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

- 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

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

- 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
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:

  `{ y = 2 } x := y {x = 2}`
The Assignment Rule – Your Turn

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

\[\{?\}\]

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

Precondition Strengthening

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

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?

\[\{x > 0 \& x < 5\} x := x \times x \{x < 25\}\]
\[\{x = 3\} x := x \times x \{x < 25\}\]
\[\{x > 0 \& x < 5\} x := x \times x \{x < 25\}\]
\[\{x \times x < 25\} x := x \times x \{x < 25\}\]

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

\[\{x = 3\} x := x \times x \{x < 25\}\]
\[\{x > 0 \& x < 5\} x := x \times x \{x < 25\}\]

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

\[\{x > 0 \& x < 5\} x := x \times x \{x < 25\}\]
**Sequencing**

\(\{P\} C_1 \{Q\} \quad \{Q\} C_2 \{R\}\)

- Example:
  \(\{z = z \land z = z\} \ x := z \quad \{x = z \land z = z\}\)
  \(\{x = z \land z = z\} \ y := z \quad \{x = z \land y = z\}\)
  \(\{z = z \land z = z\} \ x := z; \ y := z \quad \{x = z \land y = z\}\)

**Postcondition Weakening**

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

- Example:
  \(\{z = z \land z = z\} \ x := z \quad \{x = z \land z = z\}\)
  \(\{x = z \land z = z\} \ y := z \quad \{x = z \land y = z\}\)
  \(\{z = z \land z = z\} \ x := z; \ y := z \quad \{x = z \land y = z\}\)

**Rule of Consequence**

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

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

**If Then Else**

\(\{P \text{ and } B\} C_1 \{Q\} \quad \{P \text{ and } (\text{not } B)\} C_2 \{Q\}\)

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

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

(1) Reduces to (2) and (3) by Precondition Strengthening
(2) Follows from assignment axiom
(3) Because \(x<0 \Rightarrow |x| = -x\)
\{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) Reduce 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|\} \\
\{y=a\} \\
if x < 0 then y:= y-x 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: \\
\{ ? \} \ C \ \{ ? \} \\
\{ ? \} \ while \ B \ do \ C \ od \ \{ P \} \\

While \\

- The loop may never be executed, so if we want \( P \) to hold after, it had better hold before, so let’s try: \\
\{ ? \} \ C \ \{ ? \} \\
\{ P \} \ while \ B \ do \ C \ od \ \{ P \} \\

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: \\
\{ P \land B \} \ C \ \{ P \} \\
\{ P \} \ while \ B \ do \ C \ od \ \{ P \} \\

While \\

- We can strengthen the previous rule because we also know that when the loop is finished, \( \neg B \) also holds \\
- Final while rule: \\
\{ P \land B \} \ C \ \{ P \} \\
\{ P \} \ while \ B \ do \ C \ od \ \{ P \land \neg B \}
While

\{ P \text{ and } B \} \rightarrow C \rightarrow \{ P \} \\]
\{ P \} \text{ while } B \text{ do } C \text{ od } \{ P \text{ 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
  
  $\{ x \geq 0 \text{ and } x = a \}$
  $\text{fact} := 1;$
  while $x > 0$ do
    $\text{fact} := \text{fact} \times x$; $x := x - 1$
  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. $\{ x \geq 0 \text{ and } x = a \}$
     $\text{fact} := 1;$
     while $x > 0$ do
       $\text{fact} := \text{fact} \times x$; $x := x - 1$
     od

  \{ $a! = \text{fact} \times (x!)$ and not $(x > 0)$ \}
  
  and
  2. $\{ a! = \text{fact} \times (x!) \text{ and not } (x > 0) \} \Rightarrow \{ \text{fact} \text{ = a!} \}$
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. \(x \geq 0\) 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 \(x \geq 0\) 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. \(x \geq 0\) and \(x = a\)
    \(\text{fact} := 1;\)
    while \(x > 0\) do (\(\text{fact} := \text{fact} \times x; x := x - 1\)) od
    \(\{a! = \text{fact} \times (x!) \text{ and } x \geq 0\}\)
    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
    \(\{a! = \text{fact} \times (x!) \text{ and } x \geq 0\}\) and \(\{x \geq 0\}\) and \(\{x = a\}\)

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

Example

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

We will use assignment rule, sequencing rule and precondition strengthening

Example

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

This finishes the proof