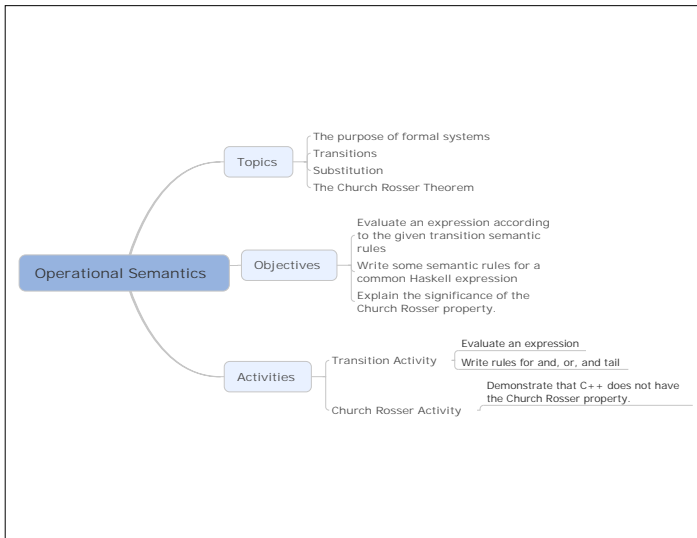


# Operational Semantics

Dr. Mattox Beckman

Illinois Institute of Technology  
Department of Computer Science

# Outline



# Objectives

You should be able to...

In order to express the meaning of a program, we need a formal language to capture these meanings. Today's semantics will use *transitions* to specify the value of an expression. By the end of lecture, you should know how to use transitional semantics.

- what the word “semantics” means.
- determine the value of an expression (i.e., be able to read)
- specify the meaning of a language (i.e., be able to write).

You should also know the Church-Rosser property and be able to give examples of languages that have it and languages that don't have it.

# Parts of a Formal System

To create a formal system, you must specify the following:

- A set of *symbols* or an *alphabet*.
- A definition of a *valid sentence*.
- A set of *transformation rules* to make new valid sentences out of old ones.
- A set of *initial valid sentences*.

You do NOT need:

- An *interpretation* of those symbols.  
They are highly recommended, but the formal system can exist and do its work without one.

# Example

Symbols  $S, (, ), Z, P, x, y$ .

## Definition of a furbitz

- $Z$  is a furbitz.  $x$  and  $y$  are variables of type furbitz.
- if  $x$  is a furbitz, then  $S(x)$  is a furbitz.
- if  $x$  and  $y$  are furbitz, then  $P(x, y)$  is a furbitz.

## Definition of the gloppit relation

- $Z$  has the gloppit relation with  $Z$ .
- If  $x$  and  $y$  have the gloppit relation, then  $S(x)$  and  $S(y)$  have the gloppit relation.
- If  $\alpha$  and  $\beta$ , then we can write  $\alpha g \beta$ .

True Sentences If  $\alpha g \beta$ , then also

- $P(S(\alpha), \beta) g P(\alpha, S(\beta))$ , and  $P(Z, \alpha) g \alpha$

# Example

Symbols  $S, (, ), Z, P, x, y$ .

## Definition of an integer

- 0 is an integer.  $x$  and  $y$  are variables of type integer.
- if  $x$  is an integer, then  $S(x)$  is an integer.
- if  $x$  and  $y$  are integers, then  $P(x, y)$  is an integer.

## Definition of the equality relation

- 0 has the equality relation with 0.
- If  $x$  and  $y$  have the equality relation, then  $S(x)$  and  $S(y)$  have the equality relation.
- If  $\alpha$  and  $\beta$ , then we can write  $\alpha = \beta$ .

True Sentences If  $\alpha = \beta$ , then also

- $P(S(\alpha), \beta) = P(\alpha, S(\beta))$ , and  $P(0, \alpha) = \alpha$

# Example

Symbols  $S, (, ), Z, P, x, y$ .

## Definition of an integer

- 1 is an integer.  $x$  and  $y$  are variables of type integer.
- if  $x$  is an integer, then  $S(x)$  is an integer.
- if  $x$  and  $y$  are integers, then  $P(x, y)$  is an integer.

## Definition of the equality relation

- 1 has the equality relation with 1.
- If  $x$  and  $y$  have the equality relation, then  $S(x)$  and  $S(y)$  have the equality relation.
- If  $\alpha$  and  $\beta$ , then we can write  $\alpha = \beta$ .

## True Sentences If $\alpha = \beta$ , then also

- $P(S(\alpha), \beta) = P(\alpha, S(\beta))$ , and  $P(1, \alpha) = \alpha$

# Transformations

- There are many ways we can specify the meaning of an expression. One way is to specify the steps that the computer will take during an evaluation.
- An *evaluation* has the following form:

$$e_1 \rightarrow e_2$$

where  $e$  is some expression, and  $e_2$  is another expression, possibly a value.

Examples:

- `if true then 4 else 38`  $\rightarrow$  `4`
- `13 + 4 * 5`  $\rightarrow$  `13 + 20`
- Note well:  $\rightarrow$  indicates *exactly one* step of evaluation.



# Preliminaries

- In transition semantics we need to be able to distinguish between *values* and *expressions*.
  - A *value* is a valid *expression* that can not be evaluated any further.
  - (Note, the converse is not true.)
- Use letters  $U$ ,  $V$ , and  $W$  to represent values.
- Use letters  $M$ ,  $N$ , and  $L$  to represent expressions.

# If Statements

Here are three semantic rules for the if statement.

- $\text{if true then } M \text{ else } N \rightarrow M$
- $\text{if false then } M \text{ else } N \rightarrow N$
- $$\frac{L \rightarrow L'}{\text{if } L \text{ then } M \text{ else } N \rightarrow \text{if } L' \text{ then } M \text{ else } N}$$

In English:

- If the conditional part is true, evaluate the first branch.
- If the conditional part is false, evaluate the second branch.
- Otherwise, if the conditional part is not yet evaluated, evaluate it one step.

# Obvious Rules

- These rules are boring. But we need to include them anyway.

$$\frac{M \rightarrow M'}{M \oplus N \rightarrow M' \oplus N} \qquad \frac{N \rightarrow N'}{V \oplus N \rightarrow V \oplus N'}$$

Where  $\oplus$  is  $+$ ,  $-$ ,  $>$ ,  $<$ , ...

- These rules are so boring that we don't include them.

$$0 + 0 \rightarrow 0 \quad 0 + 1 \rightarrow 1 \quad \dots$$

$$1 + 0 \rightarrow 1 \quad 1 + 1 \rightarrow 2 \quad \dots$$

*et cetera...*

# Example Evaluation

Evaluate: if 3 > 2 then 5 + 9 else 2 \* 4

if 3 > 2 then 5 + 9 else 2 \* 4

→ if true then 5 + 9 else 2 \* 4

→ 5 + 9

→ 14

---

# Other Notations

## Notations

$\rightarrow^0 \equiv$  The identity

$\rightarrow^1 \equiv \rightarrow$

$\rightarrow^n \equiv \rightarrow \cdot \rightarrow^{n-1}$

$\rightarrow^* \equiv \bigcup_{i=0}^{\infty} \rightarrow^i$

$\rightarrow^+ \equiv \bigcup_{i=1}^{\infty} \rightarrow^i$

$a \leftarrow b \equiv b \rightarrow a$

$\leftrightarrow \equiv \rightarrow \cup \leftarrow$

$\leftrightarrow^* \equiv (\rightarrow \cup \leftarrow)^*$

## Example

$3 \rightarrow^* 3$ , and if  $3 > 2$  then  $5 + 9$  else  $2 * 4 \rightarrow^* 14$

# Be careful with $\leftrightarrow^*$

$$a \leftrightarrow^* b \not\equiv a \leftarrow^* b \cup a \rightarrow^* b$$

For example  $a \leftrightarrow^* b$  when

$$a \leftarrow a_1 \rightarrow a_2 \rightarrow a_3 \leftarrow b_2 \leftarrow b_1 \rightarrow b$$

# Substitution

- This particular semantics does not use an environment.
- To express the meaning of variable substitution, we use the substitution operator.
- $[e_1/x]e_2$  means “Replace all occurrences of  $x$  in  $e_2$  with  $e_1$ .”
- So,  $[3/x](2 + x) \Rightarrow (2 + 3)$

## More formally...

$$\begin{aligned}
 [y/x]x &\Rightarrow y \\
 [y/x]z &\Rightarrow z \\
 [y/x](a \oplus b) &\Rightarrow [y/x]a \oplus [y/x]b \\
 [y/x](\text{if } M \text{ then } N \text{ else } O) &\Rightarrow (\text{if } [y/x]M \text{ then } [y/x]N \\
 &\quad \text{else } [y/x]O)
 \end{aligned}$$

Substitution has to be done more carefully for let.



# More formally...

$$\begin{array}{ll}
 [y/x](\text{let } x = M \text{ in } N) & \Rightarrow \text{let } x = [y/x]M \text{ in } N \\
 [y/x](\text{let } z = M \text{ in } N) & \Rightarrow \text{let } z = [y/x]M \text{ in } [y/x]N \\
 [y/x](\text{let rec } x = M \text{ in } N) & \Rightarrow \text{let rec } x = M \text{ in } N \\
 [y/x](\text{let rec } z = M \text{ in } N) & \Rightarrow \text{let rec } z = [y/x]M \\
 & \quad \text{in } [y/x]N
 \end{array}$$

# Example

Evaluate:  $\text{let } x = 2 + 3 \text{ in let } y = x * x \text{ in } x + y$

$\text{let } x = 2 + 3 \text{ in let } y = x * x \text{ in } x + y$

$\rightarrow \text{let } x = 5 \text{ in let } y = x * x \text{ in } x + y$

$\rightarrow \text{let } y = 5 * 5 \text{ in } 5 + y$

$\rightarrow \text{let } y = 25 \text{ in } 5 + y$

$\rightarrow 5 + 25$

$\rightarrow 30$

---

# Activity

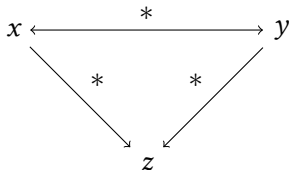
Do the Operational Semantics activity.

# Term Rewriting Systems

Transition semantics can be thought of as a *term-rewriting system*. Common questions:

- Does an expression always terminate?
- Can we tell if two expressions are equal?

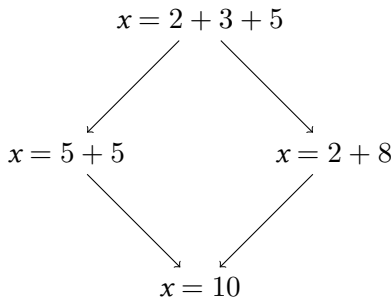
**Church-Rosser Property:** If  $x \leftrightarrow^* y$  then  $x$  and  $y$  normalize to the same value.



# Example

## Confluence

If  $x \rightarrow y_1$  and  $x \rightarrow y_2$  then  $y_1$  and  $y_2$  normalize to the same value.  
(Confluence and the Church-Rosser Property coincide.)



This is also known as the “diamond property”

# Who has it?

- Alonzo Church and J. Barkley Rosser proved that the  $\lambda$ -calculus has these properties in 1936.
- Very important for theorem provers.
- Most programming languages have this property... some of the time...
- One Benefit: you can check for equality of  $x$  and  $y$  by evaluating them.

# Activity

Do the Diamond Property Activity.