Eric Seidel

✉ eric@seidel.io ☎ (225) 276-2830 🔗 eric.seidel.io in ericlseidel gridaphobe

Profile Summary

Software Engineer and Architect with extensive experience in building scalable, efficient systems and designing cutting-edge domain-specific languages. Specializes in functional programming, programming languages, and distributed systems, with a proven ability to translate complex technical requirements into impactful solutions.

Work Experience

Lead Architect – Domain-Specific Languages *Feb 2021 to present*
Bridgewater Associates

- Led the design and development of a Scala-based Domain-Specific Language (DSL) for economic modeling and investment logic.
- Utilized advanced Scala features, including type-level programming, macros, and compiler plugins, to enable expressive and efficient modeling capabilities.
- Collaborated with domain experts to ensure the DSL met rigorous business and analytical requirements.
- Drove architectural decisions to support scalability, maintainability, and robust integrations within the firm's technology stack.

Senior Software Engineer *Aug 2017 to Feb 2021*
Bloomberg

- Member of the Engineering Champs organization, helping to guide the technical direction of the company.
- Designed and implemented low-latency caching service of feature enablement flags with real-time updates over Apache Kafka.
- Extended FORTRAN parser to support non-standard language features, helping to launch a new team doing automated refactoring of legacy codebase.
- Designed and maintained libraries and infrastructure for writing Haskell at Bloomberg.

Education

UC San Diego *2017*
PhD in Computer Science

- Applied machine-learning techniques to improve type-error localization for Hindley-Milner type systems (Seidel, Sibghat, Chaudhuri, Weimer, and Jhala 2017).
- Synthesized counter-examples to type errors using symbolic execution (Seidel, Jhala, and Weimer 2016).
- Translated refinement types into efficient, exhaustive test suites (Seidel, Vazou, and Jhala 2015).
- Verified memory safety and functional correctness of Data.Text library, discovering and fixing a memory safety error in the process (Vazou, Seidel, and Jhala 2014).

The City College of New York *2012*
BS in Computer Science

- Graduated Magna Cum Laude.
- Received Engineering Achievement Medal (top of graduating class).

Open Source Contributions and Service

Member of the GHC Steering Committee (2018 to present).

Served on the Haskell Symposium 2019 Program Committee.

Contributed **HasCallStack**, a lightweight call-stack mechanism, to GHC.

Contributed to macOS support in the Nix package manager and software distribution.

Publications

E. L. Seidel, R. Jhala, and W. Weimer (2018). "Dynamic witnesses for static type errors (or, Ill-Typed Programs Usually Go Wrong)". In: J. Funct. Programming 28

E. L. Seidel (2017). "Data-Driven Techniques for Type Error Diagnosis". PhD thesis. UC San Diego

E. L. Seidel, H. Sibghat, K. Chaudhuri, W. Weimer, and R. Jhala (Oct. 2017). "Learning to Blame: Localizing Novice Type Errors with Data-driven Diagnosis". In: Proc. ACM Program. Lang.

E. L. Seidel, R. Jhala, and W. Weimer (2016). "Dynamic Witnesses for Static Type Errors (or, Ill-Typed Programs Usually Go Wrong)". In: Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming.

T. Elliott, L. Pike, S. Winwood, P. Hickey, J. Bielman, J. Sharp, ***E. L. Seidel***, and J. Launchbury (2015). "Guilt free ivory". In: Proceedings of the 8th ACM SIGPLAN Symposium on Haskell.

E. L. Seidel, N. Vazou, and R. Jhala (2015). "Type Targeted Testing". In: Programming Languages and Systems.

N. Vazou, ***E. L. Seidel***, and R. Jhala (2014). "Liquidhaskell: Experience with refinement types in the real world". In: Proceedings of the 2014 ACM SIGPLAN symposium on Haskell.

N. Vazou, ***E. L. Seidel***, R. Jhala, D. Vytiniotis, and S. Peyton-Jones (2014). "Refinement types for Haskell". In: Proceedings of the 19th ACM SIGPLAN international conference on Functional programming.

Skills

Languages: Haskell, Scala, Rust, Python, C++

Domains: Domain-Specific Languages, Functional Programming, Type Systems, Distributed Systems