CS4022D Principles of Programming Languages
Lecture #3: Semantics - Part 1

Saleena N
NIT Calicut

August 04, 2021

Semantics

- Meaning of syntactically valid programs
- Required by programmers and implementers (writing compiler/interpreter)
- Mostly described informallly
- Approaches to Formal Semantics
  - Operational, Denotational, Axiomatic

Semantics

- Semantics of C language assignment statement: $id = E$?
  - how programs are evaluated / run-time behavior of programs ?
  - run-time behavior ?

Semantics: Boolean Expressions

Evaluation of *expr1* && *expr2*

false && *expr2*

true && *expr2*

Semantics: Boolean Expressions

Evaluation of *expr*1 && *expr*2

*expr*1 evaluates to *false*

Semantics: Boolean Expressions

Evaluation of *expr*1 && *expr*2

- Short-Circuit Evaluation Semantics
  - if *expr*1 evaluates to *false*, *expr*2 is not evaluated
- Complete Evaluation Semantics
  - both *expr*1 and *expr*2 are evaluated

Semantics: Boolean Expressions

Evaluation of *expr*1 && *expr*2

- *expr*1 evaluates to *false*
- evaluation of *expr*2 results in a run-time error???

Short-circuit / Complete - run-time behavior?

Different Semantics: Different run-time behavior

Evaluation of $f(x)$ && $g(y)$

- Short-Circuit Evaluation Semantics
  - if $f(x)$ evaluates to *false*, $g(y)$ is not evaluated
- Complete Evaluation Semantics
  - suppose $g(y)$ results in a run-time error???

Semantics - informal

- Meaning of x = a+b*c ?
- In general, $< variable > = < expr >$
  - Evaluate *expr*, to a value say *v*
  - store the value *v* to location corresponding to *variable*

Semantics of Assignment [1]

- Assignment Operator *lvalue = expression*

The value of the expression replaces that of the object referred to by the lvalue. The operands need not have the same type, but both must be int, char, float, double, or pointer. If neither operand is a pointer, the assignment takes place as expected, possibly preceded by conversion of the expression on the right. When both operands are int or pointers of any kind, no conversion ever takes place; the value of the expression is simply stored into the object referred to by the lvalue. Thus it is possible to generate pointers which will cause addressing exceptions when used.

[1] from the C language reference manual

Formal Semantics

- concerned with rigorously specifying the meaning, or behavior of programs
  - standardization
  - basis for implementation, analysis, and verification

Formal Semantics: Approaches

- Operational semantics
- Denotational semantics
- Axiomatic semantics

Operational Semantics

- meaning of a construct is specified by the computation it induces when it is executed on a machine (mostly an abstract machine).
- the interest is on how the effect of a computation is produced.
- meaning of $x = y + z$?
  - evaluate the expression $y + z$ in the current state[2]
  - assign the value to variable x, resulting in a new state [3]
- meaning of $x = y + z; a = x$?

[2] Program state: Mapping of values to variables (simple view)

[3] Assignment causes a change in state (side effect)

Operational Semantics

- Meaning of expression (1+2)*(3+4)

  $\underline{(1+2)} * (3+4) \rightarrow 3 * \underline{(3+4)} \rightarrow \underline{3*7} \rightarrow 21$

  sequence of internal steps of computation
- *Intensional Semantics* - sequence of internal steps of computation is important

Operational Semantics

- two different Operational Semantics for (1+2)*(3+4)

$$\underline{(1+2)} * (3+4) \rightarrow 3 * \underline{(3+4)} \rightarrow \underline{3*7} \rightarrow 21$$

$$(1+2) * \underline{(3+4)} \rightarrow \underline{(1+2)} * 7 \rightarrow \underline{3*7} \rightarrow 21$$

- Factorial function - differently coded functions may have different semantics

Boolean Expression - Operational Semantics

- Language Syntax

$$
\begin{aligned}
B ::= \ & \mathit{true} \\
& \mathit{false} \\
& B \wedge \ B
\end{aligned}
$$

Boolean Expression - Operational Semantics

- Language Syntax

  $B ::= \mathit{true}$

  $\quad\quad \mathit{false}$

  $\quad\quad B \wedge B$

- Some sentences (strings) in the language:

  true

  false

  $\mathit{true} \wedge \mathit{false}$

  $\mathit{true} \wedge \mathit{false} \wedge \mathit{false}$

  $\mathit{false} \wedge \mathit{false} \wedge \mathit{false} \wedge \mathit{true}$