Axiomatic Semantics

- Also called Floyd-Hoare Logic
- Based on formal logic (first order predicate calculus)
- Axiomatic Semantics is a logical system built from axioms and inference rules
- Mainly suited to simple imperative programming languages

Axiomatic Semantics

- Used to formally prove a property (post-condition) of the state (the values of the program variables) after the execution of program, assuming another property (pre-condition) of the state holds before execution

Axiomatic Semantics

- Goal: Derive statements of form \{P\} C \{Q\}
- P, Q logical statements about state, P precondition, Q postcondition, C program
- Example: \{x = 1\} x := x + 1 \{x = 2\}

Axiomatic Semantics

- An expression \{P\} C \{Q\} is a partial correctness statement
- For total correctness must also prove that C terminates (i.e. doesn’t run forever)
- Written: [P] C [Q]
- Will only consider partial correctness here
We will give rules for simple imperative language

\[
\text{<command> ::= <variable> := <term> | <command>; ... ;<command> | if <statement> then <command> else <command> | while <statement> do <command>}
\]

Could add more features, like for-loops

---

**Substitution**

- **Notation:** \( P[e/v] \) (sometimes \( P[v <- e] \))
- **Meaning:** Replace every \( v \) in \( P \) by \( e \)
- **Example:**
  \[
  (x + 2) [y - 1/x] = ((y - 1) + 2)
  \]

---

**The Assignment Rule**

\[
\{P [e/x] \} x := e \{P\}
\]

**Example:**

\[
\{ \ ? \} x := y \{x = 2\}
\]

---

**The Assignment Rule**

\[
\{P [e/x] \} x := e \{P\}
\]

**Example:**

\[
\{\boxed{2}\} x := y \{\boxed{x} = 2\}
\]

---

**The Assignment Rule**

\[
\{P [e/x] \} x := e \{P\}
\]

**Examples:**

\[
\begin{align*}
\{y = 2\} x := y \{x = 2\} \\
\{y = 2\} x := 2 \{y = x\} \\
\{x + 1 = n + 1\} x := x + 1 \{x = n + 1\} \\
\{2 = 2\} x := 2 \{x = 2\}
\end{align*}
\]
The Assignment Rule – Your Turn

What is the weakest precondition of 
\( x := x + y \) \( \{ x + y = w - x \} \)?

\[
\begin{align*}
\{ & \quad ? \quad \} \\
& x := x + y \quad \{ x + y = w - x \}
\end{align*}
\]

Precondition Strengthening

\( P \rightarrow P' \quad \{ P' \} \subseteq \{ Q \} \)

Meaning: If we can show that \( P \) implies \( P' \) (\( P \rightarrow P' \)) and we can show that \( \{ P' \} \subseteq \{ Q \} \), then we know that \( \{ P \} \subseteq \{ Q \} \). 

\( P \) is stronger than \( P' \) means \( P \rightarrow P' \).

Which Inferences Are Correct?

\[
\begin{align*}
\{ x > 0 \land x < 5 \} \\
\{ x := x \times x \} \\
\{ x < 25 \}
\end{align*}
\]

\[
\begin{align*}
\{ x := x \times x \} & \quad \{ x < 25 \} \\
\{ x = 3 \} & \quad \{ x := x \times x \} \\
\{ x < 25 \} & \quad \{ x > 0 \land x < 5 \} \\
\{ x := x \times x \} & \quad \{ x < 25 \}
\end{align*}
\]

\[
\begin{align*}
\{ x := x \times x \} & \quad \{ x < 25 \} \\
\{ x := x \times x \} & \quad \{ x < 25 \} \\
\{ x := x \times x \} & \quad \{ x < 25 \}
\end{align*}
\]
Sequencing

\{P\} C_1 \{Q\} \quad \{Q\} C_2 \{R\}
\{P\} C_1; C_2 \{R\}

Example:
\{z = z & z = z\} x := z \{x = z & z = z\}
\{x = z & z = z\} y := z \{x = z & y = z\}
\{z = z & z = z\} x := z; y := z \{x = z & y = z\}

Sequencing

\{P\} C_1 \{Q\} \quad \{Q\} C_2 \{R\}
\{P\} C_1; C_2 \{R\}

Example:
\{z = z & z = z\} x := z \{x = z & z = z\}
\{x = z & z = z\} y := z \{x = z & y = z\}
\{z = z & z = z\} x := z; y := z \{x = z & y = z\}

Postcondition Weakening

\{P\} C \{Q'\} \quad Q' \Rightarrow Q
\{P\} C \{Q\}

Example:
\{z = z & z = z\} x := z; y := z \{x = z & y = z\}
(x = z & y = z) \Rightarrow (x = y)
\{z = z & z = z\} x := z; y := z \{x = y\}

Rule of Consequence

\textit{P} \Rightarrow \textit{P}' \quad \{P'\} C \{Q'\} \quad Q' \Rightarrow Q
\{P\} C \{Q\}

Logically equivalent to the combination of
Precondition Strengthening and
Postcondition Weakening

Uses \textit{P} \Rightarrow \textit{P} and \textit{Q} \Rightarrow \textit{Q}

If Then Else

\{P \text{ and } B\} C_1 \{Q\} \quad \{P \text{ and } \neg B\} C_2 \{Q\}
\{P\} \text{ if } B \text{ then } C_1 \text{ else } C_2 \{Q\}

Example: Want
\{y=a\}
if \textit{x} < 0 then \textit{y} := \textit{y} - \textit{x} else \textit{y} := \textit{y} + \textit{x}
\{y=a+|x|\}

Suffices to show:
(1) \{y=a&x<0\} \textit{y} := \textit{y} - \textit{x} \quad \{y=a+|x|\}
(2) \{y-x=a+|x|\} \textit{y} := \textit{y} - \textit{x} \quad \{y=a+|x|\}
(3) \{y=a&x<0\} \textit{y} := \textit{y} - \textit{x} \quad \{y=a+|x|\}
(4) \{y=a\&\text{not}(x<0)\} \textit{y} := \textit{y} + \textit{x} \quad \{y=a+|x|\}

(1) Reduces to (2) and (3) by
Precondition Strengthening
(2) Follows from assignment axiom
(3) Because \textit{x}<0 \Rightarrow |x| = -\textit{x}
\{y = a \& \neg (x < 0)\} \ y := y + x \ \{y = a + |x|\}

(6) \ (y = a \& \neg (x < 0)) \rightarrow (y + x = a + |x|)
(5) \ {y + x = a + |x|} \ y := y + x \ \{y = a + |x|\}
(4) \ {y = a \& \neg (x < 0)} \ y := y + x \ \{y = a + |x|\}

(4) Reduces to (5) and (6) by Precondition Strengthening
(5) Follows from assignment axiom
(6) Because not(x < 0) \rightarrow |x| = x

If then else

(1) \ {y = a \& x < 0} y := y - x \{y = a + |x|\}
(4) \ {y = a \& \neg (x < 0)} y := y + x \{y = a + |x|\}
\begin{align*}
&\{y = a\} \\
&\text{if } x < 0 \text{ then } y := y - x \text{ else } y := y + x \\
&\{y = a + |x|\}
\end{align*}

By the if_then_else rule

While

We need a rule to be able to make assertions about while loops.
- Inference rule because we can only draw conclusions if we know something about the body
- Let's start with:
\[
\begin{array}{c}
\{ ? \} \\
\{ ? \} \\
\{ ? \} \\
\{ ? \}
\end{array}
\begin{array}{c}
C \\
while \ B \\
do \\
C
\end{array}
\{ ? \}
\{ P \}

While

We can strengthen the previous rule because we also know that when the loop is finished, \not B also holds
- Final while rule:
\[
\begin{array}{c}
\{ P \text{ and } B \} \\
\{ P \}
\end{array}
\begin{array}{c}
C \\
\{ P \}
\end{array}
\{ P \}
\begin{array}{c}
\text{while } B \\
do \\
C
\end{array}
\{ P \text{ and } \not B \}

While

The loop may never be executed, so if we want P to hold after, it had better hold before, so let's try:
\[
\begin{array}{c}
\{ ? \} \\
\{ P \}
\end{array}
\begin{array}{c}
C \\
\{ ? \}
\end{array}
\{ P \}
\begin{array}{c}
while \ B \\
do \\
C
\end{array}
\{ P \}
\]
While

\[
{ P and B } \quad C \quad { P } \\
{ P } \quad \textbf{while} \quad B \quad \textbf{do} \quad C \quad { P and not B } \\
\]

- P satisfying this rule is called a loop invariant because it must hold before and after the each iteration of the loop.

While rule generally needs to be used together with precondition strengthening and postcondition weakening.
- There is NO algorithm for computing the correct P; it requires intuition and an understanding of why the program works.

Example
- Let us prove \( \{x \geq 0 \text{ and } x = a\} \)
  
  \[
  \text{fact} := 1; \\
  \text{while } x > 0 \text{ do (fact := fact } \times x; x := x - 1) \\
  \{\text{fact} = a!\}
  \]

Example
- We need to find a condition P that is true both before and after the loop is executed, and such that
  
  \[ (P \text{ and not } x > 0) \rightarrow (\text{fact} = a!) \]

Example
- First attempt:
  
  \( \{a! = \text{fact } \times (x!}\} \)

  - Motivation:
  - What we want to compute: a!
  - What we have computed: fact
  - which is the sequential product of a down through \((x + 1)\)
  - What we still need to compute: x!

Example
- By post-condition weakening suffices to show
  1. \( \{x\geq 0 \text{ and } x = a\} \)
     
     \[
     \text{fact} := 1; \\
     \text{while } x > 0 \text{ do (fact := fact } \times x; x := x - 1) \\
     \{a! = \text{fact } \times (x!) \text{ and not } (x > 0)\}
     \]
     
     and
  2. \( \{a! = \text{fact } \times (x!) \text{ and not } (x > 0)\} \rightarrow \{\text{fact} = a!\} \)
Problem

2. \{a! = fact * (x!) and not (x > 0)\} \Rightarrow \{fact = a!\}
   - Don’t know this if x < 0
   - Need to know that x = 0 when loop terminates
   - Need a new loop invariant
   - Try adding x >= 0
   - Then will have x = 0 when loop is done

Example

Second try, combine the two:
\[ P = \{a! = fact * (x!) and x >=0\} \]
Again, suffices to show
1. \{x>=0 and x = a\}
   - fact := 1;
   - while x > 0 do (fact := fact * x; x := x –1)
   - \{P and not x > 0\}
   and
2. \{P and not x > 0\} \Rightarrow \{fact = a!\}

Example

For 2, we need
\{a! = fact * (x!) and x >=0 and not (x > 0)\} \Rightarrow \{fact = a!\}
But \{x >=0 and not (x > 0)\} \Rightarrow \{x = 0\} so
\[ fact * (x!) = fact * (0!) = fact \]
Therefore
\{a! = fact * (x!) and x >=0 and not (x > 0)\} \Rightarrow \{fact = a!\}

Example

For 1, by the sequencing rule it suffices to show
3. \{x>=0 and x = a\}
   - fact := 1
   \{a! = fact * (x!) and x >=0\}
   And
4. \{a! = fact * (x!) and x >=0\}
   - while x > 0 do (fact := fact * x; x := x –1)
   \{a! = fact * (x!) and x >=0 and not (x > 0)\}

Example

By the assignment rule, we have
\{a! = 1 * (x!) and x >= 0\}
   - fact := 1
   \{a! = fact * (x!) and x >= 0\}
Therefore, to show (3), by precondition strengthening, it suffices to show
\(x>= 0 and x = a\) \Rightarrow \(a! = 1 * (x!) and x >= 0\)
Example

(x >= 0 and x = a) \Rightarrow
(a! = 1 * (x!) and x >= 0)
holds because x = a \Rightarrow x! = a!

Have that \{a! = fact * (x!) and x >= 0\}
holds at the start of the while loop

Example

To show (4):
\{a! = fact * (x!) and x >=0\}
while x > 0 do
(fact := fact * x; x := x – 1)
\{a! = fact * (x!) and x >=0 and not (x > 0)\}
we need to show that
\{a! = fact * (x!) and x >= 0\}
is a loop invariant

Example

We need to show:
\{(a! = fact * (x!)) and x >= 0 and x > 0\}
( fact = fact * x; x := x – 1 )
\{(a! = fact * (x!)) and x >= 0\}

We will use assignment rule,
sequencing rule and precondition strengthening

Example

By the assignment rule, we have
\{(a! = fact * ((x-1)!)) and x – 1 >= 0\}
fact = fact * x
\{(a! = fact * ((x-1)!)) and x – 1 >= 0\}

By Precondition strengthening, it suffices
to show that
((a! = fact * (x!)) and x >= 0 and x > 0) \Rightarrow
((a! = (fact * x) * ((x-1)!)) and x – 1 >= 0)

Example

By the assignment rule, we have that
\{(a! = (fact * x) * ((x-1)!)) and x – 1 >= 0\}
fact = fact * x
\{(a! = fact * ((x-1)!)) and x – 1 >= 0\}

By Precondition strengthening, it suffices
to show that
((a! = fact * (x!)) and x >= 0 and x > 0) \Rightarrow
((a! = (fact * x) * ((x-1)!)) and x – 1 >= 0)

Example

However
fact * x * (x – 1)! = fact * x
and
(x > 0) \Rightarrow x – 1 >= 0
since x is an integer, so
\{(a! = fact * (x!)) and x >= 0 and x > 0\} \Rightarrow
\{(a! = (fact * x) * ((x-1)!)) and x – 1 >= 0\}
Example

Therefore, by precondition strengthening
\{(a! = \text{fact} \ast (x!)) \text{ and } x \geq 0 \text{ and } x > 0\}
\text{fact} = \text{fact} \ast x
\{(a! = \text{fact} \ast ((x-1)!)) \text{ and } x - 1 \geq 0\}

This finishes the proof