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 $\texttt{x} < 1$

Formal Specification

- Specification:
  The final value of `y` is the factorial of the initial value of `x`
- We can express the specification formally based on a formal semantics

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

- 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

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

- Step 2: The loop satisfies

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

- Step 3: The whole statement is partially correct

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

Proof: Step 1—Loop Body

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

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

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

Proof: Step 2—Loop

- Step 2: The loop satisfies

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

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

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

- 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

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

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

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

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

- 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

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 informal meaning of $\{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}$ is

> If **P** evaluates to true in the initial state σ, and
> if the execution of s from σ terminates in a state σ'
> then **Q** will evaluate to true 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

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

- In general, this assertion does not hold
  - Consider an initial state $\{\,\mathtt{x} \mapsto 2, \mathtt{y} \mapsto 0\,\}$
  - The final state will be $\{\,\mathtt{x} \mapsto 1, \mathtt{y} \mapsto 2\,\}$
- We have to express that $\mathtt{y}$ in the final state is the factorial of $\mathtt{x}$ 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, boolean expressions
  - It is common to use a richer set of expressions for assertions, for instance, to include quantification
  - We will use additional expressions when it is convenient (e.g., `x!`)
- We will use the following convenient notations
  - "$P_1 \wedge P_2$" for "P_1 `and` P_2"
  - "$P_1 \vee P_2$" for "P_1 `or` P_2"
  - "$\neg P$" for "`not` P"
  - "$P[x \mapsto e]$" for P with each free occurrence of x replaced by e
- We will use the following substitution lemma (see exercise for proof):

$$\boxed{\mathcal{B}[\![\mathbf{P}[x \mapsto e]]\!]\sigma \Leftrightarrow \mathcal{B}[\![\mathbf{P}]\!]\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]}$$

3. Axiomatic Semantics

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

$$\textsc{Skip}_{Ax}\ \frac{}{\{\ \mathbf{P}\ \}\ \texttt{skip}\ \{\ \mathbf{P}\ \}}$$

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

$$\textsc{Ass}_{Ax}\ \frac{}{\{\ \mathbf{P}[x \mapsto e]\ \}\ x\texttt{:=}e\ \{\ \mathbf{P}\ \}}$$

  - Let σ be the initial state
  - Precondition: $\mathcal{B}[\![\mathbf{P}[x \mapsto e]]\!]\sigma$, which is equivalent to $\mathcal{B}[\![\mathbf{P}]\!]\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$ (substitution lemma)
  - Final state: $\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$
  - Consequently, $\mathcal{B}[\![\mathbf{P}]\!]$ holds in the final state

- The rules are axiom schemes

Axiomatic Semantics of IMP (cont'd)

- Sequential composition $s_1 ; s_2$

$$\textsc{Seq}_{Ax}\ \frac{\{\ \mathbf{P}\ \}\ s_1\ \{\ \mathbf{Q}\ \} \qquad \{\ \mathbf{Q}\ \}\ s_2\ \{\ \mathbf{R}\ \}}{\{\ \mathbf{P}\ \}\ s_1 ; s_2\ \{\ \mathbf{R}\ \}}$$

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

$$\textsc{If}_{Ax}\ \frac{\{\ b \wedge \mathbf{P}\ \}\ s_1\ \{\ \mathbf{Q}\ \} \qquad \{\ \neg b \wedge \mathbf{P}\ \}\ s_2\ \{\ \mathbf{Q}\ \}}{\{\ \mathbf{P}\ \}\ \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{end}\ \{\ \mathbf{Q}\ \}}$$

Axiomatic Semantics of IMP (cont'd)

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

$$\text{W}\text{H}_{Ax}\ \frac{\{\ b \wedge \mathbf{P}\ \}\ s\ \{\ \mathbf{P}\ \}}{\{\ \mathbf{P}\ \}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\ \neg b \wedge \mathbf{P}\ \}}$$

  - **P** is the loop invariant

- Rule of consequence

$$\text{C}\text{ONS}_{Ax}\ \frac{\{\ \mathbf{P}'\ \}\ s\ \{\ \mathbf{Q}'\ \}}{\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}}\ \text{if } \mathbf{P} \Rightarrow \mathbf{P}' \text{ and } \mathbf{Q}' \Rightarrow \mathbf{Q}$$

  - We can strengthen preconditions
  - We can weaken postconditions
  - $\mathbf{P} \Rightarrow \mathbf{Q}$ is defined as:
    "For all states σ, $\mathcal{B}[\![\mathbf{P}]\!]\sigma$ implies $\mathcal{B}[\![\mathbf{Q}]\!]\sigma$"

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

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

where we write:

- **P** for $\texttt{x} = X_0 \land \texttt{y} = Y_0$
- **Q** for $\texttt{x} = Y_0 \land \texttt{y} = X_0$
- **Q′** for $\texttt{x} = Y_0 \land \texttt{z} = X_0$

Inference Trees: Example 2

- Consider the non-terminating loop

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

- We can build the following inference tree

$$\textsc{Cons}_{Ax}\dfrac{\textsc{Wh}_{Ax}\dfrac{\textsc{Cons}_{Ax}\dfrac{\textsc{Skip}_{Ax}\dfrac{}{\{\,\texttt{true}\,\}\ \texttt{skip}\ \{\,\texttt{true}\,\}}}{\{\,\texttt{true}\wedge\texttt{true}\,\}\ \texttt{skip}\ \{\,\texttt{true}\,\}}}{\{\,\texttt{true}\,\}\ \texttt{while true do skip end}\ \{\,\neg\texttt{true}\wedge\texttt{true}\,\}}}{\{\,\texttt{true}\,\}\ \texttt{while true do skip end}\ \{\,\neg\texttt{true}\,\}}$$

- 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:

$$
\begin{array}{l}
\{\, \mathbf{P} \,\} \\
\quad \texttt{if } b \texttt{ then} \\
\qquad \{\, b \wedge \mathbf{P} \,\} \\
\qquad\quad s_1 \\
\qquad \{\, \mathbf{Q} \,\} \\
\quad \texttt{else} \\
\qquad \{\, \neg b \wedge \mathbf{P} \,\} \\
\qquad\quad s_2 \\
\qquad \{\, \mathbf{Q} \,\} \\
\quad \texttt{end} \\
\{\, \mathbf{Q} \,\}
\end{array}
$$

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

$$
\begin{array}{l}
\{\, \mathbf{P} \,\} \\
\quad \texttt{while } b \texttt{ do} \\
\qquad \{\, b \wedge \mathbf{P} \,\} \\
\qquad\quad s \\
\qquad \{\, \mathbf{P} \,\} \\
\quad \texttt{end} \\
\{\, \neg b \wedge \mathbf{P} \,\}
\end{array}
$$

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

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

Proof Outlines: Example

- Back to our swap-example:

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

- Proof outline:

$$
\begin{array}{l}
\{\ \mathtt{x} = X_0 \wedge \mathtt{y} = Y_0\ \} \\
\Rightarrow \\
\{\ \mathtt{y} = Y_0 \wedge \mathtt{x} = X_0\ \} \\
\quad \boxed{\texttt{z := x;}} \\
\{\ \mathtt{y} = Y_0 \wedge \mathtt{z} = X_0\ \} \\
\quad \boxed{\texttt{x := y;}} \\
\{\ \mathtt{x} = Y_0 \wedge \mathtt{z} = X_0\ \} \\
\quad \boxed{\texttt{y := z}} \\
\{\ \mathtt{x} = Y_0 \wedge \mathtt{y} = X_0\ \}
\end{array}
$$

- Proof outlines are typically developed bottom-up

Verification of Factorial Statement

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

Verification of Factorial Statement

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

- Determining the loop invariant

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

Verification of Factorial Statement

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

- 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	$N*(N-1)$	$N*(N-1)*\ldots*(N-i+1)$	$N!$

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

Proof Outline for Factorial Statement

$\{ \mathtt{x} = N \}$
$\Rightarrow$
$\{ \mathtt{x} > 0 \Rightarrow \mathtt{1*x!} = N! \wedge N \geq \mathtt{x} \}$
$\quad \boxed{\mathtt{y\ :=\ 1;}}$
$\{ \mathtt{x} > 0 \Rightarrow \mathtt{y*x!} = N! \wedge N \geq \mathtt{x} \}$
$\quad \boxed{\mathtt{while\ not\ x\ =\ 1\ do}}$
$\quad \{ \mathtt{x} \neq 1 \wedge (\mathtt{x} > 0 \Rightarrow \mathtt{y*x!} = N! \wedge N \geq \mathtt{x}) \}$
$\quad \Rightarrow$
$\quad \{ \mathtt{x-1} > 0 \Rightarrow \mathtt{y*x*x-1!} = N! \wedge N \geq \mathtt{x-1} \}$
$\quad\quad \boxed{\mathtt{y\ :=\ y*x;}}$
$\quad \{ \mathtt{x-1} > 0 \Rightarrow \mathtt{y*x-1!} = N! \wedge N \geq \mathtt{x-1} \}$
$\quad\quad \boxed{\mathtt{x\ :=\ x-1}}$
$\quad \{ \mathtt{x} > 0 \Rightarrow \mathtt{y*x!} = N! \wedge N \geq \mathtt{x} \}$
$\quad \boxed{\mathtt{end}}$
$\{ \mathtt{x} = 1 \wedge (\mathtt{x} > 0 \Rightarrow \mathtt{y*x!} = N! \wedge N \geq \mathtt{x}) \}$
$\Rightarrow$
$\{ \mathtt{y} = N! \wedge N > 0 \}$

Proof Outline for Zune Example

$\{\ 0 < D\ \}$
$\Rightarrow$
$\{\ 1980 - 1980 \leq (D - D)/365 \wedge 0 < D\ \}$

```
year := 1980;
```

$\{\ \text{year} - 1980 \leq (D - D)/365 \wedge 0 < D\ \}$

```
days := D;
```

$\{\ \text{year} - 1980 \leq (D - \text{days})/365 \wedge 0 < \text{days}\ \}$

```
while L(year) and 366 < days or not L(year) and 365 < days do
```

$\{\ (L(\text{year}) \wedge 366 < \text{days} \vee \neg L(\text{year}) \wedge 365 < \text{days}) \wedge \text{year} - 1980 \leq (D - \text{days})/365 \wedge 0 < \text{days}\ \}$
$\Rightarrow$
$\{\ \text{year} - 1980 \leq (D - \text{days})/365 \wedge (L(\text{year}) \Rightarrow 366 < \text{days}) \wedge 365 < \text{days}\ \}$

```
if L(year) then
```

$\{\ L(\text{year}) \wedge \text{year} - 1980 \leq (D - \text{days})/365 \wedge (L(\text{year}) \Rightarrow 366 < \text{days}) \wedge 365 < \text{days}\ \}$
$\Rightarrow$
$\{\ \text{year} + 1 - 1980 \leq (D - (\text{days} - 366))/365 \wedge 0 < (\text{days} - 366)\ \}$

```
days := days - 366
```

$\{\ \text{year} + 1 - 1980 \leq (D - \text{days})/365 \wedge 0 < \text{days}\ \}$

```
else
```

$\{\ \neg L(\text{year}) \wedge \text{year} - 1980 \leq (D - \text{days})/365 \wedge (L(\text{year}) \Rightarrow 366 < \text{days}) \wedge 365 < \text{days}\ \}$
$\Rightarrow$
$\{\ \text{year} + 1 - 1980 \leq (D - (\text{days} - 365))/365 \wedge 0 < \text{days} - 365\ \}$

```
days := days - 365;
```

$\{\ \text{year} + 1 - 1980 \leq (D - \text{days})/365 \wedge 0 < \text{days}\ \}$

```
end;
```

$\{\ \text{year} + 1 - 1980 \leq (D - \text{days})/365 \wedge 0 < \text{days}\ \}$

```
year := year + 1
```

$\{\ \text{year} - 1980 \leq (D - \text{days})/365 \wedge 0 < \text{days}\ \}$

```
end
```

$\{\ \neg(L(\text{year}) \wedge 366 < \text{days} \vee \neg L(\text{year}) \wedge 365 < \text{days}) \wedge \text{year} - 1980 \leq (D - \text{days})/365 \wedge 0 < \text{days}\ \}$
$\Rightarrow$
$\{\ \text{year} - 1980 \leq D/365\ \}$

3. Axiomatic Semantics

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

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

  - 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 Skip_{Ax}
  - 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 Cons_{Ax}, because no other rule applies to `skip`
  - From Cons_{Ax}, 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 "$\Rightarrow$"
  - 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:

$$\textsc{Seq}_{Ax}\ \dfrac{\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \} \qquad \textsc{Skip}_{Ax}\ \dfrac{}{\{\ \mathbf{Q}\ \}\ \texttt{skip}\ \{\ \mathbf{Q}\ \}}}{\{\ \mathbf{P}\ \}\ s;\texttt{skip}\ \{\ \mathbf{Q}\ \}}$$

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: $\textsc{Seq}_{Ax}$ and $\textsc{Cons}_{Ax}$
  - We continue by case distinction

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

- Case Seq_{Ax}
  - 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 Cons_{Ax} to obtain $\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$

- Case Cons_{Ax}
  - 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 Cons_{Ax} to obtain a tree for $\{\ \mathbf{P}\ \}\ s\ \{\ \mathbf{Q}\ \}$

3. Axiomatic Semantics

Total Correctness

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

> If **P** evaluates to true in the initial state σ
> then the execution of s from σ terminates
> and **Q** will evaluate to true 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 an expression that evaluates to a value in 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 `x`

While Rule for Total Correctness

- For simplicity, we consider loop variants that evaluate to values in $\mathbb{N}$
  - We use arithmetic expressions of IMP as loop variants
  - 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:

$$\boxed{\textsc{WhTot}_{Ax} \frac{\{\ b \land \mathbf{P} \land e = Z\ \}\ s\ \{\ \Downarrow \mathbf{P} \land e < Z\ \}}{\{\ \mathbf{P}\ \}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\ \Downarrow \neg b \land \mathbf{P}\ \}}\ \text{if } b \land \mathbf{P} \Rightarrow 0 \leq e}$$

  where Z is a logical variable

- Other well-founded sets are also possible and useful

Total Correctness of Factorial

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

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

Proof Outline for Factorial Statement

$$\{\, x = N \wedge x > 0 \,\}$$
$$\Rightarrow$$
$$\{\, x > 0 \wedge 1{*}x! = N! \,\}$$
$$\quad \boxed{\texttt{y := 1;}}$$
$$\{\, x > 0 \wedge y{*}x! = N! \,\}$$
$$\quad \boxed{\texttt{while not x = 1 do}}$$
$$\quad \{\, x \neq 1 \wedge x > 0 \wedge y{*}x! = N! \wedge x = Z \,\}$$
$$\quad \Rightarrow$$
$$\quad \{\, x - 1 > 0 \wedge (y{*}x){*}(x-1)! = N! \wedge x - 1 < Z \,\}$$
$$\qquad \boxed{\texttt{y := y*x;}}$$
$$\quad \{\, x - 1 > 0 \wedge y{*}(x-1)! = N! \wedge x - 1 < Z \,\}$$
$$\qquad \boxed{\texttt{x := x-1}}$$
$$\quad \{\, \Downarrow x > 0 \wedge y{*}x! = N! \wedge x < Z \,\}$$
$$\quad \boxed{\texttt{end}}$$
$$\{\, \Downarrow x = 1 \wedge x > 0 \wedge y{*}x! = N! \,\}$$
$$\Rightarrow$$
$$\{\, \Downarrow y = N! \,\}$$

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

(Failing) Proof Attempt for Zune Bug

$\{\ \texttt{true}\ \}$
$\boxed{\texttt{while 365 < days do}}$
$\{\ 365 < \texttt{days} \land \texttt{days} = Z\ \}$
$\boxed{\texttt{if } L(\texttt{year}) \texttt{ then}}$
$\{\ L(\texttt{year}) \land 365 < \texttt{days} \land \texttt{days} = Z\ \}$
$\boxed{\texttt{if 366 < days then}}$
$\{\ 366 < \texttt{days} \land L(\texttt{year}) \land 365 < \texttt{days} \land \texttt{days} = Z\ \}$
$\Rightarrow$
$\{\ \texttt{days} - 366 < Z\ \}$
$\boxed{\texttt{days := days - 366;year := year + 1}}$
$\{\ \Downarrow \texttt{days} < Z\ \}$
$\boxed{\texttt{else}}$
$\{\ \neg(366 < \texttt{days}) \land L(\texttt{year}) \land 365 < \texttt{days} \land \texttt{days} = Z\ \}$
$\Rightarrow$
$\{\ \texttt{days} < Z\ \}$
$\boxed{\texttt{skip}}$
$\{\ \Downarrow \texttt{days} < Z\ \}$
$\boxed{\texttt{end}}$
$\{\ \Downarrow \texttt{days} < Z\ \}$
$\boxed{\texttt{else}}$
$\{\ \neg L(\texttt{year}) \land 365 < \texttt{days} \land \texttt{days} = Z\ \}$
$\Rightarrow$
$\{\ \texttt{days} - 365 < Z\ \}$
$\boxed{\texttt{days := days - 365; year := year + 1}}$
$\{\ \Downarrow \texttt{days} < Z\ \}$
$\boxed{\texttt{end}}$
$\{\ \Downarrow \texttt{days} < Z\ \}$
$\boxed{\texttt{end}}$
$\{\ \Downarrow \neg(365 < \texttt{days})\ \}$
$\Rightarrow$
$\{\ \Downarrow \texttt{true}\ \}$

Termination Proof for Corrected Zune Example

$\{\ \mathtt{true}\ \}$

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

$\{\ \mathtt{true}\ \}$

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

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

$\Rightarrow$

$\{\ \mathtt{days} = Z\ \}$

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

$\{\ L(\mathtt{year}) \land \mathtt{days} = Z\ \}$

$\Rightarrow$

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

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

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

$\boxed{\mathtt{else}}$

$\{\ \neg L(\mathtt{year}) \land \mathtt{days} = Z\ \}$

$\Rightarrow$

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

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

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

$\boxed{\mathtt{end;}}$

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

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

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

$\boxed{\mathtt{end}}$

$\{\ \Downarrow \neg(L(\mathtt{year}) \land 366 < \mathtt{days} \lor \neg L(\mathtt{year}) \land 365 < \mathtt{days})\ \}$

$\Rightarrow$

$\{\ \Downarrow \mathtt{true}\ \}$

Side condition: $(L(\mathtt{year}) \land 366 < \mathtt{days} \lor \neg L(\mathtt{year}) \land 365 < \mathtt{days}) \land \mathtt{true} \Rightarrow 0 \leq \mathtt{days}$

3. Axiomatic Semantics

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

$$\text{WHU}_{Ax}\ \frac{\{\, b \wedge \mathbf{P} \wedge e = Z \,\}\ s\ \{\Downarrow \mathbf{P} \wedge e < Z \,\}}{\{\, \mathbf{P} \wedge 0 \leq e \,\}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\Downarrow \neg b \wedge \mathbf{P} \,\}}$$

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

$$\text{CONS}\ \dfrac{\text{WHU}\ \dfrac{\text{CONS}\ \dfrac{\text{ASS}\ \dfrac{}{\{\, \texttt{true} \wedge \mathtt{x} - 1 < Z \,\}\ \mathtt{x{:=}x} - 1\ \{\Downarrow \texttt{true} \wedge \mathtt{x} < Z \,\}}}{\{\, \texttt{true} \wedge \texttt{true} \wedge \mathtt{x} = Z \,\}\ \mathtt{x{:=}x} - 1\ \{\Downarrow \texttt{true} \wedge \mathtt{x} < Z \,\}}}{\{\, \texttt{true} \wedge 0 \leq \mathtt{x} \,\}\ \texttt{while true do}\ \mathtt{x{:=}x} - 1\ \texttt{end}\ \{\Downarrow \neg\texttt{true} \wedge \texttt{true} \,\}}}{\{\, 0 \leq \mathtt{x} \,\}\ \texttt{while true do}\ \mathtt{x{:=}x} - 1\ \texttt{end}\ \{\Downarrow \texttt{true} \,\}}$$

- 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

$$\textsc{Whi}_{Ax}\ \frac{\{\, b \wedge \mathbf{P} \wedge e = Z \,\}\ s\ \{\, \Downarrow \mathbf{P} \wedge e < Z \,\}}{\{\, \mathbf{P} \,\}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\, \Downarrow \neg b \wedge \mathbf{P} \,\}}\ \text{if } \mathbf{P} \Rightarrow 0 \leq e$$

- 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

$$\forall \sigma, \sigma' \in \text{State} : \mathcal{B}[\![\mathbf{P}]\!]\sigma = tt \wedge \langle s, \sigma \rangle \rightarrow \sigma' \Rightarrow \mathcal{B}[\![\mathbf{Q}]\!]\sigma' = tt$$

- 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

Soundness Proof

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

$$\vdash \{ \mathbf{P} \} \, s \, \{ \mathbf{Q} \} \wedge \mathcal{B}[\![\mathbf{P}]\!]\sigma = tt \wedge \langle s, \sigma \rangle \rightarrow \sigma' \Rightarrow \mathcal{B}[\![\mathbf{Q}]\!]\sigma' = tt$$

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

Soundness Proof: Base Cases

- Case $\textsc{Ass}_{Ax}$
  - Assume $\langle x := e, \sigma \rangle \rightarrow \sigma'$
  - We have to prove $\mathcal{B}[\![\mathbf{P}[x \mapsto e]]\!]\sigma = tt \Rightarrow \mathcal{B}[\![\mathbf{P}]\!]\sigma' = tt$
  - From the natural semantics, we get $\langle x := e, \sigma \rangle \rightarrow \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$
  - By the substitution lemma, we have $\mathcal{B}[\![\mathbf{P}[x \mapsto e]]\!]\sigma = tt \Leftrightarrow \mathcal{B}[\![\mathbf{P}]\!]\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma] = tt$
- Case $\textsc{Skip}_{Ax}$: Trivial

Soundness Proof: Composition

- Consider arbitrary states σ and σ'' where $\mathcal{B}[\![\mathbf{P}]\!]\sigma = tt$ 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 $\mathcal{B}[\![\mathbf{P}]\!]\sigma = tt$, we get $\mathcal{B}[\![\mathbf{Q}]\!]\sigma' = tt$
- From $\models \{\ \mathbf{Q}\ \}\ s_2\ \{\ \mathbf{R}\ \}$, $\langle s_2, \sigma'\rangle \to \sigma''$, and $\mathcal{B}[\![\mathbf{Q}]\!]\sigma' = tt$, we get $\mathcal{B}[\![\mathbf{R}]\!]\sigma'' = tt$

Soundness Proof: Conditional

- Case 1: $\mathcal{B}[\![b]\!]\sigma = tt$
  - Consider arbitrary states σ and σ' where $\mathcal{B}[\![\mathbf{P}]\!]\sigma = tt$ 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 $\models \{ b \wedge \mathbf{P} \}\ s_1\ \{ \mathbf{Q} \}$
  - From $\mathcal{B}[\![\mathbf{P}]\!]\sigma = tt$ and $\mathcal{B}[\![b]\!]\sigma = tt$, we get $\mathcal{B}[\![b \wedge \mathbf{P}]\!]\sigma = tt$
  - From $\models \{ b \wedge \mathbf{P} \}\ s_1\ \{ \mathbf{Q} \}$ and $\mathcal{B}[\![b \wedge \mathbf{P}]\!]\sigma = tt$, we get $\mathcal{B}[\![\mathbf{Q}]\!]\sigma' = tt$
- Case 2: $\mathcal{B}[\![b]\!]\sigma = ff$ is analogous

Soundness Proof: Loop

- We have to prove

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

  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 $\mathcal{B}[\![\mathbf{P}]\!]\sigma = tt$ 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 $\models \{\ b \wedge \mathbf{P}\ \}\ s\ \{\ \mathbf{P}\ \}$, we get $\mathcal{B}[\![\mathbf{P}]\!]\sigma' = tt$
  - 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 $\mathcal{B}[\![\neg b \wedge \mathbf{P}]\!]\sigma'' = tt$
- Case 2: $\mathcal{B}[\![b]\!]\sigma = ff$
  - From the natural semantics, we get $\sigma = \sigma''$
  - $\mathcal{B}[\![\mathbf{P}]\!]\sigma = tt$ and $\mathcal{B}[\![b]\!]\sigma = ff$ imply $\mathcal{B}[\![\neg b \wedge \mathbf{P}]\!]\sigma'' = tt$

Soundness Proof: Consequence

- Consider arbitrary states σ and σ' where $\mathcal{B}[\![\mathbf{P}]\!]\sigma = tt$ holds and $\langle s, \sigma \rangle \to \sigma'$
- We have $\models \{ \mathbf{P'} \}\ s\ \{ \mathbf{Q'} \}$, $\mathbf{P} \Rightarrow \mathbf{P'}$, and $\mathbf{Q'} \Rightarrow \mathbf{Q}$
- From $\mathcal{B}[\![\mathbf{P}]\!]\sigma = tt$ and $\mathbf{P} \Rightarrow \mathbf{P'}$, we get $\mathcal{B}[\![\mathbf{P'}]\!]\sigma = tt$
- By applying the induction hypothesis, we get $\mathcal{B}[\![\mathbf{Q'}]\!]\sigma' = tt$
- From $\mathcal{B}[\![\mathbf{Q'}]\!]\sigma' = tt$ and $\mathbf{Q'} \Rightarrow \mathbf{Q}$, we get $\mathcal{B}[\![\mathbf{Q}]\!]\sigma' = tt$

3. Axiomatic Semantics

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

$$\begin{array}{ll} \mathcal{B}[\![wp(s, \mathbf{Q})]\!]\sigma = tt & \Leftrightarrow \exists \sigma' : (\langle s, \sigma \rangle \rightarrow \sigma' \wedge \mathcal{B}[\![\mathbf{Q}]\!]\sigma') \\ \mathcal{B}[\![wlp(s, \mathbf{Q})]\!]\sigma = tt & \Leftrightarrow \forall \sigma' : (\langle s, \sigma \rangle \rightarrow \sigma' \Rightarrow \mathcal{B}[\![\mathbf{Q}]\!]\sigma') \end{array}$$

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

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

$$\textsc{Cons}_{Ax}\ \frac{\{ \mathit{wlp}(s, \mathbf{Q}) \}\ s\ \{ \mathbf{Q} \}}{\{ \mathbf{P} \}\ s\ \{ \mathbf{Q} \}}$$

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

Completeness Proof: Base Cases

- Case assignment:
  - From the natural semantics, we get $\langle x := e, \sigma \rangle \rightarrow \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$
  - From the definition of *wlp* and the substitution lemma, we get $\mathcal{B}[\![wlp(x := e, \mathbf{Q})]\!]\sigma \Leftrightarrow \mathcal{B}[\![\mathbf{Q}]\!]\sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma] \Leftrightarrow \mathcal{B}[\![\mathbf{Q}[x \mapsto e]]\!]\sigma$
  - Therefore, we get $wlp(x := e, \mathbf{Q}) \Leftrightarrow \mathbf{Q}[x \mapsto e]$
  - Using $\textsc{Ass}_{Ax}$ and $\textsc{Cons}_{Ax}$, we can infer $\vdash \{\ wlp(x := e, \mathbf{Q})\ \}\ x := e\ \{\ \mathbf{Q}\ \}$

- Case skip:
  - From the natural semantics, we get $wlp(\texttt{skip}, \mathbf{Q}) \Leftrightarrow \mathbf{Q}$
  - 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 $\mathcal{B}[\![wlp(s_1 ; s_2, \mathbf{Q})]\!]\sigma = tt$ for some σ and show that $\mathcal{B}[\![wlp(s_1, wlp(s_2, \mathbf{Q}))]\!]\sigma = tt$

Completeness Proof: Composition (2)

- If there is no σ' such that $\langle s_1, \sigma \rangle \rightarrow \sigma'$ then $\mathcal{B}[\![\, \mathit{wlp}(s_1, \mathit{wlp}(s_2, \mathbf{Q})) \,]\!]\sigma = \mathit{tt}$ follows immediately from the definition of *wlp*
- Otherwise, we have to show $\mathcal{B}[\![\, \mathit{wlp}(s_2, \mathbf{Q}) \,]\!]\sigma' = \mathit{tt}$
- Again, if there is no σ'' such that $\langle s_2, \sigma' \rangle \rightarrow \sigma''$ then $\mathcal{B}[\![\, \mathit{wlp}(s_2, \mathbf{Q}) \,]\!]\sigma' = \mathit{tt}$ follows immediately from the definition of *wlp*
- Otherwise, we have to show $\mathcal{B}[\![\, \mathbf{Q} \,]\!]\sigma''$
- $\mathcal{B}[\![\, \mathbf{Q} \,]\!]\sigma''$ follows from $\mathcal{B}[\![\, \mathit{wlp}(s_1 ; s_2, \mathbf{Q}) \,]\!]\sigma = \mathit{tt}$ 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 $\mathbf{P} \equiv (b \wedge wlp(s_1, \mathbf{Q})) \vee (\neg b \wedge wlp(s_2, \mathbf{Q}))$
- We have $b \wedge \mathbf{P} \Rightarrow wlp(s_1, \mathbf{Q})$ and $\neg b \wedge \mathbf{P} \Rightarrow wlp(s_2, \mathbf{Q})$
- We derive

$$\textsc{If}\ \dfrac{\textsc{Cons}\ \dfrac{\{\ wlp(s_1, \mathbf{Q})\ \}\ s_1\ \{\ \mathbf{Q}\ \}}{\{\ b \wedge \mathbf{P}\ \}\ s_1\ \{\ \mathbf{Q}\ \}} \qquad \textsc{Cons}\ \dfrac{\{\ wlp(s_2, \mathbf{Q})\ \}\ s_2\ \{\ \mathbf{Q}\ \}}{\{\ \neg b \wedge \mathbf{P}\ \}\ s_2\ \{\ \mathbf{Q}\ \}}}{\{\ \mathbf{P}\ \}\ \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{end}\ \{\ \mathbf{Q}\ \}}$$

Completeness Proof: Conditional (2)

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

Completeness Proof: Loop

- Define $\mathbf{P} \equiv wlp(\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, \mathbf{Q})$
- We will prove
  (1) $(\neg b \wedge \mathbf{P}) \Rightarrow \mathbf{Q}$
  (2) $(b \wedge \mathbf{P}) \Rightarrow wlp(s, \mathbf{P})$
- By the induction hypothesis, we get $\vdash \{\ wlp(s, \mathbf{P})\ \}\ s\ \{\ \mathbf{P}\ \}$
- From (2), we get $\vdash \{\ b \wedge \mathbf{P}\ \}\ s\ \{\ \mathbf{P}\ \}$
- By $\textsc{Wh}_{Ax}$, we get $\vdash \{\ \mathbf{P}\ \}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\ \neg b \wedge \mathbf{P}\ \}$
- From (1), we get $\vdash \{\ \mathbf{P}\ \}\ \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}\ \{\ \mathbf{Q}\ \}$

Completeness Proof: Loop (2)

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

Completeness Proof: Loop (3)

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

Completeness Proof: Loop (4)

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

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