Notes Functional Programming Presentation

Bas Bossink

April 2010

Contents

- Taxonomy = classification

Introduction

Disclaimer

- No *real* project experience with FP
- Just read some books and did a few excercises

Inventory

- Who has used a functional language?
- Who has used a functional language after completing their education?
- Who has written a LINQ query?

Anecdote

- Euclid greek mathematician 300 BC known for:
  - Elements, 13 books containing from the ground up geometry and number theory
  - Euclidean algorithm, efficient algorithm to calculate gcd

> When Ptolemy the First asked Euclid if there was no shorter road to geometry than the Elements, he replied, "there is no royal road to geometry."

About

- Opensource GFDL
- On github
- Cross platform
- Tools used:
  - pandoc (Haskell markup 2 markup transformer)
    - S5 (requires firefox)
    - just HTML, CSS, javascript
  - dot (graph layout engine)
  - Rake (ruby build automation)
  - git (source control)
  - vim (ide)
  - gnuplot (plotting graphs)
  - perl
  - Miktex/latex (generate notes in pdf)
  - imagemagick (svg to png conversion)
    - native svg in firefox ok on linux but broken on windows

Definition

first class citizen

- comparison to OO:
  - objects can be created
  - passed around
  - returned from methods
  - mutated
  - transformed into new objects
- support for operations on functions:
  - composition
  - (partial) application

functions are mathematical

- same input, same output

Example

- for each x there's 0 or 1 y value, never 2 several x's can have the same y

Non-example

- same input x, can have multiple y values

History

Lambda calculus

- Research in foundations of mathematics
- formalise function definition, application, recursion
- typed is computationally weaker but logically consistent

Timeline

- Stripped down history of programming languages
- Lisp one of the first programming languages
  - predates C by 14 years
  - based on untyped lambda calculus
- Scheme attempt to clean-up Lisp
- Erlang also influenced by Prolog
- CLOS Common Lisp Object System: OO added to Lisp
- OCaml is OO added to Caml
- Scala runs on both JVM and .Net
- F# syntax compatible with OCaml
- Haskell 2010 is about to be released
  - Haskell is designed by committee
  - Haskell was designed to focus research of typed, lazy functional languages
- Clojure a Lisp on the JVM and .Net

Taxonomy

- Languages can be classified in different ways
- Next is rough sketch, no scale
- To see who your friends are

Static/Dynamic, Strong/Weak

- Functional languages are mostly strongly typed
- Two basic camps still exist
  - no equivalent of Duck Typing, no *objects* to recieve the message

Lazy/Eager, Single/Multi-paradigm

- Erlang, really functional, Armstrong says: "Concurrency Oriented"
- Common Lisp, CLOS, Common Lisp Object System
- F#:
  - Funtional
  - Imperative
  - Object-Oriented
  - Language-Oriented
- Scala:
  - Functional
  - Object-Oriented
- Lazy here means by default, F# and Scala support lazy with special syntax

Features

Read Eval Print Loop (REPL)

- Also found in:
  - Python
  - Ruby
  - Scala
- F# no interpreter but on the fly compilation

Type inference

- Lately mostly doing Haskell, preperation a bit disappointing F# inference does not use info about signature sqrt Haskell does.
- Haskell :type ghci is different from type of bound expression

Pattern matching

- Examples
- F# supports **Active Patterns** functions to be run as part of pattern matching Makes it possible to match over arbitrary types and parameterize pattern matching.

Algabraic data types

- What's in a name
- Abstraction of tuple/enum

  an algebraic data type (sometimes also called a variant type[1]) is a datatype each of whose values is data from other datatypes wrapped in one of the constructors of the datatype. Any wrapped datum is an argument to the constructor

- Somewhat similar to `union`

Function Composition

- Examples

Anonymous function (lamdba expressions)

- Examples

Closure

> In computer science, a closure is a first-class function with free variables that are bound in the lexical environment. Such a function is said to be "closed over" its free variables.

Java JSE 7 proposed syntax quite ugly

Sequence Expressions (List Comprehensions)

- Think PowerShell ranges on steriods
- Also available in Python and Erlang
- F# generalizes list comprehensions to Sequence Expressions with syntactic sugar for seq, list and array

Partial function application

- Also known as Currying
- show arrows in types

Higher order functions

- functions that take functions as arguments or return functions

Functional Concepts

Tail Recursion

- example
- needed optimization for arbitrary length recursion

Continuation passing style

- example
- naive is **not** tail recursive
  - large and/or unbalanced tree's -> stackoverflow
- continuation passing, pass the function that should be invoked with answer
- both sizeCont left (fun leftSize -> .. are tail recursive

Memoization

- think caching in a closure
- example
- caching in a local function

Monads

- hiding complexity
- think control structures
- chaining computations
- jukky cs example

When to use it?

Stateless

- web/REST

When not to use it?

- a lotta state
- interoperabillity, e.g. Haskell, Erlang only interoperate with C
- team knowledge