Course Wrapup

Prof. Clarkson

Summer 2016

Today’s music: Theme Songs to Red Cliff 赤壁 Parts I and II by Alan 阿兰
Functional Programming

- We've had 8 lectures and 8 labs over 1 week on OCaml
- Things we've learned that might have been new:
  - using a top-level
  - immutable variable bindings
  - higher-order functions, esp. map and fold
  - pattern matching
  - tuples, variants, algebraic datatypes
  - implementing interpreters

- Regular semester course CS 3110 has about 25 lectures and 25 labs over 16 weeks
  ...so don't worry if you don't feel like an expert yet
TWO IDEAS I HOPE YOU’LL REMEMBER...
1. Syntax and Semantics

• Every language feature can be defined in isolation from other features, with rules for:
  – syntax
  – static semantics (typing rules)
  – dynamic semantics (evaluation rules)

• Divide-and-conquer!

• Entire language can be defined mathematically and precisely

• Learning to think about software in this “PL” way will make you a better programmer even when you go back to old ways
  – And give you the mental tools and experience you need for a lifetime of confidently picking up new languages and ideas
2. Benefits of immutability

- Programming becomes simpler: No need to think about aliasing or copying or pointers...

- Can even program in a functional style inside of imperative languages

- Concurrent programming easier with immutable data

  "Maximum reliance on immutable objects is widely accepted as a sound strategy for creating simple, reliable code."
  [http://docs.oracle.com/javase/tutorial/essential/concurrency/immutable.html]

- But: mutability is appropriate when you need to model inherently state-based phenomena, or implement some efficient data structures
What next?

• Study more OCaml:
  – Read *Real World OCaml*
  – Solve 99 OCaml Problems:  

• Study another functional(-inspired) language:
  – Haskell, Racket, Scala, Clojure, ...

• Study more about programming languages:
  – Compilers (how to implement programming languages)  
    e.g., *Modern Compiler Implementation in ML* by Andrew Appel  
  – Theory of programming languages (how to define and reason about programming languages)  
    e.g., *Types and Programming Languages* by Benjamin C. Pierce
Problem Set

• Spend the final day of this course on a problem set
• An individual homework
• Solve as many problems as you can before 4 pm
• You do not need to solve all the problems
• Submit by 4 pm deadline
• You may use online resources, esp. course notes and OCaml manual
• But do not search for answers to these specific problems
• Prof. Clarkson office hours in Gates 461:
  – now until noon
  – 1:30-3:30 pm
THE END