

# MyLang and my PhD

Kevin Atkinson

20th November 2004

Designating a new programming language is a huge task, furthermore there might not be any actual research in that area. Therefore designing a new language for a PhD may not be such a good idea. However, this is what I currently want to do for my PhD. Or at very least I want to work on some aspect of it for my PhD. The language in question here is MyLang, a new systems programming language. Details on this new language are presented in a separate paper. This paper will focus on possible PhD work involved creating of MyLang.

One of the key features of MyLang will be the strong support for macros. Macros in MyLang will be very similar to Lisp style macros but even more powerful. In fact MyLang macro will likely be able to directly manipulate the compile time environment, and thus aren't technically macros but rather Compile Time Functions (CTF). In fact CTF will be used to create a good part of the language. The core language will be kept as simple as possible. For my PhD work I would like explore how powerful I can make CTF and how simple I can make the core languages. Seen this way MyLang is more of a general purpose language framework than a particular language. In fact, since CTF will be defining much, if not all, of the syntax of the language, it will be possible to define a completely different language on top of the MyLang core. The development of this framework in the main area I would like to work on for my PhD work.

Another possible, although less likely, PhD topic to explore is the development of the type system for MyLang. The C++ type system is very powerful and flexible, but rather inelegant when compared to the type system of functional programming languages such as Haskell. However, Haskell's type system is not nearly as powerful as C++'s. For example, Haskell lacks any sort of ad-hoc overloading. Many

extensions have been proposed to Haskell's type system, yet none of them, that I have worked with, give me the flexibility that I am use to with C++. One reason for the lack of flexibility of Haskell's type system is that any extension must also be compatible with the type inference system in Haskell. Supporting things like ad-hoc overloading while at the same time supporting complete type inference proves to be a very difficult problem. I, however, hope to be able to support both, by only supporting partial type inference. For example, by only support type inference for local variables within a function.

By now I hope to have shown that developing a new language for a PhD is not such a bad idea. I can either focusing on developing a powerful language framework or possible focus on the type system of MyLang. I have not done an extensive search for previous research in those two areas but I am confidant that there is work to be done in both areas.