CptS 355- Programming Language Design

Functional Programming

Instructor: Sakire Arslan Ay

WASHINGTON STATE UNIVERSITY

World Class. Face to Face.

Discussion:

Consider everything we learned so far in CptS 355 about functional programming and Haskell .

- What was your biggest takeaway ?
- What did you struggle the most?
- Do you plan to expand your knowledge in functional programming?

Functional Programming

Why spend 40% of course using *functional languages?*

- Rely heavily on recursion;
- Functions are treated as first class objects;
- Higher-order functions are very convenient;
- One-of-a-kind types via constructs like *data* types.

Because:

1. These features help to write correct, elegant, efficient software
2. Functional languages have always been ahead of their time
3. Functional languages well-suited to where computing is going

Ahead of their time

- Garbage collection (Java didn't exist in 1995)
- Generics (`List<T>` in Java, C#), much more like ML / Haskell than C++
- XML for universal data representation (like Racket/Scheme/LISP/…)
- Higher-order functions (Javascript, C#, Python, Ruby, now Java, …)
- Type inference (C#, Scala, …)
- Recursion (a big fight in 1960 about this)
- …

The future may resemble the past

- Maybe pattern-matching, currying, etc. will be next

What is happening recently

Other popular functional programming languages (alphabetized)

- Clojure http://clojure.org
- Erlang http://www.erlang.org
- F# http://tryfsharp.org
- Haskell http://www.haskell.org
- OCaml http://ocaml.org
- Scala http://www.scala-lang.org

Some “industry users” lists (surely more exist):

- http://www.haskell.org/haskellwiki/Haskell_in_industry
- http://www.ocaml.org/learn/companies.html
- In general, see http://cufp.org

What is happening recently?

Popular adoption of concepts:

- C# (function closures, type inference, …)
- Java 8 (closures)
- MapReduce / Hadoop

The Languages Together

- Haskell, Python, Java, and Postscript are a useful *combination* for us

	dynamically typed	statically typed
functional	-	Haskell
object-oriented	Python	Java

- *Haskell*: static type checking, polymorphic types, pattern-matching, user-defined datatypes
- *Python*: dynamic type checking, generators/streams
- *Java*: classes, pure OOP

Is this real programming?

- The way we use Haskell can make them seem almost “silly” precisely because lecture and homework focus on language constructs

- “Real” programming needs file I/O, string operations, floating-point, graphics, project managers, testing frameworks, threads, build systems, …
  - Many elegant languages have all that and more

A note on reality

Reasonable questions when deciding to use/learn a language:

- What libraries are available for reuse?
- What tools are available?
- What can get me a job?
- What does my boss tell me to do?
- What is the de facto industry standard?
- What do I already know?