COMP 150PP: Contributions of probabilistic programming languages

October 19, 2016

Contributions of $\lambda_\circ$

As far as we understand $\lambda_\circ$, it seems to be the Ramsey/Pfeffer sampling monad, plus the ability to sample floating-point numbers uniformly from the unit interval, plus the mysterious Bayes operator, which we don't understand (and which has a nondeterministic, statistical semantics). In this context,

1. What do you think is the contribution of the paper? Why would the referees agree to publish it?

Church

Questions for class: using Church

Both the probability monad and $\lambda_\circ$ offer statically typed languages in which you describe either deterministic computation or probabilistic computation, with a hard distinction between them. The description of a probabilistic computation denotes a distribution. Inference, when it happens at all, seems to be something that is done somewhat "outside" the language.

2. How does Church compare? Is it true that expressions can be "arbitrarily composed"? If so, what implications does that have for programmers? How is the programmer's experience likely to differ from his or her experience using the probability monad or $\lambda_\circ$?

3. For inference $\lambda_\circ$ provides the `bayes` operator. We agreed to stipulate that we don't quite understand what kinds of inferences we can draw (that is, we don't understand what sorts of prior/posterior distributions are available, and we don't understand exactly what observations are available). Using Church's `query` or `lex-query` operation, do you understand what sorts of prior/posterior distributions you can work with, and what sorts of observations you can make? Are they sufficient for the dice problems?

4. Using `query` (or `lex-query`) and `flip`, how would you code the following, familiar inference problem?

   > I roll a 6-sided die to produce a number N, then toss N coins. When asked, I inform you that exactly three of the coins are heads. What do you infer as the posterior distribution over N?

Questions for class: semantics

5. In the discussion of procedure values in section 2, is there a difference between an "ordinary procedure" and a closure?

6. Are there any surprises in Figure 1? Anything different from what you've seen elsewhere?

7. What is meant by an "evaluation history"? Is there some way to define it precisely by appealing only to the language, or is it necessary to appeal to an implementation? (You might consult the proof sketch at the beginning of Section 2.2, which ought to remind you of some pictures drawn in class and some formalisms discussed in class.)

   The end of the second page describes a computation that maps evaluation histories to a distribution on values, which is a function of an expression `e` and environment `env`. Can you explain this paragraph using the vocabulary of COMP 105 and COMP 150PP?

8. What is meant by "expressions are values"?

Other questions

9. Section 2.1 says "we believe that the right notion of purity in a stochastic language is *exchangeability*." Does this claim hide anything as precise as a definition or a theorem? Can you relate this claim to any of the algebraic laws we developed for the probability monad? Can you relate it to any properties of $\lambda_\circ$.

10. If Church "views evaluation as sampling," which is claimed in the introduction, can it be different from $\lambda_\circ$ or the sampling monad?

11. Section 5 says that in Church, generative processes (which is to say, probabilistic models) are first-class objects that can be arbitrarily composed and abstracted. Similarly, in Haskell, values in the probability monad are first-class objects that can be arbitrarily composed and abstracted. Do we understand anything about our ability to use such values in a first-class way? What would that look like? How would it be useful? Would it be different in the two languages?

12. What do you think of the explanation of the "stochastic memoizer," `mem`? Would you rather see a formal definition?

    How do you see the role of the "stochastic memoizer," `mem`? How would you use it? What mechanisms of the probability monad or of $\lambda_\circ$ would you use for the same purpose?