**Declaration of plagiarism**

**Name(s):** Stephen Mc Donagh

**Programme:** Case3

**Module Code:** CA341

**Assignment Title:** Comparing logic programming and functional programming

**Submission Date:** 11-12-2017

**Module Coordinator:** David Sinclair

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged, and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

**Introduction**

Functional programming is a paradigm where you focus on transforming data using expressions that doesn’t contain any side effects. This means that if you call the same function twice the result will be the same both times. For example if I called add(1,2) it would always compute the same answer. This is in contract to procedures which depends on a local state which may produce different answers at different stages. By eliminating side effects and making your functions ‘pure functions’ you then make it much easier to understand the behaviour of the program. Looping in a functional language is usually done using recursion. Recursive functions call themselves until reaching the base case.

Logic programming is a paradigm based on formal logic in which program statements express facts and rules about problems. The basic program unit in a logic program is a relation and there are no procedures. A normal logic program consists of a vast number of relations and each of which usually have a simple definition. There are no control flow sequencers in Prolog. The only way to repeat an evaluation is by using recursion

I decided to do my functional style in Haskell as we had used it this year in our computability & complexity module and I felt I was comfortable with that. I then decided to do my logical part of this assignment in prolog as I had some knowledge in that from 2nd year.

**Comparison**

**Functions/predicates:**

One of the biggest differences between a functional and logic program paradigm is that functional programming uses functions and that logic programming uses predicates. A predicate is not a function as it does not return anything. Depending on the arguments it will either be true or false. When you write a prolog predicate you are defining a horn clause. An example would be

predicateExample:-

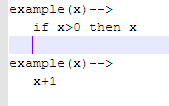
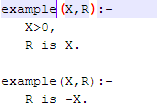
first,second,third.

Would return true if first, second, third are true.

**Backtracking:**

Another difference between the two programming paradigms is that Haskell does not allow you to fail and move on to the next definition. In comparison with prolog if my first definition of a predicate fails, it will move onto the next definition if there is one and try that one. Examples of this are seen below.

Haskell(doesn’t Work) Prolog(works)

Given the argument -3 the prolog version will fail the first predicate and then move onto the next one.

**Execution:**

Adding to my previous point in the functional program the execution of the code will stop if there is an error which is not the same for logic programming which will keep running even if there is an error.

**Usage:**

After doing a little research on ow much both paradigms are used, I have seen that functional programming quite a bit more. From my research I found that functional programming is used in industry, compilers, software development and artificial intelligence. In comparison I noticed that logic programming is not used quite as much and from what I found it is only really used in applications such as database management systems.

**Some similarities:**

We can say that both functional and logic programming are based on mathematical notions. Functional programming has mathematical functions and logical programming had logical declarations.

Both are non-imperative programming languages. This means that both oversee only the result that needs to be computed and not how it is meant to be done.

Both are “error free” and high-level because they are very simple. The focus of both programming paradigms is just on the input and output.

**Advantages/disadvantages:**

**Functional:**

Advantages:

* Functional languages provide protected environment
* Functional languages don’t suffer from side affects
* Functional languages avoid large amounts of code.

Disadvantages:

* Functional languages can be less efficient than others
* Can need a lot of time and memory
* Negative impact on debugging

**Logic:**

Advantages:

* Does not depend on implementation making programs more flexible
* Quick to develop as it relies on true/false statements
* Quite easy to maintain
* Useful for complex ideas

Disadvantages:

* Not used very much
* Predicates not easy to read
* Limited in what type of programs can be solved as only true/false statements are used

**Conclusion:**

From doing this assignment I found the functional part much easier. I find functional languages a lot simpler to understand and to solve problems with. This may be since Haskell is fresh in my mind, but I struggled a lot with my prolog part of this assignment and was not able to figure out a fully working solution.

My solution for prolog only works with 2 strings and anything more it will cause the program to not work. Whereas because I am more confident with Haskell I was able to get a working solution much faster and I believe works well.

From what I could see, both solutions are relatively efficient. But I did some research and found that by using memorization I could make both my programs more efficient. Unfortunately, I was unable to grasp the concept before the due date.