Discussion questions for *Imperative Functional Programming*
COMP 150FP

September 20, 2013

Today's questions

1. Most of the figures in the paper contain translations that prove equivalence of expressive power. For example, Figure 1 shows the `Dialogue`s can simulate the I/O monad and vice versa, but using the I/O monad to simulate `Dialogue`s causes both duplicated work and a space leak. How, then, could anyone possibly be interested in the I/O monad?
2. What's the central claim of the paper? What kinds of evidence would you accept as supporting such a claim? How do you judge the merits of the claim advanced in this paper?
3. To what degree do you think this paper has lasting value?
4. Peyton Jones and Wadler claim that although the I/O monad may look like C, it's actually better than C. What's an interesting thing you can do with the I/O monad that you can't do in C?

   Suggestions:

   - Remember what Hughes said about reuse and composition. You might think about what kinds of applications make heavy use of I/O or interaction and could benefit from some composable, reusable parts.
   - Try to come up with an actual programming example. You don't have to write code, but do try to develop the example in enough detail that you *could* write code if you had the time.
5. Mutable variables are lumped in with the I/O monad. Why?
6. What are the limitations of the techniques proposed in this paper? Which limitations appear significant?
7. This paper and the Hutton/Meijer paper both use the "monad" abstraction. (Hutton and Meijer helpfully explain `do` notation, which has become standard.) Generalizing from these two examples, how would you answer the question "what is a monad?"[1]
8. In the context of your answers to question 7, can you say what the monad laws accomplish?

I've set two programming-practice problems on monads.

Questions to think about later

9. Is there something the I/O monad can't do that you want to do?
10. How would you improve on the I/O monad?
11. Is it reasonable to treat the "world" type as a basic type like integer, float, and double? How would this work?
12. Why is it safe to abandon the "world" values at code-generation time?

[1] It is OK to have more than one answer. Sometimes a candy is also a breath mint.