Axiomatic semantics

- used to prove the correctness of programs
- state-based analysis
  
  \[ \text{state} = \text{mappings from variables to values same as environment} \]

- statements in the program viewed as state transitions

- logical formulae (‘assertions’) to represent sets of states

  \[ x == 1 \] represents all states in which the value of \( x \) is 1
Triples

\{P\} \mathbin{\stackrel{S}{\longrightarrow}} \{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 \}
  \text{while } (x > 1) \ \{ x = x + 1; \}
\{ x == 0 \}
Examples of invalid triples

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

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

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

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

**Inference Rules**:

\[
\frac{E_1, \ldots, E_n}{E}
\]

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

**Example**: Modus ponens in logic

\[
\frac{P, \ P \Rightarrow Q}{Q}
\]
Implication

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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

\[
\frac{P_1 \Rightarrow P, \{P\} \mathcal{S} \{Q\}}{\{P_1\} \mathcal{S} \{Q\}}
\]

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

So is

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

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

\[
\{\text{false}\} \mathcal{S} \{Q\} \text{ is valid for every statement } \mathcal{S}
\]

and assertion \(Q\)
Postcondition weakening

$$\{P\} \subseteq \{Q\}, \quad Q \Rightarrow Q_1 \quad \Rightarrow \quad \{P\} \subseteq \{Q_1\}$$

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

So is

$$\{x > 10\} \ x = x - 3 \ {\{x > 6\}}$$

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

$$\{P\} \subseteq \{\text{true}\} \quad \text{is valid for every statement S}$$

and assertion $P$
If-then-else

\[
\begin{align*}
\{ P \land C \} & \quad S_1 \quad \{ Q \}, \\
\{ P \land \neg C \} & \quad S_2 \quad \{ Q \}
\end{align*}
\]

\[
\{ P \} \text{ if } C \quad \{ S_1 \} \quad \text{else} \quad \{ S_2 \} \quad \{ Q \}
\]

\[
\{ x > y \} \\
\quad \text{if } (x == 6) \quad \{ x = x - 1; \} \quad \text{else} \quad \{ y = y + 1; \}
\]

\[
\{ x \geq y \}
\]
If-then

\[ \{P \land C\} \subseteq \{Q\}, \quad (P \land \neg C) \Rightarrow Q \]
\[ \{P\} \text{ if } C \{S\} \{Q\} \]

\( \{x > 5\} \)

if \((x == 6)\) then \(\{x = x - 1\}\)

\(\{x > 4\}\)
Sequencing

\[
\begin{align*}
\{P\} \quad & S_1 \quad \{Q\}, \\
\{Q\} \quad & S_2 \quad \{R\} \\
\{P\} \quad & S_1; \quad S_2 \quad \{R\}
\end{align*}
\]

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

is valid because

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

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

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

$I$ is called the *loop invariant*. 
Strongest loop invariant

true is always a loop invariant (why?)

Consider the (valid) triple

\[(x == 50) \land (y == 0)\]
while \((x > 0)\) \{x = x - 1; y = y + 2;\}
\[(x == 0) \land (y == 100)\]

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

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

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.

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

\[-(x > 0)\] does not imply \[(x == 0)\]
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>VCs for ({P \land C} S {Q}) and ((P \land \neg C) \Rightarrow Q)</td>
</tr>
<tr>
<td>Triple</td>
<td>VC</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>{P} S₁ {Q} S₂ {R}</td>
<td>VCs for</td>
</tr>
<tr>
<td></td>
<td>{P} S₁ {Q} and</td>
</tr>
<tr>
<td></td>
<td>{Q} S₂ {R}</td>
</tr>
<tr>
<td>{P} 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</td>
</tr>
<tr>
<td></td>
<td>{I \land C} S {I}</td>
</tr>
</tbody>
</table>
Weakest Liberal Preconditions

\[ \{ \ldots \} \models \{ Q \} \]

Think of \( Q \) as a set of states

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

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

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

\begin{align*}
x &= U; \\
S &= 0; \\
\text{while } (x \geq 1) & \\
\{S &= S + 2x - 1; \\
x &= x - 1; \\
\}
\end{align*}

\{S == U^2\}
\{ U \geq 0 \} \quad (U \text{ is the input})

\begin{align*}
&x = U; \\
&S = 0; \\
&\text{while } (x \geq 1) \\
&\quad \{ S = S + 2*x - 1; \\
&\quad \quad x = x - 1; \\
&\quad \} \quad \{ x \geq 0, \quad x^2 + S = U^2 \} \\
\end{align*}

\{ S = U^2 \}