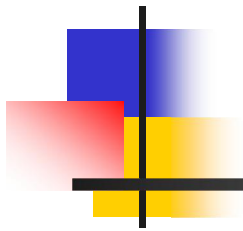


Programming Languages and Compilers (CS 421)



Reza Zamani

<http://www.cs.illinois.edu/class/cs421/>

Based in part on slides by Mattox Beckman, as updated by Vikram Adve, Gul Agha and Munawar Hafiz



Semantics

- n Expresses the meaning of syntax
- n Static semantics
 - n Meaning based only on the form of the expression without executing it
 - n Usually restricted to type checking / type inference
 - n Could include Data Flow Analysis.



Dynamic semantics

- n Method of describing meaning of executing a program
- n Several different types:
 - n Operational Semantics
 - n Axiomatic Semantics
 - n Algebraic Semantics
 - n Denotational Semantics (Scott–Strachey semantics)



Dynamic Semantics

- n Different languages better suited to different types of semantics
- n Different types of semantics serve different purposes



Operational Semantics

- n Start with a simple notion of machine
- n Describe how to execute (implement) programs of language on virtual machine, by describing how to execute each program statement (ie, following the *structure* of the program)
- n Meaning of program is how its execution changes the state of the machine
- n Useful as basis for implementations



Axiomatic Semantics

- n Also called Floyd-Hoare Logic
- n Based on formal logic (first order predicate calculus)
- n Axiomatic Semantics is a logical system built from *axioms* and *inference rules*
- n Mainly suited to simple imperative programming languages



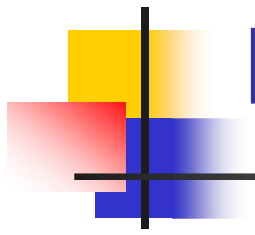
Axiomatic Semantics

- n 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 before execution
- n Written :
 {Precondition} Program {Postcondition}
- n Source of idea of *loop invariant*



Denotational Semantics

- n Construct a function M assigning a mathematical meaning to each program construct
- n Lambda calculus often used as the range of the meaning function
- n Meaning function is compositional: meaning of construct built from meaning of parts
- n Useful for proving properties of programs



Denotational Semantics

- n Construct a function M assigning a mathematical meaning to each program construct
- n Meaning function is compositional: meaning of construct built from meaning of parts
- n Useful for proving properties of programs



Natural Semantics

- n Aka Structural Operational Semantics, aka “Big Step Semantics”
- n Provide value for a program by rules and derivations, similar to type derivations
- n Rule conclusions look like

$$(C, m) \Downarrow m'$$

or

$$(E, m) \Downarrow v$$



Simple Imperative Programming Language

$n \ I \in \textit{Identifiers}$

$n \ N \in \textit{Numerals}$

$n \ B ::= \text{true} \mid \text{false} \mid B \ \& \ B \mid B \ \text{or} \ B \mid \text{not } B$
 $\mid E < E \mid E = E$

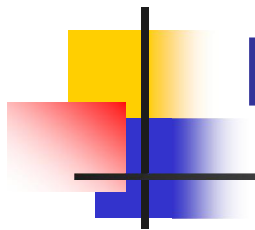
$n \ E ::= N \mid I \mid E + E \mid E * E \mid E - E \mid - E$

$n \ C ::= \text{skip} \mid C; C \mid I := E$
 $\mid \text{if } B \text{ then } C \text{ else } C \text{ fi} \mid \text{while } B \text{ do } C \text{ od}$



Natural Semantics of Atomic Expressions

- n Identifiers: $(I, m) \Downarrow m(I)$
- n Numerals are values: $(N, m) \Downarrow N$
- n Booleans: $(\text{true}, m) \Downarrow \text{true}$
 $(\text{false}, m) \Downarrow \text{false}$



Booleans:

$$\frac{(B, m) \Downarrow \text{false}}{(B \ \& \ B', m) \Downarrow \text{false}}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (B', m) \Downarrow b}{(B \ \& \ B', m) \Downarrow b}$$

$$\frac{(B, m) \Downarrow \text{true}}{(B \ \text{or} \ B', m) \Downarrow \text{true}}$$

$$\frac{(B, m) \Downarrow \text{false} \quad (B', m) \Downarrow b}{(B \ \text{or} \ B', m) \Downarrow b}$$

$$\frac{(B, m) \Downarrow \text{true}}{(\text{not } B, m) \Downarrow \text{false}}$$

$$\frac{(B, m) \Downarrow \text{false}}{(\text{not } B, m) \Downarrow \text{true}}$$



Relations

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \sim V = b}{(E \sim E', m) \Downarrow b}$$

- n By $U \sim V = b$, we mean does (the meaning of) the relation \sim hold on the meaning of U and V
- n May be specified by a mathematical expression/equation or rules matching U and V



Arithmetic Expressions

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \text{ op } V = N}{(E \text{ op } E', m) \Downarrow N}$$

where N is the specified value for $U \text{ op } V$



Commands

Skip: $(\text{skip}, m) \Downarrow m$

Assignment:
$$\frac{(E, m) \Downarrow V}{(I ::= E, m) \Downarrow m[I \leftarrow V]}$$

Sequencing:
$$\frac{(C, m) \Downarrow m' \quad (C', m') \Downarrow m''}{(C; C', m) \Downarrow m''}$$



If Then Else Command

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}$$

$$\frac{(B, m) \Downarrow \text{false} \quad (C', m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}$$



While Command

$$\frac{(B, m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m}$$

$$\frac{(B, m) \Downarrow \text{true} \quad (C, m) \Downarrow m' \quad (\text{while } B \text{ do } C \text{ od}, m) \Downarrow m''}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m''}$$



Example

(if $x > 5$ then $y := 2 + 3$ else $y := 3 + 4$ fi,
 $\{x \rightarrow 7\}) \Downarrow ?$



Example

$$(x > 5, \{x \rightarrow 7\}) \Downarrow ?$$

$$(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\ \{x \rightarrow 7\}) \Downarrow ?$$



Example

$? > ? = ?$

$(x, \{x \rightarrow 7\}) \Downarrow? \quad (5, \{x \rightarrow 7\}) \Downarrow?$

$(x > 5, \{x \rightarrow 7\}) \Downarrow?$

$(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow ?$



Example

$7 > 5 = \text{true}$

$(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5$

$(x > 5, \{x \rightarrow 7\}) \Downarrow ?$

$(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow ?$



Example

$7 > 5 = \text{true}$

$(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5$

$(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}$

$(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow ?$



Example

$$\begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \\
 \hline
 \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}
 \qquad
 \begin{array}{c}
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \\
 \Downarrow ? \\
 .
 \end{array}$$



Example

$$\begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \\
 \hline
 \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}
 \qquad
 \begin{array}{c}
 \hline
 (2 + 3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \\
 \Downarrow ? \\
 \hline
 \end{array}$$



Example

$$\begin{array}{c}
 \text{? + ? = ?} \\
 \hline
 (2, \{x \rightarrow 7\}) \Downarrow? \quad (3, \{x \rightarrow 7\}) \Downarrow? \\
 \hline
 \begin{array}{cc}
 7 > 5 = \text{true} & (2+3, \{x \rightarrow 7\}) \Downarrow? \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 & (y := 2 + 3, \{x \rightarrow 7\}) \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} & \Downarrow? \\
 \hline
 \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}
 \end{array}$$



Example

$$\begin{array}{c}
 2 + 3 = 5 \\
 \hline
 (2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3 \\
 \hline
 (2+3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 \begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \\
 \hline
 \end{array}
 \end{array}$$

$$\begin{array}{c}
 \hline
 (\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}$$



Example

$$\begin{array}{c}
 2 + 3 = 5 \\
 \hline
 (2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3 \\
 \hline
 (2+3, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \Downarrow ? \\
 \hline
 \begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \\
 \hline
 \end{array}
 \end{array}$$

(if $x > 5$ then $y := 2 + 3$ else $y := 3 + 4$ fi,
 $\{x \rightarrow 7\}) \Downarrow ?$



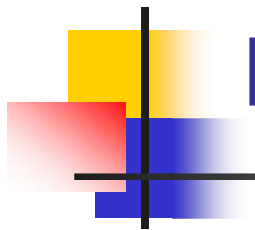
Example

$$\begin{array}{c}
 2 + 3 = 5 \\
 \frac{(2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3}{(2+3, \{x \rightarrow 7\}) \Downarrow 5} \\
 \frac{7 > 5 = \text{true} \quad (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5}{(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}} \quad \frac{(y := 2 + 3, \{x \rightarrow 7\}) \Downarrow 5}{\Downarrow \{y \rightarrow 5, x \rightarrow 7\}} \\
 \hline
 (\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow ?
 \end{array}$$



Example

$$\begin{array}{c}
 2 + 3 = 5 \\
 \hline
 (2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3 \\
 \hline
 (2+3, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (y := 2 + 3, \{x \rightarrow 7\}) \\
 \Downarrow \{y \rightarrow 5, x \rightarrow 7\} \\
 \hline
 \begin{array}{c}
 7 > 5 = \text{true} \\
 \hline
 (x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5 \\
 \hline
 (x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \\
 \hline
 \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\
 \{x \rightarrow 7\}) \Downarrow \{y \rightarrow 5, x \rightarrow 7\}
 \end{array}
 \end{array}$$



Let in Command

$$\frac{(E, m) \Downarrow v \quad (C, m[I \leftarrow v]) \Downarrow m'}{(\text{let } I = E \text{ in } C, m) \Downarrow m''}$$

Where $m''(x) = m'(x)$ for $x \neq I$ and
 $m''(\Lambda) = m(\Lambda)$ if $m(\Lambda)$ is defined,
otherwise $m''(\Lambda)$ is undefined



Example

$$\begin{array}{c}
 \frac{(x, \{x \rightarrow 5\}) \Downarrow 5 \quad (3, \{x \rightarrow 5\}) \Downarrow 3}{(x+3, \{x \rightarrow 5\}) \Downarrow 8} \\
 \frac{(5, \{x \rightarrow 17\}) \Downarrow 5 \quad (x := x+3, \{x \rightarrow 5\}) \Downarrow \{x \rightarrow 8\}}{(\text{let } x = 5 \text{ in } (x := x+3), \{x \rightarrow 17\}) \Downarrow ?}
 \end{array}$$



Example

$$\frac{\frac{(x, \{x \rightarrow 5\}) \Downarrow 5 \quad (3, \{x \rightarrow 5\}) \Downarrow 3}{(x+3, \{x \rightarrow 5\}) \Downarrow 8}}{(5, \{x \rightarrow 17\}) \Downarrow 5 \quad (x := x+3, \{x \rightarrow 5\}) \Downarrow \{x \rightarrow 8\}} \Downarrow 17$$

(let x = 5 in (x := x+3), {x -> 17}) \Downarrow 17



Comment

- n Simple Imperative Programming Language introduces variables *implicitly* through assignment
- n The let-in command introduces scoped variables *explicitly*
- n Clash of constructs apparent in awkward semantics



Interpretation Versus Compilation

- n A **compiler** from language L1 to language L2 is a program that takes an L1 program and for each piece of code in L1 generates a piece of code in L2 of same meaning
- n An **interpreter** of L1 in L2 is an L2 program that executes the meaning of a given L1 program
- n Compiler would examine the body of a loop once; an interpreter would examine it every time the loop was executed



Interpreter

- n An *Interpreter* represents the operational semantics of a language L1 (source language) in the language of implementation L2 (target language)
- n Built incrementally
 - n Start with literals
 - n Variables
 - n Primitive operations
 - n Evaluation of expressions
 - n Evaluation of commands/declarations



Interpreter

- n Takes abstract syntax trees as input
 - n In simple cases could be just strings
- n One procedure for each syntactic category (nonterminal)
 - n eg one for expressions, another for commands
- n If Natural semantics used, tells how to compute final value from code
- n If Transition semantics used, tells how to compute next "state"
 - n To get final value, put in a loop



Natural Semantics Example

n $\text{compute_exp } (\text{Var}(v), m) = \text{look_up } v \ m$

n $\text{compute_exp } (\text{Int}(n), _) = \text{Num } (n)$

n ...

n $\text{compute_com}(\text{IfExp}(b, c1, c2), m) =$
 if $\text{compute_exp } (b, m) = \text{Bool}(\text{true})$
 then $\text{compute_com } (c1, m)$
 else $\text{compute_com } (c2, m)$



Natural Semantics Example

- n $\text{compute_com}(\text{While}(b,c), m) =$
 - if $\text{compute_exp}(b,m) = \text{Bool}(\text{false})$
 - then m
 - else compute_com
 $(\text{While}(b,c), \text{compute_com}(c,m))$
- n May fail to terminate - exceed stack limits
- n Returns no useful information then