Fabian Ritter

ritter.x2a@gmail.com

Web: compilers.cs.uni-saarland.de/people/ritter/ GitHub: github.com/fabian-r

LinkedIn: linkedin.com/in/fabian-ritter-x2a/ ORCID: 0000-0001-9227-0910

Education

Saarland University, Saarbrücken, Germany

since 2017: **PhD Candidate** in the Compiler Design Lab, advised by Sebastian Hack.
I work on inferring, improving, and understanding low-level performance models of CPUs using a variety of techniques: from evolutionary algorithms to linear optimization and satisfiability modulo theories to abstract interpretation and differential testing. Before that, I co-developed an LLVM-based framework to efficiently enforce memory safety in C programs.

2016 – 2017: **Doctoral Preparatory Phase** at Saarbrücken Graduate School of Computer Science.

2012 – 2015: **Bachelor of Science** in Computer Science with minor in Mathematics.
Thesis: Compiler Optimization using Symbolic Abstraction

Leibniz Gymnasium, St. Ingbert, Germany

2004 – 2012: **Abitur** with majors in Mathematics, English, German, Physics, and Politics.

Academic Activities

Publications

- AnICA: Analyzing Inconsistencies in Microarchitectural Code Analyzers. F. Ritter and S. Hack. 2022. In: Proceedings of the ACM on Programming Languages (OOPSLA)
- PICO: A Presburger In-bounds Check Optimization for Compiler-based Memory Safety Instrumentations. T. Jung, F. Ritter, and S. Hack. 2021. In: ACM Transactions on Architecture and Code Optimization (TACO)
- PMEvo: Portable Inference of Port Mappings for Out-of-Order Processors by Evolutionary Optimization. F. Ritter and S. Hack. 2020. In: ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)

Scholarships

2017 – 2020: Fellow of the International Max Planck Research School (IMPRS) for Computer Science

2016 – 2017: Scholarship holder of the Saarbrücken Graduate School of Computer Science

2012 – 2015: Member of the Bachelor Honors Program, Saarland University

Teaching

2017 – 2023: Lecturer's assistant at Saarland University for the lecture Compiler Construction.

2014 – 2017: Tutor/Teaching assistant at Saarland University for the lectures:

- Compiler Construction
- Nebenläufige Programmierung (concurrent programming)
- Grundzüge der Theoretischen Informatik (theoretical computer science)
- Systemarchitektur (system architecture)

Skills

Technical

- C/C++, Python: experienced, used in research projects, course work, private projects, especially using the LLVM compiler framework
- Satisfiability Modulo Theories (SMT) solvers, (Integer) Linear Programming (ILP) solvers: used extensively in research projects
- Scala: used in research projects and private projects
- Java, C#, Verilog, Standard ML: used in course work
- LaTeX: used for writing academic documents, designing slides and posters

Languages

- German: native
- English: proficient

Projects

Understanding and Improving Microarchitectural Performance Models (2021 – 2023)

- We investigate how microarchitectural throughput models differ and where their shortcomings are. Based on differential testing and abstract interpretation, we developed a tool to find and fix problems in microarchitectural models: https://github.com/cdl-saarland/AnICA
- Technologies: Python, C++, x86 Assembly, LLVM

Inferring Port Mappings of Out-of-Order Processors (2018 – 2023)

- We infer the instruction-to-execution-port mapping of modern out-of-order processors by Intel, AMD, and ARM from experiments with time measurements. Our mechanisms for experiment design and port mapping inference use formal methods as well as learning-based approaches.
- Find our research artifact here: https://github.com/cdl-saarland/pmevo-artifact
- Technologies: Python, C++, x86 Assembly, Gurobi (ILP solver), SMT solvers

Memory Safety in C (2017 – 2021)

- We explore ways to make C a memory-safe programming language without sacrificing too much performance. Our static program analyses and dynamic program instrumentations are implemented as a general framework for memory safety instrumentations in the LLVM compiler infrastructure: https://github.com/cdl-saarland/MemInstrument
- Technologies: C++, LLVM, C

Supporting Transcendental Functions in Daisy, a Sound Verification Tool for the Precision of Floating-Point Computations (2016)

- I extended Daisy to soundly estimate round-off errors caused by floating-point operations for trigonometric and exponential functions. This required developing algorithms to compute sound rational bounds for real-valued results of transcendental functions.
- Technologies: Scala

Sprattus: A Unified Framework for Rapid Prototyping of Static and Dynamic Program Analyses (2015 – 2016)

- I implemented the analyses required for classical compiler optimizations in an LLVM-based static analysis framework using symbolic abstraction. This allowed me to investigate how combining these analyses influences transformation quality in the Clang compiler.
- Furthermore, I designed new analysis domains concerning accessed memory ranges and allocated memory regions in the symbolic abstraction framework. This involved formalizing a structured memory model for LLVM IR for use in symbolic abstraction. The resulting analysis can validate the absence of memory safety violations in C programs.
- Technologies: LLVM, C++, C, Z3 (SMT solver)