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

- 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 kind 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 kind

- Compose axioms and inference rules to build proofs for complex programs

- 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 <expression> then <command> else <command> fi  
  |  while <expression> 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:

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

The Assignment Rule

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

Example:

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

The Assignment Rule

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

Examples:

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

What is a valid precondition of

\[ x := x + y \{ x + y = w - x \} \]?

\[
\{ \quad ? \quad \}
\]

\[ x := x + y \]

\[ \{ x + y = w - x \} \]

Precondition Strengthening

\[
P \Rightarrow P' \quad \{P'\} \subset \{Q\}
\]

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

\( P \) is stronger than \( P' \) means \( P \Rightarrow P' \)

Which Inferences Are Correct?

\[
\{ x > 0 \land x < 5 \} \quad x := x \times x \{ x < 25 \}
\]

\[
\{ x = 3 \} \quad x := x \times x \{ x < 25 \}
\]

\[
\{ x > 0 \land x < 5 \} \quad x := x \times x \{ x < 25 \}
\]

\[
\{ x \times x < 25 \} \quad x := x \times x \{ x < 25 \}
\]

\[
\{ x > 0 \land x < 5 \} \quad x := x \times x \{ x < 25 \}
\]

\[
\{ x \times x < 25 \} \quad x := x \times x \{ x < 25 \}
\]

\[
\{ x > 0 \land x < 5 \} \quad x := x \times x \{ x < 25 \}
\]
Sequencing

\[
\begin{align*}
(P) \quad C_1 (Q) & \quad (Q) \quad C_2 (R) \\
(P) \quad C_1 ; \quad C_2 (R)
\end{align*}
\]

- Example:
  \[
  \begin{align*}
  \{z = z & z = z\} & \quad x := z \quad \{x = z & z = z\} \\
  \{x = z & z = z\} & \quad y := z \quad \{x = z & y = z\} \\
  \{z = z & z = z\} & \quad x := z \quad ; \quad y := z \quad \{x = z & y = z\}
  \end{align*}
  \]

Postcondition Weakening

\[
\begin{align*}
(P) \quad C (Q') & \quad \quad Q' \quad \Rightarrow \quad Q \\
(P) \quad C (Q)
\end{align*}
\]

Example:

\[
\begin{align*}
\{z = z & z = z\} & \quad x := z \quad \{x = z & y = z\} \\
\{x = z & y = z\} & \quad y := z \quad \{x = z & y = z\} \\
\{z = z & z = z\} & \quad x := z \quad ; \quad y := z \quad \{x = z & y = z\}
\end{align*}
\]

Rule of Consequence

\[
\begin{align*}
P \quad \Rightarrow \quad P' \quad \quad \{P'\} \quad C (Q') & \quad \quad Q' \quad \Rightarrow \quad Q \\
(P) \quad C (Q)
\end{align*}
\]

- Logically equivalent to the combination of Precondition Strengthening and Postcondition Weakening
- Uses \(P \quad \Rightarrow \quad P'\) and \(Q' \quad \Rightarrow \quad Q\)

If Then Else

\[
\begin{align*}
(P \quad \text{and} \quad B) \quad C_1 (Q) & \quad \quad (P \quad \text{and} \quad \text{not} \quad B)) \quad C_2 (Q) \\
(P) \quad \text{if} \quad B \quad \text{then} \quad C_1 \quad \text{else} \quad C_2 \quad \text{fi} (Q)
\end{align*}
\]

- Example: Want \(y = a\)
  
  if \(x < 0\) then \(y := y - x\) else \(y := y + x\) fi

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

Suffices to show:

1. \(\{y = a \quad \text{and} \quad x < 0\} \quad y := y - x \quad \{y = a + |x|\}\)
2. \(\{y - x = a + |x|\}\)
3. \(\{y = a + |x|\}\)

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

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

If Then Else

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

\begin{align*}
\text{if } x < 0 \text{ then } y := y \cdot x \text{ else } y := y + x \\
\text{if } x < 0 \text{ then } y := y + x
\end{align*}

By the **IfThenElse 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:

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

While

- The loop may never be executed, so if we want \(P\) to hold after, it had better hold before, so let’s try:

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

While

- If all we know is \(P\) when we enter the **while** loop, then we all we know when we enter the body is \((P \land B)\)
- If we need to know \(P\) when we finish the **while** loop, we had better know it when we finish the loop body:

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

While

- We can strengthen the previous rule because we also know that when the loop is finished, \(\neg B\) also holds
- Final **while** rule:

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

\{(P \land B) \land C \land \{P\}\}\ \textbf{while} \ B \ \textbf{do} \ C \ \textbf{od} \ \{P \land \lnot B\}

- \(P\) satisfying this rule is called a \textit{loop invariant} because it must hold before and after each iteration of the loop

\textbf{Counting up to n}

\[
x := 0; \quad \text{while } (x < n) \{ \quad P \equiv x \leq n \land 0 \leq n \quad \}
\]

\[
x := x + 1
\]

Want to show: \(x \geq n \land n \geq 0\)

\textbf{Sum of numbers 1 to n}

\[
x := 0 \quad \text{while } y < n \{ \quad P \equiv x = 1 + \ldots + y \quad \land \ y \leq n \quad \land \ 0 \leq n \quad \}
\]

\[
x := x + y \
\]

Want to show: \(x = 1 + \ldots + n\)

\textbf{Fibonacci}

\[
x = 0; \quad y = 1;
\]

\[
z = 1;
\]

\[
\text{while } (z < n) \{ \quad P \equiv y = \text{fib } z \quad \land \ x = \text{fib } (z-1) \quad \land \ z \leq n \quad \land \ 1 \leq n \quad \}
\]

Want to show: \(y = \text{fib } n\)

\textbf{List Length}

\[
x = \text{lst}; \quad y = 0
\]

\[
\text{while } (x \neq []) \{ \quad P \equiv y + \text{len } x = \text{len } \text{lst} \quad \land \ y \leq n \quad \land \ 0 \leq n \quad \}
\]

\[
x := \text{tl } x; \
y := y + 1
\]

Want to show: \(y = \text{len } \text{lst}\)
Example (Use of Loop Invariant in Full Proof)

Let us prove
\{x \geq 0 \text{ and } x = a\}

\begin{verbatim}
    fact := 1;
    while x > 0 do (fact := fact * x; x := x - 1) od
\end{verbatim}
\{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:
\(P = \{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: \(x!\)

Problem!! (Dead End)

2. \(a! = \text{fact} \times (x!)\) 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

By post-condition weakening suffices to show
1. \(\{x \geq 0 \text{ and } x = a\}
\begin{verbatim}
    fact := 1;
    while x > 0 do (fact := fact * x; x := x - 1) od
\end{verbatim}
\{a! = \text{fact} \times (x!) \text{ and not } (x > 0)\}

And

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

Example

Second try, let us combine the two:
\(P = a! = \text{fact} \times (x!)\) and \(x \geq 0\)

We need to show:
1. \(\{x \geq 0 \text{ and } x = a\}
\begin{verbatim}
    fact := 1;
    (P)
    while x > 0 do (fact := fact * x; x := x - 1) od
\end{verbatim}
\{P \text{ and not } x > 0\}

And

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

{(x>=0 and x = a) (*this was part 1 to prove*)

  fact := 1;
  while x > 0 do (fact := fact * x; x := x -1) od
  {a! = fact * (x!) and x >=0 and not (x>0)}

} For Part 1, by sequencing rule it suffices to show

3. {x>=0 and x = a}
   {a! = fact * (x!) and x >=0 }

And

4. {a! = fact * (x!) and x >=0}
   while x > 0 do
       (fact := fact * x; x := x -1) od
   {a! = fact * (x!) and x >=0 and not (x > 0)}

Example

(Part 3 – Assignment) Suffices to show that

a! = fact * (x!) and x >= 0

holds before the while loop is entered

(Part 4 – While Loop) And that if

(a! = fact * (x!)) and x >= 0 and x > 0

holds before we execute the body of the loop, then

(a! = fact * (x!)) and x >= 0

holds after we execute the body (part 4)

Example

(Part 3) By the assignment rule, we have

{a! = fact * (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) => (a! = 1 * (x!) and x >= 0)

It holds because x = a => x! = a!.

So, we have that a! = fact * (x!) and x >= 0
holds at the start of the while loop!

Example

To prove (Part 4):

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

while x > 0 do
    (fact := fact * x; x := x -1) od
{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

We will use assignment rule, sequencing rule and
precondition strengthening rule

Example

We look into the loop body:

• (fact := fact * x; x := x - 1)

By the sequencing rule, we need to show 2 things:

• By the assignment rule, show

((a! = fact * (x!)) and x >= 0 and x > 0)

• By the assignment rule, show

(Q)

• By the assignment rule, show

(Q)

• By the assignment rule, show

(Q)

x := x - 1

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

Example

We look into the loop body:

• (Fact := fact * x; x := x - 1)

By the sequencing rule, we need to show 2 things:

• By the assignment rule, show

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

• By the assignment rule, show

{Q}

• From the assignment rule, we know:

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

x := x - 1

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

We look into the loop body:
- \((\text{fact} := \text{fact} \times x; x := x - 1)\)

By the sequencing rule, we need to show 2 things:
- By the assignment rule, show
\[\{(a! = \text{fact} \times (x!)) \text{ and } x \geq 0 \text{ and } x > 0\} \Rightarrow \{\text{fact} = \text{fact} \times x\}\]

\[\{(a! = \text{fact} \times ((x-1)!)) \text{ and } x - 1 \geq 0\}\]
- From the assignment rule, we know:
\[\{(a! = \text{fact} \times ((x-1)!)) \text{ and } x - 1 \geq 0\}\]

Example

Second try, let us combine the two:
\[P = \{a! = \text{fact} \times (x!) \text{ and } x \geq 0\}\]

We need to show:
1. \((x > 0 \text{ and } x = a)\)
   \[\text{fact} := 1; \{P\} \text{ while } x > 0 \text{ do } (\text{fact} := \text{fact} \times x; x := x - 1) \text{ od} \{P \text{ and not } x > 0\}\]

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

Example

We proved that \(\{a! = \text{fact} \times (x!)\} \text{ and } x > 0\) is the loop invariant
We proved the sequence rule for the assignment and while statements
We applied postcondition weakening to prove the final predicate

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