Samuel Breese, breess@rpi.edu

Education

Rensselaer Polytechnic Institute:
B.S. (summa cum laude, 3.91 GPA), Computer Science, 6/18
Ph.D., Computer Science, In Progress
Autodidacticism

Knowledge: Programming Languages

Mastery of:

Haskell 2010, C, Forth, R5RS Scheme

Extensive experience with:

Idris, Purescript, Coq, Python, C++, Emacs Lisp, Common Lisp, Clojure, x86 Intel-syntax assembly, POSIX-compatible shell

Significant experience with:

Java, JavaScript (up to ES6), Go, Scala, Erlang, PHP, Standard ML/OCaml, Agda, Perl 5, LLVM IR, Zilog Z80 machine code

Some experience with:

Elm, R, Factor, K, Elixir, Shen, Frege, Julia, Rust, Prolog, Octave, Ruby, Perl 6, non-POSIX shells (e.g. Plan 9's `rc` shell)

Knowledge: Technologies

Very familiar with Unix system administration, from unconventional kernel builds and init systems to window managers and common applications. Particularly familiar with optimizing kernel and system space and time usage on GNU/Linux. General familiarity with the software ecosystem of Linux-based (and to a lesser extent BSD-based and Plan 9-based) systems overall, especially Gentoo and GuixSD.

Working experience with Linux kernel development, particularly within the TCP/IP stack.

Very familiar with common parsing tools, from parser generators like Yacc/Bison to parser combinator systems like Parsec.

Generally very familiar with the library and tooling ecosystem for many programming languages, especially Haskell, Clojure, Common Lisp, and Scheme.

Knowledgeable about low-level graphics APIs (i.e. framebuffer/DRM) on GNU/Linux systems.

Skilled at developing web crawling and REST API interaction tools in any context. Experience with WebSockets and other modern browser APIs.

Versed in a variety of version control systems, such as Git, Mercurial, Subversion, and Bazaar, as well as other common software development tools (e.g. continuous integration, testing frameworks, Agile).

Skilled with LaTeX.

Knowledge: Mathematics

Experienced in and comfortable with modern algebra and category theory, comfortable with algebraic/category-theoretic abstraction as applied to statically-typed functional programming and other domains.

Experienced with the formal specification of programming language semantics.

Knowledgeable of the type-theoretic foundations of statically-typed functional programming, especially in Martin-Löf's intuitionistic type theory.

Experience

From 2015 until 2018 I worked in a research and development context on Submitty, Rensselaer Polytechnic Institute's in-house open source homework submission and automatic grading system. In this position I developed a framework of tools for performing static analysis upon student source code using Haskell, including a sophisticated plagiarism detection tool. I also developed significant development and deployment infrastructure around this tool and others, including a unit testing framework, continuous integration, and automatic deployment. This work was done on a part-time basis during the school year and full-time for two summers.

From Spring 2016 until Spring 2018 I worked as an undergraduate TA for Rensselaer Polytechnic Institute's Data Structures course (CSCI 1200). As an undergraduate TA I worked in a grading role within weekly lab sections in addition to a teaching role during weekly office hours. Similarly, **since Fall 2018** I have worked as a graduate TA for Rensselaer's Programming Languages course (CSCI 4430).

Since Summer 2018 I have worked in a research role within Rensselaer's Worldwide Computing Lab, where I research, develop, implement, and deploy solutions based on formal methods and type theory in safety critical systems, particularly in aviation. This mostly involves developing tools in the Coq interactive theorem prover and other mathematics software, as well as writing and publishing research papers about the same.

Projects

Over the past decade, I have contributed to the open source community through existing projects as well as novel projects. These projects range in scope from the Python and Elixir standard libraries to small collections of utilities for personal use. A subset of more recent contributions can be found on GitHub at `https://github.com/chameco`. Some examples of these projects are given here:

Hitman - in which I developed a Markdown parser for Clojure as an example of best-practice usage of a popular parser-generator library. This project developed considerable community momentum, inspiring several forks for different markup languges in addition to being featured in Read-Eval-Print-λove, a bi-monthly newsletter for the Lisp family of languages.

Solid - in which I wrote and released a bytecode-interpreted scripting language under the MIT license. This language supports modern functionality like lexical closure, higher-order programming, a prototype object system similar to JavaScript, and compilation to a novel bytecode format for a custom virtual machine in order to facilitate some basic optimizations. This project received immensely positive reception, remaining the top trending C language project on GitHub for weeks, and was used by a third party as a scripting language on embedded platforms.

Reliquary - in which I developed (and continue to develop) a dependently-typed concatenative programming language in Haskell. The language compiles to a dependently-typed lambda calculus (similar to the Calculus of Constructions), for which I have also implemented an evaluator and type-checker.

Links

Website: `https://chame.co`
This Document: `https://chame.co/cv.pdf`
GitHub: `https://github.com/chameco`
Lab: `http://wcl.cs.rpi.edu`