Hongbo Zhang

bobzhang1988 at gmail dot com

PDF, HTML

I am a senior software engineer specialized in *functional programming*, *meta-programming* and *transpilers*.

I was a contributor to *OCaml* compiler, the creator of *Fan*, which is a Lispy style meta-programming system for OCaml and creator of OCamlscript, which compiles OCaml to highly readable JavaScript.

Recently I am impressed by the capability of modern Javascript, the whole Fan and OCamlscript compiler is transformed into JS, so that you can play with it in your browser without running a server: *iFan*, *OCamlscript*.

The rise of *ReactJS*, *NodeJS* sparks my interest in front-end programming as well.

SKILLS

Expert: OCaml, compiler front end, compile-time meta-programming
Proficient: Python, Haskell, C/C++, F#, C#, Common Lisp, Clojure
Working knowledge: Coq, X86 Assembly, LLVM IR, Perl, Ruby
Editor: More than 12 years in Emacs, ELisp

EDUCATION

University of Pennsylvania, Master in Computer Sci., *2011 - 2013*

Research Interests: Programming Language Design, Domain Specific Languages and metaprogramming

Tsinghua University, Bachelor degree of Electrical Engineering., *2005-2009*

GPA ranked #1 out of 200+ in the last year

OPEN SOURCE CONTRIBUTION

Creator of OCamlscript

OCamlscript is OCaml's JavaScript backend which aims to generate *readable* and *editable* JavaScript code. It's ultimate goal is to generate code as readable as typescript.

OCamlscript maps one OCaml module into one JavaScript es6 module, which also provides an easy FFI. The OCaml to JavaScript compiler is also compiled into JavaScript which is available.

Creator of *Fan*

Fan is a is a compile-time metaprogramming system for OCaml, which is specialized in helping make writing compilers easier. It is *bootstrapped* by itself. Major features:

- Structural typing system for the representation of abstract syntax.
- Overloaded quasiquotation support.
- First class parser, lexer as Delimited Domain Specific Languages
- Reflective macro system

The Fan compiler itself is also *compiled to JavaScript* which can be running in the brower directly.

OCaml Compiler maintainer *2012 - 2013*.

The compiler-dev team is composed of less than 20 people who are mainly accomplished professors in INRIA, I am honored to help maintain the part of Camlp4 and front end of OCaml.

The administrator of OCaml Beginners mailing list

Invited to the Panel Session of ML workshop *2012*

- Topic: Syntactic Meta Programming for ML.
- Slides: Camlp4 a better macro system

Github Repo

PAST EXPERIENCE

Bloomberg L.P. *2014 - Senior Functional Programmer*

- UI generation for algebraic data type based on in-house OCaml compiler extension. Mapping OCaml's algebraic data type into a specific JSON format which fits into Bloomberg Terminal UI
- POC: OCaml backend ported to AIX mainframe and most OCaml foundational libraries and OCaml to existing C/C++ API bindings
- Develop a YAML based declarative build system to replace the old style Makefiles

Bloomberg L.P. *2013 Financial Software Developer Summer Intern*

Automatic Termsheet Generation

By making use of run-time reflection in *OCaml*, a flexible multiple pass transformation on top of financial algebra was introduced so that it could be exported to various backends in the future, for example, Microsoft Excel, HTML. More than *60,000* lines (written in *C++*, *OCaml*, *Javascript* and an internal domain specific language) were pushed into production.

Microsoft Research Asia *2009-2010 Research Assistant*

Designed and implemented a DSL, Wukong based on FRP, which helps designer express animation in a declarative way.

Hulu Recommendation Algorithm Group *2010 Algorithm Designer, part-time*

Designed and developed the recommendation algorithms based on *collaborative filtering* which recommends appropriate videos based on the user's watch history using *F#*.

University of Pennsylvania *2012 - 2013 Teaching Assistant*

- Compilers *CIS 341 Spring 2013*
  One of the major contributors to the Quaker OAT compiler framework. The compiler compiles the *OAT* programming language to a subset of *LLVM IR*, and also translates such subset into *X86 Assembly* without using LLVM backend.
  OAT is a subset of C++ without multiple inheritance and templates, it supports objects, dynamic dispatching and most features of C programming language.
- Advanced Functional Programming in Haskell *CIS 552 Fall 2012*