Revenge of the Nerds

Yanan Xiao

Masdar Institute of Science and Technology

Software Engineering Course Presentation

1 Introduction

2 Functional Programming at a Glimpse

3 More Details

4 The Dream Language

"Programming languages have almost caught up with 1958."

Paul Graham

A Bit of History

Question about *computation*.

If we had machines that have **infinite** computational power, what problems would we be able to solve?

Lambda calculus.

- A formal system developed by Alonzo Church.
- Essentially a programming language for one of those imaginary machines.
- Equivalent in power with Turing Machine.

Lisp.

- Invented by John McCarthy as an implementation of Alonzo's lambda calculus, in 1958.
- Lisp machine developed by programmers from MIT AI lab, as a native hardware implementation.

Functional Programming ABC

Alonzo Church

- A practical implementation of Alonzo Church's ideas.
- A set of **ideas**, not a set of strict guidelines.
- **A function is a very basic unit in functional programming.**

Functional or Object-Oriented?

Objects are little capsules, containing ...

- Some internal states.
- A collection of method calls.

Functional programming tries to ...

- Avoid state changes.
- Works with data flowing between **functions**.

In this manner, functional programming can be considered the opposite of object-oriented programming.

Theoretical and Practical Advantages

Functional design may seem like an odd constraint to work under[1]. Why should you avoid objects and side effects? Some sharp benefits are:

- Formal provability.
- Modularity.
- Composability.
- Ease of debugging and testing.

[1] And it is indeed.

- Formal provability. It's easier to construct a **mathematical proof** that a functional program is correct.
- Modularity. It forces you to break apart your problem into **small pieces**.
- Composability. Over time you will form a personal library of utilities. It's because of the modularity benefit.
- Ease of debugging and testing. For debugging: functions are generally small and clearly specified. For testing: each function is a potential subject fir a unit test.

Concurrency

A functional program is ready for concurrency without any further modifications.

Toy Code in Concurrency

```
String s1 = somewhatLongOperation1();
String s2 = somewhatLongOperation2();
String s3 = concatenate(s1, s2);
```

As shown above, even if your application is inherently single threaded, the **compiler**[2] can still optimize functional programs to run on multiple CPUs.

[2] The compiler plays a vital role.

Hot Code Deployment

In a functional program all state is stored on the stack in the arguments passed to functions. All we'd really have to do is run a diff between the code in production and the new version, and deploy the new code.

Higher Order Functions

Functional languages offer a different kind of abstraction tools that make you forget you've ever **liked** modifying variables. One such tool is the capability to work with **higher order functions**.

Add Function

```
class add_function_t{
    int add(int i, int j) {
        return i + j;
    }
}

add_function_t add = new add_function_t();
```

In functions we trust.

Currying

In a functional language one does not need design patterns because the language is likely so high level, you end up programming in concepts that eliminate design patterns all together.

Currying

```
int pow(int i, int j);
int square(int i)
{
    return pow(i, 2);
}
```

Recall: **functions** are passed around as arguments.

In functional programming, it looks like this.

```
square = int pow(int i, 2);
```

This will automatically create a function *square* for us with one argument.

Lazy Evaluation

Operations can only be run when another function depends on them[3]. Some advantages are ...

- Optimization. Code can be reasoned about mathematically[4].
- Abstracting control structures. Seemingly impossible code can be implemented.
- Infinite data structures. With functional programming, we can simply define an infinite list of Fibonacci numbers.

Disadvantage. It's "lazy". No strict evaluation can be guaranteed.

[3] Thank you! Compiler.

[4] Revenge of the nerds!

Continuation

A "continuation" is a parameter we may choose to pass to our function that specifies where the function should return.

Continuation

```
int i = add(5, 10);
int j = square(i);
### A continuation approach ###
int j = add(5, 10, square);
```

At any given point you can ask for a **current continuation**, which is simply the information on the stack.

Real World Applications

The Erlang programming language, developed by Ericsson.

- originally used in fault-tolerant telecommunications systems.
- be popular among companies like T-Mobile, Nortel, Facebook, Electricite de France and WhatsApp.

The Scheme programming language, a dialect of lisp, is taught at a graduate level course[5] in MIT.

[5] `http://mitpress.mit.edu/sicp/`

The dream language is clean and terse. It has an interactive top level that starts up fast. You can write programs to solve common problems with very little code. Nearly all the code in any program you write is code that's specific to your application. Everything else has been done for you.

Excerpt from "Hackers and Painters"