

## Programming Languages and Compilers (CS 421)

---

Elsa L Gunter  
2112 SC, UIUC  
<http://www.cs.uiuc.edu/class/sp07/cs421/>

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

## 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

Elsa L. Gunter

## Dynamic semantics

---

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

Elsa L. Gunter

## Dynamic Semantics

---

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

Elsa L. Gunter

## 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

Elsa L. Gunter

## 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

Elsa L. Gunter

## 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*

Elsa L. Gunter

## 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

Elsa L. Gunter

## 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
- Useful for proving properties of programs

Elsa L. Gunter

## 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
$$(C, m) \Downarrow m'$$
$$(E, m) \Downarrow v$$

Elsa L. Gunter

## Simple Imperative Programming Language

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

Elsa L. Gunter

## Natural Semantics of Atomic Expressions

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

Elsa L. Gunter

## Booleans:

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

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

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

Elsa L. Gunter

## 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  $V$

Elsa L. Gunter

## 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  $U \text{ op } V$

Elsa L. Gunter

## Commands

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

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

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

Elsa L. Gunter

## If Then Else Command

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

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

Elsa L. Gunter

## While Command

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

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

Elsa L. Gunter

## Example

---


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

Elsa L. Gunter

## Example

---


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

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

Elsa L. Gunter

## Example

---

? > ? = ?

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

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

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

Elsa L. Gunter

## Example

---

7 > 5 = true

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

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

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

Elsa L. Gunter

## Example

---

7 > 5 = true

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

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

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

Elsa L. Gunter

## Example

---

7 > 5 = true

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

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

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

$$\Downarrow ?$$

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

Elsa L. Gunter

### Example

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

Elsa L. Gunter

### Example

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

Elsa L. Gunter

### Example

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

Elsa L. Gunter

### Example

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

Elsa L. Gunter

### Example

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

Elsa L. Gunter

### Example

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

Elsa L. Gunter

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

Elsa L. Gunter

## Example

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

Elsa L. Gunter

## Example

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

Elsa L. Gunter

## 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

Elsa L. Gunter

## 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

Elsa L. Gunter

## 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

Elsa L. Gunter

## 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

Elsa L. Gunter

## 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  $\text{compute\_com}(c2, m)$

Elsa L. Gunter

## 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

Elsa L. Gunter