

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

Natural Semantics

- Aka Structural Operational 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

2

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

3

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

4

Booleans:

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

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

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

11/10/15

5

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

6

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

7

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

8

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

9

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

10

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

11

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

12

Example: Arith Relation

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

11/10/15

13

Example: Identifier(s)

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

11/10/15

14

Example: Arith Relation

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

11/10/15

15

Example: If Then Else Rule

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

11/10/15

16

Example: Assignment

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

11/10/15

17

Example: Arith Op

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

11/10/15

18

Example: Numerals

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

11/10/15

19

Example: Arith Op

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

11/10/15

20

Example: Assignment

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

11/10/15

21

Example: If Then Else Rule

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

11/10/15

22

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

23

Example

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

11/10/15

24

Example

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

11/10/15

25

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

26

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

27

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

28

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

29

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

30

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 then

11/10/15

31

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

32

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

33

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

34

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

35

Transitions for Expressions

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

11/10/15

36

Boolean Operations:

- Operators: (short-circuit)

$$\frac{(B, m) \rightarrow (B'', m)}{(false \ \& \ B, m) \rightarrow (false, m) \quad (true \ \& \ B, m) \rightarrow (B, m)}$$

$$\frac{(B, m) \rightarrow (B'', m)}{(B \ \& \ B', m) \rightarrow (B'' \ \& \ B', m)}$$

$$\frac{(B, m) \rightarrow (B'', m)}{(true \ or \ B, m) \rightarrow (true, m) \quad (false \ or \ B, m) \rightarrow (B, m)}$$

$$\frac{(B, m) \rightarrow (B'', m)}{(B \ or \ B', m) \rightarrow (B'' \ or \ B', m)}$$

$$\frac{(B, m) \rightarrow (B', m)}{(not \ true, m) \rightarrow (false, m) \quad (not \ false, m) \rightarrow (true, m)}$$

$$\frac{(B, m) \rightarrow (B', m)}{(not \ B, m) \rightarrow (not \ B', m)}$$

11/10/15

37

Relations

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

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

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

11/10/15

38

Arithmetic Expressions

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

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

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

11/10/15

39

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

40

Commands

$$(skip, m) \rightarrow m$$

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

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

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

11/10/15

41

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

42

If Then Else Command

(if true then C else C' fi, m) \rightarrow (C , m)

(if false then C else C' 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

43

While Command

(while B do C od, m) \rightarrow
 (if B then C ; while B do C 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.

11/10/15

44

Example Evaluation

- First step:

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

11/10/15

45

Example Evaluation

- First step:

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

11/10/15

46

Example Evaluation

- First step:

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

11/10/15

47

Example Evaluation

- First step:

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

11/10/15

48

Example Evaluation

- First step:

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

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

11/10/15

49

Example Evaluation

- Second Step:

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

- Third Step:

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

11/10/15

50

Example Evaluation

- Fourth Step:

$$\frac{(2 + 3, \{x \rightarrow 7\}) \rightarrow (5, \{x \rightarrow 7\})}{(y := 2 + 3, \{x \rightarrow 7\}) \rightarrow (y := 5, \{x \rightarrow 7\})}$$

- Fifth Step:

$$(y := 5, \{x \rightarrow 7\}) \rightarrow \{y \rightarrow 5, x \rightarrow 7\}$$

11/10/15

51

Example Evaluation

- Bottom Line:

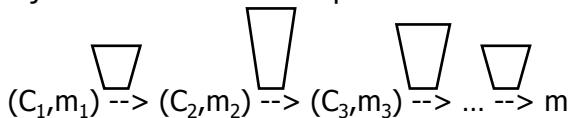
$$(if\ x > 5\ then\ y := 2 + 3\ else\ y := 3 + 4\ fi,\ \{x \rightarrow 7\}) \rightarrow (if\ 7 > 5\ then\ y := 2 + 3\ else\ y := 3 + 4\ fi,\ \{x \rightarrow 7\}) \rightarrow (if\ true\ then\ y := 2 + 3\ else\ y := 3 + 4\ fi,\ \{x \rightarrow 7\}) \rightarrow (y := 2 + 3,\ \{x \rightarrow 7\}) \rightarrow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

11/10/15

52

Transition Semantics Evaluation

- A sequence of steps with trees of justification for each step



- Let \rightarrow^* be the transitive closure of \rightarrow
- Ie, the smallest transitive relation containing \rightarrow

11/10/15

53

Adding Local Declarations

- Add to expressions:
 - $E ::= \dots \mid \text{let } I = E \text{ in } E' \mid \text{fun } I \rightarrow E \mid EE'$
 - $\text{fun } I \rightarrow E$ is a value
- Could handle local binding using state, but have assumption that evaluating expressions doesn't alter the environment
- We will use substitution here instead
- Notation:** $E[E' / I]$ means replace all free occurrence of I by E' in E

11/10/15

54

Call-by-value (Eager Evaluation)

$$\frac{(\text{let } I = V \text{ in } E, m) \rightarrow (E[V/I], m)}{(E, m) \rightarrow (E', m)} \quad \frac{}{(\text{let } I = E \text{ in } E', m) \rightarrow (\text{let } I = E' \text{ in } E')}$$

$$\frac{((\text{fun } I \rightarrow E) V, m) \rightarrow (E[V/I], m)}{((\text{fun } I \rightarrow E) E', m) \rightarrow ((\text{fun } I \rightarrow E) E', m)}$$

Call-by-name (Lazy Evaluation)

- $(\text{let } I = E \text{ in } E', m) \rightarrow (E' [E/I], m)$
- $((\text{fun } I \rightarrow E') E, m) \rightarrow (E' [E/I], m)$
- Question: Does it make a difference?
- It can depending on the language

Church-Rosser Property

- Church-Rosser Property: If $E \rightarrow^* E_1$ and $E \rightarrow^* E_2$, if there exists a value V such that $E_1 \rightarrow^* V$, then $E_2 \rightarrow^* V$
- Also called **confluence** or **diamond property**
- Example:

$$\begin{array}{c}
 E = 2 + 3 + 4 \\
 \swarrow \quad \searrow \\
 E_1 = 5 + 4 \quad E_2 = 2 + 7 \\
 \searrow \quad \swarrow \\
 V = 9
 \end{array}$$

Does It always Hold?

- No. Languages with side-effects tend not be Church-Rosser with the combination of call-by-name and call-by-value
- Alonzo Church and Barkley Rosser proved in 1936 the λ -calculus does have it
- Benefit of Church-Rosser: can check equality of terms by evaluating them (Given evaluation strategy might not terminate, though)