CS 225, Professor Darais
Final Project Proposal
Zachary Bechhoefer, Noah Mintz Roberts

For our project, we plan to write a database object interface (DBO: database object) generator in C, for OCaml. The reason simply being that every major platform needs one. As an example, Ruby and PHP, and various JVM languages owe their success in part to their ability to form an MVC stack for industry use. In this sense, it is apparent that having mainstream use cases is a key factor in the success of a programming language. While ML certainly has a niche in academia, the point of functional programming is to ultimately streamline the software development process via better compile-time debugging. If functional programming is to ever advance past its current incarnation as an ivory-tower technology, it will need to be introduced to the larger technological community. Like Ruby, Java (through Java EE), and languages before it, OCaml can break into industry through the web, starting with a model for an MVC stack.

On Indeed.com there are many jobs for Scala and Groovy, but not for functional JVM languages like Clojure or Kotlin. Even though Clojure and Kotlin are held in high regard by the programming world, the gap between functional purity and utilitarian side effects impede adoption of otherwise great functional languages. A lack of industry adoption prevents the growth of a platform, which ultimately results in would-be innovative languages falling by the wayside (as many JVM languages have). Ocsigen has ventured into the fray by giving database access in OCaml, but they have not written an automated generator. Automatic code generation for database access allows the bug-free database-access bootstrapping that web developers (like myself, Zach) crave, and give languages their first ‘in’ with industry.

The process for writing this model appears to be straightforward. The OCaml C library, and the *ocamlc* compiler can be used to write modules with all the utility of C. However, the C module would itself be written in C++, which would directly read a database, and write C to access it. So the process would be: C++ program reading the database, and writing C access code, ocamlc compiling that C into a module, and that module being used by OCaml to operate on the database. This is not far from what existing generators do, except other generators, for example the ADO.NET DBO for the .NET MVC framework, are generated to C#, in C#. Rails on the other hand, is interpreted using a C program, so clearly this is a practical soltuion. Alternatively, we could leverage Ocsigen libraries, and directly generate code from those libraries. Either way, there is either existing code to use, or similar productions in use.

Disclaimer: We are aware that OCaml is intended to compile type-checking languages, rather than being a development platform. C++ was probably meant to be an application language, but look, now it probably sees as much use as a part of operating systems and compilers as it does for applications, so tools are re-purposed.
Other disclaimer: Haskell already has a lot of hardware access implemented, but is still not popular. Let’s attribute this to Cabal being a difficult package manager, and Haskell being a difficult language.
Other other disclaimer: Disregard Elm being a functional web development language. If you look at it closely, it is obviously not suited for any serious development.