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

- Approach: For each type of language statement, give an axiom or inference rule stating how to derive assertions of form \( \{P\} C \{Q\} \)
  - where \( C \) is a statement of that type
- Compose axioms and inference rules to build proofs for complex programs

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
Language

- We will give rules for simple imperative language

<command>::= <variable> := <term>
|  <command>; ... ;<command>
|  if <statement> then <command> else <command> fi
|  while <statement> do <command> od

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

Example:
\[ \{ y = 2 \} \ x := y \ \{ x = 2 \} \]
\[ \{ y = 2 \} \ x := 2 \ \{ y = x \} \]
\[ \{ x + 1 = n + 1 \} \ x := x + 1 \ \{ x = n + 1 \} \]
\[ \{ 2 = 2 \} \ x := 2 \ \{ x = 2 \} \]
The Assignment Rule – Your Turn

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

\[
\begin{cases}
\text{?} & \\
x := x + y & \\
\{ x + y = w - x \} & 
\end{cases}
\]

Precondition Strengthening

\[ P \Rightarrow P' \{ P' \} \subseteq \{ Q \} \]

- 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?

- \( \{ x > 0 \& x < 5 \} x := x * x \{ x < 25 \} \)
- \( \{ x = 3 \} x := x * x \{ x < 25 \} \)
- \( \{ x > 0 \& x < 5 \} x := x * x \{ x < 25 \} \)
- \( \{ x > 0 \& x < 5 \} x := x * x \{ x < 25 \} \)
- \( \{ x > 0 \& x < 5 \} x := x * x \{ x < 25 \} \)
- \( \{ x * x < 25 \} x := x * x \{ x < 25 \} \)
- \( \{ x = 3 \} x := x * x \{ x < 25 \} \)
- \( \{ x > 0 \& x < 5 \} x := x * x \{ x < 25 \} \)
- \( \{ x = 3 \} x := x * x \{ x < 25 \} \)
- \( \{ x * x < 25 \} x := x * x \{ x < 25 \} \)
Sequencing

\[
\{P\} C_1 \{Q\} \{Q\} C_2 \{R\} \\
\{P\} C_1; C_2 \{R\}
\]

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

\[
\{P\} C \{Q'\} Q' \Rightarrow Q \\
\{P\} C \{Q\}
\]

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

Rule of Consequence

\[
P \Rightarrow P' \{P'\} C \{Q'\} Q' \Rightarrow Q \\
\{P\} C \{Q\}
\]

Logically equivalent to the combination of Precondition Strengthening and Postcondition Weakening

Uses \(P \Rightarrow P'\) and \(Q' \Rightarrow Q\)

If Then Else

\[
\{P \text{ and } B\} C_1 \{Q\} \{P \text{ and } \neg B\} C_2 \{Q\} \\
\{P\} \text{ if } B \text{ then } C_1 \text{ else } C_2 \text{ fi } \{Q\}
\]

Example: Want
\[
\{y=a\} \text{ if } x < 0 \text{ then } y := y-x \text{ else } y := y+x \text{ fi} \\
\{y=a+|x|\}
\]

Suffices to show:
\[
(1) \{y=a\&x<0\} y := y-x \{y=a+|x|\} \\
(2) \{y-x=a+|x|\} y := y-x \{y=a+|x|\} \\
(3) \{y=a+x<0\} y := y-x \{y=a+|x|\}
\]

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

(6) \( (y = a \& \lnot (x < 0)) \rightarrow (y + x = a + |x|) \)

(5) \( \{ y + x = a + |x| \} \; y := y + x \; \{ y = a + |x| \} \)

(4) \( \{ y = a \& \lnot (x < 0) \} \; y := y + x \; \{ y = a + |x| \} \)

(4) Reduces to (5) and (6) by Precondition Strengthening

(5) Follows from assignment axiom

(6) Because \( \lnot (x < 0) \rightarrow |x| = x \)

---

**If then else**

(1) \( \{ y = a \& x < 0 \} \; y := y - x \; \{ y = a + |x| \} \)

(4) \( \{ y = a \& \lnot (x < 0) \} \; y := y + x \; \{ y = a + |x| \} \)

\[ \text{if } x < 0 \text{ then } y := y - x \text{ else } y := y + x \]

\[ \{ y = a + |x| \} \]

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:

\[
\begin{array}{c}
\{ ? \} \quad C \quad \{ ? \} \\
\{ ? \} \quad \text{while } B \text{ do } C \text{ od } \quad \{ P \}
\end{array}
\]

---

We can strengthen the previous rule because we also know that when the loop is finished, \( \lnot B \) also holds

Final while rule:

\[
\begin{array}{c}
\{ P \text{ and } B \} \quad C \quad \{ P \} \\
\{ P \} \quad \text{while } B \text{ do } C \text{ od } \quad \{ P \text{ and } \lnot B \}
\end{array}
\]

---

If all we know is \( P \) when we enter the while loop, then we all we know when we enter the body is \( (P \text{ and } B) \)

If we need to know \( P \) when we finish the while loop, we had better know it when we finish the loop body:

\[
\begin{array}{c}
\{ P \text{ and } B \} \quad C \quad \{ P \} \\
\{ P \} \quad \text{while } B \text{ do } C \text{ od } \quad \{ P \}
\end{array}
\]
Example
Let us prove \( \{x \geq 0 \text{ and } x = a\} \)

\[ \text{fact} := 1; \]
\[ \text{while } x > 0 \text{ do } (\text{fact} := \text{fact } \times x; x := x - 1) \text{ od } \]
\( \{\text{fact} = a!\} \)

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: \( \text{fact} \)

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. \( \{x \geq 0 \text{ and } x = a\} \)

\[ \text{fact} := 1; \]
\[ \text{while } x > 0 \text{ do } (\text{fact} := \text{fact } \times x; x := x - 1) \text{ od } \]
\( \{a! = \text{fact } \times (x!) \text{ and not } (x > 0)\} \)

and
2. \( \{a! = \text{fact } \times (x!) \text{ and not } (x > 0)\} \Rightarrow \{\text{fact} = a!\} \)
Problem

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

Example

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

Example

For 2, we need
\( \{ a! = \text{fact} \times (x!) \text{ and } x \geq 0 \text{ and not } (x > 0) \} \Rightarrow \{ \text{fact} = a! \} \)
But \( \{ x \geq 0 \text{ and not } (x > 0) \} \Rightarrow \{ x = 0 \} \) so
\( \text{fact} \times (x!) = \text{fact} \times (0!) = \text{fact} \)
Therefore
\( \{ a! = \text{fact} \times (x!) \text{ and } x \geq 0 \text{ and not } (x > 0) \} \Rightarrow \{ \text{fact} = a! \} \)

Example

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

Example

Suffices to show that
\( \{ a! = \text{fact} \times (x!) \text{ and } x \geq 0 \} \)
holds before the while loop is entered and that if
\( \{ a! = \text{fact} \times (x!) \text{ and } x \geq 0 \text{ and } x > 0 \} \)
holds before we execute the body of the loop, then
\( \{ a! = \text{fact} \times (x!) \text{ and } x \geq 0 \} \)
holds after we execute the body

Example

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

\(x \geq 0 \text{ and } x = a \Rightarrow \\
(a! = 1 \times (x!) \text{ and } x \geq 0)
\) holds because \(x = a \Rightarrow x! = a!\)

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

\(11/29/22\)
\(43\)

Example

\(11/29/22\)
\(44\)

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

Example

\(11/29/22\)
\(45\)

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

We will use assignment rule,
sequencing rule and precondition strengthening

\(11/29/22\)
\(46\)

Example

\(11/29/22\)
\(47\)

By the assignment rule, we have
\(\{(a! = (fact \times x) \times ((x - 1)!) \text{ and } x - 1 \geq 0\}\)
\(\text{fact } = \text{fact } \times x\)
\(\{(a! = fact \times ((x - 1)!) \text{ and } x - 1 \geq 0\}\}

By Precondition strengthening, it suffices to show that
\(\{(a! = fact \times (x!)) \text{ and } x \geq 0 \text{ and } x > 0\} \Rightarrow \\
\{(a! = (fact \times x) \times ((x - 1)!) \text{ and } x - 1 \geq 0\}\}

\(11/29/22\)
\(48\)

Example

\(11/29/22\)
\(47\)

By the assignment rule, we have that
\(\{(a! = (fact \times x) \times ((x - 1)!) \text{ and } x - 1 \geq 0\}\)
\(\text{fact } = \text{fact } \times x\)
\(\{(a! = fact \times ((x - 1)!) \text{ and } x - 1 \geq 0\}\}

By Precondition strengthening, it suffices to show that
\(\{(a! = fact \times (x!)) \text{ and } x \geq 0 \text{ and } x > 0\} \Rightarrow \\
\{(a! = (fact \times x) \times ((x - 1)!) \text{ and } x - 1 \geq 0\}\}

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

Therefore, by precondition strengthening

\{(a! = \text{fact} * (x!)) \text{ and } x >= 0 \text{ and } x > 0\}

\text{fact} = \text{fact} * x

\{(a! = \text{fact} * ((x-1)!)) \text{ and } x – 1 >= 0\}

This finishes the proof