

Programming Languages and Compilers (CS 421)

Elsa L Gunter
2112 SC, UIUC



<https://courses.engr.illinois.edu/cs421/sp2023>

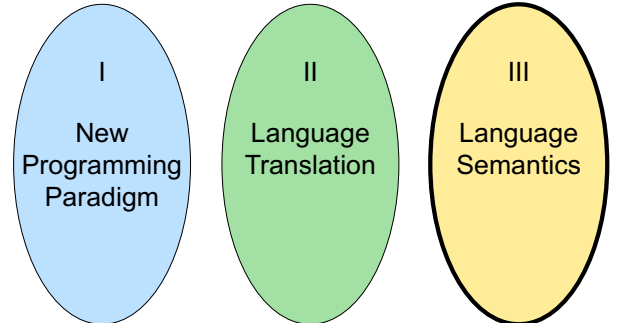
Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

4/12/23

1

Programming Languages & Compilers

Three Main Topics of the Course

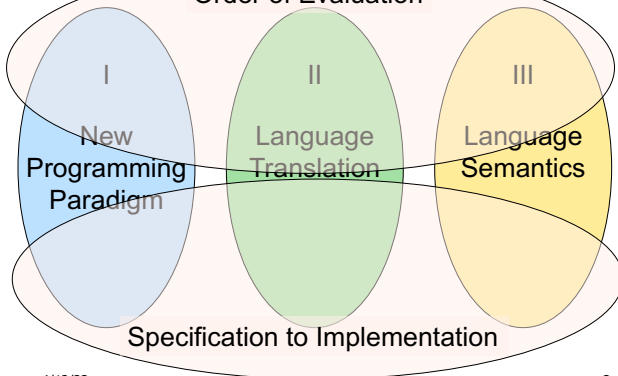


4/12/23

2

Programming Languages & Compilers

Order of Evaluation



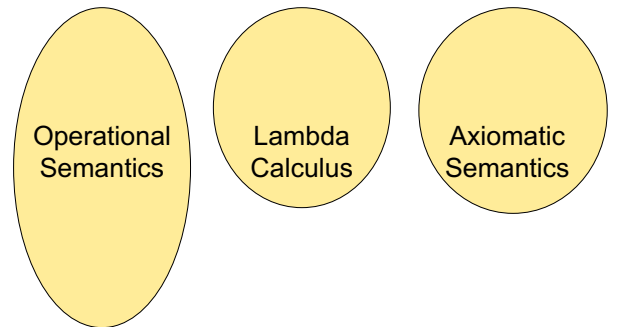
Specification to Implementation

4/12/23

3

Programming Languages & Compilers

III : Language Semantics

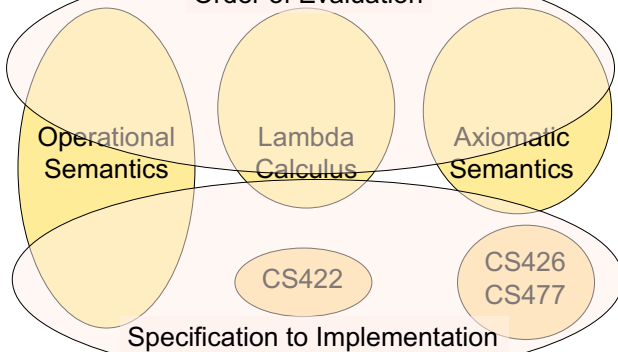


4/12/23

4

Programming Languages & Compilers

Order of Evaluation



Specification to Implementation

4/12/23

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

4/12/23

6

Dynamic semantics

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

4/12/23

7

Dynamic Semantics

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

4/12/23

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

4/12/23

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

4/12/23

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*

4/12/23

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

4/12/23

12

Natural Semantics

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

$$\begin{array}{c} (C, m) \Downarrow m' \\ \text{or} \\ (E, m) \Downarrow v \end{array}$$

4/12/23

14

Simple Imperative Programming Language

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

4/12/23

15

Natural Semantics of Atomic Expressions

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

4/12/23

16

Booleans:

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

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

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

4/12/23

17

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

4/12/23

18

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 $U \text{ op } V$

4/12/23

19

Commands

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

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

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

4/12/23

21

If Then Else Command

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

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

4/12/23

22

While Command

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

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

4/12/23

23

Example: If Then Else Rule

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

4/12/23

24

Example: If Then Else Rule

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

4/12/23

25

Example: Arith Relation

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

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

4/12/23

26

Example: Identifier(s)

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

4/12/23

27

Example: Arith Relation

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

4/12/23

28

Example: If Then Else Rule

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

4/12/23

29

Example: Assignment

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

4/12/23

30

Example: Arith Op

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

4/12/23

31

Example: Numerals

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

4/12/23

32

Example: Arith Op

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

4/12/23

33

Example: Assignment

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

4/12/23

34

Example: If Then Else Rule

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

4/12/23

35

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

4/12/23

36

Example

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

4/12/23

37

Example

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

4/12/23

38



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

4/12/23

39



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

4/12/23

40



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

4/12/23

41