Chapter 11

Functional Programming

Lecture Objective

Today's primary lesson:

- Understand the how functional programming languages differ from imperative programming languages

Problem Solving

1901 mathematician David Hilbert

- proposed 23 problems to solve within next 100 years
- 10th problem: Given a polynomial with integer coefficients (called a Diophantine equation), does it have integer roots?

https://en.wikipedia.org/wiki/Hilbert%27s_problems

Polynomials and Integer Roots

Do the following polynomials have integer roots?

- $x^2 - x = 0$
- $3x^2 - 7x - 40 = 0$
- $5x^4 + 2x^2 + 52 = 0$
- $3x^2y - 7xy + 40 = 0$
- $10x^3y^2 - 9x^2y - 3xy + 29 = 0$

Polynomials and Integer Roots

Do the following polynomials have integer roots?

- $x^2 - x = 0$ — Yes, x=1
- $3x^2 - 7x - 40 = 0$ — Yes, x=5
- $5x^4 + 2x^2 + 52 = 0$ — No, no negatives
- $3x^2y - 7xy + 40 = 0$ — Yes, x=5, y=-1
- $10x^3y^2 - 9x^2y - 3xy + 29 = 0$ — ?

Can you write a program which takes an arbitrary polynomial as input and correctly outputs "yes, the polynomial has integer roots" or "no, the polynomial does not have integer roots"?

Effective Procedure

Assumption was that all 23 problems were solvable by some mechanical process in a finite number of operations.

Effective Procedure

A procedure for achieving some desired result is called "effective" (mechanical) if:

- It is set out in terms of a finite number of exact instructions;
- If, when carried out without error, produces the result in a finite number of steps;
- Can (in practice or in principle) be carried out by a human being unaided by any machinery save paper and pencil;
- Demands no insight or ingenuity on the part of the human being carrying it out.

Definitions Effective Procedure

- Alan Turing – automaton - Turing Machine
- Alonzo Church – Lambda Calculus (λ Calculus)
- Stephen Kleene – Recursive Functions (also invented regular expressions)
- Emil Post – combinatorics – Rewriting Systems
- others

Church-Turing Thesis

The previous formalisms are equally powerful and any intuitively appealing model of computing will be equally powerful

Foundations of Language Styles

Imperative	Functional
Turing Machines	λ Calculus

Foundations of Language Styles

Imperative

Turing Machines

computation occurs by changing state

Functional

λ Calculus

computation occurs by 'applying' functions to parameters (which, themselves can be functions applied to parameters)

Also called "applicative programming style"

Example Imperative Program

```
int gcd (int a, int b) {
    while (a!=b) {
        if (a>b) a = a-b;
        else b=b-a;
    }
    return a;
}
```

Example Scheme Program

```
(define gcd
   (lambda (a b)
      (cond ((= a b) a)
            ((> a b) (gcd (- a b) b))
             (else  (gcd (-b a)  a)))))
```

Functional Programming in Industry

http://homepages.inf.ed.ac.uk/wadler/realworld/

Functional Programming in Industry

Priceline, http://www.priceline.com/ , is largely Lisp based

Cleartrip, http://www.cleartrip.com/, is built almost entirely using ANSI Common Lisp

Functional Languages

- Pure functional languages: Miranda, Haskell, pH, Sisal, Single Assignment C
- Nearly pure: Erlang
- Includes imperative features: Lisp, Scheme (dialect of Lisp), Scala (combination of object-oriented and functional)

Lisp

- LISt Processing
- Good for manipulating symbolic data
- Originally used most in AI
- Statically typed Lisp is increasingly being used in scientific and business applications
- Has atoms (symbols or numeric literals) and lists (uses parenthesis to show)
  - car – content of the address part of the register (“first”)
  - cdr – contents of the destination data part of the register (“rest”)
- Scheme is a dialect of LISP which is small and has static scope

Lisp (Scheme)

- Homoiconic
  - Programs look like ordinary data structures
  - Programs can be created, modified and executed “on the fly”
- Self-definition – the operational semantics of Lisp can be defined in terms of an interpreter written in Lisp
- Interaction via “read-eval-print” loop

Imperative Features of Lisp (Scheme)

- Assignment
- Sequence of statements
- Iteration