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

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.

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

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

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 the weakest precondition of**

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

\[ \{x = 3\} \ x := x + 3 \ \{x < 10\} \]
\[ \{2 = 2\} \ x := 2 \ \{x = 2\} \]

**1725 minutes**

Precondition Strengthening

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

- Meaning: If we can show that \( P \) implies \( P' \) (\( P \Rightarrow P' \)) and we can show that \( \{P'\} \ C \{Q\} \), then we know that \( \{P\} \ C \{Q\} \)
- \( P \) is stronger than \( P' \) means \( P \Rightarrow P' \)

Examples:

\[ x = 3 \Rightarrow x < 7 \ \{x < 7\} \ x := x + 3 \ \{x < 10\} \]
\[ \{x = n\} \ x := x + 1 \ \{x = n+1\} \]
\[ x = 2 \Rightarrow 2 = 2 \ \{2 = 2\} \ x := 2 \ \{x = 2\} \]
Which Inferences Are Correct?

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

Sequencing

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

Example:

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

Postcondition Weakening

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

Example:

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

Rule of Consequence

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

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

\[ \{P \text{ and } B\} C_1 \{Q\} \quad \{P \text{ and } \neg B\} C_2 \{Q\} \]

Example: Want

\[ \{y=a\} \]

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

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

Suffices to show:

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

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

(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) Reduces to (5) and (6) by Precondition Strengthening
(5) Follows from assignment axiom
(6) Because \( \neg(x<0) \Rightarrow |x| = x \)

If then else

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

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:

\[ \{ \ ? \ \} \quad \text{C} \quad \{ \ ? \ \} \]

\[ \{ \ ? \ \} \quad \text{while} \quad \text{B} \quad \text{do} \quad \text{C} \quad \text{od} \quad \{ \ P \ \} \]
While

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

- If all we know is \( P \) when we enter the \textbf{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 \textbf{while} loop, we had better know it when we finish the loop body:
  
  \[
  \{ P \text{ and } B \} \ C \ { P } \\
  \{ P \} \text{ while } B \text{ do } C \text{ od } \{ P \}
  \]

- We can strengthen the previous rule because we also know that when the loop is finished, \( \text{not } B \) also holds

- Final \textbf{while} rule:
  
  \[
  \{ P \text{ and } B \} \ C \ { P } \\
  \{ P \} \text{ while } B \text{ do } C \text{ od } \{ P \text{ and } \text{not } B \}
  \]