Axiomatic Semantics
CS 6371: Advanced Programming Languages

Kevin W. Hamlen

April 21, 2022
Operational Semantics
- large-step and small-step varieties
- formally defines the operation of a machine that executes the program

Denotational Semantics
- defines the mathematical object (i.e., function) that a program denotes

Static Semantics (Type Theory)
- a static analysis that prevents certain runtime errors ("stuck states")

Today: Axiomatic Semantics
Goal: We wish to prove complete correctness of mission-critical code.
  - Type-theory too weak* (just proves soundness)
  - Operational semantics requires us to step outside the derivation system to prove things about derivations. Non-derivation parts cannot be machine-checked.
  - Denotational semantics creates a massive mathematical object that encodes all memory states (too hard to reason about).

Solution: Axiomatic Semantics
  - inference rules that encapsulate the entire correctness proof into a derivation
  - Derivation is fully machine-checkable, so no reliance on (error-prone) humans writing perfect proofs or perfectly checking proofs.

* Actually, advanced type systems like $\lambda C$ encode an entire axiomatic semantics into the type system, but let’s classify that as type theory + axiomatic semantics.
Two Kinds of Correctness

- **Partial Correctness**
  - Notation: \( \{A\}c\{B\} \) (called a *Hoare triple*)
  - If \( A \) is true before executing \( c \), and if \( c \) terminates, then \( B \) is true after executing \( c \).
  - \( A \) is *precondition*, and \( B \) is *postcondition*

- **Total Correctness**
  - Notation: \([A]c[B]\)
  - If \( A \) is true before executing \( c \), then \( c \) eventually terminates and \( B \) is true once it does.
Examples

1. \( \{x \leq 10\} \textbf{while } x \leq 10 \textbf{ do } x := x + 1\{?\} \)
Examples

1. \( \{ x \leq 10 \} \textbf{while} x \leq 10 \textbf{ do } x := x + 1 \{ x = 11 \} \)
Examples

1. \( \{ x \leq 10 \} \textbf{while } x \leq 10 \textbf{ do } x := x + 1 \{ x = 11 \} \)

2. \([ x \leq 10 ] \textbf{while } x \leq 10 \textbf{ do } x := x + 1 [?] \)
Examples

1. \( \{ x \leq 10 \} \textbf{while} \ x \leq 10 \ \textbf{do} \ x := x + 1 \{ x = 11 \} \)

2. \( [ x \leq 10 ] \textbf{while} \ x \leq 10 \ \textbf{do} \ x := x + 1 \{ x = 11 \} \)
Examples

1. \( \{ x \leq 10 \} \textbf{while} \ x <= 10 \ \textbf{do} \ x := x + 1 \{ x = 11 \} \)

2. \([ x \leq 10] \textbf{while} \ x <= 10 \ \textbf{do} \ x := x + 1 \{ x = 11 \} \)

3. \([T] \textbf{while} \ x <= 10 \ \textbf{do} \ x := x + 1 \{ \text{?} \} \)
Examples

1. \{x \le 10\} while x <= 10 \ do \ x := x + 1 \{x = 11\}
2. \[x \le 10\] while x <= 10 \ do \ x := x + 1 \[x = 11\]
3. \[T\] while x <= 10 \ do \ x := x + 1 \[x \ge 11\]
Examples

1. $\{x \leq 10\} \text{while} \ x \leq 10 \ \text{do} \ x := x + 1\{x = 11\}$
2. $[x \leq 10] \text{while} \ x \leq 10 \ \text{do} \ x := x + 1\{x = 11\}$
3. $[T] \text{while} \ x \leq 10 \ \text{do} \ x := x + 1\{x \geq 11\}$
4. $[x = \bar{i}] \text{while} \ x \leq 10 \ \text{do} \ x := x + 1\{?\}$
Examples

1. \( \{x \leq 10\} \text{while } x \leq 10 \text{ do } x := x + 1 \{ x = 11 \} \)
2. \([x \leq 10] \text{while } x \leq 10 \text{ do } x := x + 1 \{ x = 11 \} \)
3. \([T] \text{while } x \leq 10 \text{ do } x := x + 1 \{ x \geq 11 \} \)
4. \([x = \bar{i}] \text{while } x \leq 10 \text{ do } x := x + 1 \{ x = \max(11, \bar{i}) \} \)
Examples

1. \( \{ x \leq 10 \}\textbf{while} x \leq 10 \ \textbf{do} \ x := x + 1 \{ x = 11 \} \)
2. \([x \leq 10]\textbf{while} x \leq 10 \ \textbf{do} \ x := x + 1 \{ x = 11 \}\)
3. \([T]\textbf{while} x \leq 10 \ \textbf{do} \ x := x + 1 \{ x \geq 11 \}\)
4. \([x = \tilde{i}]\textbf{while} x \leq 10 \ \textbf{do} \ x := x + 1 \{ x = \max(11, \tilde{i}) \}\)
5. \(\{T\}\textbf{while} \text{true} \ \textbf{do} \ \textbf{skip} \{ F \}\)
Examples

1. \( \{x \leq 10\} \textbf{while } x \leq 10 \textbf{ do } x := x + 1 \{x = 11\} \)
2. \( [x \leq 10] \textbf{while } x \leq 10 \textbf{ do } x := x + 1 \{x = 11\} \)
3. \( [T] \textbf{while } x \leq 10 \textbf{ do } x := x + 1 \{x \geq 11\} \)
4. \( [x = \bar{i}] \textbf{while } x \leq 10 \textbf{ do } x := x + 1 \{x = \max(11, \bar{i})\} \)
5. \( \{T\} \textbf{while true do skip} \{F\} \)
   - \{any \(A\}\} any non-terminating program\{any \(B\)\}
Examples

1. \(\{x \leq 10\} \textbf{while} \ x \leq 10 \ \textbf{do} \ x := x + 1 \{x = 11\}\)
2. \([x \leq 10] \textbf{while} \ x \leq 10 \ \textbf{do} \ x := x + 1 \{x = 11\}\)
3. \([T] \textbf{while} \ x \leq 10 \ \textbf{do} \ x := x + 1 \{x \geq 11\}\)
4. \([x = \bar{i}] \textbf{while} \ x \leq 10 \ \textbf{do} \ x := x + 1 \{x = \max(11, \bar{i})\}\)
5. \([T] \textbf{while} \ \text{true} \ \textbf{do} \ \text{skip}\{F\}\)
   - \{any A\} any non-terminating program \{any B\}\)
6. \{F\} any program \{any B\}
Language of Assertions

- First-order logic with arithmetic:

  arithmetic exps \( a ::= n \mid v \mid \overline{v} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \ast a_2 \)

  assertions \( A ::= T \mid F \mid a_1 = a_2 \mid a_1 \leq a_2 \mid A_1 \land A_2 \)

  \( \mid A_1 \lor A_2 \mid \neg A \mid A \Rightarrow A_2 \mid \forall \overline{v}.A \mid \exists \overline{v}.A \)

- Meta-variables (\( \overline{v} \)) are **mathematical** variables (not program variables) that have fixed (arbitrary) integer values across all assertions.

- From these one can construct all functions and logical operators, so we will freely use extensions to the above.

  But if you write something extremely exotic, I reserve the right to challenge you on whether it can actually be expressed using the above.
First published by Tony Hoare [1969]
- First and most famous axiomatic semantics
- “An axiomatic basis for computer programming”
- Often cited as one of the greatest CS papers of all time (only 6 pages long!)
- Optional: read the original paper (linked from course web site)

Adaption to SIMPL consists of...
- six axioms (rules) describing SIMPL programs
- inference rules of first-order logic
- axioms of arithmetic (e.g., Peano arithmetic)
Skip Rule

\[
\{A\} \text{skip}\{?\} \tag{1}
\]
Skip Rule

\[ \{A\} \text{skip} \{A\}(1) \]
Sequence Rule

\[
\{A\} c_1 ; c_2 \{B\} \quad (2)
\]
Sequence Rule

\[
\frac{\{A\}c_1\{C\} \quad \{C\}c_2\{B\}}{\{A\}c_1; c_2\{B\}} \quad (2)
\]
Conditional Rule

\{A\} \textbf{if} b \ \textbf{then} \ c_1 \ \textbf{else} \ c_2 \ \{B\}
Conditional Rule

\[
\begin{align*}
\{ A \} c_1 \{ B \} & \quad (3a) \\
\{ A \} & \text{if } b \text{ then } c_1 \text{ else } c_2 \{ B \} \\
\{ A \} c_2 \{ B \} & \quad (3b) \\
\{ A \} & \text{if } b \text{ then } c_1 \text{ else } c_2 \{ B \}
\end{align*}
\]
Conditional Rule

\[
\begin{align*}
\{A\}c_1\{B\} & \Rightarrow \{A\} \text{if } b \text{ then } c_1 \text{ else } c_2 \{B\} \\
\{A\}c_2\{B\} & \Rightarrow \{A\} \text{if } b \text{ then } c_1 \text{ else } c_2 \{B\}
\end{align*}
\]

(3a) (3b)

Problem: These rules can derive false assertions (unsound)!

\[
\begin{align*}
\{T\}x:=0\{x = 0\} & \Rightarrow \{T\} \text{if } x \leq 0 \text{ then } x:=0 \text{ else skip} \{x = 0\} \\
\end{align*}
\]

(3a)
Conditional Rule

\[
\begin{array}{c}
\{ A \} c_1 \{ B \} \\
\{ A \} c_2 \{ B \}
\end{array}
\frac{\{ A \} \text{if } b \text{ then } c_1 \text{ else } c_2 \{ B \}}{(3)}
\]
Conditional Rule

\[
\begin{array}{c}
\{A\}c_1\{B\} \quad \{A\}c_2\{B\} \\
\{A\}\text{if } b \text{ then } c_1 \text{ else } c_2\{B\}
\end{array}
\] (3)

Problem: This rule cannot derive some true assertions (incomplete)!

\[
\begin{array}{c}
\{T\}x := 0\{x \geq 0\} \quad \{T\}\text{skip} \{x \geq 0\} \\
\{T\}\text{if } x \leq 0 \text{ then } x := 0 \text{ else skip} \{x \geq 0\}
\end{array}
\] (3)
Conditional Rule

\[
\frac{\{A \land b\}c_1\{B\} \quad \{A \land \neg b\}c_2\{B\}}{\{A\}\text{if } b \text{ then } c_1 \text{ else } c_2\{B\}} \tag{3}
\]

Solves completeness problem without sacrificing soundness:

\[
\frac{\{T \land x \leq 0\} x := 0 \{x \geq 0\} \quad \{T \land \neg(x \leq 0)\} \text{skip} \{x \geq 0\}}{\{T\} \text{if } x \leq 0 \text{ then } x := 0 \text{ else skip} \{x \geq 0\}} \tag{3}
\]
Assignment Rule

\[
\{ A \} v := a \{ ? \} ^{(4)}
\]
Assignment Rule

\[
\{A \} \nu := a \{ ? \}^{(4)}
\]

Usage example:

\[
\{ x > 10 \} x := x + 1 \{ x > 11 \}
\]
Assignment Rule

\[
\{A\}v := a\{B\}^{(4)}
\]

where \( B = A \) with all \( a \)'s replaced with \( v \)?

Usage example:

\[
\{x > 10\}x := x + 1\{x > 11\}
\]
Assignment Rule

\[ \{A\} v := a\{B\} \quad (4) \]

where \( B = A \) with all \( a \)'s replaced with \( v \)?

Usage example:

\[ \{x > 10\} x := x + 1 \{x > 11\} \]

\[ \xrightarrow{\text{equivalent}} \]

\[ \{x + 1 > 11\} x := x + 1 \{x > 11\} \]
Assignement Rule

\[
\{\ ? \ \}v := a\{B\}^{(4)}
\]

Usage example:

\[
\{x > 10\}x := x + 1\{x > 11\}
\]

\[
\equiv
\]

\[
\{x + 1 > 11\}x := x + 1\{x > 11\}
\]
Assignment Rule

\[
\{B[a/v]\}v := a\{B\}^{(4)}
\]

Usage example:

\[
\{x > 10\}x := x + 1\{x > 11\}
\]

\[
\equiv \downarrow \quad\quad\downarrow
\]

\[
\{x + 1 > 11\}x := x + 1\{x > 11\}
\]
While Rule

\{A\} while \ b \ do \ c \{B\}
While Rule

\[
\begin{align*}
\{A\}& \textbf{if } b \textbf{ then } (c; \textbf{while } b \textbf{ do } c) \textbf{ else skip} \{B\} \\
\{A\} & \textbf{while } b \textbf{ do } c \{B\} 
\end{align*}
\]
While Rule

\[
\begin{align*}
\{ A \land b \} c ; \textbf{while } b \textbf{ do } c \{ B \} & \quad \{ A \land \neg b \} \textbf{skip} \{ B \} \\
\{ A \} \textbf{if } b \textbf{ then } (c ; \textbf{while } b \textbf{ do } c) \textbf{ else } \textbf{skip} \{ B \} & \quad \{ A \} \textbf{while } b \textbf{ do } c \{ B \}
\end{align*}
\]
While Rule

\[
\frac{\{A \land b\} c \{C\} \quad \{C\} \textbf{while} b \textbf{ do } c \{B\}}{\{A \land b\} c ; \textbf{while} b \textbf{ do } c \{B\}} \quad \tag{2} \\
\frac{\{A\} \textbf{if} b \textbf{ then } (c ; \textbf{while} b \textbf{ do } c) \textbf{ else} \textbf{skip} \{B\}}{\{A\} \textbf{while} b \textbf{ do } c \{B\}} \quad \tag{3} \\
\frac{\{A\} \textbf{while} b \textbf{ do } c \{B\}}{\{A\} \textbf{while} b \textbf{ do } c \{B\}} \quad \tag{5}
\]
While Rule

\[ \vdash \]

\[ \{ A \land b \} c \{ C \} \quad \{ C \} \text{while} \ b \ \text{do} \ c \{ B \} \quad (5) \]

\[ \{ A \land b \} c; \text{while} \ b \ \text{do} \ c \{ B \} \quad (2) \]

\[ \{ A \land \neg b \} \text{skip} \{ B \} \quad (3) \]

\[ \{ A \} \text{if} \ b \ \text{then} \ (c; \text{while} \ b \ \text{do} \ c) \ \text{else} \ \text{skip} \{ B \} \quad (5) \]

\[ \{ A \} \text{while} \ b \ \text{do} \ c \{ B \} \quad (5) \]
While Rule

\{A\} \textbf{while} \ b \ \textbf{do} \ c \{ \ ? \}
While Rule

\[
\{ A \land b \} c \{ A \} \\
\{ A \} \text{while } b \text{ do } c \{ ? \} \tag{5}
\]
While Rule

\[
\begin{align*}
\{A \land b\} c \{A\} \\
\{A\} \textbf{while} b \textbf{ do } c \{\neg b \land A\}^{(5)}
\end{align*}
\]
While Rule

\[
\begin{align*}
\{ A \land b \} c \{ A \} \\
\{ A \} \text{while } b \text{ do } c \{ \neg b \land A \}^{(5)}
\end{align*}
\]
While Rule

\[
\begin{align*}
\{ I \land b \} & \quad c \quad \{ I \} \\
\{ I \} & \quad \textbf{while} \quad b \quad \textbf{do} \quad c \quad \{ \neg b \land I \} \quad (5)
\end{align*}
\]

\( I \) is called a \textbf{loop invariant}
Rule of Consequence

Recall that we earlier needed a way to prove (derive) equivalence of assertions:

\[
\{ x > 10 \} x := x + 1 \{ x > 11 \}
\]

equivalent

\[
\{ x + 1 > 11 \} x := x + 1 \{ x > 11 \}
\]
Recall that we earlier needed a way to prove (derive) equivalence of assertions:

\[
\{ x > 10 \} x := x + 1 \{ x > 11 \}
\]

equivalent

\[
\{ x + 1 > 11 \} x := x + 1 \{ x > 11 \}
\]

Rule of Consequence:

\[
\begin{array}{c}
\{ A' \} c \{ B' \} \\
\{ A \} c \{ B \}
\end{array}
\]

\[(6)\]
Recall that we earlier needed a way to prove (derive) equivalence of assertions:

\[
\{x > 10\} x := x + 1 \{x > 11\}
\]

\[
\{x + 1 > 11\} x := x + 1 \{x > 11\}
\]

Rule of Consequence:

\[
\begin{array}{c}
\models A \Rightarrow A' \\
\{A'\}c\{B'\}
\end{array}
\]

\[
\{A\}c\{B\}
\]

\(\models\) with nothing to the left means implication is **universally true** (i.e., not merely true in this program or loop)

- \(\models A \Rightarrow A'\) ← Assumptions may be safely **weakened**
Recall that we earlier needed a way to prove (derive) equivalence of assertions:

\[
\{ x > 10 \} x := x + 1 \{ x > 11 \}
\]

\[
\{ x + 1 > 11 \} x := x + 1 \{ x > 11 \}
\]

Rule of Consequence:

\[
\begin{align*}
\models A & \Rightarrow A' & \{ A' \} c \{ B' \} & \models B' & \Rightarrow B \\
& \models B' & \Rightarrow B & \{ A \} c \{ B \}
\end{align*}
\]

\((6)\)

\[\models\] with nothing to the left means implication is **universally true** (i.e., not merely true in this program or loop)

- \(\models A \Rightarrow A' \) ← Assumptions may be safely **weakened**
- \(\models B' \Rightarrow B \) ← Conclusions (goals) may be safely **strengthened**
Rule of Consequence Example

\[
\{ x > 10 \} x := x + 1 \{ x > 11 \}
\]
When you write axiomatic derivations in this class:
You are not required to write out the derivations of consequence premises (| = A).
I assume those are derivable using the laws of propositional logic and integer arithmetic.
But make sure your implications \( X \Rightarrow Y \) are universally true!

**Rule of Consequence Example**

\[
\begin{array}{c}
\vdash x > 10 \Rightarrow x + 1 > 11 \\
\{x + 1 > 11\} x := x + 1 \{x > 11\} \quad (4) \\
\{x > 10\} x := x + 1 \{x > 11\}
\end{array}
\]
Rule of Consequence Example

\[
\vdash x > 10 \Rightarrow x + 1 > 11 \\
\{x + 1 > 11\} \ x := x + 1 \{x > 11\} \quad (4) \\
\{x > 10\} \ x := x + 1 \{x > 11\} \\
\vdash x > 11 \Rightarrow x > 11 \quad (6)
\]

When you write axiomatic derivations in this class:

- You are **not** required to write out the derivations of consequence premises (\(\vdash A\)).
- I assume those are derivable using the laws of propositional logic and integer arithmetic.
- But make sure your implications \(X \Rightarrow Y\) are **universally true**!
Axiomatic Semantics of SIMPL

\( \{ A \} \text{skip} \{ A \} \) \(^{(1)}\)

\( \{ B[a/v] \} v := a \{ B \} \) \(^{(4)}\)

\[
\frac{\{ A \} c_1 \{ C \} \quad \{ C \} c_2 \{ B \}}{\{ A \} c_1 ; c_2 \{ B \}} \quad (2)
\]

\[
\frac{\{ I \land b \} c \{ I \}}{\{ I \} \text{while } b \text{ do } c \{ \neg b \land I \}} \quad (5)
\]

\[
\frac{\{ A \land b \} c_1 \{ B \} \quad \{ A \land \neg b \} c_2 \{ B \}}{\{ A \} \text{if } b \text{ then } c_1 \text{ else } c_2 \{ B \}} \quad (3)
\]

\[
\frac{A \Rightarrow A' \quad \{ A' \} c \{ B' \}}{\{ A \} c \{ B \}} \quad (6)
\]