COMP 150PP Class Exercise: Semantics

September 21, 2016

Semantics review: A simple functional language

Let's imagine a little functional language of expressions, with this syntax:

$$e \Rightarrow \lambda x.e \mid e_1\ e_2 \mid x \mid v \mid \textbf{if}\ e_1\ e_2\ e_3$$

Let's further imagine that this languages is endowed with a rich collection of values, including booleans, lists, primitive functions, closures, pairs, sums, and several kinds of numbers. And let's assume that the initial basis contains everything necessary for the basic types, plus a rich set of arithmetic and relational primitives.

1. Please give the language an operational semantics using the judgment form $\langle e, \rho \rangle \Downarrow v$.

A simple probabilistic language

Please tackle the three questions in this section:

2. Extend our language so it can encode probabilistic simulations. You may add syntax, primitive functions, or both. Do *not* try to support inference.
3. Think about what could reasonably be considered an operational semantics for a probabilistic language. I can imagine two reasonable alternatives.

   Then, write the evaluation judgment and rules of such a semantics.

Another way to define a language is by translating it into a known formalism. This is the technique used in *denotational* semantics. And we have spent the last three class periods developing a suitable formalism for expressing probabilistic computation. (Actually, three very similar formalisms.) Allow me to extend the in-class formalism with the following:

Type `Exp`	The syntax of the functional language above
Type `Value`	The values of the functional language above
Constructor `Clo :: Name -> Exp -> Value`	Builds closures
Type `Env`	Finite map from `Name` to `Value`
Function `applyDPrim :: Value -> Value -> Value`	Deterministic primitives
Function `applyPrim :: Value -> Value -> Dist Value`	General primitives (if needed)

4. Write the translation `eval :: Exp -> Env -> Dist Value`. You can start with the deterministic language, then add the probabilistic extensions.

Bonus question

This one is intended for you to think about at home, although if you get stuck writing semantics, it may help you get unstuck:

5. Design a type system for our language. To keep things simple, write nondeterministic typing rules, so that we can imagine type inference without requiring any annotations.