JARED ROESCH

roeschinc@gmail.com

Education

M.S. Computer Science (Systems Emphasis) — Spring 2013 - Present
University of California Santa Barbara

B.S. Creative Studies (Computer Science Emphasis, 3rd year) — Fall 2010 - Present
University of California Santa Barbara

Selected Classes

- CS 263: Implementation of Modern Programming Languages (Graduate Level)
- CS 260: Advanced Topics in Program Analysis (Graduate Level)

Experience

Research Assistant, PL Lab UCSB — July 2012 - Present
Working under Professor Ben Hardekopf. I have spent most of my time helping graduate students with existing projects, while gaining the needed background knowledge to lead my own research. The lab's research is largely focused on Program Analysis techniques, and their applications.

Software Engineer, Zentopy Inc. — May 2011 - Present
I've done many things at Zentopy from architecting and writing both the production and prototype API and authentication system, as well as benchmarks, tests, and internal tools. I've worked on the front-end doing everything from a little design and a lot of Javascript, to GUI work on C/C++ client, including porting it from Windows to OS X. One of my largest tasks was building and integrating a filesystem layer between our API and our storage mechanisms. I've stopped working full-time to focus on school, and now act as a part time consultant.

CMPSC 1B/1L, College of Creative Studies — Winter 2013
I guest lectured on both Haskell and Scala in the College of Creative Studies. The class is the second class in the introductory series, and is intended to cover a variety practical topics, that correspond to roughly one year of curriculum in the College of Engineering.

Cloud Computing, College of Creative Studies — Spring 2012
I guest lectured, and acted as an effective Teaching Assistant for this class in Spring 2012, having taken it the previous year. The class covered topics that related to building and maintaing web services. My lectures were focused on introducing students to programming languages such as Ruby, and Scala, as well as the requiste tools such as Ruby on Rails and other web frameworks.

Computer Theorem Proving, College of Creative Studies — Spring 2012
I co-taught a class last spring on Computer Theorem Proving, focused functional programming, type systems, and logic. We use Haskell as our introduction to the concepts behind computer theorem proving, and spent the rest of the quarter in the proof assistant Coq.

Skills and Interests

Topics:

- Compilation, Program Analysis, Programming Language Theory, Type Systems, Lambda Calculus.
- Web services, data storage, AWS, front-end engineering and design, NoSQL.

Frameworks:

- Ruby: Ruby on Rails, Sinatra.
- Javascript: jQuery, Backbone.js, Underscore.js, Node.js.
- Scala: Play!, Unfiltered.

Languages:

- Advanced: Haskell, Scala, Coffeescript, Javascript, Ruby.
- Proficient: Python, C, C++, Java, C#, Clojure, Objective-C.
- Beginner: Rust, Erlang.

Tools:

- Bash, (C — R — M)ake, Leiningen, SBT, MongoDB, PostgreSQL, ZShell, Cabal.

Projects

Javascript Interpreter Implementation — PL Lab 2012-2013

I have spent the last few months working in collaboration with Qualcomm Research on new techniques for engineering JavaScript interpreters. The PL Lab at UCSB is leveraging our experience with analysis, both dynamic and static, to improve the efficiency of Javascript Engines.

Analyze-JS — PL Lab 2012

A framework for doing Abstract Interpretation on Javascript. I have helped run experiments, and write modifications to improve the quality of the framework, so that our analysis is fast and memory efficient.

Small-Step Interpreters — PL Lab 2012

We built a set of interpreters over the summer to compare and contrast the efficiency of implementation techniques. We were examining the differences between interpreters based on big-step operational semantics, and small-step operational semantics, along with bytecode interpreters(like the JVM, the CLR, ect).

Coffeescript Import — 2012

A build tool that extended the Coffeescript language with a preprocessor directive to allow automatic dependency ordering. One of the major headaches of writing Javascript is maintaining dependency ordering. There are a few established solutions but most are either bloated, or dynamic, adding additional headaches to development. The tool was intended to be dumb and easy to use tool, eschewing complexity and configurability for simplicity.