Alexander G. Bakst

CONTACT INFORMATION

1514 7th Ave., Unit 301
San Diego, CA 92101

(858) 354-4091
alexander.bakst@gmail.com
abakst.github.io

EDUCATION

University of California, San Diego, La Jolla, CA

Ph.D., Computer Science, December, 2017
Dissertation: "Sequentialization and Synchronization for Distributed Programs"
Web: https://escholarship.org/uc/item/0hh6k2zn

Massachusetts Institute of Technology, Cambridge, Massachusetts

S.B., Computer Science, May, 2008
M.Eng., Computer Science, May, 2009
Thesis: "Enabling Diagnostics in User Interfaces for CAD Applications"
Web: http://hdl.handle.net/1721.1/53142

PROFESSIONAL EXPERIENCE

Cetora, Inc *Senior Researcher* **May 2022 - Present**
Research and development of a static analyzer for smart contracts. Owned core static analysis used by EVM analyzer, provided technical leadership and contributed to the development of front-ends and analyses for other target languages.

Galois, Inc *Research engineer* **May 2020 - May 2022**
Software development in the areas of verification, symbolic execution, computer security, cryptography, hardware design, and high-assurance software development.

Qualcomm Technologies *Senior Engineer* **October 2017 - April 2019**
Responsible for maintenance and feature development in support of the shader compiler for Qualcomm GPU chipsets, focusing on support for the production Vulkan shader compiler.

Oracle Corporation *Software Developer* **June 2009 - May 2011**
Developed a cluster filesystem and a dynamic volume manager (Oracle ACFS and Oracle ADVM). I contributed to both products on Linux, IBM AIX, Solaris, and Windows 2003 and 2008.

PUBLICATIONS & REPORTS

HARDENS Final Report
`https://github.com/GaloisInc/HARDENS`
Joseph Kiniry, Alexander Bakst, Simon Hansen, Michal Podhradsky, Andrew Bivin

Weird Machines as Insecure Compilation
arXiv
Jennifer Paykin, Eric Mertens, Mark Tullsen, Luke Maurer, Benoit Razet, Alexander Bakst, Scott Moore

Pretend Synchrony: Synchronous Verification of Asynchronous Distributed Programs
POPL 2019: Principles of Programming Languages
Klaus von Gleissenthall, Rami Gökhan Kıcı, Alexander Bakst, Deian Stefan, Ranjit Jhala

Verifying Distributed Programs via Canonical Sequentialization
OOPSLA 2017: Object-Oriented Programming, Systems, Languages & Applications
Alexander Bakst, Klaus von Gleissenthall, Rami Gökhan Kıcı

Predicate Abstraction for Linked Data Structures
VMCAI 2016: International Conference on Verification, Model Checking, and Abstract Interpretation
Alexander Bakst, and Ranjit Jhala

Bounded Refinement Types
ICFP 2015: ACM SIGPLAN International Conference on Functional Programming
Niki Vazou, Alexander Bakst, and Ranjit Jhala

Deterministic Parallelism via Liquid Effects
PLDI 2012: ACM SIGPLAN Conference on Programming Language Design and Implementation
Ming Kawaguchi, Patrick Rondon, Alexander Bakst, and Ranjit Jhala

CSolve: Verifying C Programs with Liquid Types (tool description)
CAV 2012: Computer Aided Verification
Patrick Rondon, Alexander Bakst, Ming Kawaguchi, and Ranjit Jhala

Academic Experience

University of California, San Diego, La Jolla, CA
Graduate Student **September 2011 - September 2017**
My current research is on developing a new approach to verifying distributed systems, by automatically transforming a system into a simpler, single-threaded program that summarizes the behaviors of the original program.

Microsoft Research, Redmond, WA
Research Intern **June 2012 - September 2012**
I worked with Chris Hawblitzel at Microsoft Research on Verve, a computer-verified memory-safe operating system. We used several language-based techniques in order to specify and verify the memory-safety of Verve on multicore processors.

Massachusetts Institute of Technology, Cambridge, Massachusetts
Graduate Student **September 2008 - June 2009**
Master of Engineering research done as an intern at Autodesk. I augmented geometric solvers in Autodesk Civil 3D in order to enable the development of user interfaces that would be able to guide users through the design process. I developed a method to allow the solvers to explore the solution space of the problem. The user is then presented with various corrections to infeasible designs, or valid ranges for unspecified parameters.

Teaching Experience

University of California, San Diego
CSE 130: Programming Languages, CSE 131: Compilers

Massachusetts Institute of Technology
6.005: Elements of Software Construction
Teaching assistant duties included leading a discussion section, holding office hours, and grading student work.