COMP 150PP Class Exercise:
A Probabilistic Language Based on Sampling Functions

October 5, 2016

Comparing designs, again

If P is a probability monad, then the Haskell code presented by Ramsey and Pfeffer offers these functions:

```
-- functions in all probability monads:
return  :: a -> P a
(>>=)   :: P a -> (a -> P b) -> P b
                          -- pronounced "bind"
choose  :: Probability -> P a -> P a -> P a

-- function only in the support monad:
support :: P a -> [a]

-- function only in the expectation monad:
expectation:: (a -> Double) -> P a -> Double

-- function only in the sampling monad:
sample  :: RandomGen g => P a -> g -> (a, g)
```

The language $\lambda_\circ$ of Park, Pfenning, and Thrun is described in Figures 1 and 2 on page 4:8. A sampling semantics is given in Figure 3 on page 4:9. I also recommend close scrutiny of the example at the end of Section 8.2 on page 4:35, particularly the derivation of Update Equation (4)—for me, this example illuminates both the strengths and the limitations of $\lambda_\circ$.

1. If you're given a Haskell expression in the sampling monad, how would you translate it into $\lambda_\circ$? (Assume that you are given a translation from *pure* Haskell expressions into *terms* of $\lambda_\circ$. For our purposes, a *pure* expression is one whose type does not mention the probability monad, and all of whose subexpressions are also pure.)

   If you need to extend $\lambda_\circ$ to express the translation, do so.

2. If you're given an *expression* or a *term* in $\lambda_\circ$, how would you translate it into Haskell code that uses the probability monad?

   If you need to extend the probability monad to express the translation, do so.

Deeper analysis of the paper

3. Park, Pfenning, and Thrun claim that their representation scheme is "sufficient for all practical purposes." Using all of the example problems from the dice world, say whether you agree or disagree.

4. On page 4:13, Park, Pfenning, and Thrun list their major achievements as

   - A unified representation scheme for probability distributions
   - Rich expressiveness (can encode a lot of distributions)
   - High versatility in encoding probability distributions

   Is there anything missing here that you would like a probabilistic language to have or to do?

An observation and some questions to ponder as class ends

5. In the last paragraph of Section 3.2, which spans pages 4:10 and 4:11, I observe some old wine in new bottles: once you get into the I/O monad, you can't escape the I/O monad. Something similar appears to be true of the probability monad... except for that pesky expectation thing.

6. Language people like abstraction, and this paper talks about it a lot. In light of the discussion on page 4:25, how much abstraction do you see, really?

7. Can inference from observation be encoded using the `bayes` operator? Can the `bayes` operator be encoded using inference from observation?

8. What, really, is the contribution of this paper?

9. What is the most important problem left unsolved?