**Operational Semantics**

- Describe meaning by executing program on machine
  - Machine can be actual or simulated
- Program meaning determined by change in machine
  - Machine state characterized by memory, registers, etc.
- Operational Semantics requires a virtual machine
  - Different hardware requires different simulator
  - Semantic definition varies from machine to machine
- Best used for low-level languages

**Dynamic Semantics**

**Axiomatic Semantics**
**Axiomatic Semantics**

Based on formal logic (predicate calculus)

Originally for program verification
- Define axioms or inference rules for each statement type
- Allows for transformation of one expression to another

Evaluation of axiomatic semantics:
- Good tool for correctness proofs
- Excellent framework for reasoning about programs
- Less useful for language users and compiler writers
- Difficult to develop rules for all language statements

---

**Axiomatic Semantics**

- **Assertions**
  - Logical expressions or predicates (test for validity)

- **Axioms**
  - Logical statements assumed to be true

- **Inference Rule**
  - Inferring the truth of one assertion based on the truth of another
Axiomatic Semantics

Assertions

• States relationships and constraints among variables that are true at that point in execution

• Precondition: \( \{ P \} \)
  - Assertion before a statement

• Postcondition: \( \{ Q \} \)
  - Assertion following a statement

• Weakest Precondition:
  - least restrictive precondition that will guarantee the postcondition

Precondition/Postcondition Form

\( \{ P \} \text{ S } \{ Q \} \)

Example:

\( \{ b > 10 \} \text{ x = b + 1 } \{ x > 1 \} \)

Weakest precondition:

\( P = Q_{x \rightarrow E} \) (Take \( Q \) and substitute \( E \) for \( x \))

\( P = \{ b + 1 > 1 \} \)

\( P = \{ b > 0 \} \) (Weakest Precondition for \( S \))
Axiomatic Semantics

Proving Program Correctness
- Last postcondition for whole program is desired result.
- Work back through the program to the first statement
  - Computing weakest precondition for first program statement
- Program is correct if
  - Precondition on first statement satisfies resulting weak precondition

Axiomatic Semantics

The Rule of Consequence
\[
\{ x > 4 \} \ x = x - 3 \ { x > 1 } \\
\{ x > 5 \} \ x = x - 3 \ { x > 0 } \\
\{ x > 5 \} \text{ implies } \{ x > 4 \} \\
\{ x > 1 \} \text{ implies } \{ x > 0 \} \\
\]

\[
{P}S{Q}, \ P' \Rightarrow P, \ Q \Rightarrow Q' \\
\]

\[
{P'}S{Q'} \\
\]
Axiomatic Semantics

The Rule of Consequence

\[ \{x > 4\} \rightarrow x = x - 3 \{x > 1\}, \{x > 5\} \Rightarrow \{x > 4\}, \{x > 1\} \Rightarrow \{x > 0\} \]

\[ \{x > 5\} \quad x = x - 3 \quad \{x > 1\} \]

\[
\begin{array}{c}
\{P\} S \{Q\}, \quad P' \Rightarrow P, \quad Q \Rightarrow Q' \\
\hline
\{P'\} S \{Q'\}
\end{array}
\]

Axiomatic Semantics

Inference Rule for Sequences

For a sequence \( S_1; S_2 \):

\[
\begin{align*}
\{P_1\} & \quad S_1 \quad \{P_2\} \\
\{P_2\} & \quad S_2 \quad \{P_3\}
\end{align*}
\]

\[
\begin{array}{c}
\{P_1\} S_1 \{P_2\}, \quad \{P_2\} S_2 \{P_3\} \\
\hline
\{P_1\} S_1; S_2 \quad \{P_3\}
\end{array}
\]

\[
\begin{align*}
& \{x < 2\} \\
y & = 3 \ast x + 1; \\
& \{y < 7\} \\
x & = y + 3; \\
& \{x < 10\}
\end{align*}
\]
**Axiomatic Semantics**

**Inference Rule for Selections**
- Boolean (predicate test) becomes part of precondition
  \[
  \text{if (boolean) then do } S_1 \text{ else do } S_2
  \]

\[
\{ B \text{ and } P \} S_1 \{ Q \}, \{ \neg B \text{ and } P \} S_2 \{ Q \}
\]

\[
\{ P \} \text{ if } B \text{ then } S_1 \text{ else } S_2 \{ Q \}
\]

**Axiomatic Semantics**

**Inference Rule for Logical Pretest Loops**

For the loop construct:

\[
\{ P \} \text{ while } B \text{ do } S \text{ end } \{ Q \}
\]

\[
\{ I \text{ and } B \} S \{ I \}
\]

\[
\{ I \} \text{ while } B \text{ do } S \text{ end } \{ I \text{ and } \neg B \}
\]

where \( I \) is the loop invariant (the inductive hypothesis)
Axiomatic Semantics

The loop invariant \( I \) must meet the following conditions

- \( P \Rightarrow I \)
  - The loop invariant must be true initially
- \( \{I\} B \{I\} \)
  - Evaluation of Boolean does not change validity of \( I \)
- \( \{I \land B\} S \{I\} \)
  - Executing body of loop does not change \( I \)
- \( (I \land (\neg B)) \Rightarrow Q \)
  - If \( I \) is true and \( B \) is false, \( Q \) is implied
- The loop terminates (can be difficult to prove)
- It is a weakened version of loop postcondition and is a precondition
  - \( I \) must be weak enough to be satisfied prior to the beginning of the loop, but when combined with the loop exit condition, it must be

Dynamic Semantics

Denotational Semantics
Denotational Semantics

Based on recursive function theory

The most abstract semantics description method
- Define a mathematical object for each language entity
- Define a function that maps instances of the language entities onto instances of the corresponding mathematical objects

Meaning of language constructs
- Defined by only the values of the program's variables

Denotational Semantics

- Can be used to prove the correctness of programs
- Provides a rigorous way to think about programs
- Can be an aid to language design
- Has been used in compiler generation systems

Comparison
- Operational semantics
  - State changes are defined by coded algorithms;
- Denotational semantics
  - State changes are defined by rigorous mathematical functions
Denotational Semantics

State of a program is the values of all its variables

\[ s = \{<i_1, v_1>, <i_2, v_2>, \ldots, <i_n, v_n>\} \]

<variable name, variable value> pairs

Let VARMAP be a function that,
- when given a variable name and a state,
- returns the current value of the variable

\[ \text{VARMAP}(i_j, s) = v_j \]