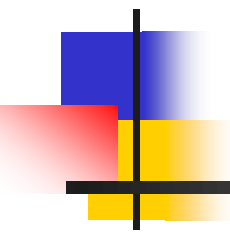


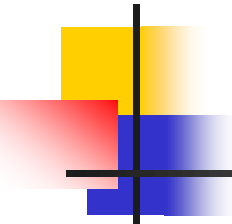
Programming Languages and Compilers (CS 421)



William Mansky

<http://courses.engr.illinois.edu/cs421/>

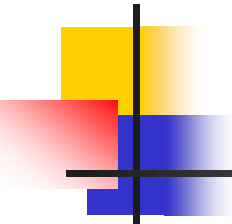
Based in part on slides by Mattox Beckman, as updated by Vikram Adve, Gul Agha, Elsa Gunter, and Dennis Griffith



Compilers: Big Picture

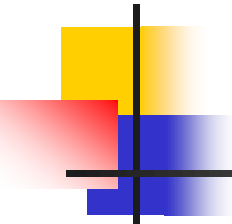
- We want to turn strings (code) into computer-readable instructions
- Done in phases
- Turn strings into abstract syntax trees (lex and parse)
- Translate abstract syntax trees into executable instructions (interpret or compile)

- Syntax gives the shape/structure of a language
- Semantics assigns meaning to syntax
- Static semantics: what does the text of a program mean?
 - e.g. type checking/type inference
- Dynamic semantics: what does a program do when it executes?



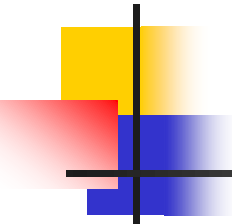
Dynamic semantics

- Several different types:
 - Operational Semantics
 - Axiomatic Semantics
 - Denotational Semantics
- Different languages better expressed by different kinds of semantics
- Different kinds of semantics serve different purposes



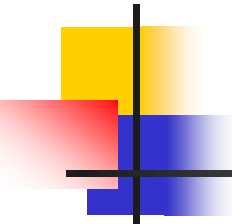
Operational Semantics

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



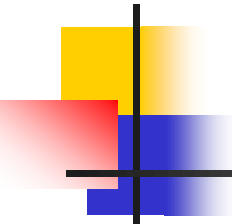
Axiomatic Semantics

- Axiomatic Semantics is a logical system built from *axioms* and *inference rules*
- Common example is Floyd-Hoare Logic
- Assume precondition on state, prove postcondition based on statement
 - E.g. $\{x = 0\} \ y = x + 1 \ \{x = 0, y = 1\}$



Denotational Semantics

- Construct a function $\mathcal{M}$ assigning a mathematical meaning to each program construct
- Meaning function is compositional: meaning of construct built from meaning of parts
- One possible kind of “meaning” is lambda calculus
- Useful for proving properties of programs



Natural Semantics

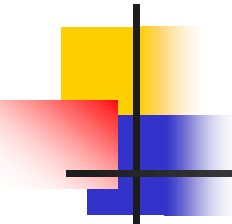
- A.k.a. “Big-Step Semantics”
- Provide value for a program by rules and derivations, similar to type derivations
- Rule conclusions look like

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

or

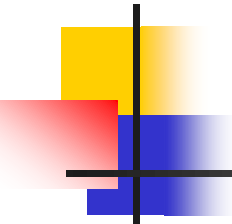
$$(E, m) \Downarrow v$$

where v is a *value*, m is a *map* from variables to values (“environment”)



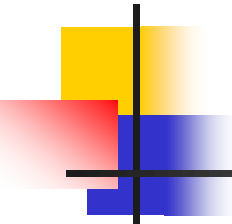
Simple Imperative Programming Language

- $I \in \textit{Identifiers}$
- $N \in \textit{Numerals}$
- $B ::= \text{true} \mid \text{false} \mid B \ \& \ B \mid B \ \text{or} \ B \mid \text{not} \ B$
 $\mid E < E \mid E = E$
- $E ::= N \mid I \mid E + E \mid E * E \mid E - E \mid - E$
- $C ::= \text{skip} \mid C; C \mid I := E$
 $\mid \text{if } B \text{ then } C \text{ else } C \text{ fi} \mid \text{while } B \text{ do } C \text{ od}$



Natural Semantics of Atomic Expressions

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



Booleans:

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

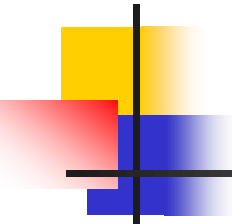
$$\frac{(B, m) \Downarrow \text{true} \quad (B', m) \Downarrow b}{(B \ \& \ B', m) \Downarrow b}$$

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

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

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

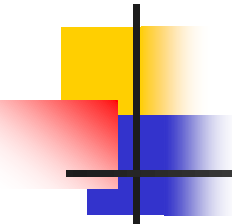
$$\frac{(B, m) \Downarrow \text{false}}{(\text{not } B, m) \Downarrow \text{true}}$$



Relations

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

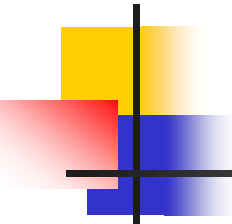
- b is true if the relation $\sim$ holds on the meaning of v and v' , false otherwise
- May be specified by a mathematical expression/equation or rules matching v and v'



Arithmetic Expressions

$$\frac{(E, m) \Downarrow v \quad (E', m) \Downarrow v' \quad v \text{ op } v' = n}{(E \text{ op } E', m) \Downarrow n}$$

where n is the specified value for $v \text{ op } v'$

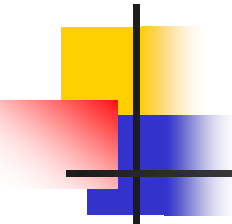


Commands

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

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

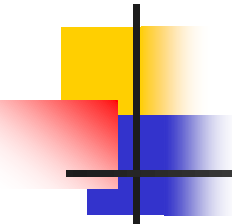
Sequencing:
$$\frac{(C, m) \Downarrow m' \quad (C', m') \Downarrow m''}{(C; C', m) \Downarrow m''}$$



If Then Else Command

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

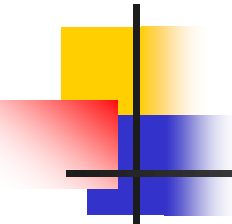
$$\frac{(B, m) \Downarrow \text{false} \quad (C', m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}$$



While Command

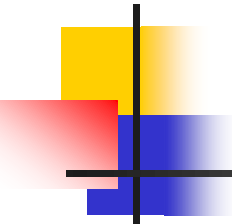
$$\frac{(B, m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m}$$

$$\frac{\begin{array}{cc} (B, m) \Downarrow \text{true} & (C, m) \Downarrow m' \\ (\text{while } B \text{ do } C \text{ od}, m') \Downarrow m'' \end{array}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m''}$$



Example: If Then Else Rule

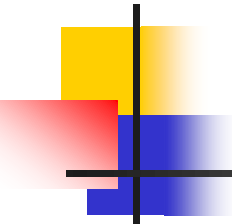
(if $x > 5$ then $y := 2 + 3$ else $y := 3 + 4$ fi,
 $\{x \rightarrow 7\}) \Downarrow ?$



Example: If Then Else Rule

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

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

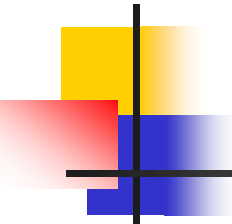


Example: Arith Relation

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

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

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



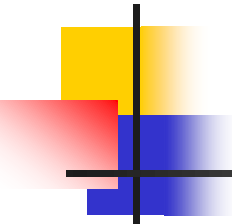
Example: Identifier(s)

$7 > 5 = \text{true}$

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

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

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



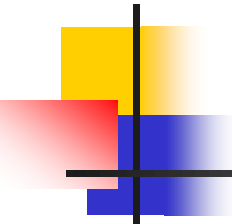
Example: Arith Relation

$$7 > 5 = \text{true}$$

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

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

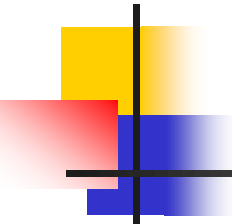
$$\begin{array}{c} \text{(if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi,} \\ \{x \rightarrow 7\}) \Downarrow ? \end{array}$$



Example: If Then Else Rule

$7 > 5 = \text{true}$

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



Example: Assignment

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

Example: Arith Op

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

Example: Numerals

$$2 + 3 = 5$$

$$\frac{(2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3}{(2 + 3, \{x \rightarrow 7\}) \Downarrow ?}$$

$$7 > 5 = \text{true}$$

$$(2 + 3, \{x \rightarrow 7\}) \Downarrow ?$$

$$\frac{(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5}{(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true} \quad (y := 2 + 3, \{x \rightarrow 7\}) \Downarrow ?}$$

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

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

Example: Arith Op

$$2 + 3 = 5$$

$$\frac{(2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3}{(2 + 3, \{x \rightarrow 7\}) \Downarrow 5}$$

$$7 > 5 = \text{true}$$

$$(2 + 3, \{x \rightarrow 7\}) \Downarrow 5$$

$$\frac{(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5}{(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}} \quad (y := 2 + 3, \{x \rightarrow 7\})$$

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

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

Example: Assignment

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

Example: If Then Else

$$2 + 3 = 5$$

$$\frac{(2, \{x \rightarrow 7\}) \Downarrow 2 \quad (3, \{x \rightarrow 7\}) \Downarrow 3}{(2 + 3, \{x \rightarrow 7\}) \Downarrow 5}$$

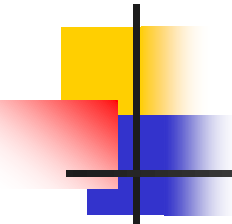
$$7 > 5 = \text{true}$$

$$(2 + 3, \{x \rightarrow 7\}) \Downarrow 5$$

$$\frac{(x, \{x \rightarrow 7\}) \Downarrow 7 \quad (5, \{x \rightarrow 7\}) \Downarrow 5}{(x > 5, \{x \rightarrow 7\}) \Downarrow \text{true}} \quad (y := 2 + 3, \{x \rightarrow 7\})$$

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

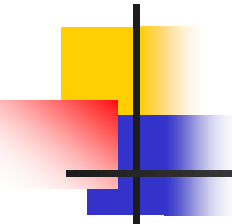
$$\frac{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow \{x \rightarrow 7, y \rightarrow 5\}}{(\text{if } x > 5 \text{ then } y := 2 + 3 \text{ else } y := 3 + 4 \text{ fi, } \{x \rightarrow 7\}) \Downarrow \{x \rightarrow 7, y \rightarrow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



Let in Command

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

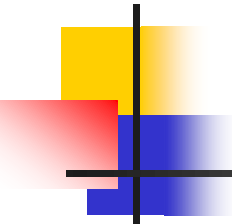
- First evaluate value for I
- Then evaluate C given that value for I
- Finally, restore old value for I



Example

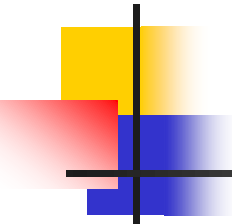
$$\frac{\frac{(x, \{x \rightarrow 5\}) \Downarrow 5 \quad (3, \{x \rightarrow 5\}) \Downarrow 3}{(x+3, \{x \rightarrow 5\}) \Downarrow 8}}{(5, \{x \rightarrow 17\}) \Downarrow 5 \quad (x := x+3, \{x \rightarrow 5\}) \Downarrow \{x \rightarrow 8\}} \Downarrow ?$$

$(\text{let } x = 5 \text{ in } (x := x+3), \{x \rightarrow 17\}) \Downarrow ?$



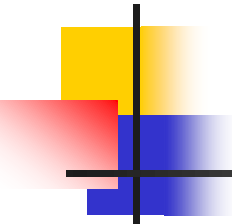
Example

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



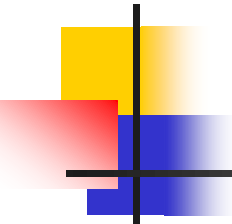
Notes on Let

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



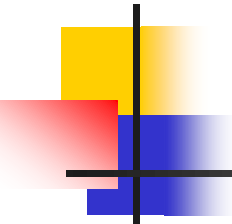
Running Code

- A **compiler** for language L1 takes an L1 program and generates code in a target language L2 (e.g., assembly) with same meaning
- An **interpreter** of L1 executes the meaning of a given L1 program directly
- Compiling tends to be more efficient
- Compiler examines the body of a loop once; interpreter examines it every time the loop is executed



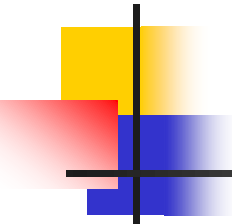
Interpreter

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



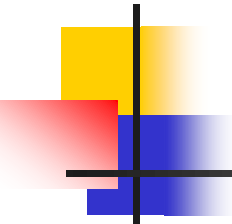
Interpreter

- Takes abstract syntax trees as input
 - In simple cases could be just strings
- One procedure for each syntactic category (nonterminal)
 - E.g. one for expressions, another for commands
- If natural semantics used, tells how to compute final value from code
- If transition semantics used, tells how to compute next “state”
 - To get final value, put in a loop



Natural Semantics Example

- $\text{compute_exp}(\text{Var}(v), m) = \text{look_up } v \ m$
- $\text{compute_exp}(\text{Int}(n), _) = \text{Num } (n)$
- ...
- $\text{compute_com}(\text{IfExp}(b, c1, c2), m) =$
 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

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