

Programming Languages and Compilers (CS 421)

Elsa L Gunter
2112 SC, UIUC

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

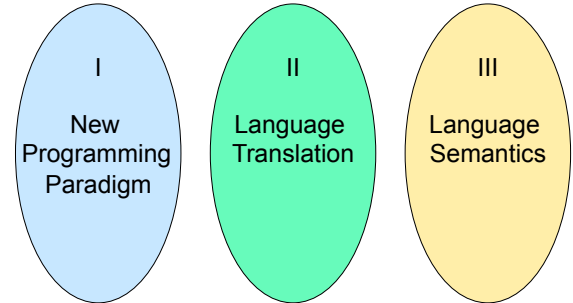
Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

11/10/15

1

Programming Languages & Compilers

Three Main Topics of the Course

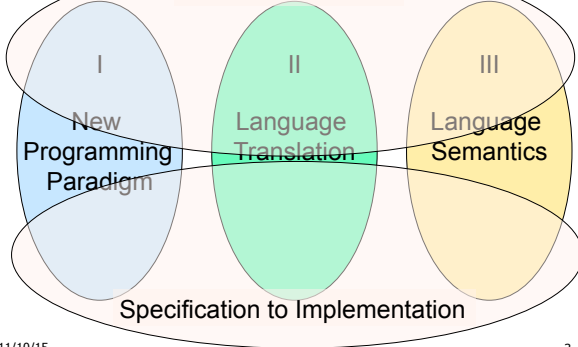


11/10/15

2

Programming Languages & Compilers

Order of Evaluation

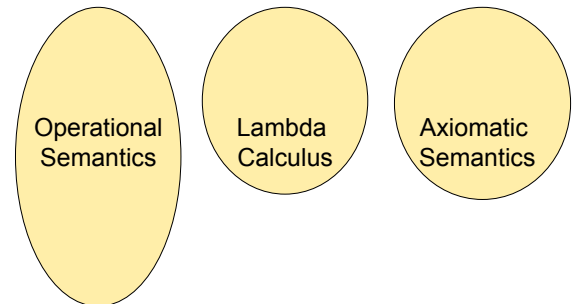


11/10/15

3

Programming Languages & Compilers

III : Language Semantics

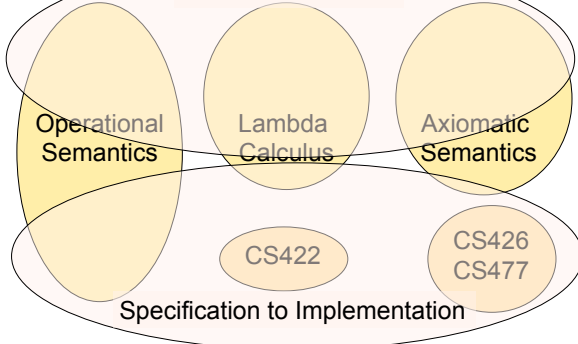


11/10/15

4

Programming Languages & Compilers

Order of Evaluation



11/10/15

5

Semantics

- Expresses the meaning of syntax
- Static semantics
 - Meaning based only on the form of the expression without executing it
 - Usually restricted to type checking / type inference

11/10/15

6

Dynamic semantics

- Method of describing meaning of executing a program
- Several different types:
 - Operational Semantics
 - Axiomatic Semantics
 - Denotational Semantics

11/10/15

7

Dynamic Semantics

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

11/10/15

8

Operational Semantics

- Start with a simple notion of machine
- 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)
- Meaning of program is how its execution changes the state of the machine
- Useful as basis for implementations

11/10/15

9

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

11/10/15

10

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

11/10/15

11

Denotational Semantics

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

11/10/15

12

Natural Semantics

- Aka “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

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

11/10/15

13

Simple Imperative Programming Language

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

11/10/15

14

Natural Semantics of Atomic Expressions

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

11/10/15

15

Booleans:

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

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

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

11/10/15

16

Relations

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

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

11/10/15

17

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 $U \ \text{op} \ V$

11/10/15

18

Commands

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

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

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

11/10/15

19

If Then Else Command

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

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

11/10/15

20

While Command

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

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

11/10/15

21

Example: If Then Else Rule

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

11/10/15

22

Example: If Then Else Rule

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

11/10/15

23

Example: Arith Relation

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

11/10/15

24

Example: Identifier(s)

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

11/10/15

25

Example: Arith Relation

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

11/10/15

26

Example: If Then Else Rule

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

11/10/15

27

Example: Assignment

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

11/10/15

28

Example: Arith Op

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

11/10/15

29

Example: Numerals

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

11/10/15

30

Example: Arith Op

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

11/10/15

31

Example: Assignment

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

11/10/15

32

Example: If Then Else Rule

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

11/10/15

33

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

11/10/15

34

Example

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

11/10/15

35

Example

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

11/10/15

36

Comment

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

11/10/15

37

Interpretation Versus Compilation

- 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
- An **interpreter** of L1 in L2 is an L2 program that executes the meaning of a given L1 program
- Compiler would examine the body of a loop once; an interpreter would examine it every time the loop was executed

11/10/15

38

Interpreter

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

11/10/15

39

Interpreter

- Takes abstract syntax trees as input
 - In simple cases could be just strings
- One procedure for each syntactic category (nonterminal)
 - eg 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

11/10/15

40

Natural Semantics Example

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

11/10/15

41

Natural Semantics Example

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

11/10/15

42

Transition Semantics

- Form of operational semantics
- Describes how each program construct transforms machine state by *transitions*
- Rules look like $(C, m) \rightarrow (C', m')$ or $(C, m) \rightarrow m'$
- C, C' is code remaining to be executed
- m, m' represent the state/store/memory/environment
 - Partial mapping from identifiers to values
 - Sometimes m (or C) not needed
- Indicates exactly one step of computation

11/10/15

43

Expressions and Values

- C, C' used for commands; E, E' for expressions; U, V for values
- Special class of expressions designated as *values*
 - Eg 2, 3 are values, but 2+3 is only an expression
- Memory only holds values
 - Other possibilities exist

11/10/15

44

Evaluation Semantics

- Transitions successfully stops when E/C is a value/memory
- Evaluation fails if no transition possible, but not at value/memory
- Value/memory is the final *meaning* of original expression/command (in the given state)
- Coarse semantics: final value / memory
- More fine grained: whole transition sequence

11/10/15

45

Simple Imperative Programming Language

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

11/10/15

46

Transitions for Expressions

- Numerals are values
- Boolean values = {true, false}
- Identifiers: $(I, m) \rightarrow (m(I), m)$

11/10/15

47

Boolean Operations:

- Operators: (short-circuit)

$$\begin{array}{l} (\text{false} \ \& \ B, m) \rightarrow (\text{false}, m) \quad \frac{(B, m) \rightarrow (B'', m)}{(B \ \& \ B', m) \rightarrow (B'' \ \& \ B', m)} \\ (\text{true} \ \& \ B, m) \rightarrow (B, m) \\ (\text{true} \ \text{or} \ B, m) \rightarrow (\text{true}, m) \quad \frac{(B, m) \rightarrow (B'', m)}{(B \ \text{or} \ B', m) \rightarrow (B'' \ \text{or} \ B', m)} \\ (\text{false} \ \text{or} \ B, m) \rightarrow (B, m) \\ (\text{not} \ \text{true}, m) \rightarrow (\text{false}, m) \quad \frac{(B, m) \rightarrow (B', m)}{(\text{not} \ B, m) \rightarrow (\text{not} \ B', m)} \\ (\text{not} \ \text{false}, m) \rightarrow (\text{true}, m) \end{array}$$

11/10/15

48

Relations

$$\frac{(E, m) \rightarrow (E', m)}{(E \sim E', m) \rightarrow (E' \sim E', m)}$$

$$\frac{(E, m) \rightarrow (E', m)}{(V \sim E, m) \rightarrow (V \sim E', m)}$$

$(U \sim V, m) \rightarrow (\text{true}, m)$ or (false, m)
depending on whether $U \sim V$ holds or not

11/10/15

49

Arithmetic Expressions

$$\frac{(E, m) \rightarrow (E', m)}{(E \text{ op } E', m) \rightarrow (E' \text{ op } E', m)}$$

$$\frac{(E, m) \rightarrow (E', m)}{(V \text{ op } E, m) \rightarrow (V \text{ op } E', m)}$$

$(U \text{ op } V, m) \rightarrow (N, m)$ where N is the
specified value for $U \text{ op } V$

11/10/15

50

Commands - in English

- skip means done evaluating
- When evaluating an assignment, evaluate the expression first
- If the expression being assigned is already a value, update the memory with the new value for the identifier
- When evaluating a sequence, work on the first command in the sequence first
- If the first command evaluates to a new memory (ie completes), evaluate remainder with new memory

11/10/15

51

Commands

$$(\text{skip}, m) \rightarrow m$$

$$\frac{(E, m) \rightarrow (E', m)}{(I ::= E, m) \rightarrow (I ::= E', m)}$$

$$(I ::= V, m) \rightarrow m[I \leftarrow V]$$

$$\frac{(C, m) \rightarrow (C'', m')}{(C; C', m) \rightarrow (C''; C', m')} \quad \frac{(C, m) \rightarrow m'}{(C; C', m) \rightarrow (C', m')}$$

11/10/15

52

If Then Else Command - in English

- If the boolean guard in an if_then_else is true, then evaluate the first branch
- If it is false, evaluate the second branch
- If the boolean guard is not a value, then start by evaluating it first.

11/10/15

53

If Then Else Command

$$(\text{if true then } C \text{ else } C' \text{ fi}, m) \rightarrow (C, m)$$

$$(\text{if false then } C \text{ else } C' \text{ fi}, m) \rightarrow (C', m)$$

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

11/10/15

54



While Command

$(\text{while } B \text{ do } C \text{ od}, m) \rightarrow$
 $(\text{if } B \text{ then } C; \text{ while } B \text{ do } C \text{ od else skip fi}, m)$

In English: Expand a While into a test of the boolean guard, with the true case being to do the body and then try the while loop again, and the false case being to stop.