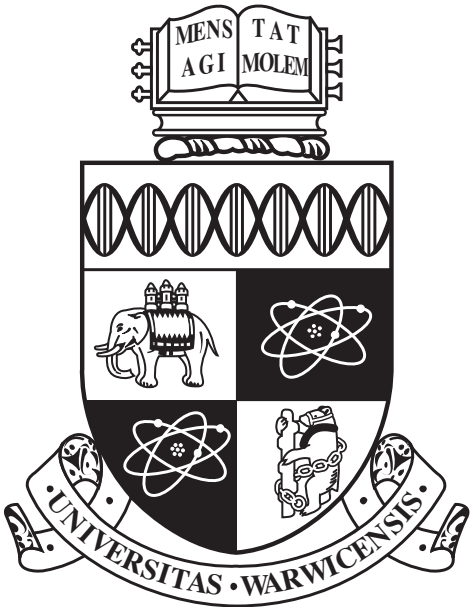
University of Warwick
Department of Computer Science

CS141

Functional Programming



Cem Yilmaz
January 10, 2022

Contents

1 What is Functional Programming

1.1 History

In 1928, David Hilbert posed the question "Given any true mathematical statement, is there an algorithm for verifying that it is true?". This is now known as the "Entscheidungsproblem", that is, the decision problem. E.g.

$$1 = 1$$
$$2 = 2$$
$$2 < 3$$
$$1 < 3$$

In 1931, Kurt Gödel came up with the paradox with the statement "This statement is not provable". If we assume not provable, there is a double negation which implies it is provable. This contradicts the assumption. If we assume true, we have a true statement that is not provable. This is called the incompleteness theorem. This answered the question that mathematics cannot answer every statement.

In 1936, Alonzo Church came up with $\lambda-$calculus as a system for describing algorithms. Kurt Gödel then believed that he can do better. He believed that some algorithms would not be able to be described with $\lambda-$calculus. Kurt then came up with a system of recursive functions. Alonzo Church then claimed that any algorithm that can be described using recursive functions can also be described using $\lambda-$calculus.

Alan Turing then came along and then came up with his own system of describing algorithms utilising Turing machines. However, he also showed that anything described using Turing machine could be described with $\lambda-$calculus. However, notice that despite these being different systems, they were also equivalent in describing algorithms.

1.2 Today

Today, programming as you know it, for example:

$$\prod_{i=1}^{4} = 1 \times 2 \times 3 \times 4$$

If we wanted to turn this to the roughly equivalent program in a language such as Java or C

```
int x = 1;
for (int = 1; i <= 4; i++) {
  x *= i;
}
```

Listing 1: Product in Java

Table 1: Table of results

Variable	Value
x	1
x	2
x	6
x	24

However, in a function language, we can express it as

```
product[1..4]
```

Listing 2: Product in Haskell

That is, product is a function. You can also expand this to be

```
product[1,2,3,4]
1 * product[2,3,4]
1 * 2 * product[3,4]
1 * 2 * 3 * product[4]
1 * 2 * 3 * 4
24
```

Listing 3: Expanded Product

The definition of product function is rather simple:

```
product [n] = n -- Described in terms of 2 equations. First it checks if there a single item
    and returns it
product(n:ns) = n * product ns -- Takes the the first item in the list and keeps the rest
```

Listing 4: Product function

Let us now compare imperative programming with functional

Imperative	Functional
Mutation of state	Reduction of expression
Tell the computer how you want to do something	Tell the computer what you want to compute and let it work out how to do it
Statements executed in order specified	Sub-expressions can often be evaluated in an arbitrary order
Loops	Recursion

1.3 Programming Paradigms

In history, there used to be a clear distinction between programming paradigms. However, today, this has changed. That is,

- Java / C are now multi-paradigm: They're imperative, object-oriented, functional, etc.
- Python, JavaScript, C++ have similarly multi-paradigm.

As such, learning functional programming will be useful as paradigms have blended together.

1.4 What Haskell is good for

1.4.1 Web Services

Furthermore, a particularly nice application to functional programming is that they're good at web services.

- Lots of cool frameworks for developing web applications
- Easy to embed domain-specific languages for routing, templates, etc.
- Servant: describe web service as a type, automatically generate client programs
- One of the coursework assignments uses a web service written in Haskell to provide a browser-based interface

1.4.2 Domain-specific language

Domain-specific languages are also a thing. For example, you can write music in a Haskell library to describe music.

```
import Mezoo

v1 = d qn :|: g qn :|: fs qn :|: g en :|: a en :|: bf qn :|: a qn :|: g hn
v2 = d qn :|: ef qn :|: d qn :|: bf_ en :|: a_ en :|: b_ qn :|: a_ qn :|: g_ hn

main = playLive (v1 :-: v2)
```

Listing 5: Music in Haskell

What's cooler is that if we try to compile, we would get an error that the composition is not harmonic, that is, if it does not sound good, it does not compile. In particular,

- Major sevenths are not permitted in harmony: Bb and B_
- Direction motion in a perfect octave is forbidden: Bb and B_, then A and A_
- Parallel octaves are forbidden: A and A_, then G and G_

1.4.3 Games

For example, the game magic cookies utilises functional reactive programming to describe its game logic. It is the same code across different platforms. It is also good for time-travel debugging.

1.4.4 System Software

- XMonad: Window manager
- OS: Mirage (OCaml), House (Haskell) and more