Levent Erkök

16581 NW Rossetta St
Portland, OR, 97229
(503) 645-7862

http://leventerkok.github.io/
erkokl@gmail.com

Profile Ph.D. in Computer Science, specializing in programming language semantics, functional programming, and formal reasoning about software and hardware. *Current affiliation:* Formal verification engineer at Apple, Beaverton, OR. *Citizenship:* United States.

Interests Using programming language based methodologies for tackling hard industrial problems. A common theme is to integrate high-level declarative programming techniques with (semi-)automated formal verification tools such as SAT/SMT solvers, model checkers, and theorem provers. The overarching goal is to build frameworks that can be used for modeling, reasoning about, and automatically generating software/hardware systems.

Education

2002 Ph.D. in Computer Science, OGI School of Science and Engineering, OHSU; USA.

1998 M.Sc. in Computer Science, The University of Texas at Austin; USA.

1996 M.Sc. in Computer Engineering, Middle East Technical University, Ankara, Turkey.

1994 B.Sc. in Computer Engineering, Middle East Technical University, Ankara, Turkey.

Ph.D. Dissertation *Title:* "Value Recursion in Monadic Computations." *Thesis advisor:* Prof. John Launchbury. *Synopsis:* We investigate the interaction between two fundamental notions in computing: recursion and effects. Contributions include a mathematical axiomatization of the interaction on the theoretical side, and a programming language binding construct on the practical side. (A copy of the thesis can be seen at http://leventerkok.github.io/papers/erkok-thesis.pdf.)

Software I am a strong believer in functional programming, (semi-)automated theorem proving, and open source software. I've contributed my own share to the Haskell community over the years. I'm the main developer of SBV, a framework that aims to integrate constraint solving based on SMT solvers and programming in Haskell. See http://leventerkok.github.io for a full list of open-source software projects I was involved with. Few representative works are:

– SBV (SMT Based Verification) is an Embedded DSL (Domain Specific Language), allowing symbolic execution, verification, and code-generation from Haskell programs. SBV provides a seamless integration between Haskell and SMT solvers. More info at: http://hackage.haskell.org/package/sbv.

– SBV-Plugin is a GHC (Glasgow Haskell Compiler) plugin for automatic SMT based verification of Haskell programs. http://hackage.haskell.org/package/sbvPlugin.

– HArduino is a Haskell library that lets you program your Arduino board using the Firmata protocol. More info at: http://hackage.haskell.org/package/hArduino.

– CrackNum is a Haskell library and executable that lets you encode and decode IEEE-754 Floating-Point numbers, helping demystify many of the intricacies of the format. https://hackage.haskell.org/package/crackNum.

Experience Post-graduate employment is listed below, in reverse chronological order.

3/2019–Current Formal verification engineer at Apple. Focusing on formal verification of Apple's Hardware designs that go into all classes of devices.

8/2011–3/2019 Formal verification engineer at Intel. Focusing on formally verifying core functionality of Intel's future generation many core microprocessor offerings. Products I worked on span client, server, and HPC domains. Verifying the floating point circuitry, error-correcting capabilities, internal communication protocols, amongst many other Verilog artifacts. Developed and used both in-house tools, and vendor tools such as JasperGold. Being a corporate environment, publications from this era include those in Intel's internal conferences, regarding many practical applications of formal methods based techniques to actual circuit designs. In collaboration with several colleagues, we also published a paper in IEEE Transactions on Computers, regarding how one can design high-radix division/square-root algorithms with correctness proofs.

12/2006–7/2011 Member of Technical Staff, Galois Inc., Portland, OR., USA.
Galois is a privately held company specializing in high assurance methodologies and technology transfer (http://www.galois.com). Responsibilities included research and development of interfaces to modern SAT/SMT solvers from theorem provers, such as between MiniSat/Yices and Isabelle/HOL. Focusing on adding formal verification capabilities to the Cryptol (http://www.cryptol.net) compiler for reasoning about functional correctness.

A large portion of my work at Galois was done in close collaboration with Dr. John Matthews, where we have developed a system for compiling Cryptol programs to higher-order logic (targeting Isabelle/HOL). This work was aimed at tackling difficult verification tasks that can not be handled by automated solvers and push button techniques. The large majority of the projects I work on at Galois involves programming in the functional programming language Haskell, using a variety of tools including SAT/SMT solvers (such as ABC and Yices), theorem provers (such as Isabelle), and compiler generator tools (parsers, lexers etc.), amongst others. We published the results of some of our work in various conferences, including FMCAD'09, PLPV'09, TPHOLs'08, and AFM'08.

4/2006–12/2006 Research Scientist at Intel Corporation, Hillsboro, OR., USA.
Member of "Trusted Platforms Lab" at Intel Corporation, focusing on developing/applying formal methods based techniques to create trusted computing platforms for Intel's future microprocessor offerings.

Most of my work at Intel's trusted platforms lab was done in close collaboration with Dr. Flemming Andersen. Projects included using functional programming techniques to develop a formal model of a hypervisor, and use of light-weight formal methods (mainly model-checking) to formally specify and verify design artifacts. The main languages/tools used were Haskell, and the SMV and SPIN model checkers. In particular, we have used model-checking techniques to formally specify and verify a proposed leader-selection algorithm in a many-core processor environment, leading to a more detailed understanding of the algorithm at its early design stages. Based on this work, we have published several Intel internal research reports, and an external paper in the MIKES'07 workshop.

10/2002–4/2006 Sr. Automation/Software Engineer at Intel Corporation, Hillsboro, OR., USA.
Responsibilities included design, development, and integration of automation systems that drive modern semiconductor processing plants, along with pathfinding work to enhance automation capabilities in semiconductor manufacturing industry.

While most of the programming work was done using C++ on Windows based development environments, I was also able to use model checkers to verify temporal properties of controller scripts that were deployed in Intel's semiconductor manufacturing fabs. In particular, this work led to a publication in Intel's internal DTTC conference in 2004.

Publications In reverse chronological order. Almost all of these papers are electronically available at http://leventerkok.github.io.

1. Warren E. Ferguson, Jesse Bingham, Levent Erkök, John R. Harrison, and Joe Leslie-Hurd. Digit serial methods with applications to division and square root. *IEEE Transactions on Computers*, 67(3):449–456, March 2018.

2. Levent Erkök and Noura Amjadi. Counting ECC failures using Formal Methods to substantiate RAS claims. In *Intel Design and Test Technology Conference, DTTC'17*, 2017.

3. Levent Erkök, Flemming Andersen, and John Matthews. Formal verification of SECDED-ECC RTL using functional programs as golden models. In *Intel Design and Test Technology Conference, DTTC'12*, 2012.

4. Iavor Diatchki, Lee Pike, and Levent Erkök. Practical considerations in control-flow integrity monitoring. In *SECTEST'2011, Proceedings of the The Second International Workshop on Security Testing*. IEEE, March 2011.

5. Levent Erkök, Magnus Carlsson, and Adam Wick. Hardware/software co-verification of cryptographic algorithms using Cryptol. In *Formal Methods in Computer Aided Design, FMCAD'09, Austin, TX, USA*, pages 188–191. IEEE, November 2009.

6. Levent Erkök and John Matthews. High assurance programming in Cryptol. In *CSIIRW '09: Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research*, Oak Ridge, TN, USA, April 2009. ACM.

7. Levent Erkök and John Matthews. Pragmatic equivalence and safety checking in Cryptol. In *Programming Languages meets Program Verification, PLPV'09, Savannah, Georgia, USA*, pages 73–81. ACM Press, January 2009.

8. Lukas Bulwahn, Alexander Krauss, Florian Haftmann, Levent Erkök, and John Matthews. Imperative functional programming with Isabelle/HOL. In *Theorem Proving in Higher Order Logics, 21th International Conference, TPHOLs 2008, Montreal, Canada.*, volume 5170 of *LNCS*, pages 134–149. Springer, August 2008.

9. Levent Erkök and John Matthews. Using Yices as an automated solver in Isabelle/HOL. In *Automated Formal Methods'08, Princeton, New Jersey, USA*, pages 3–13. ACM Press, July 2008.

10. Rebekah Leslie, Levent Erkök, and Flemming Andersen. Formalizing information flow in a Haskell hypervisor. In *Microkernels for Embedded Systems Workshop, MIKES'07, Sydney, Australia*, January 2007.

11. Levent Erkök. Application of formal methods in factory automation to reduce configuration-induced misprocessing risks. *International Sematech APC/AEC Symposium XVI*, Poster Session, September 2004.

12. Levent Erkök. Formal verification of controller transactions using computation tree logic. *Intel Design and Test Technology Conference*, DTTC'04, August 2004.

13. Levent Erkök. *Value recursion in monadic computations*. PhD thesis, OGI School of Science and Engineering, OHSU, Portland, Oregon, October 2002.

14. Levent Erkök and John Launchbury. A recursive do for Haskell. In *Proceedings of the Haskell Workshop'02, Pittsburgh, Pennsylvania, USA*, pages 29–37. ACM Press, October 2002.

15. Levent Erkök, John Launchbury, and Andrew Moran. Semantics of value recursion for monadic input/output. *Journal of Theoretical Informatics and Applications*, 36(2):155–180, 2002.

16. Levent Erkök, John Launchbury, and Andrew Moran. Semantics of *fixIO*. In *Fixed Points in Computer Science Workshop, FICS'01, Florence, Italy*, September 2001.

17. Levent Erkök and John Launchbury. Recursive monadic bindings. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP'00*, pages 174–185. ACM Press, September 2000.

18. Levent Erkök and John Launchbury. A recursive do for Haskell: Design and implementation. Technical Report CSE-00-014, Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology, August 2000.

19. Levent Erkök and John Launchbury. Recursive monadic bindings: Technical development and details. Technical Report CSE-00-011, Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology, June 2000.

20. Levent Erkök. *A partial evaluator and a post-optimizer for a flow chart language*. Master of Science thesis, Middle East Technical University, Ankara, Turkey, July 1997.

21. O. Burçhan Bayazıt, Levent Erkök, Okan Uludağ, and Fatoş Yarman Vural. Gray Level Texture Generation by Binary Markov Random Field Model with Morphological Operations. Technical Report 95-10, Department of Computer Engineering, Middle East Technical University, July 1995.