# Concluding ML

11-19-2004

## Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?  I'll be putting up an early description of assignment #9 up soon.
- How well do you have to use the octree?

## ML is Functional

- ML is a functional language like Scheme and when programmed in a functional way, there are no side effects.
- Also, being functional, we have the ability to easily create higher order functions.  This means that we can pass functions into other functions or have functions return functions. A common case of the latter is currying which ML makes quite easy to do.
- Repetition is achieved through recursion.

## ML is Strongly, Statically Typed

- One of the main ways that ML differs from Scheme is in the typing system.
- Scheme is completely dynamically typed. Outside of unbalanced parentheses, pretty much all errors are runtime errors.
- ML has static typing for everything, but in general it doesn't require you to tell it the types of things. It figures that out for you. It also has polymorphic types so it makes your code as general as possible.
- Learning to understand error messages.

## Patterns

- Another way ML differs from Scheme is the use of patterns. We use patterns in functions, case statements, and exception handling.  The basic idea is that we can have different chunks of code execute depending on the pattern of an argument. This can almost eliminate if statements.
- Patterns also give us a nice way to pull parts out of constructs like lists, tuples, records, and datatypes. This can almost eliminate functions like hd, tl, and #1.

## Custom Datatypes in ML

- Another significant difference between ML and Scheme is that ML allows us to build custom datatypes. These constructs are similar to enums or unions in C, but are type safe and far more useful. They allow us to build recursive data types without having pointers.
- Being able to use these datatypes requires having patterns.  That's a big part of why this style of datatype isn't used in other languages.

## Modules in ML

- ML also provides us with constructs for doing larger scale programming, particularly in the form of structures and things related to them.
- By providing our own signatures to structures we can hide some of their details.
- Using functors ML lets us create new structures from existing ones.
- Further hiding of implementation is given with local declarations, abstract types, and opaque signatures.

## Looking at Code

- Let's go look at the code for the XML parser that I have put on the web. We can also look at the binary search tree code that we did. Between the two, we can see most of the interesting features of ML in use.

## Minute Essay

- What features of ML do you like the best? What features do you like the least? If you could translate some ML features to another programming language, what features would you want to put in your favorite language?
- Remember that assignment #8 is due on Monday.