CS 314: Principles of Programming Languages

Functional Programming with OCaml

What is a functional language?

A functional language:

- defines computations as **mathematical functions**
- discourages use of mutable **(program) state**

State: the information maintained by a computation

Mutable: can be changed

```
{x = 1}
x = x + 1;
{x = 2}
```

Functional vs. Imperative

Imperative languages:

- *Focuses on how to execute, defines control flow as statements that change a program state.*

Functional languages:

- *Treats programs as evaluating mathematical functions and avoids state and mutable data.*

Imperative Programming

Commands specify **how** to compute, by destructively changing state:

```
x = x+1;
a[i] = 42;
p.next = p.next.next;
```

The fantasy of changing state(mutability):

- It's easy to reason about: the machine does this, then this...
- Machines are good at complicated manipulation of state

Imperative Programming: Reality

Thread 1 on CPU 1

```
x = x+1;
a[i] = 42;
p.next = p.next.next;
```

Thread 2 on CPU 2

```
x = x+1;
a[i] = 42;
p.next = p.next.next;
```

- There is no single state
    - Programs have many threads, spread across many cores, spread across many processors, spread across many computers...
    - each with its own view of memory

Imperative Programming

Functions/methods have **side effects**:

```
int cnt = 0;//global

int f(Node *r) {
     r->data = cnt;
     cnt++;
     return cnt;
}
```

- mutability breaks referential transparency: ability to replace an expression with its value without affecting the result.

```
f(x) + f(x) + f(x) ≠  3 f (x)
```

Functional programming

Expressions specify **what** to compute

- Variables never change value
    - Like mathematical variables
- Functions (almost) never have side effects

The reality of immutability:

- No need to think about state
- Easier (and more powerful) ways to build correct programs and concurrent programs

Functional vs. Imperative

Functional languages:

- *Higher* level of abstraction
- *Easier* to develop robust software
- *Immutable* state: easier to reason about software

Imperative languages:

- *Lower* level of abstraction
- *Harder* to develop robust software
- *Mutable* state: harder to reason about software

Key Features of Functional Programming

- First-class functions
  - Functions can be parameters to other functions ("higher order") and return values, and stored as data
- Favor immutability ("assign once")
- Data types and pattern matching
  - Convenient for certain kinds of data structures
- Type inference
  - No need to write types in the source language
    - But the language is statically typed
  - Supports parametric polymorphism
    - *Generics* in Java, *templates* in C++

- Like Java, ...: exceptions and garbage collection

Why study functional programming?

Functional languages predict the future:

- Garbage collection
  - Java [1995], LISP [1958]
- Parametric polymorphism (generics)
  - Java 5 [2004], ML [1990]
- Higher-order functions
  - C#3.0 [2007], Java 8 [2014], LISP [1958]
- Type inference
  - C++11 [2011], Java 7 [2011] and 8, ML [1990]
- Pattern matching
  - ML [1990], Scala [2002], Java *X* [?]
    - http://cr.openjdk.java.net/~briangoetz/amber/pattern-match.html

Why study functional programming?

Functional languages in the real world

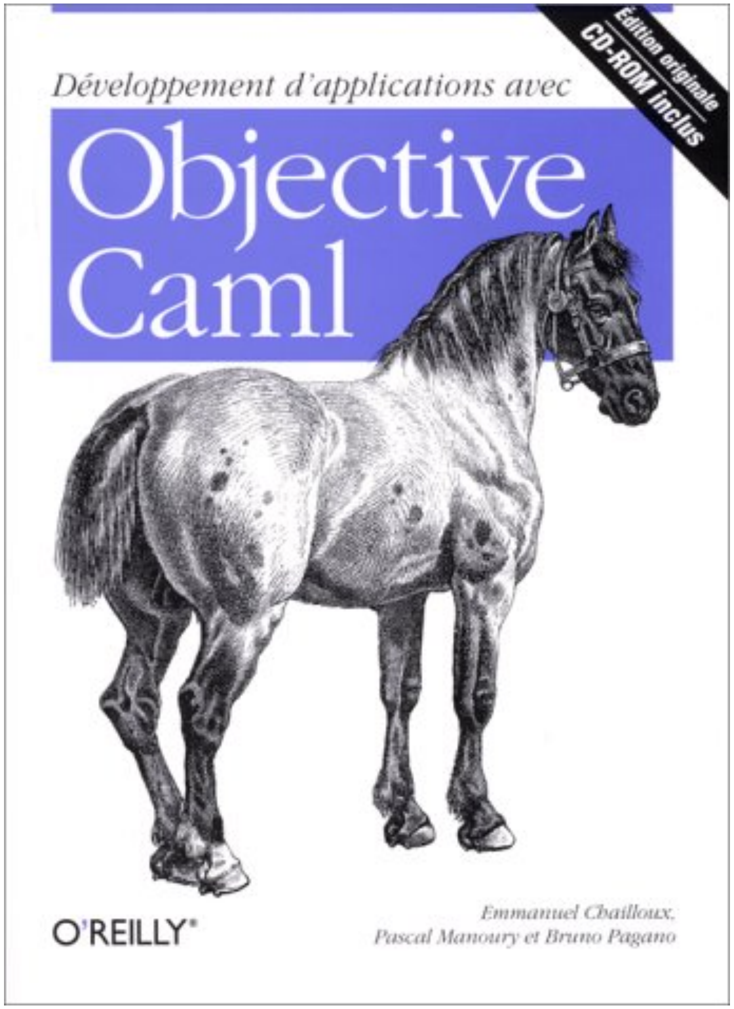
- Java 8 ORACLE
- F#, C# 3.0, LINQ Microsoft
- Scala twitter foursquare Linked in
- Haskell facebook BARCLAYS at&t
- Erlang facebook amazon T · · Mobile ·
- OCaml facebook Bloomberg CiTRIX

  https://ocaml.org/learn/companies.html Jane Street

ML-style (Functional) Languages

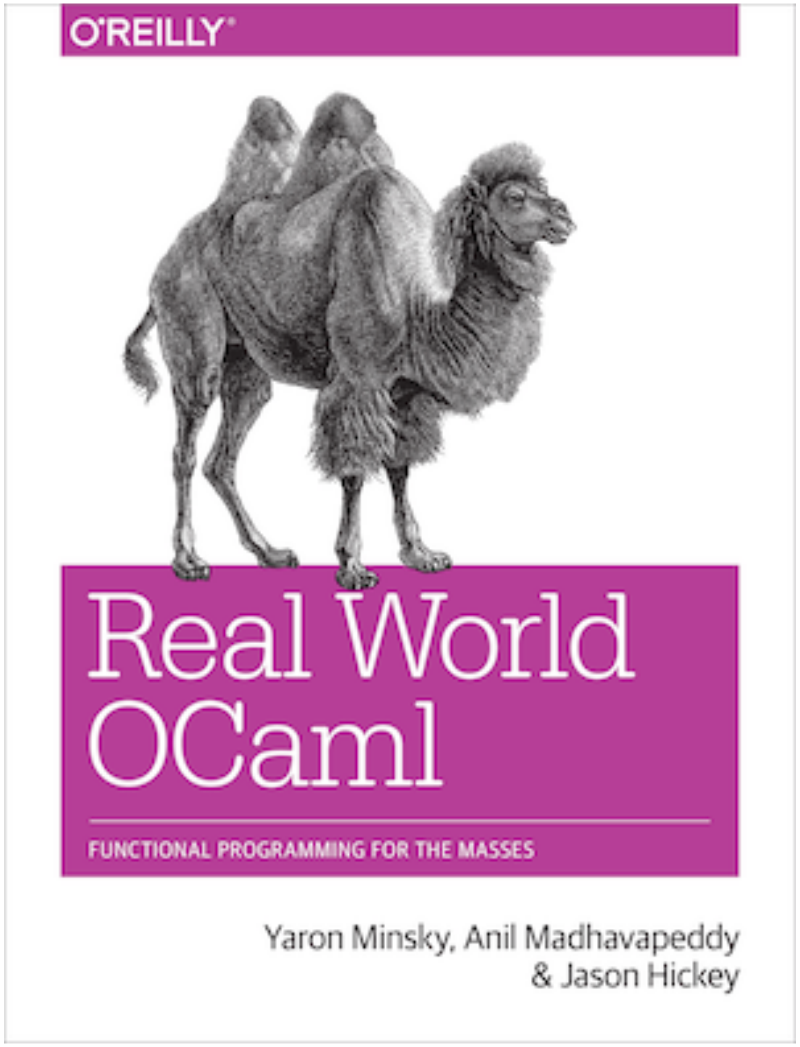
- ML (Meta Language)
  - Univ. of Edinburgh, 1973
  - Part of a theorem proving system LCF
- Standard ML
  - Bell Labs and Princeton, 1990; Yale, AT&T, U. Chicago
- OCaml (Objective CAML)
  - INRIA, 1996
    - French Nat'l Institute for Research in Computer Science
  - O is for "objective", meaning objects (which we'll ignore)
- Haskell (1998): *lazy* functional programming
- Scala (2004): functional and OO programming

Useful Information on OCaml language



- Translation available on the class webpage
  - *Developing Applications with Objective Caml*
- Webpage also has link to another book
  - *Introduction to the Objective Caml Programming Language*

More Information on OCaml


O'REILLY
Real World OCaml
FUNCTIONAL PROGRAMMING FOR THE MASSES
Yaron Minsky, Anil Madhavapeddy & Jason Hickey

- Book designed to introduce and advance understanding of OCaml
  - Authors use OCaml in the real world
  - Introduces new libraries, tools
- Free HTML online
  - realworldocaml.org

Coding Guidelines

- We will not grade on style, but style is important
- Recommended coding guidelines:

- https://ocaml.org/learn/tutorials/guidelines.html