A Bit of Haskell and Concluding Remarks

12-6-2004

Opening Discussion

- What are some of the neat features we talked about in Haskell?
- Do you have any questions about the assignment?

Sections and Prefix Operators

- Haskell allows the partial evaluation of infix operators (since they are curried).
  - (+x) is like \y -> y+x
- Similarly, by putting an operator in parentheses we get a prefix operator. This is just like putting “op” before an operator in ML.
- A prefix operator can be changed to an infix operator by putting backwards single quotes around it.
  - 5 `elementOf` lst
- Precedence of infix ops can be set.

Lazy Evaluation

- Haskell can have the infinite lists we saw before because it uses lazy evaluation. That implies that a function doesn't evaluate its arguments unless it actually needs them.
- Let's look at some code that demonstrates the effect of lazy evaluation.

let and where

- Haskell and a let similar to ML, but without the end.
- Haskell also has a where clause that can go after all the matches in a function or case statement. This allows you to define a value that is used across many patterns in a function.
  - f x y | y<z = ... | y==z = ... | y>z = ... where z=x*x

Type Classes

- Haskell allows you to define things called type classes. A type class produces a type of polymorphism that is more restrictive than the parametric polymorphism of the parameterized types.
- You define a set of functions that must be defined for types in that class.
- You can also use inheritance to make subtypes of type classes.

Course Recap

- During this course you have learned how to program in two distinct functional programming languages: Scheme and ML.
- In doing this, you have learned how to think in a functional way doing things like using recursion to repeat processes. You have also mastered the use of immutable lists to store and process data.
- Hopefully you have also improved your general programming capability and can tackle more different types of problems.

The Meaning of Types

- One benefit of programming in both Scheme and ML has been that you have been exposed to two very different ways of handling type information.
- Hopefully this has led you to having a much deeper understanding of what types really are and their role in all programming languages.
- With any luck, the next time you write a program in C, C++, Java, or some other language, you will think more about types.

Course Objectives

- While the most significant aspect of this course was to give you an understanding of functional programming, I also had other objectives that I laid out at the beginning of the semester.
- The most important of these was to make you think. Next year you might well not remember the syntax a case or cond, but you should still take with you the concepts of the functional approach and be able to view problems from a different perspective.

Course Evaluation

- There is no minute essay today. Instead, you will fill out course evaluations. These are very important. Next semester, I will read them and I use the feedback to help me the next time I teach the course. The University also takes them very seriously so please give them thought.
- I'll send out an e-mail about a final review time.
- Assignment #9 is due today and #10 is due on the 16th.