Language Engineering - A nice set of notes

Josh Felmeden

November 8, 2019

Contents

1 Introduction to Semantics

Semantics are really complex and they actually exist in the real world as problems that can arise when the semantics are unclear. In the example of the Derek Bentley case, Bentley tells Chris (who is holding a gun, and a policeman standing in front of him to 'let him have it!'. Here, it appears that he could be talking about the gun, or to kill him. The same kind of thing can happen in computing when we are unsure of the references of certain objects.

Here are some examples learned from natural languages:

- Syntactic complexity
  - Jack built the house the malt the rat the cat killed ate lay in
- Syntactic ambiguity
  - Let him have it, Chris!
- Semantic Complexity
  - It depends on what the meaning of the word 'is' is!
- Semantic ambiguity
  - I haven't slept for ten days
- Semantic undefinedness
  - Colourless green ideas sleep furiously
- Interaction of syntax and semantics
  - Time flies like an arrow, fruit flies like a banana.

We can apply these things to computing terms, too.

- Syntactic complexity

  ```
  x-=y = (x=x+y) - y      //switches variables x and y
  ```

- Syntactic ambiguity

  ```
  if (...) if (...) ..; else ..       //dangling else
  ```

- Semantic Complexity

  ```
  y = x++ + x++       //sequence points
  ```

- Semantic ambiguity

  ```
  (x%2=1) ? "odd" : "even"        //unspecified in C89 if x<0
  ```

- Semantic undefinedness

```
while(x/x)      //division error or infinite loop
```

- Interaction of syntax and semantics

```
A * B       //lever hack
```

To put this another way:

- **Syntax**: concerned with the form of expressions and whether or not the program actually *compiles*
- **Semantics**: concerned with the meaning of expressions and what the program does when it *runs*
- **Pragmatics**: concerned with issues like design patterns, program style, industry standards, etc.

2 Structural Operational Semantics

We're going to look at doing some compilation (of the *while*) language.

2.1 Termination and looping

The execution of the statement S in state σ terminates iff there exists a finite derivation sequence from $\langle S, \sigma \rangle$. The derivation sequence looks like:

$$\langle S, \theta \rangle \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n \text{ where } \gamma_n \text{ is terminal } \sigma' \text{ or stuck } \langle S', \sigma' \rangle$$

The while language never gets stuck, but some language might if we try to divide by zero because we don't know how to process this.

The execution of the statement S in a state θ loops iff there exists an infinite derivation sequence from $\langle S, \sigma \rangle$

$$\langle S, \sigma \rangle \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \dots$$

S always terminates iff its execution terminates in all states σ.

S always loops if the execution loops in all states σ.

The execution of statement S in state σ terminates successfully iff it ends with a terminal configuration.

Note while has no stuck configurations, so termination implies successful termination!

2.2 Determinism and Equivalence

The structural operation semantics is (strongly) **deterministic** iff $\langle S, \sigma \rangle \Rightarrow \gamma$ and $\langle S, \sigma \rangle \Rightarrow \gamma'$ imply that $\gamma = \gamma'$ for all $S, \sigma, \gamma, \gamma'$

It is **weakly deterministic** iff $\langle S, \sigma \rangle \Rightarrow^* \sigma'$ and $\langle S, \sigma \rangle \Rightarrow^* \sigma''$ imply that $\sigma' = \sigma''$ for all $S, \sigma, \sigma', \sigma''$. This is different from the strong determinism above because it says that for every successfully terminating branch, (it doesn't matter how we get there) we get to the same final state.

Two statements are **semantically equivalent** whenever it holds that for *all states* σ

$$\langle S_1, \sigma \rangle \Rightarrow^* \gamma \text{ iff } \langle S_2, \sigma \rangle \Rightarrow^* \gamma \text{ whenever } \gamma \text{ is terminal or stuck}$$

This means that there is an infinite derivation sequence from $\langle S_1, \sigma \rangle$ iff there is an infinite derivation from $\langle S_2, \sigma \rangle$.

Note! The length of these could be different (because of the * again.)

For a deterministic structural operational semantics, we can define a semantic function as follows:

- $S_{sos}[[.]]$ Stm $\rightarrow$ (State $\nrightarrow$ State)

3 Chain-Complete Partial Orders

Sets can have upper and lower bounds depending on where they come in the order of chains. For example, those at the top of the set are the upper bound of every element since they are unable to have an upper bound themselves, while the ones at the bottom are the lower bound, since they have no lower bound themselves.

3.1 Definitions of PO-Set (partially ordered set)

A PO-Set (D, $\sqsubseteq$) is called a *chain-complete* partially ordered set (ccpo) whenever the least upper bound $\sqcup y$ exists for all <u>chains</u> $Y \subseteq D$.

- To be a CCPO, each chain must have an upper bound

Furthermore, a PO-Set (D, $\sqsubseteq$) is called a *complete lattice* whenever the least upper bound $\sqcup Y$ exists for all <u>subsets</u> $Y \subseteq D$.

- To be a lattice, each chain must have a **

Note that every CCPO (D, $\sqsubseteq$) has a (necessarily unique) element denoted $\bot = \sqcup\, \emptyset$. This means that it is given by the lease upper bound of the empty chain. First observe $\emptyset$ is a chain, since we know that $\emptyset \subseteq D$ by the basic set properties and $d \sqsubseteq e$ vacuously holds for all $d, e \in \emptyset$ (of which there are none). So, the lease upper bound $\bot$ **

The question is: is our relation a chain-complete partial order (from the slides)? The answer, simply, is no.

- This is because we have a whole bunch of items at the end. There is no least element of the ordering, so it cannot be a CCPO (this is an important part of the CCPO).

What we can do is the **lifted** relation, which we obtain by adding a least element $\perp$. But, does this now make it a CCPO?