Eliza Brandt

jackbrandt11@gmail.com | 630-998-4473 | StackOverflow: Eliza Brandt | Github: tavrinky | Website

Skills

- **Skills**: Haskell, PostgreSQL, Python, TypeScript, Javascript, Git

Experience

Astro Financial — Remote
Software Engineer (Back-end) — *Jun 2022 - Nov 2022*

- **Feature Development**: Developed Reward Redemption and BaaS Integration
  - Developed the requirements for and built functionality for users to redeem their accumulated points for rewards from a selected creator.
  - Built part of the initial integration for the app's integration with a BaaS provider
  - Managed the migration to a new version of the BaaS provider alone
  - Communicated issues reflecting disagreements between documentation and the BaaS service with the vendor
- **Codebase and Communal Knowledge Improvement**:
  - Caught a major misuse of Beam's parameterization over queries that interfered with mulit-table queries
  - Improved the backend tests so that they could be executed with QuickCheck, making them more than just unit tests, but contracts

SimSpace Corporation — Boston, MA
Software Engineer (Back-end) — *Oct 2019 - Aug 2020*

- **Metrics Scraping**: Designed and developed tool for automated scraping of metrics
  - Interfaces between endpoints, PostgreSQL, and VMware's vCenter
  - Used with Grafana integration for a metrics dashboard
- **Data Generation**: Designed and developed tool for load-testing of various scales
  - Built a configurable tool for testing the speed of generating varying quantities of entities for benchmarking purposes
- **Assisted with hiring**:
  - Interviewed several candidates and gave feedback on performance
  - Reviewed take-home interviews for several candidates

Conduent Inc. — Morrisville, NC
Software Engineer (Blockchain Platform) — *Feb 2019 - Jul 2019*

- **Katip Logging**: Introduced and developed utilities for a logging framework
  - Developed extensions to the Katip library for ease of use among beginner Haskell programmers
  - Developed a tutorial explaining the use and limitations of the logging framework
- **Blockchain abstracted and Petri Nets**: Built blockchain models in Haskell using Petri Nets
  - Used message-passing to model local state transitions on mock chains
  - Used the rich type tools of Haskell to provide compile-time guarantees that messages were not passed to nonexistent members of the chain
- Advised blockchain team on the best methods for implementing errors and exceptions into Haskell
- Developed and wrote a Haskell style guide for internal use

Projects

- Flashcards: A Yesod website for studying languages
  - Quizzed the vocabulary words you were most likely to error on
  - Based off Anki, designed for better ergonomics
  - Main implementation uses Haskell and SQLite
  - Other test implementations use Flask and SQLAlchemy; Node and PostgreSQL
- **T and L**: Add-on to Haskell's type system
  - Used to simplify isomorphisms, e.g. $a \to b =$ Vec (L a) b
  - When implemented, would permit coercibility of nontrivial (i.e. not just newtype wrapped but otherwise equivalent types) types for reduction of boilerplate and increased optimization potential
  - Equivalence between types is decidable, and easy enough for a human to read (due to polynomial representation)
- **G**: Exploration of type equivalencies permitted by T and L, as a generalization of common data structures
  - Generalization developed after noticing similarity between indexed cofree comonads and k-ary trees
  - Cofree comonads are easily used for representing a variety of data structures depending on choice of functor, and are easily adapted to algebraic representation, and so were an excellent motivating example for the use of T and L
- PDominion: API for simulating the Dominion card games, with the ruleset for the base game, all the base cards, and several basic player strategies already developed, written in Python
- **HDiff**: Symbolic differentiator written in Haskell, using an AST over mathematical expressions, looking to generalize over more broadly representative mathematical objects, rather than just Haskell's Num and Floating typeclasses

In progress papers on T and L and G available on request.