 - 1980 \leq (D - (\mathtt{days} - 366))/365 \land 0 < (\mathtt{days} - 366)\ \}$
`days := days - 366`
$\{\ \mathtt{year} + 1 - 1980 \leq (D - \mathtt{days})/365 \land 0 < \mathtt{days}\ \}$
`else`
$\{\ \neg L(\mathtt{year}) \land \mathtt{year} - 1980 \leq (D - \mathtt{days})/365 \land (L(\mathtt{year}) \Rightarrow 366 < \mathtt{days}) \land 365 < \mathtt{days}\ \}$
$\Rightarrow$
$\{\ \mathtt{year} + 1 - 1980 \leq (D - (\mathtt{days} - 365))/365 \land 0 < \mathtt{days} - 365\ \}$
`days := days - 365;`
$\{\ \mathtt{year} + 1 - 1980 \leq (D - \mathtt{days})/365 \land 0 < \mathtt{days}\ \}$
`end;`
$\{\ \mathtt{year} + 1 - 1980 \leq (D - \mathtt{days})/365 \land 0 < \mathtt{days}\ \}$
`year := year + 1`
$\{\ \mathtt{year} - 1980 \leq (D - \mathtt{days})/365 \land 0 < \mathtt{days}\ \}$
`end`
$\{\ \neg(L(\mathtt{year}) \land 366 < \mathtt{days} \lor \neg L(\mathtt{year}) \land 365 < \mathtt{days}) \land \mathtt{year} - 1980 \leq (D - \mathtt{days})/365 \land 0 < \mathtt{days}\ \}$
$\Rightarrow$
$\{\ \mathtt{year} - 1980 \leq D/365\ \}$

3. Axiomatic Semantics

3.1 Hoare Logic

- 3.1.1 Proofs of Program Correctness
- 3.1.2 Assertion Language
- 3.1.3 Inference System
- 3.1.4 Properties of the Semantics
- 3.1.5 Extensions

3.2 Soundness and Completeness

Induction on the Shape of Inference Trees

- Properties of the axiomatic semantics are typically proved by induction on the shape of the inference tree
  - Analogous to induction of the shape of derivation trees in natural semantics
  - Note: structural induction on the shape of the statement does not work because of the rule of consequence

1. Induction base: Prove that the property holds for all the simple inference trees by showing that it holds for the axioms of the inference system

2. Induction step: Prove that the property holds for all composite inference trees:
   - Induction hypothesis: For each rule, assume that the property holds for its premises
   - Prove that it also holds for the conclusion, provided that the conditions of the rule are satisfied

Proving Properties: Example

- We prove the lemma

$$\text{If} \vdash \{\ \mathbf{P}\ \}\ \texttt{skip}\ \{\ \mathbf{Q}\ \}\ \text{then}\ \mathbf{P} \Rightarrow \mathbf{Q}$$

  - If there exists an inference tree for $\{\ \mathbf{P}\ \}\ \texttt{skip}\ \{\ \mathbf{Q}\ \}$ then $\mathbf{P} \Rightarrow \mathbf{Q}$
  - We do induction on the shape of the inference tree for $\{\ \mathbf{P}\ \}\ \texttt{skip}\ \{\ \mathbf{Q}\ \}$

Proving Properties: Example (cont'd)

- Induction base:
  The inference tree for $\{\ \mathbf{P}\ \}\ \texttt{skip}\ \{\ \mathbf{Q}\ \}$ is an axiom instance
  - The only axiom that can form this tree is the `skip` axiom
  - We get $\mathbf{P} = \mathbf{Q}$ and, thus, $\mathbf{P} \Rightarrow \mathbf{Q}$
- Induction step:
  The inference tree for $\{\ \mathbf{P}\ \}\ \texttt{skip}\ \{\ \mathbf{Q}\ \}$ is a composite tree
  - We consider all rules that could be used at the root of the inference tree
  - The only applicable rule is the rule of consequence, because no other rule applies to `skip`
  - From the rule of consequence, we know that there exists an inference tree for $\{\ \mathbf{P}'\ \}\ \texttt{skip}\ \{\ \mathbf{Q}'\ \}$, where $\mathbf{P} \Rightarrow \mathbf{P}'$ and $\mathbf{Q}' \Rightarrow \mathbf{Q}$
  - By applying the induction hypothesis to $\{\ \mathbf{P}'\ \}\ \texttt{skip}\ \{\ \mathbf{Q}'\ \}$, we get $\mathbf{P}' \Rightarrow \mathbf{Q}'$
  - Now we have $\mathbf{P} \Rightarrow \mathbf{P}'$, $\mathbf{P}' \Rightarrow \mathbf{Q}'$, and $\mathbf{Q}' \Rightarrow \mathbf{Q}$ and, thus, $\mathbf{P} \Rightarrow \mathbf{Q}$

Semantic Equivalence

Two statements s_1 and s_2 are provably equivalent if for all preconditions **P** and postconditions **Q** we have

$$\vdash \{ \mathbf{P} \}\ s_1\ \{ \mathbf{Q} \} \text{if and only if} \vdash \{ \mathbf{P} \}\ s_2\ \{ \mathbf{Q} \}$$

- Example: s and s;`skip` are equivalent
- Proof for "⇒"
  - We know there is an inference tree for $\{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}$
  - We extend that tree using the `skip`-axiom and the rule for sequential composition:

$$\frac{\{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \} \quad \{ \mathbf{Q} \}\ \texttt{skip}\ \{ \mathbf{Q} \}}{\{ \mathbf{P} \}\ s;\texttt{skip}\ \{ \mathbf{Q} \}}$$

Semantic Equivalence: Proof for "$\Leftarrow$"

- The proof runs by induction on the shape of the inference tree for $\{\ \mathbf{P}\ \}\ s;\mathtt{skip}\ \{\ \mathbf{Q}\ \}$
- Induction base:
  The inference tree for $\{\ \mathbf{P}\ \}\ s;\mathtt{skip}\ \{\ \mathbf{Q}\ \}$ is an axiom instance
  - There is no axiom that can form an inference tree for a sequential composition
  - Therefore, the property holds trivially for all base cases
- Induction step:
  The inference tree for $\{\ \mathbf{P}\ \}\ s;\mathtt{skip}\ \{\ \mathbf{Q}\ \}$ is a composite tree
  - We consider all rules that could be used at the root of the inference tree
  - There are two applicable rules: the rule for sequential composition and the rule of consequence
  - We continue by case distinction

Semantic Equivalence: Proof for "$\Leftarrow$" (cont'd)

- Case sequential composition rule
  - We know there are inference trees for $\{\, \mathbf{P} \,\}\ s\ \{\, \mathbf{R} \,\}$ and $\{\, \mathbf{R} \,\}\ \mathtt{skip}\ \{\, \mathbf{Q} \,\}$ for some predicate $\mathbf{R}$
  - Applying the auxiliary lemma to $\{\, \mathbf{R} \,\}\ \mathtt{skip}\ \{\, \mathbf{Q} \,\}$ yields $\mathbf{R} \Rightarrow \mathbf{Q}$
  - We extend the inference tree for $\{\, \mathbf{P} \,\}\ s\ \{\, \mathbf{R} \,\}$ using the rule of consequence to obtain $\{\, \mathbf{P} \,\}\ s\ \{\, \mathbf{Q} \,\}$

- Case rule of consequence
  - We know that there exists an inference tree for $\{\, \mathbf{P}' \,\}\ s\mathtt{;skip}\ \{\, \mathbf{Q}' \,\}$ where $\mathbf{P} \Rightarrow \mathbf{P}'$ and $\mathbf{Q}' \Rightarrow \mathbf{Q}$
  - By applying the induction hypothesis to $\{\, \mathbf{P}' \,\}\ s\mathtt{;skip}\ \{\, \mathbf{Q}' \,\}$, we know there is an inference tree for $\{\, \mathbf{P}' \,\}\ s\ \{\, \mathbf{Q}' \,\}$
  - We extend the tree for $\{\, \mathbf{P}' \,\}\ s\ \{\, \mathbf{Q}' \,\}$ using the rule of consequence to obtain a tree for $\{\, \mathbf{P} \,\}\ s\ \{\, \mathbf{Q} \,\}$

3. Axiomatic Semantics

3.1 Hoare Logic

3.1.1 Proofs of Program Correctness
3.1.2 Assertion Language
3.1.3 Inference System
3.1.4 Properties of the Semantics
3.1.5 Extensions

3.2 Soundness and Completeness

Total Correctness

- The meaning of $\{ \mathbf{P} \}\ s\ \{ \Downarrow \mathbf{Q} \}$ is

  If **P** holds in the initial state σ
  then the execution of s from σ terminates
  and **Q** will hold in the final state

- This meaning describes total correctness, that is, termination is required
- All rules except the rule for loops are analogous

Loop Variants

- Termination is proved using loop variants
- A loop variant is a function from a state to a well-founded set, for instance, $\mathbb{N}$
- Each iteration decreases the value of the loop variant
- The loop has to terminate when a minimal value of the well-founded set is reached
- Example

```
x := 5;
while x # 0 do x := x - 1 end
```

  - Possible loop variant $v : \text{State} \to \mathbb{N}$ where $v(\sigma) = \sigma(\mathtt{x})$

While Rule for Total Correctness

- For simplicity, we consider loop variants that map states to $\mathbb{N}$
  - We use arithmetic expressions of IMP to express loop variants
  - The expression e denotes the loop variant $\mathcal{A}[\![e]\!]$; we prove explicitly that $\mathcal{A}[\![e]\!] \in \mathbb{N}$ before each iteration
  - Intuition: loop variant gives an upper bound on the number of iterations
- Rule:

$$\frac{\{\, \mathcal{B}[\![b]\!] \wedge \mathbf{P} \wedge \mathcal{A}[\![e]\!] = Z \,\}\ s\ \{\, \Downarrow \mathbf{P} \wedge \mathcal{A}[\![e]\!] < Z \,\}}{\{\, \mathbf{P} \,\}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\, \Downarrow \neg\mathcal{B}[\![b]\!] \wedge \mathbf{P} \,\}}$$

$$\text{Condition: } \mathbf{P} \wedge \mathcal{B}[\![b]\!] \Rightarrow 0 \leq \mathcal{A}[\![e]\!]$$

  where:
  - e is an arithmetic expression
  - Z is a logical variable
- Other well-founded sets are also possible and useful

Total Correctness of Factorial

$\{ \mathtt{x} = N \ \wedge \mathtt{x} > 0 \}$
`y:=1;while not x = 1 do y:=y * x;x:=x - 1 end`
$\{ \Downarrow \mathtt{y} = N! \}$

- Invariant: $\mathbf{P} \equiv \mathtt{x} > 0 \wedge \mathtt{y} \times \mathtt{x}! = N!$
- Variant: $\mathtt{x}$
- Side condition: $\mathtt{x} > 0 \wedge \mathtt{y} \times \mathtt{x}! = N! \wedge \mathtt{x} \neq 1 \Rightarrow 0 \leq \mathtt{x}$

Proof Outline for Factorial Statement

$\{ \ x = N \wedge x > 0 \ \}$

$\Rightarrow$

$\{ \ x > 0 \wedge 1 \times x! = N! \ \}$

$\boxed{\texttt{y := 1;}}$

$\{ \ x > 0 \wedge y \times x! = N! \ \}$

$\boxed{\texttt{while not x = 1 do}}$

$\{ \ x \neq 1 \wedge x > 0 \wedge y \times x! = N! \wedge x = Z \ \}$

$\Rightarrow$

$\{ \ x - 1 > 0 \wedge (y \times x) \times (x - 1)! = N! \wedge x - 1 < Z \ \}$

$\boxed{\texttt{y := y*x;}}$

$\{ \ x - 1 > 0 \wedge y \times (x - 1)! = N! \wedge x - 1 < Z \ \}$

$\boxed{\texttt{x := x-1}}$

$\{ \Downarrow x > 0 \wedge y \times x! = N! \wedge x < Z \ \}$

$\boxed{\texttt{end}}$

$\{ \Downarrow x = 1 \wedge x > 0 \wedge y \times x! = N! \ \}$

$\Rightarrow$

$\{ \Downarrow y = N! \ \}$

Zune Bug Revisited

```
//---------------------------
// Split total days since
// Jan. 01, ORIGINYEAR
// into year, month and day
//---------------------------
BOOL ConvertDays(UINT32 days, ...) {
  int year = ORIGINYEAR; /* =1980 */

  while (365 < days) {
    if (IsLeapYear(year)) {
      if (366 < days) {
        days -= 366; year += 1;
      }
    } else {
      days -= 365; year += 1;
    }
  }
  ... }
```

- Invariant: $\mathbf{P} \equiv \mathit{true}$
- Variant: `days`
- Side condition: $\mathit{true} \wedge 365 < \texttt{days} \Rightarrow 0 \leq \texttt{days}$

(Failing) Proof Attempt for Zune Bug

$\{ \ \mathit{true} \ \}$

$\boxed{\texttt{while 365 < days do}}$

$\{ \ 365 < \texttt{days} \land \texttt{days} = Z \ \}$

$\boxed{\texttt{if } L(\texttt{year}) \texttt{ then}}$

$\{ \ L(\texttt{year}) \land 365 < \texttt{days} \land \texttt{days} = Z \ \}$

$\boxed{\texttt{if 366 < days then}}$

$\{ \ 366 < \texttt{days} \land L(\texttt{year}) \land 365 < \texttt{days} \land \texttt{days} = Z \ \}$

$\Rightarrow$

$\{ \ \texttt{days} - 366 < Z \ \}$

$\boxed{\texttt{days := days - 366;year := year + 1}}$

$\{ \Downarrow \texttt{days} < Z \ \}$

$\boxed{\texttt{else}}$

$\{ \ \neg(366 < \texttt{days}) \land L(\texttt{year}) \land 365 < \texttt{days} \land \texttt{days} = Z \ \}$

$\Rightarrow$

$\{ \ \texttt{days} < Z \ \}$

$\boxed{\texttt{skip}}$

$\{ \Downarrow \texttt{days} < Z \ \}$

$\boxed{\texttt{end}}$

$\{ \Downarrow \texttt{days} < Z \ \}$

$\boxed{\texttt{else}}$

$\{ \ \neg L(\texttt{year}) \land 365 < \texttt{days} \land \texttt{days} = Z \ \}$

$\Rightarrow$

$\{ \ \texttt{days} - 365 < Z \ \}$

$\boxed{\texttt{days := days - 365; year := year + 1}}$

$\{ \Downarrow \texttt{days} < Z \ \}$

$\boxed{\texttt{end}}$

$\{ \Downarrow \texttt{days} < Z \ \}$

$\boxed{\texttt{end}}$

$\{ \Downarrow \neg(365 < \texttt{days}) \ \}$

$\Rightarrow$

$\{ \Downarrow \mathit{true} \ \}$

Termination Proof for Corrected Zune Example

$\{$ *true* $\}$

$\boxed{\texttt{days := D;}}$

$\{$ *true* $\}$

$\boxed{\texttt{while } L(\texttt{year}) \texttt{ and 366 < days or not } L(\texttt{year}) \texttt{ and 365 < days do}}$

$\{ (L(\texttt{year}) \wedge 366 < \texttt{days} \vee \neg L(\texttt{year}) \wedge 365 < \texttt{days}) \wedge \texttt{days} = Z \}$

$\Rightarrow$

$\{ \texttt{days} = Z \}$

$\boxed{\texttt{if } L(\texttt{year}) \texttt{ then}}$

$\{ L(\texttt{year}) \wedge \texttt{days} = Z \}$

$\Rightarrow$

$\{ \texttt{days} - 366 < Z \}$

$\boxed{\texttt{days := days - 366}}$

$\{ \Downarrow \texttt{days} < Z \}$

$\boxed{\texttt{else}}$

$\{ \neg L(\texttt{year}) \wedge \texttt{days} = Z \}$

$\Rightarrow$

$\{ \texttt{days} - 365 < Z \}$

$\boxed{\texttt{days := days - 365;}}$

$\{ \Downarrow \texttt{days} < Z \}$

$\boxed{\texttt{end;}}$

$\{ \Downarrow \texttt{days} < Z \}$

$\boxed{\texttt{year := year + 1}}$

$\{ \Downarrow \texttt{days} < Z \}$

$\boxed{\texttt{end}}$

$\{ \Downarrow \neg(L(\texttt{year}) \wedge 366 < \texttt{days} \vee \neg L(\texttt{year}) \wedge 365 < \texttt{days}) \}$

$\Rightarrow$

$\{ \Downarrow$ *true* $\}$

Side condition: $\mathit{true} \wedge (L(\texttt{year}) \wedge 366 < \texttt{days} \vee \neg L(\texttt{year}) \wedge 365 < \texttt{days}) \Rightarrow 0 \leq \texttt{days}$

3. Axiomatic Semantics

3.1 Hoare Logic

3.2 Soundness and Completeness

3.2.1 Proof of Soundness
3.2.2 Proof of Completeness

Motivation

- Developing an axiomatic semantics is difficult
- Soundness:
  If a property can be proved then it does indeed hold
  - An unsound inference system is useless
- Completeness:
  If a property does hold then it can be proved
  - With an incomplete inference system, a program might be correct, but we cannot prove it

Unsoundness: Example

$$\frac{\{\ \mathcal{B}[\![b]\!] \wedge \mathbf{P} \wedge \mathcal{A}[\![e]\!] = Z\ \}\ s\ \{\ \Downarrow \mathbf{P} \wedge \mathcal{A}[\![e]\!] < Z\ \}}{\{\ \mathbf{P} \wedge 0 \leq \mathcal{A}[\![e]\!]\ \}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\ \Downarrow \neg\mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}}$$

- With $e \equiv \texttt{x}$, we can derive:

$$\dfrac{\dfrac{\dfrac{\{\ \mathit{true} \wedge \texttt{x} - 1 < Z\ \}\ \texttt{x:=x} - 1\ \{\ \Downarrow \mathit{true} \wedge \texttt{x} < Z\ \}}{\{\ \mathit{true} \wedge \mathit{true} \wedge \texttt{x} = Z\ \}\ \texttt{x:=x} - 1\ \{\ \Downarrow \mathit{true} \wedge \texttt{x} < Z\ \}}}{\{\ \mathit{true} \wedge 0 \leq \texttt{x}\ \}\ \texttt{while true do x:=x} - 1\ \texttt{end}\ \{\ \Downarrow \neg\mathit{true} \wedge \mathit{true}\ \}}}{\{\ 0 \leq \texttt{x}\ \}\ \texttt{while true do x:=x} - 1\ \texttt{end}\ \{\ \Downarrow \mathit{true}\ \}}$$

- This derivation is not sound (the derived triple does not hold)
- The rule does not ensure that the loop variant is non-negative before each loop iteration

Incompleteness: Example

$$\frac{\{\ \mathcal{B}[\![b]\!] \wedge \mathbf{P} \wedge \mathcal{A}[\![e]\!] = Z\ \}\ s\ \{\ \Downarrow \mathbf{P} \wedge \mathcal{A}[\![e]\!] < Z\ \}}{\{\ \mathbf{P}\ \}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\ \Downarrow \neg\mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}}$$

Condition: $\mathbf{P} \Rightarrow 0 \leq \mathcal{A}[\![e]\!]$

- With this rule, we cannot prove that the following loop always terminates

```
while 0 < x do
  x := x - 1
end
```

  - The loop variant is `x`
  - The strongest possible loop invariant is *true* (because we want to show termination for all initial states)
  - This loop invariant is not strong enough to show the side condition

Soundness and Completeness

- Soundess and completeness can be proved w.r.t. an operational or denotational semantics

The partial correctness assertion $\{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}$ is valid—written as $\models \{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}$— iff

$$\forall \sigma, \sigma' \in \text{State} : \mathbf{P}(\sigma) = tt \wedge \langle s, \sigma \rangle \to \sigma' \Rightarrow \mathbf{Q}(\sigma') = tt$$

- Soundness: $\vdash \{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \} \Rightarrow \models \{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}$
- Completeness: $\models \{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \} \Rightarrow \vdash \{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}$

Theorem

Soundess and completeness theorem

For all partial correctness assertions $\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$ of IMP we have

$$\vdash \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \} \Leftrightarrow\ \models \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$$

3. Axiomatic Semantics

3.1 Hoare Logic

3.2 Soundness and Completeness

3.2.1 Proof of Soundness

3.2.2 Proof of Completeness

Soundness Proof

- We prove $\vdash \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \} \Rightarrow \models \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$
- That is, we have to show

$$\vdash \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \} \land \mathbf{P}(\sigma) = tt \land \langle s, \sigma \rangle \to \sigma' \Rightarrow \mathbf{Q}(\sigma') = tt$$

- The proof runs by induction on the shape of the inference tree for $\vdash \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$

Soundness Proof: Base Cases

- Case assign-axiom
  - Assume $\langle x := e, \sigma \rangle \to \sigma'$
  - We have to prove $(\mathbf{P}[x \mapsto e])\sigma = tt \Rightarrow \mathbf{P}(\sigma') = tt$
  - From the natural semantics, we get $\langle x := e, \sigma \rangle \to \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$
  - We have $(\mathbf{P}[x \mapsto e])\sigma = tt \Leftrightarrow \mathbf{P}(\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]) = tt$
- Case skip-axiom: Trivial

Soundness Proof: Composition

- Consider arbitrary states σ and σ'' where $\mathbf{P}(\sigma) = tt$ holds and $\langle s_1 ; s_2, \sigma \rangle \to \sigma''$
- From the natural semantics, we know that there is a state σ' such that $\langle s_1, \sigma \rangle \to \sigma'$ and $\langle s_2, \sigma' \rangle \to \sigma''$
- From the induction hypothesis, we get $\models \{ \mathbf{P} \}\ s_1\ \{ \mathbf{Q} \}$ and $\models \{ \mathbf{Q} \}\ s_2\ \{ \mathbf{R} \}$
- From $\models \{ \mathbf{P} \}\ s_1\ \{ \mathbf{Q} \}$, $\langle s_1, \sigma \rangle \to \sigma'$, and $\mathbf{P}(\sigma) = tt$, we get $\mathbf{Q}(\sigma') = tt$
- From $\models \{ \mathbf{Q} \}\ s_2\ \{ \mathbf{R} \}$, $\langle s_2, \sigma' \rangle \to \sigma''$, and $\mathbf{Q}(\sigma') = tt$, we get $\mathbf{R}(\sigma'') = tt$

Soundness Proof: Conditional

- Case 1: $\mathcal{B}[\![b]\!]\sigma = tt$
  - Consider arbitrary states σ and σ' where $\mathbf{P}(\sigma) = tt$ holds and $\langle \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}, \sigma \rangle \rightarrow \sigma'$
  - From the natural semantics, we get $\langle s_1, \sigma \rangle \rightarrow \sigma'$
  - From the induction hypothesis, we get $\models \{ \mathcal{B}[\![b]\!] \wedge \mathbf{P} \} \; s_1 \; \{ \mathbf{Q} \}$
  - From $\mathbf{P}(\sigma) = tt$ and $\mathcal{B}[\![b]\!]\sigma = tt$, we get $(\mathcal{B}[\![b]\!] \wedge \mathbf{P})\sigma = tt$
  - From $\models \{ \mathcal{B}[\![b]\!] \wedge \mathbf{P} \} \; s_1 \; \{ \mathbf{Q} \}$ and $(\mathcal{B}[\![b]\!] \wedge \mathbf{P})\sigma = tt$, we get $\mathbf{Q}(\sigma') = tt$
- Case 2: $\mathcal{B}[\![b]\!]\sigma = ff$ is analogous

Soundness Proof: Loop

- We have to prove

$$\begin{aligned}&\vdash \{\ \mathbf{P}\ \}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\ \neg\mathcal{B}[[b]] \wedge \mathbf{P}\ \} \wedge \\ &\mathbf{P}(\sigma) = tt \wedge \langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, \sigma\rangle \to \sigma'' \\ &\Rightarrow (\neg\mathcal{B}[[b]] \wedge \mathbf{P})\sigma''\end{aligned}$$

  where σ and σ'' are arbitrary states

- The proof runs by induction on the shape of the derivation tree for $\langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, \sigma\rangle \to \sigma''$

Soundness Proof: Loop (cont'd)

- Case 1: $\mathcal{B}[\![b]\!]\sigma = tt$
  - From the natural semantics, we get $\langle s, \sigma \rangle \to \sigma'$ and $\langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, \sigma' \rangle \to \sigma''$
  - From $\mathbf{P}(\sigma) = tt$ and $\mathcal{B}[\![b]\!]\sigma = tt$, we get $(\mathcal{B}[\![b]\!] \wedge \mathbf{P})\sigma = tt$
  - By applying the induction hypothesis of the outer induction to $\models \{\ \mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}\ s\ \{\ \mathbf{P}\ \}$, we get $\mathbf{P}(\sigma') = tt$
  - Now we can apply the induction hypothesis of the nested induction to $\langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, \sigma' \rangle \to \sigma''$ to get $(\neg\mathcal{B}[\![b]\!] \wedge \mathbf{P})\sigma'' = tt$
- Case 2: $\mathcal{B}[\![b]\!]\sigma = ff$
  - From the natural semantics, we get $\sigma = \sigma''$
  - $\mathbf{P}(\sigma) = tt$ and $\mathcal{B}[\![b]\!]\sigma = ff$ imply $(\neg\mathcal{B}[\![b]\!] \wedge \mathbf{P})\sigma'' = tt$

Soundness Proof: Consequence

- Consider arbitrary states σ and σ' where $\mathbf{P}(\sigma) = tt$ holds and $\langle s, \sigma \rangle \rightarrow \sigma'$
- We have $\models \{ \mathbf{P}' \}\ s\ \{ \mathbf{Q}' \}$, $\mathbf{P} \Rightarrow \mathbf{P}'$, and $\mathbf{Q}' \Rightarrow \mathbf{Q}$
- From $\mathbf{P}(\sigma) = tt$ and $\mathbf{P} \Rightarrow \mathbf{P}'$, we get $\mathbf{P}'(\sigma) = tt$
- By applying the induction hypothesis, we get $\mathbf{Q}'(\sigma') = tt$
- From $\mathbf{Q}'(\sigma') = tt$ and $\mathbf{Q}' \Rightarrow \mathbf{Q}$, we get $\mathbf{Q}(\sigma') = tt$

3. Axiomatic Semantics

3.1 Hoare Logic

3.2 Soundness and Completeness

3.2.1 Proof of Soundness

3.2.2 Proof of Completeness

Weakest (Liberal) Preconditions

- The weakest precondition of a statement s and a postcondition $\mathbf{Q}$ is the weakest predicate that has to hold in the initial state of an execution of s to guarantee that $\mathbf{Q}$ holds in the final state
  - The weakest precondition $wp(s, \mathbf{Q})$ guarantees termination
  - The weakest liberal precondition $wlp(s, \mathbf{Q})$ does not guarantee termination

$$\begin{aligned} wp(s, \mathbf{Q})\sigma = tt \quad &\Leftrightarrow \exists \sigma' : (\langle s, \sigma \rangle \rightarrow \sigma' \wedge \mathbf{Q}(\sigma')) \\ wlp(s, \mathbf{Q})\sigma = tt \quad &\Leftrightarrow \forall \sigma' : (\langle s, \sigma \rangle \rightarrow \sigma' \Rightarrow \mathbf{Q}(\sigma')) \end{aligned}$$

- In the following, we consider partial correctness

wlp-Lemma

Lemma: For every statement s and predicate $\mathbf{Q}$ we have

1. $\models \{\ wlp(s, \mathbf{Q})\ \}\ s\ \{\ \mathbf{Q}\ \}$
2. $\models \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \} \Rightarrow (\mathbf{P} \Rightarrow wlp(s, \mathbf{Q}))$

- Proof 1:
  - Let $wlp(s, \mathbf{Q})\sigma = tt$ and $\langle s, \sigma \rangle \rightarrow \sigma'$
  - From the definition of *wlp*, we get $\mathbf{Q}(\sigma')$
- Proof 2:
  - Let $\mathbf{P}(\sigma) = tt$ and $\langle s, \sigma \rangle \rightarrow \sigma'$
  - From $\models \{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$, we get $\mathbf{Q}(\sigma') = tt$
  - From the definition of *wlp*, we get $wlp(s, \mathbf{Q})\sigma'$

Completeness Proof

- We prove $\models \{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \} \Rightarrow \vdash \{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}$
- It suffices to infer $\vdash \{ \mathit{wlp}(s, \mathbf{Q}) \}\ s\ \{ \mathbf{Q} \}$
  - By $\models \{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}$, the *wlp*-lemma implies $\mathbf{P} \Rightarrow \mathit{wlp}(s, \mathbf{Q})$

$$\frac{\{ \mathit{wlp}(s, \mathbf{Q}) \}\ s\ \{ \mathbf{Q} \}}{\{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}}$$

- We prove $\vdash \{ \mathit{wlp}(s, \mathbf{Q}) \}\ s\ \{ \mathbf{Q} \}$ by structural induction on s

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Completeness Proof: Base Cases

- Case assign-axiom
  - From the natural semantics, we get $\langle x\texttt{:=}e, \sigma\rangle = \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$
  - From the definition of *wlp*, we get $wlp(x\texttt{:=}e, \mathbf{Q})\sigma \Leftrightarrow \mathbf{Q}(\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma])$
  - Therefore, we get $wlp(x\texttt{:=}e, \mathbf{Q}) = \mathbf{Q}[x \mapsto e]$
  - We can infer $\vdash \{\ \mathbf{Q}[x \mapsto e]\ \}\ x\texttt{:=}e\ \{\ \mathbf{Q}\ \}$
- Case skip-axiom:
  - From the natural semantics, we get $wlp(\texttt{skip}, \mathbf{Q}) = \mathbf{Q}$
  - We can infer $\vdash \{\ \mathbf{Q}\ \}\ \texttt{skip}\ \{\ \mathbf{Q}\ \}$

Completeness Proof: Composition

- By the induction hypothesis, we get $\vdash \{\ wlp(s_2, \mathbf{Q})\ \}\ s_2\ \{\ \mathbf{Q}\ \}$ and $\vdash \{\ wlp(s_1, wlp(s_2, \mathbf{Q}))\ \}\ s_1\ \{\ wlp(s_2, \mathbf{Q})\ \}$
- We can infer $\vdash \{\ wlp(s_1, wlp(s_2, \mathbf{Q}))\ \}\ s_1\,;s_2\ \{\ \mathbf{Q}\ \}$
- It remains to prove that $wlp(s_1\,;s_2, \mathbf{Q}) \Rightarrow wlp(s_1, wlp(s_2, \mathbf{Q}))$
- We assume that $wlp(s_1\,;s_2, \mathbf{Q})\sigma = tt$ and show that $wlp(s_1, wlp(s_2, \mathbf{Q}))\sigma = tt$

Completeness Proof: Composition (2)

- If there is no σ' such that $\langle s_1, \sigma\rangle \rightarrow \sigma'$ then $\mathit{wlp}(s_1, \mathit{wlp}(s_2, \mathbf{Q}))\sigma = \mathit{tt}$ follows immediately from the definition of *wlp*
- Otherwise, we have to show $\mathit{wlp}(s_2, \mathbf{Q})\sigma' = \mathit{tt}$
- Again, if there is no σ'' such that $\langle s_2, \sigma'\rangle \rightarrow \sigma''$ then $\mathit{wlp}(s_2, \mathbf{Q})\sigma' = \mathit{tt}$ follows immediately from the definition of *wlp*
- Otherwise, we have to show $\mathbf{Q}(\sigma'')$
- $\mathbf{Q}(\sigma'')$ follows from $\mathit{wlp}(s_1\,;s_2, \mathbf{Q})\sigma = \mathit{tt}$ and $\langle s_1\,;s_2, \sigma\rangle \rightarrow \sigma''$

Completeness Proof: Conditional

- By the induction hypothesis, we get $\vdash \{\ wlp(s_1, \mathbf{Q})\ \}\ s_1\ \{\ \mathbf{Q}\ \}$ and $\vdash \{\ wlp(s_2, \mathbf{Q})\ \}\ s_2\ \{\ \mathbf{Q}\ \}$
- Define $\mathbf{P} \equiv (\mathcal{B}[\![b]\!] \wedge wlp(s_1, \mathbf{Q})) \vee (\neg\mathcal{B}[\![b]\!] \wedge wlp(s_2, \mathbf{Q}))$
- We have $\mathcal{B}[\![b]\!] \wedge \mathbf{P} \Rightarrow wlp(s_1, \mathbf{Q})$ and $\neg\mathcal{B}[\![b]\!] \wedge \mathbf{P} \Rightarrow wlp(s_2, \mathbf{Q})$
- We derive

$$\dfrac{\dfrac{\{\ wlp(s_1, \mathbf{Q})\ \}\ s_1\ \{\ \mathbf{Q}\ \}}{\{\ \mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}\ s_1\ \{\ \mathbf{Q}\ \}} \qquad \dfrac{\{\ wlp(s_2, \mathbf{Q})\ \}\ s_2\ \{\ \mathbf{Q}\ \}}{\{\ \neg\mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}\ s_2\ \{\ \mathbf{Q}\ \}}}{\{\ \mathbf{P}\ \}\ \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{end}\ \{\ \mathbf{Q}\ \}}$$

Completeness Proof: Conditional (2)

- We have $\mathbf{P} \equiv (\mathcal{B}[\![b]\!] \wedge \mathit{wlp}(s_1, \mathbf{Q})) \vee (\neg\mathcal{B}[\![b]\!] \wedge \mathit{wlp}(s_2, \mathbf{Q}))$
- It remains to show that
  $\mathit{wlp}(\texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}, \mathbf{Q})\sigma = \mathit{tt} \Rightarrow \mathbf{P}(\sigma) = \mathit{tt}$
- Case 1: $\mathcal{B}[\![b]\!]\sigma = \mathit{tt}$
  - If there is no σ' such that $\langle s_1, \sigma\rangle \rightarrow \sigma'$ then $\mathit{wlp}(s_1, \mathbf{Q})\sigma = \mathit{tt}$ follows immediately from the definition of *wlp*
  - Otherwise, we have to prove $\mathbf{Q}(\sigma')$
  - From $\mathit{wlp}(\texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}, \mathbf{Q})\sigma = \mathit{tt}$ and $\langle \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}, \sigma\rangle \rightarrow \sigma'$, we get $\mathbf{Q}(\sigma')$
- Case 2: $\mathcal{B}[\![b]\!]\sigma = \mathit{ff}$ is analogous

Completeness Proof: Loop

- Define $\mathbf{P} \equiv wlp(\texttt{while } b \texttt{ do } s \texttt{ end}, \mathbf{Q})$
- We will prove

  (1) $(\neg\mathcal{B}[\![b]\!] \wedge \mathbf{P}) \Rightarrow \mathbf{Q}$

  (2) $(\mathcal{B}[\![b]\!] \wedge \mathbf{P}) \Rightarrow wlp(s, \mathbf{P})$

- By the induction hypothesis, we get $\vdash \{\ wlp(s, \mathbf{P})\ \}\ s\ \{\ \mathbf{P}\ \}$
- From (2), we get $\vdash \{\ \mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}\ s\ \{\ \mathbf{P}\ \}$
- Be the while rule, we get $\vdash \{\ \mathbf{P}\ \}\ \texttt{while } b \texttt{ do } s \texttt{ end}\ \{\ \neg\mathcal{B}[\![b]\!] \wedge \mathbf{P}\ \}$
- From (1), we get $\vdash \{\ \mathbf{P}\ \}\ \texttt{while } b \texttt{ do } s \texttt{ end}\ \{\ \mathbf{Q}\ \}$

Completeness Proof: Loop (2)

- We prove (1): $(\neg\mathcal{B}[\![b]\!] \wedge \mathbf{P}) \Rightarrow \mathbf{Q}$
- Assume $(\neg\mathcal{B}[\![b]\!] \wedge \mathbf{P})\sigma = tt$
- Then we have $\langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, \sigma \rangle = \sigma$
- By $wlp(\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, \mathbf{Q})\sigma = tt$ and the definition of *wlp*, we get $\mathbf{Q}(\sigma) = tt$

Completeness Proof: Loop (3)

- We prove (2): $(\mathcal{B}[\![b]\!] \wedge \mathbf{P}) \Rightarrow wlp(s, \mathbf{P})$
- We assume $(\mathcal{B}[\![b]\!] \wedge \mathbf{P})\sigma = tt$ and show that $wlp(s, \mathbf{P})\sigma = tt$
- If there is no σ' such that $\langle s, \sigma \rangle \rightarrow \sigma'$ then $wlp(s, \mathbf{P})\sigma = tt$ follows immediately from the definition of *wlp*
- Otherwise, we have to show $\mathbf{P}(\sigma') = tt$

Completeness Proof: Loop (4)

- Case 1: There is no σ'' such that $\langle \texttt{while } b \texttt{ do } s \texttt{ end}, \sigma' \rangle = \sigma''$
  - By the definition of *wlp*, we get that $wlp(\texttt{while } b \texttt{ do } s \texttt{ end}, \mathbf{Q})\sigma' = tt$ and, thus, $\mathbf{P}(\sigma') = tt$
- Case 2: There is a σ'' such that $\langle \texttt{while } b \texttt{ do } s \texttt{ end}, \sigma' \rangle = \sigma''$
  - From $\langle s, \sigma \rangle \rightarrow \sigma'$ and $\langle \texttt{while } b \texttt{ do } s \texttt{ end}, \sigma' \rangle = \sigma''$, we get $\langle \texttt{while } b \texttt{ do } s \texttt{ end}, \sigma \rangle = \sigma''$
  - By $\mathbf{P}(\sigma) = tt$ and $\langle \texttt{while } b \texttt{ do } s \texttt{ end}, \sigma \rangle = \sigma''$, we get $\mathbf{Q}(\sigma'') = tt$
  - By $\mathbf{Q}(\sigma'') = tt$ and $\langle \texttt{while } b \texttt{ do } s \texttt{ end}, \sigma' \rangle = \sigma''$, we get $wlp(\texttt{while } b \texttt{ do } s \texttt{ end}, \mathbf{Q})\sigma' = tt$ and, thus, $\mathbf{P}(\sigma') = tt$

Summary: Axiomatic Semantics

- Axiomatic semantics
  - expresses specific properties of the effect of executing a program
  - Some aspects of the computation may be ignored
- Axiomatic semantics is used to verify programs
  - Partial correctness
  - Total correctness
  - Other properties, e.g., resource consumption
- The inference system should be sound and complete