CMSC330: Operational Semantics

Chris Kauffman

Last Updated:
Tue Nov 7 09:15:45 AM EST 2023

Logistics

Reading

TBD

Goals

- Notation and Mechanics of Operational Semantics
- A few Applications
- Practice Problems

Assignments

- Project 6 Posted: Lambda Calculus Interpreter, Due 15-Nov
  - Lexer / Parser
  - Evaluator with...
  - Eager + Lazy Reductions
- Exam 2 Thursday
  - OCaml coding
  - Lexing/Parsing/Eval
  - Operational Semantics

OCaml Practicum?

Finishing up Operational Semantics Discussion is the priority right now but time permitting, may post a video of solving a practical problem in OCaml. Though there are other, Rusty, fish to fry...

How many of you viewed / benefited from the Python Practicum video?

Announcements

Exam 2 Review Session

“We will be hosting an exam 2 review session Tuesday 07-Nov at 6pm in IRB0324.”

https://piazza.com/class/lkimk0rc39wfi/post/1463

Semantics Informally and Formally

semantics (noun): *The branch of linguistics and logic concerned with meaning. There are a number of branches and subbranches of semantics, including formal semantics, which studies the logical aspects of meaning,*

Natural Languages

- Populations of humans ascribe a shared meaning to words
- Meanings vary according to population and period

Programming Language Semantics

What does the following syntax DO in language X?

Informal Semantics

- Creator of Language X describes in words what its syntax does
- Write a parser + interpreter / compiler that reflects that meaning
- May add features, update, alter semantics

  Python 2005: `print "Hello!"` prints Hello!

  Python 2009: `print "Hello!"` prints Syntax Error

Formal Semantics

- Attempts to describe with some mathematical rigor the meaning of Programming language statements
- Comes in several flavors, equipped with jargon / notation
- Useful to quickly describe to humans small features of languages for comparison
- Used by some in proofs about properties of languages and programs in those languages, also to guide development of language interpreters

Operational Semantics

- Several flavors of Formal Semantics exist of which **Operational Semantics (OpSem)** is one
- OpSem focuses on relating syntax of language to behavior of an abstract machine
- High variance on which machine to target, how machine operations are described, etc.
  - Provide actual assembly instructions
  - Describe instructions in an abstract machine
  - Describe what would happen in another PL
  - Describe in English sentences what is happening
- Referred to as the **Meta-Language**: description of what the target language does
- The persistent character is usually the **notation** used which is new and takes some getting used to

OpSem Notation

- Specifics of notation for OpSem vary
- Will turn to some standing slides for CMSC330 for the moment to ensure compatibility with Prof Bakalian's treatment
- Posted as "Reference Slides", come from Spring 2021 Offering of CMSC330 with other materials here: https://www.cs.umd.edu/class/spring2021/cmsc330/

L'Maco: Practice with OpSem

L'Maco has familiar ideas with slightly unfamiliar syntax

Sample Expressions

```
add 5 and 2
```

```
with 7 as z
add z and 2
```

```
with add 1 and 2 as x
with add x and 7 as y
add x and y
```

CFG for L'Maco

$$
\begin{aligned}
W &\rightarrow \texttt{with}\, E\, \texttt{as}\, V\; W \\
E &\rightarrow C \mid V \mid \texttt{add}\, E\, \texttt{and}\, E \\
V &\rightarrow \text{variable name} \\
C &\rightarrow \text{constant number}
\end{aligned}
$$

L'Maco with Environments

The following (with) and (add) rules specify the semantics of L'Maco using Environments;

```
A;E1=>N1   A,V:N1; E2 => N2                        A(x)=>v
--------------------------- (with)                 ------- (var-lookup)
A; with E1 as V E2 => N2                           A; x=>v


A; E1=>N1   A; E2=>N2  N1+N2 is N3
---------------------------------- (add)           ---- (constants)
    A; add E1 and E2 => N3                         C=>C
```

Note use of environments: (with) rule allows extension of environments with new bindings

Exercise: L'Maco Big Derivation

Fill in the first step in this derivation

Hint: work left to right...

```
         ????????             ????????????????
================================================================
[]; with add 1 and 2 as x with add x and 7 as y add x and y =>13
```

Reference Rules

```
A;E1=>N1   A,V:N1; E2 => N2          A; E1=>N1   A; E2=>N2  N1+N2 is N3
--------------------------- (with)   ---------------------------------- (add)
A; with E1 as V E2 => N2                 A; add E1 and E2 => N3
```

Answers: L'Maco Big Derivation

- According to the CFG syntax, the (with)-rule is applicable first
- Matches the general idea of "bind name, use name"
- Leads to the first steps in the derivation tree

```
[]; add 1 and 2=>3                    [x:3]; with add x and 7 as y add x and y =>13
===================================================================================
    []; with add 1 and 2 as x with add x and 7 as y add x and y =>13
```

Complete the Left Branch with the (add) rule

Reference Rules

```
A;E1=>N1   A,V:N1; E2 => N2          A; E1=>N1   A; E2=>N2  N1+N2 is N3
--------------------------- (with)   ---------------------------------- (add)
A; with E1 as V E2 => N2                 A; add E1 and E2 => N3
```

Answers: L'Maco Big Derivation Left Branch

```
==== ====
1=>1 2=>2 3 is 1+2
==================
[]; add 1 and 2=>3                    [x:3]; with add x and 7 as y add x and y =>13
================================================================================
    []; with add 1 and 2 as x with add x and 7 as y add x and y =>13
```

Complete the Right Branch

It's of some girth but starts with another (with)

Reference Rules

```
A;E1=>N1   A,V:N1; E2 => N2            A; E1=>N1   A; E2=>N2  N1+N2 is N3
--------------------------- (with)   ---------------------------------- (add)
A; with E1 as V E2 => N2                   A; add E1 and E2 => N3
```

Answers: L'Maco Big Derivation

```
                    ==========  ====               ===============  ================
                    [x:3] x=>3  7=>7 10 is 3+7     [x:3,y:10] x=>3  [x:3,y:10] y=>10 13 is 3+10
==== ====           ==========================     ============================================
1=>1 2=>2 3 is 1+2     [x:3] add x and 7=>10                   [x:3,y:10]; add x and y=>13
==================     ============================================================
[]; add 1 and 2=>3               [x:3]; with add x and 7 as y add x and y =>13
==============================================================================
    []; with add 1 and 2 as x with add x and 7 as y add x and y =>13
```