Computing Science (CMPUT) 325

Nonprocedural Programming

Martin Müller

Department of Computing Science
University of Alberta
mmueller@ualberta.ca

Winter 2016

Part I

Functional Programming

Topics for Today - First Lecture

- Main Points
- What is nonprocedural programming?
- Goals of course - What will I learn?
- Basic concepts about functions

Main Points

- Labs start next week with Lisp tutorial
- All course information is on eClass course site, `https://eclass.srv.ualberta.ca/course/view.php?id=27790`
- Now: we quickly review information on eClass - course outline, policies, coursework,...
- Homework this week: Read information on eClass in detail

What is Nonprocedural Programming?

- First: what it is not...
- Most popular languages (Java, C/C++, Python,...) heavily use variables with destructive assignment
  - `int x=3;`
  - `x=5;`
- Imperative programming: statements change **program state**
- Directly corresponds to changing content of registers and memory locations

Problems with Changing Program State Directly

- Parallel programming - what if two threads write to the same variable?
- Side effects, or: Who changed my variable?
- In unstructured programs anyone can write anywhere. Modern languages help by **encapsulating** access
- Example: in object-oriented style, use only get/set methods to change state, use `private` fields in object to restrict access
- Still, in a large program it can be hard to figure out who changed the overall state, and how/when it happened

Functional Programming

- Wouldn't it be nice if variables were "write-once" and never changed?
- **Pure functional programming** uses exactly this coding style
- Well written functional programs can be cleaner, and easier to use in new computing environments, such as multithreaded or distributed
- You can use a functional style even in classical languages such as Java and C/C++
- It is often a good idea!
- Many recent changes to C++ and Java introduced (very old) techniques from functional programming

Functional Programming in C++

`http://bartoszmilewski.com/2014/06/09/the-functional-revolution-in-c/`

> *There was a sea change at this year's C++Now. The cool kids were all talking about functional programming, and the presentation "Functional Data Structures in C++" earned me the most inspiring session award.*

Functional Programming in Java

`https://pragprog.com/book/vsjava8/functional-programming-in-java` (from the book ad):

> *Functional Programming in Java will help you quickly get on top of the new, essential Java 8 language features and the functional style that will change and improve your code.*

`http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html` (about Java in 2015)

> *... won the TIOBE Index programming language award of the year. ... Java has become a language that integrates modern language features such as* ***lambda expressions*** *...*

Procedural vs Declarative Programming vs Constraint Programming

- Procedural: step by step instructions, programmer specifies control flow in detail
- Declarative: specify the logic of your problem, let program figure out the solution steps
- Logic programming is a declarative style. It can lead to very short and elegant programs
- Prolog is by far the most popular logic programming language
- In Constraint Programming you specify constraints on the solution, then let a solver find solutions

Broad Goals of this Course

- Introduce you to major nonprocedural programming paradigms
  - Functional programming with Lisp
  - Logic Programming with Prolog
  - Constraint Programming using an extension of Prolog
- Understand the foundations of functional and logic programming
- Be able to use these ideas, even in other languages, for solving practical problems

Functions

- functions are **mappings** from an input domain (with one or more inputs) to a co-domain
- The result of a mapping is unique, i.e. a function cannot have two different results
- Example: sin: $\mathbb{R} \rightarrow \mathbb{R}$
  - Mapping from one real number to another, e.g. $\sin(\pi/2) = 1$
- Example: $f : \{1, 2, 3, 4\} \rightarrow \{T, F\}$
  - Mapping from set $\{1, 2, 3, 4\}$ to set $\{T, F\}$, e.g. $f(3) = F$
- Example: add: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
  - Mapping from two real numbers to one, e.g. add(1.5, 2.8) = 4.3

Abstract Definition - What is a Function?

- $f : A_1 \times A_2 \times ... \times A_n \rightarrow B$
- The product of A_i's is called the **domain**
- B is called the **co-domain** of the function

Functions in Computing Science

- We can view most programs as computing a mapping from its input to its output. So, a program computes a function
- $f : input \rightarrow output$

Function Definition

- A function can be defined in many different ways.
- A function definition describes what the mapping is.
- Example: $f(x) = x^2$

Function Definition - Finite Domain

- If domain is finite:
- Function can always be defined by enumerating:
  - the mapping between elements of the domain
  - elements of the co-domain.

Finite Domain Example

- Example: $f : \{1, 2, 3, 4\} \rightarrow \{T, F\}$
  - $f(1) \rightarrow T$
  - $f(2) \rightarrow F$
  - $f(3) \rightarrow F$
  - $f(4) \rightarrow T$
- If we view $\{T, F\}$ as representing true and false, then this is a boolean function

Function Defined by Other Function(s)

- Often we define new functions from existing ones
- $g(x) = x \times x$
  - Here, g is defined using “built-in” or “primitive” multiplication function $\times$
- $f(x, y) = g(x) + g(y)$
  - Here, f is defined using function g

Total and Partial Function

- Function defined for every element in a domain:
  - **total function**
  - otherwise, **partial function**
- Example: $\sin(x)$ is total function for domain of real numbers
- Example: $\log(x)$ is partial function for domain of real numbers
  - not defined for $x \leq 0$ (as a function from reals to reals)

Function Composition

- A function can be defined by a composition of other functions
- Example: $f(x) = \log(g(x))$
- With composition, be careful that the domains and co-domains match up.
- Example:
  - $f : \{1, 2, 3, 4\} \to \{T, F\}$
  - $g : \{T, F\} \to \{7, 8\}$
  - $g(f(x))$ is a mapping from $\{1, 2, 3, 4\}$ to $\{7, 8\}$
- Here, the co-domain of f is the same as the domain of g
- In general, the co-domain of the "inner" function must be a subset of the domain of the "outer" function

Function Composition - Abstract Example

- $f : A \to B$ and $g : C \to D$ where $B \subseteq C$.
- Computing $g(f(x))$ is a mapping from A to D
- $x \in A, \quad f(x) \in B \subseteq C, \quad g(f(x)) \in D$
- We can define a new function for this composition:
- $g \circ f : A \to D$
- The function definition of $g \circ f$ is:
- $(g \circ f)(x) = g(f(x))$

Function Composition - Programming Example

- Read data from a file f, process the data, and write the result:
- processFile(f) = writeResult(processData(readFile(f)))

Higher Order Functions

- "Ordinary" functions work with data such as numbers, lists, strings, booleans, structs etc.
- Higher order functions have other functions as input and/or output
- The composition function $\circ$ above is an example
- The mapping here is from a pair of functions to one new function
- $\circ : (A \to B) \times (C \to D) \to (A \to D)$
  - Here, the domain is $(A \to B) \times (C \to D)$
  - The co-domain is $(A \to D)$

Higher Order Functions (continued)

- Note: instead of writing the application of this higher order function as $\circ(f, g)$, it is common to write $g \circ f$
- Higher order functions are very useful to write concise and generic code
- Other example: symbolic differentiation `diff(`f, x`)`:
  - f is a function
  - x is a variable
  - `diff` is a higher-order function which returns a new function:
  - the derivative $\frac{df}{dx}$

Topic Summary - Functions

- Functions are mappings from domain to co-domain
- In functional programming, a program is a function of the input which computes the output
- Functions can be defined by enumerating all values, or by composition from other functions
- It is useful to have a small set of primitive functions to start with
- Higher-order functions have other functions as input and/or as output