Larry **Diehl**

Formal Methods Researcher & Engineer

144 Hawkeye Ct. Apt. 215 – Iowa City, IA – 52246

407-718-7665 | larrytheliquid@gmail.com | www.larrytheliquid.com | larrytheliquid | @larrytheliquid | larrytheliquid

Education

Portland State University — *Portland, OR*

Ph.D. in Computer Science — *2012 - 2017*

- Advised by Tim Sheard.
- Defended on May 8, 2017.

University of Central Florida — *Orlando, FL*

B.S. in Information Systems Technology — *2005 - 2009*

- Minor in Computer Science.
- University Honors.

Languages

Programming Agda, Coq, Cedille, Haskell, Javascript, Ruby

Spoken English, German

Experience

University of Iowa — *Iowa City, IA*

Postdoctoral Researcher — *Jun 2017 - Current*

- Research on generic programming and zero-cost reuse in Curry-style dependent type theory.
- Contributed to the development of the dependently typed Cedille language.

Portland State University — *Portland, OR*

Graduate Research Assistant — *Aug 2012 - May 2017*

- Generic dependently typed programming over type theoretic models using Agda.
- Formal correctness proofs of programming languages (especially semantic termination) using Agda.
- Implementation of dependently typed languages (Ditto and Spire) using Haskell.
- Co-authored and awarded NSF/CISE/CCF grant #1320934.

Engine Yard — *San Francisco, CA*

Software Engineer — *May 2009 - Aug 2012*

- Worked on a cloud hosting platform on top of Amazon Web Services (AWS).
- Ruby web application and API programming using Ruby on Rails and Sinatra.
- Ruby system automation using Chef.
- Unit testing using RSpec.
- Integration testing using Cucumber and Selenium.

IZEA — *Orlando, FL*

Software Engineer — *Jan 2007 - Aug 2008*

- Worked on a social media advertising platform.
- Ruby web application and API programming using Ruby on Rails.
- Unit testing using RSpec.

Bear Den Designs — *Jacksonville, FL*

Software Engineer — *May 2006 - Jan 2007*

- Worked on medical resident management software.
- Ruby web application programming using Ruby on Rails.
- Unit testing using Test::Unit.

Publications

Generic Zero-Cost Reuse for Dependent Types — *L. Diehl, D. Firsov, & A. Stump*

International Conference on Functional Programming (ICFP) — *2018*

Zero-Cost Coercions for Program and Proof Reuse — *L. Diehl & A. Stump*

arXiv Draft — *2018*

Fully Generic Programming over Closed Universes of Inductive-Recursive Types — *L. Diehl*

Ph.D. Thesis — *2017*

Generic Lookup and Update for Infinitary Inductive-Recursive Types — *L. Diehl & T. Sheard*

Proceedings of the 1st International Workshop on Type-Driven Development — *2016*

Hereditary Substitution by Canonical Evaluation (SbE) — *L. Diehl & T. Sheard*

Technical Report — *2014*

Generic Constructors and Eliminators from Descriptions: Type Theory as a Dependently Typed Internal DSL — *L. Diehl & T. Sheard*

Proceedings of the 10th ACM SIGPLAN workshop on Generic programming — *2014*

Leveling Up Dependent Types: Generic Programming over a Predicative Hierarchy of Universes — *L. Diehl & T. Sheard*

Proceedings of the 2013 ACM SIGPLAN workshop on Dependently-typed programming — *2013*

Verified Stack-Based Genetic Programming via Dependent Types — *L. Diehl*

Proceedings of AAIP 2011 4th International Workshop on Approaches and Applications of Inductive Programming — *2011*

Software

Cedille — *Agda*

Dependently Typed Programming Language — *2018*

- Curry-style type theory.
- Closed universe of types.
- Impredicative quantification, intersection types, and heterogeneous equality.
- Interactive editing and navigation.

Ditto — *Haskell*

Dependently Typed Programming Language — *2015*

- Open universe of types.
- Dependent pattern matching.
- Implicit arguments via dynamic pattern unification and constraint postponement.
- Mutual functions, induction-recursion, and induction-induction.
- Eta-equality for functions.
- Interactive holes and case splitting.
- Novel enhanced form of coverage checking.

Spire — *Haskell*

Dependently Typed Programming Language — *2013*

- Proof of concept.
- Closed universe of types.
- Generic constructors and eliminators.

Lemmachine — *Agda*

Formal Web Framework — *2010*

- Proof of concept.
- Request headers correct w.r.t. previous headers.
- Response headers and code correct w.r.t. previous request and headers.
- Verified HTTP parser.

Dataflow — *Ruby*

Dataflow Concurrency Library — *2009*

- Dataflow concurrency for Ruby inspired by the Oz programming language.