Axiomatic semantics

- used to prove the correctness of programs
- state-based analysis

\[ state = \text{mappings from variables to values} \]

same as \textit{environment}

- statements in the program viewed as state transitions
- logical formulae (‘assertions’) to represent sets of states

\[ x == 1 \text{ represents } all \text{ states} \text{ in which the value of } x \text{ is } 1 \]
Triples

\{P\} S \{Q\}

- $P$ and $Q$ are logical formulae (assertions)

  - $P$ is the \textit{precondition}

  - $Q$ is the \textit{postcondition}

- “if the statement $S$ is executed in any state satisfying $P$, and $S$ \underline{successfully terminates}, then the resulting state satisfies $Q$”

  - \textit{partial correctness}
Examples of valid triples

\{x == 10\} x = x - 3 \{x == 7\}

\{x > 10\} x = x - 3 \{x > 7\}

\{x > 10\} x = x - 3 \{x > 2\}

\{x > 10 \land y == 2\} x = x - 3 \{x > 7\}

\{x > 10\}

while (x > 1) \{x = x + 1;\}

\{x == 0\}
Invalid triples

\[
\{x == 10\} \quad x = x - 3 \quad \{x == 8\}
\]

\[
\{x > 10\} \quad x = x - 3 \quad \{x > 10\}
\]

\[
\{x > 10\} \quad x = x - 3 \quad \{y > 2\}
\]
Axioms, Inference rules

**Axioms**: Assertions whose truth is taken at face value

**Inference Rules**: 

\[
E_1, \ldots, E_n \quad \Rightarrow \quad E
\]

“If \( E_1, \ldots, E_n \) are theorems, then so is \( E \)”

**Example**: *Modus ponens* in logic

\[
P, \quad P \Rightarrow Q \quad \Rightarrow \quad Q
\]
Implication

\[ P \Rightarrow Q \quad \text{if } P \text{ then } Q \]

\[ P \text{ only if } Q \]

<table>
<thead>
<tr>
<th>( P )</th>
<th>( Q )</th>
<th>( P \Rightarrow Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

logically equivalent to

\[ \neg P \lor Q \]
Precondition strengthening

\[
P_1 \Rightarrow P, \quad \{P\} \Downarrow \{Q\} \quad \Rightarrow \quad \{P_1\} \Downarrow \{Q\}
\]

\{x > 10\} \ x = x - 3 \ \{x > 7\} \quad \text{is valid}

So is

\{x > 11\} \ x = x - 3 \ \{x > 7\}

since \ (x > 11) \Rightarrow (x > 10)

\{\text{false}\} \Downarrow \{Q\} \quad \text{is valid for every statement} \ S \text{ and assertion} \ Q
Postcondition weakening

\[
\{P\} \ s \ \{Q\}, \quad Q \Rightarrow Q_1 \quad \frac{Q}{\{P\} \ s \ \{Q_1\}}
\]

\[
\{x > 10\} \ x = x - 3 \ \{x > 7\} \quad \text{is valid}
\]

So is

\[
\{x > 10\} \ x = x - 3 \ \{x > 6\}
\]

since \((x > 7) \Rightarrow (x > 6)\)

\[
\{P\} \ s \ \{\text{true}\} \quad \text{is valid for every statement} \ S \text{ and assertion} \ P
\]
If-then-else

\[
\begin{align*}
\{P \land C\} & \mathcal{S}_1 \{Q\}, & \{P \land \neg C\} & \mathcal{S}_2 \{Q\} \\
\{P\} & \text{if } C \{S_1\} \text{ else } \{S_2\} \{Q\}
\end{align*}
\]
If-then-else

\{
\begin{align*}
x &> y \\
\text{if} \ (x == 6) \ \{x &= x - 1;\} \ \text{else} \ \{y &= y + 1;\}
\end{align*}
\}\{x \geq y\}
If-then

\[
\begin{align*}
\{P \land C\} &\subseteq \{Q\}, \quad (P \land \neg C) \Rightarrow Q \\
\{P\} \text{ if } C &\subseteq \{S\} \subseteq \{Q\}
\end{align*}
\]

\{x > 5\} \\
if (x == 6) then \{x = x - 1\} \\
\{x > 4\}
Sequencing

\[
\{P\} \text{ } S_1 \{Q\}, \quad \{Q\} \text{ } S_2 \{R\}
\]

\[
\{P\} \text{ } S_1; \text{ } S_2 \{R\}
\]

\[
\{x > y\} \quad x = x - 1; \quad y = y - 1 \quad \{x > y\}
\]

is valid because

\[
\{x > y\} \quad x = x - 1 \quad \{x \geq y\} \quad \text{and}
\]

\[
\{x \geq y\} \quad y = y - 1 \quad \{x > y\}
\]

are valid triples
Assignment

\[ \{Q[E/x]\} \ x = E \{Q\} \]

\(Q[E/x]\) represents the result of replacing (simultaneously) all occurrences of \(x\) in \(Q\) by \(E\)

\[\{x - 1 \geq y\} \ x = x - 1 \ \{x \geq y\}\]

\[\{(y + 1)^2 \geq 10\} \ x = y + 1 \ \{x^2 \geq 10\}\]
While statements

\[
\begin{align*}
\{I \land C\} & \text{ S } \{I\} \\
\{I\} & \text{ while } (C) \{\text{S}\} \{I \land \neg C\}
\end{align*}
\]

$I$ is called the \textit{loop invariant}.
Strongest loop invariant

\textit{true} is always a loop invariant (why?)

Consider the (valid) triple

\[
\{(x == 50) \land (y == 0)\}
\]

\[
\text{while } (x > 0) \{x = x - 1; y = y + 2;\}
\]

\[
\{(x == 0) \land (y == 100)\}
\]

\(x \geq 0\) is a loop invariant, since

\[
\{x > 0\} \quad x = x - 1; \ y = y + 2 \quad \{x \geq 0\}
\]

is valid. But this is not good enough for a proof.

The required invariant \(I\) is

\[
(x \geq 0) \land (2x + y == 100)
\]
Example continued

\{(x > 0) \land (2x + y == 100)\}

\[x = x - 1;\]
\{(x \geq 0) \land (2x + y + 2 == 100)\}

\[y = y + 2;\]
\{(x \geq 0) \land (2x + y == 100)\}

There are two triples here, both of which are valid.

\[I \land \neg C\]

\[\equiv (x \geq 0) \land (2x + y == 100) \land \neg(x > 0)\]

\[\equiv (x == 0) \land (2x + y == 100)\]

\[\equiv (x == 0) \land (y == 100)\]
Example continued

The precondition

\[(x == 50) \land (y == 0)\]

clearly implies the invariant

\[(x \geq 0) \land (2x + y == 100)\]

This proves the validity of the triple.
Example continued

\[(2x + y \equiv 100)\] by itself is also an invariant, but this is not strong enough.

\[\neg(x > 0)\] does not imply \[(x \equiv 0)\]
Example: Multiplication

\{ U \geq 0 \} 

// U and V are input values
x = U;

y = 0;

while (x > 0)
    \{ x = x - 1;
        y = y + V;
    \};

\{ y == UV \}
Example: Multiplication

\{ U \geq 0 \}

// U and V are input values
x = U;

y = 0;

while (x > 0)
    \{ x = x - 1;
    y = y + V;
\} \{ x \geq 0, \ xV + y == UV \};

\{ y == UV \}
Complete Annotation

\[ \{ U \geq 0 \} \]

\[ x = U; \]

\[ \{ x == U, x \geq 0 \} \]

\[ y = 0; \]

\[ \{ x == U, x \geq 0, y == 0 \} \]

while (x > 0)

\[ \{ x = x - 1; \]  

\[ \{ x \geq 0, xV + y + V == UV \} \]

\[ y = y + V; \]

\[ } \{ x \geq 0, xV + y == UV \}; \]

\[ \{ y == UV \} \]
Proving the loop correct

We have to prove that

$$\{I \land C\} \text{ loop-body } \{I\}$$

is valid.

$$\{x \geq 0, \, xV + y == UV, \, x > 0\}$$

$$x = x - 1;$$

$$\{x \geq 0, \, xV + y + V == UV\}$$

$$y = y + V;$$

$$\{x \geq 0, \, xV + y == UV\}$$
Once we have proved

\[ \{I\} \text{ while } (C) \{S\} \{I \land \neg C\} \]

we have to prove

\[ I \land \neg C \Rightarrow (y == UV) \]

That is,

\[ (x \geq 0 \land (xV + y == UV) \land \neg(x > 0)) \Rightarrow (y == UV) \]
# Verification conditions

Lemmas to be proved to verify a completely annotated program

<table>
<thead>
<tr>
<th>Triple</th>
<th>VC</th>
</tr>
</thead>
<tbody>
<tr>
<td>{P} \ x = E \ {Q}</td>
<td>(P \Rightarrow Q[E/x])</td>
</tr>
<tr>
<td>{P} if C \ {S_1} else \ {S_2} \ {Q}</td>
<td>VCs for ({P \land C} \ S_1 \ {Q}) and ({P \land \neg C} \ S_2 \ {Q})</td>
</tr>
<tr>
<td>{P} if C \ {S} \ {Q}</td>
<td>((P \land \neg C) \Rightarrow Q) and VCs for ({P \land C} \ S \ {Q})</td>
</tr>
<tr>
<td>Triple</td>
<td>VC</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>({P} \ S_1 \ {Q} \ S_2 \ {R})</td>
<td>VCs for ({P} \ S_1 \ {Q}) and ({Q} \ S_2 \ {R})</td>
</tr>
<tr>
<td>({P} \text{ while } (C) {S} {I} \ {Q})</td>
<td>(P \Rightarrow I)</td>
</tr>
<tr>
<td></td>
<td>(I \land \neg C \Rightarrow Q)</td>
</tr>
<tr>
<td></td>
<td>VCs for ({I \land C} \ S \ {I})</td>
</tr>
</tbody>
</table>
Think of $Q$ as a set of states

$$\{ \ldots \} \mathrel{S} \{ Q \}$$

$$wlp(S, Q) = \{ \sigma \mid \text{if we execute } S \text{ in state } \sigma \text{ and the computation terminates, then the resulting state is in } Q \}$$
Weakest Liberal Preconditions

\[ wlp(x = E, Q) = Q[E/x] \]

\[ wlp(\text{if } C \text{ then } R \text{ else } S, Q) = (C \land wlp(R, Q)) \lor (\neg C \land wlp(S, Q)) \]

\[ wlp(\text{if } C \text{ then } R, Q) = (C \land wlp(R, Q)) \lor (\neg C \land Q) \]

\[ wlp(S_1; S_2, Q) = wlp(S_1, wlp(S_2, Q)) \]

If \( S \) is a loop-free (compound) statement, then the VC for

\[ \{P\} S \{Q\} \]

is

\[ P \Rightarrow wlp(S, Q) \]
\( U \geq 0 \) \hspace{1cm} (U \text{ is the input})

\[
\begin{align*}
&\text{i} = 1; \\
&\text{S} = 0; \\
&\text{while (i} \leq \text{U)} \\
&\quad \{\text{S} = \text{S} + 2*\text{i} - 1; \\
&\quad \quad \text{i} = \text{i} + 1; \\
&\quad \}\{i \leq U + 1, \ S == (i - 1)^2\} \\
&\{S == U^2\}
\end{align*}
\]
\[ \{ U \geq 0 \} \quad (U \text{ is the input}) \]

\[ i = 1; \]
\[ \{ U \geq 0, \ i == 1 \} \]

\[ S = 0; \]
\[ \{ U \geq 0, \ i == 1, \ S == 0 \} \]

while (i <= U)
\[ \{ S = S + 2*i - 1; \]
\[ \{ i \leq U, \ S == i^2 \} \]

\[ i = i + 1; \]
\[ \} \quad \{ i \leq U + 1, \ S == (i - 1)^2 \} \]

\[ \{ S == U^2 \} \]
VCs for the loop

\[(U \geq 0 \land i == 1 \land S == 0) \Rightarrow (i \leq U + 1 \land S == (i - 1)^2)\]

\[(i \leq U + 1 \land S == (i - 1)^2 \land \neg i \leq U) \Rightarrow S == U^2\]

and VCs for

\[\{i \leq U + 1, S == (i - 1)^2, i \leq U\}\]

\[S = S + 2*i - 1;\]

\[i = i + 1;\]

\[\{i \leq U + 1, S == (i - 1)^2\}\]
Another squaring program

\[
\{ U \geq 0 \} \quad (U \text{ is the input})
\]

\[
x = U;
S = 0;
\text{while } (x \geq 1)
\{\]
S = S + 2*x - 1;
x = x - 1;
\}
\{ x \geq 0, x^2 + S == U^2 \}\;
\]

\[
\{ S == U^2 \}
\]