CS421: Programming Languages Final Project

Saad Rasheed - srashee2

Reviewing A History of Haskell

A History of Haskell: Being Lazy with Class

Introduction

A History of Haskell is a research paper that reads like a script. This paper genuinely goes from the beginnings of the creators of haskell, the origins, the use cases, language specifications, environments, tools and applications. Lastly it touches on the impact of Haskell in education, open source and industry. Of course because this project is about reviewing this article most of this paper will be about the most important bits of this summary in this paper.

Origin

Haskell's origin was simple. It's main goal was to bring all the different sorts of functional programming languages in the world into a unified one with agreed upon characteristics. So the self proclaimed FPLang Committee was formed to do exactly that.

The author also explores in depth the different meetings in chronological order to give context to the creation of Haskell. It was quite interesting to see the attitude taken both when creating the langauge. It was quite a humorous attitude in the paper and considering the addition of jokes during Haskell's release I think the creators didn't take themselves too seriously. Ironically, Haskell seems to be just that. We often hear people say "Semantics don't matter", well in this case, they absolutely do.

Language

This section includes the many characteristics of Haskell and the motivations. We begin by exploring the basics of functional programming. In this section the following concepts are discussed: functions, function applications, currying, anonymous functions, prefix operators, infix operators, namespaces, comments and list comprehensions.

Next the contributions for specific features and characteristics of the language were discussed. I have summarized a few of the most interesting contributions.

Implicit parameters were developed by Lewis, Shields, Meijer and Launchbury (Lewis et al.,2000). The idea behind introducing this is to essentially make pretty the function parameters that would be obvious in a function.

Typeclasses and their extensibility is covered in depth in this paper, and that is no shock. Typeclasses are definitely the main distinguishing feature of Haskell. We understand from this class where the problems of such a feature exist, and in this paper ambiguity and type defaulting are covered in depth. The author has quite a lot of contrived examples in the paper for reference.

Implementation and Tools

This section covers compilers, programming environments and tools. The article states the most widely accepted Haskell compiler is GHC or Glasgow Haskell Compiler. This is not

new to us since we've been using GHC and stack for our programs in this course. We visit some other not so known compilers such as Helium, UHC, jhc and the york haskell compiler.

Haskell has not had many tools for the programming environment at the writing of this article. A few like VS code extension and Haskell refactorer were mentioned but at the time of this project things have changed quite a lot. In the next section I will be exploring different Haskell tools and environments built with Haskell and or for Haskell. At the helm of this project, I installed xMonad, a window manager written in Haskell and a Haskell language server for all development purposes.

xMonad

I have also explored Haskell in the open source community by installing xMonad which is a window manager for linux written purely in Haskell. The configuration for xMonad along with any changes to the environment would need Haskell code changes and recompilation of the binary. This allowed me to get more hands on experience using Haskell in the open source environment.

Although I had quite a lot of fun setting up this environment, in practice, managing different versions of stack, cabal and ghc was a nightmare when the window manager, the code you're working on and the system's package manager have different versions of each other's libraries. This might have just been an inherent flaw of what I was trying to do with this project though.

Overall, despite the bumps in the road, the overall experience was amazing. It's a privilege to be able to learn and work with something in the same language. The Haskell

language server was also an extremely cool tool to use. It gave a description of functions just hovering over any Haskell code and since everything I was writing was Haskell, it was always useful.

Conclusion

In conclusion, Haskell has had quite an amazing journey and continues to do so. The language is definitely a unique challenging experience for many people including myself and I hope we see this language continue to produce some interesting things.