Programming Languages and Compilers (CS 421)

Sasa Misailovic
4110 SC, UIUC

https://courses.engr.illinois.edu/cs421/fa2017/CS421A

Based in part on slides by Mattox Beckman, as updated by Vikram Adve, Gul Agha, and Elsa L Gunter
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 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
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 <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:

\[
\{\_ = 2\} \ x := y \ \{x = 2\}
\]
The Assignment Rule

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

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 a valid precondition of

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

  \[
  \begin{align*}
  \{ & ? \} \\
  x & := x + y \\
  \{ & x + y = w - x \}
  \end{align*}
  \]
The Assignment Rule – Your Turn

What is a valid precondition of

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

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

\[ x := x + y \]

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

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

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

- $P$ is *stronger* than $P'$ means $P \implies P'$
Precondition Strengthening

**Examples:**

\[
\begin{align*}
\text{x = 3} & \implies \text{x < 7} \quad \{\text{x < 7}\} \quad \text{x := x + 3} \quad \{\text{x < 10}\} \\
\{\text{x = 3}\} & \quad \text{x := x + 3} \quad \{\text{x < 10}\}
\end{align*}
\]

**True** \(\implies\) **2 = 2**

\[
\begin{align*}
\text{True} & \implies \text{2 = 2} \quad \{\text{2 = 2}\} \quad \text{x:= 2} \quad \{\text{x = 2}\} \\
\{\text{True}\} & \quad \text{x:= 2} \quad \{\text{x = 2}\}
\end{align*}
\]

\[
\begin{align*}
\text{x=n} & \implies \text{x+1=n+1} \quad \{\text{x+1=n+1}\} \quad \text{x:=x+1} \quad \{\text{x=n+1}\} \\
\{\text{x=n}\} & \quad \text{x:=x+1} \quad \{\text{x=n+1}\}
\end{align*}
\]
Which Inferences Are Correct?

\[
\begin{align*}
\{x > 0 \land x < 5\} & \quad x := x \ast x \quad \{x < 25\} \\
\{x = 3\} & \quad x := x \ast x \quad \{x < 25\} \\
\{x > 0 \land x < 5\} & \quad x := x \ast x \quad \{x < 25\} \\
\{x \ast x < 25\} & \quad x := x \ast x \quad \{x < 25\} \\
\{x > 0 \land x < 5\} & \quad x := x \ast x \quad \{x < 25\}
\end{align*}
\]
Which Inferences Are Correct?

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

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

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

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

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

\[
\begin{align*}
\{P\} & \mathbf{C}_1 \{Q\} & \{Q\} & \mathbf{C}_2 \{R\} \\
\{P\} & \mathbf{C}_1; \mathbf{C}_2 \{R\} \\
\end{align*}
\]

- Example:

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

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

Example:

\[\{z = z \land z = z\} \ x := z \ \{x = z \land z = z\}\]
\[\{x = z \land z = z\} \ y := z \ \{x = z \land y = z\}\]
\[\{z = z \land z = z\} \ x := z; \ y := z \ \{x = z \land y = z\}\]
Postcondition Weakening

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

Example:

\[
\begin{align*}
\{z = z \land z = z\} & \implies x := z; y := z \implies \{x = z \land y = z\} \\
(x = z \land y = z) & \implies (x = y) \\
\{z = z \land z = z\} & \implies x := z; y := z \implies \{x = y\}
\end{align*}
\]
Rule of Consequence

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

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

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

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

- 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. \(\{y=a\&x<0\} \quad y := y - x \quad \{y=a+|x|\}\) and
2. \(\{y=a\&\text{not}(x<0)\} \quad y := y + x \quad \{y=a+|x|\}\)
\{y=a \& x<0\} \ y:=y-x \ \{y=a+|x|\}

(3) \quad (y=a \& x<0) \Rightarrow y-x=a+|x|

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

(1) \quad \{y=a \& x<0\} \quad y:=y-x \quad \{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 \& \text{not}(x < 0)\} \ y := y + x \ \{y = a + |x|\}

(6) \quad (y = a \& \text{not}(x < 0)) \rightarrow (y + x = a + |x|)

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

(4) \quad \{y = a \& \text{not}(x < 0)\} \quad y := y + x \quad \{y = a + |x|\}

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

(5) Follows from **assignment** axiom

(6) Because \(\text{not}(x < 0) \rightarrow |x| = x\)
If Then Else

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

(4) \{y=a \& \text{not}(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 \textbf{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:

    $$\{ \ ? \ \} \ \text{C} \ \{ \ ? \ \}$$

    $$\{ \ ? \ \} \ \text{while} \ \text{B} \ \text{do} \ \text{C} \ \text{od} \ \{ \ P \ \}$$
While

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

\[
\begin{align*}
\{ & \quad ? \quad \} \quad C \quad \{ & \quad ? \quad \} \\
\hline
\{ \ P \ \} \quad \textbf{while} \quad B \quad \textbf{do} \quad C \quad \textbf{od} \quad \{ \ P \ \}
\end{align*}
\]
While

- 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:

\[
\{ P \text{ and } B \} \, C \, \{ P \} \\
\{ P \} \, \textbf{while} \, B \, \textbf{do} \, C \, \textbf{od} \, \{ P \}
\]
While

- We can strengthen the previous rule because we also know that when the loop is finished, not B also holds
- Final **while** rule:

\[
\begin{align*}
\{ P \text{ and } B \} & \quad C & \quad \{ P \} \\
\{ P \} & \quad \textbf{while} \quad B & \quad \textbf{do} & \quad C \quad \textbf{od} & \quad \{ P \text{ and } \text{not } B \} \nonumber
\end{align*}
\]
While

}{ P and B } C } P } 

}{ P } while B do C od } P and not B }

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

- **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.
Counting up to n

\[ x := 0; \]
\[ \text{while } (x < n) \{ \]
\[ \quad x := x + 1 \]
\[ \} \]

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

\( P \equiv x \leq n \land 0 \leq n \)
Sum of numbers 1 to n

\[
x := 0 \\
y := 0 \\
while y < n \{ \\
y := y + 1; \\
x := x + y \\
\}
\]

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

\[ \text{x} = 0; \text{y} = 1; \text{z} = 1; \]

while \( (z < n) \) {
  \[ \text{y} := \text{x} + \text{y}; \]
  \[ \text{x} := \text{y} - \text{x}; \]
  \[ z := z + 1 \]
}

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

\[ P \equiv \begin{align*}
  & y = \text{fib} \; z \\
  & \land \; x = \text{fib} \; (z-1) \\
  & \land \; z \leq n \\
  & \land \; 1 \leq n
\end{align*} \]
List Length

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

\[
\text{while} \ (x \neq []) \{ \\
\quad x := \text{tl} \ x; \\
\quad y := y + 1 \\
\}
\]

Want to show: \( y = \text{len lst} \)

\[
P \equiv y + \text{len} \ x = \text{len lst}
\]
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

\{fact = a!\}
\end{verbatim}
Example

- We need to find a condition $P$ that is true both before and after the loop is executed, and such that

\[(P \text{ and not } x > 0) \implies (\text{fact} = a!)\]
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: \( x! \)
Example

By post-condition weakening suffices to show

1. \{x \geq 0 \text{ and } x = a\}

   \begin{align*}
   \text{fact} & := 1; \\
   \text{while } x > 0 \text{ do (fact := fact } \times x; \ x := x - 1) \text{ od}
   \end{align*}

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

And

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

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

- 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, let us combine the two:

\[ P \equiv a! = \text{fact} \times (x!) \text{ and } x \geq 0 \]

We need to show:

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

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

And

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

\{x \geq 0 \text{ and } x = a\} \text{ (*this was part 1 to prove*)}

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

\{a! = fact * (x!) \text{ and } x \geq 0 \text{ and not } (x > 0)\}

For Part 1, by sequencing rule it suffices to show

3. \quad \{x \geq 0 \text{ and } x = a\}
   \begin{align*}
   \text{fact} & := 1 \\
   \{a! = \text{fact} * (x!) \text{ and } x \geq 0\}
   \end{align*}

And

4. \quad \{a! = \text{fact} * (x!) \text{ and } x \geq 0\}
   \begin{align*}
   \text{while } x > 0 \text{ do} \\
   \quad (\text{fact} := \text{fact} * x; x := x - 1) \text{ od} \\
   \{a! = \text{fact} * (x!) \text{ and } x \geq 0 \text{ and not } (x > 0)\}
   \end{align*}
Example

- (Part 3 – Assignment) Suffices to show that
  \[ a! = \text{fact} \times (x!) \text{ and } x \geq 0 \]
  holds before the while loop is entered

- (Part 4 – While Loop) 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 (part 4)
(Part 3) 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

\[(x \geq 0 \text{ and } x = a) \implies (a! = 1 \times (x!) \text{ and } x \geq 0)\]

It holds because \( x = a \implies x! = a! \).

So, we have that \( a! = \text{fact} \times (x!) \text{ and } x \geq 0 \)
holds at the start of the while loop!
Example

To prove (Part 4):

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

we need to show that \( (a! = \text{fact} \ast (x!)) \text{ and } x \geq 0 \) is a loop invariant

- We will use assignment rule, sequencing rule and precondition strengthening rule
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
    \[\{(a! = \text{fact} \times (x!)) \text{ and } x \geq 0 \text{ and } x > 0\}\]
    \[\text{fact} = \text{fact} \times x\]
    \[\{Q\}\]

  - By the assignment rule, show
    \[\{Q\}\]
    \[x := x - 1\]
    \[\{(a! = \text{fact} \times (x!)) \text{ and } x \geq 0\}\]
Example

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

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

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

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

- By **Precondition strengthening**, it suffices to show that
  \[
  ((a! = \text{fact} \ast (x!)) \text{ and } x \geq 0 \text{ and } x > 0) \implies
  ((a! = (\text{fact} \ast x) \ast ((x-1)!)) \text{ and } x - 1 \geq 0)
  \]

From algebra we know that \(\text{fact} \ast x \ast (x - 1)! = \text{fact} \ast x!\)
and \((x > 0) \implies x - 1 \geq 0\) since \(x\) is an integer, so
\[
\{(a! = \text{fact} \ast (x!)) \text{ and } x \geq 0 \text{ and } x > 0\} \implies
\{(a! = (\text{fact} \ast x) \ast ((x-1)!)) \text{ and } x - 1 \geq 0\}
Second try, let us combine the two:

\[ P \equiv a! = \text{fact} \times (x!) \text{ and } x \geq 0 \]

We need to show:

1. \{ x \geq 0 \text{ and } x = a \}

   \[
   \text{fact} := 1; \quad \checkmark
   \]

   \{ P \}

   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 Part 2, we need
  \[(a! = \text{fact} \times (x!) \text{ and } x \geq 0 \text{ and not } (x > 0)) \Rightarrow (\text{fact} = a!)]

Since we know \[(x \geq 0 \text{ and not } (x > 0)) \Rightarrow (x = 0)\] so
  \[\text{fact} \times (x!) = \text{fact} \times (0!)]
And since from algebra we know that \(0! = 1\),
  \[\text{fact} \times (0)! = \text{fact} \times 1 = \text{fact}\]

- Therefore, we can prove:
  \[(a! = \text{fact} \times (x!) \text{ and } x \geq 0 \text{ and not } (x > 0)) \Rightarrow (\text{fact} = a!)]
Example

- We proved that \((a! = \text{fact} \times (x!))\) and \(x \geq 0\) 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!

\[
\{x \geq 0 \text{ and } x = a\}
\]

\[
\text{fact} := 1;
\]

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

\[
\{\text{fact} = a!\}