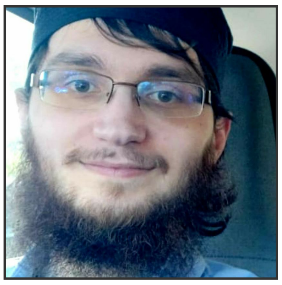
Ömer **Şakar**

PhD Student · ChEOPS project

Enschede, Netherlands

(+31) 53 489 7473 | work: o.f.o.sakar@utwente.nl, personal: o.f.o.sakar@gmail.com | OmerSakar | omersakar

| `omersakar.github.io`



Education

PhD Student at the University of Twente *Enschede, Netherlands*

University of Twente *Jun. 2020 - Present*

- ChEOPS project: Integrating Verification into GPU Program Development
- Formal Methods and Tools group
- Faculty of Electrical Engineering, Mathematics and Computer Science

Master of Science in Technical Computer Science (Technische Informatica) *Enschede, Netherlands*

University of Twente *Nov. 2017 - April 2020*

- Specialization: *Software Technology*
- Title of Thesis: *Extending support for axiomatic data types in VerCors*.
- Grade: 9.0

Bachelor of Science in Technical Computer Science (Technische Informatica) *Enschede, Netherlands*

University of Twente *Sept. 2014 - Nov. 2017*

- Title of Thesis: *Correlating the 2012 Dutch House of Representatives Elections based on Twitter mentions of Parties and their Party Leader*.
- Grade: 8.0

Work Experience

PhD Student at the University of Twente *Enschede, Netherlands*

University of Twente *Jun. 2020 - Present*

- ChEOPS project: Integrating Verification into GPU Program Development
- Formal Methods and Tools group
- Faculty of Electrical Engineering, Mathematics and Computer Science

Teaching Assistant (TA, Student Assistent) *Enschede, Netherlands*

University of Twente *2016 - 2020*

- Teaching Assistant for the Bachelor Technical Computer Science courses.
- TA during the following modules. Modules are comprised of multiple courses following a general topic.
  - Pearls of Computer Science (2016/2017, 2017/2018, 2018/2019, 2019/2020). An introduction to Computer Science.
  - Software Systems (2017/2018, 2018/2019, 2019/2020). Introduction to designing, implementing and testing software.
  - Network Systems (2017/2018, 2018/2019). A module with courses on computer networking.
  - Data & Information (2017/2018, 2018/2019). A module with courses on structured/unstructured data, databases, scripting frameworks and services such as RESTful services.

Developer Quizzard *Enschede, Netherlands*

University of Twente *Feb. 2017 - May 2019*

- Quizzard: A teaching quiz application.
- Mainly back end development.
- Technologies used: Java, Maven, PostgreSQL, Spring Boot, Thymeleaf, Hibernate, React.
- GitLab page: `https://git.tworem.nl/tnb/quizzard`

Back end developer Groufty *Enschede, Netherlands*

University of Twente *Feb. 2016 - June 2016*

- Groufty: group peer review system.
- Technologies used: Java, Maven, PostgreSQL, Spring Boot, Thymeleaf, Hibernate.
- GitHub page: `https://github.com/utwente/Groufty`

Publications

Preserving provability over GPU program optimizations with annotation-aware transformations. -

Formal Methods in System Design *2025*

- Authors: Ömer Şakar, Mohsen Safari, Marieke Huisman, and Anton Wijs.
- Abstract: GPU programs are widely used in industry. To obtain the best performance, a typical development process involves the manual or semi-automatic application of optimizations prior to compiling the code. Such optimizations can introduce errors. To avoid the introduction of errors, we can augment GPU programs with (pre- and postcondition-style) annotations to capture functional properties. However, keeping these annotations correct when optimizing GPU programs is labor-intensive and error-prone. This paper presents an approach to automatically apply optimizations to GPU programs while preserving provability by defining annotation-aware transformations. It applies frequently-used GPU optimizations, but besides transforming code, it also transforms the annotations. The approach has been implemented in the Alpinist tool and we evaluate Alpinist in combination with the VerCors program verifier, to automatically apply optimizations to a collection of verified programs and reverify them.

Deductive Verification of SYCL in VerCors. *Aveiro, Portugal*

International Conference on Software Engineering and Formal Methods 2024 *November 4-8, 2024*

- Authors: Ellen Wittingen, Marieke Huisman, and Ömer Şakar.
- Abstract: SYCL is a C++ programming model for the development of heterogeneous programs. It uses the concept of kernels, where multiple instances of a computation are executed concurrently on a computing unit. This concurrency entails that the set of possible program behaviours can be of considerable size, which makes these programs error-prone. Formal verification could be used to ensure the correctness of all these possible program behaviours. However, there exist no formal verification tools for SYCL. In this paper, SYCL support is added to VerCors, a formal verification tool for concurrent software, by encoding SYCL constructs into VerCors' internal language COL. To the extent of our knowledge, this is the first deductive verification tool for SYCL. We show how SYCL's basic- and ND-range kernels are encoded, along with the encoding and challenges related to scheduling kernels and the execution order of those kernels. In addition, we discuss how SYCL's buffers and data accessors are encoded, focusing on the challenges related to it, in particular enabling memory transfer between host and device. The usability of the added SYCL support and how it was evaluated are discussed as well.
- DOI: `https://doi.org/10.1007/978-3-031-77382-2_11`

The VerCors verifier: a progress report *Montreal, Canada*

International Conference on Computer-Aided Verification 2024 *July 22–27, 2024*

- Authors: Lukas Armborst, Pieter Bos, Lars B van den Haak, Marieke Huisman, Robert Rubbens, Ömer Şakar, and Philip Tasche.
- Abstract: This paper gives an overview of the most recent developments on the VerCors verifier. VerCors is a deductive verifier for concurrent software, written in multiple programming languages, where the specifications are written in terms of pre-/postcondition contracts using permission-based separation logic. In essence, VerCors is a program transformation tool: it translates an annotated program into input for the Viper framework, which is then used as verification back-end. The paper discusses the different programming languages and features for which VerCors provides verification support. It also discusses how the tool internally has been reorganised to become easily extendible, and to improve the connection and interaction with Viper. In addition, we also introduce two tools built on top of VerCors, which support correctness-preserving transformations of verified programs. Finally, we discuss how the VerCors verifier has been used on a range of realistic case studies.
- DOI: `http://dx.doi.org/10.1007/978-3-031-65630-9_1`

First Steps towards Deductive Verification of LLVM IR *Luxembourg City, Luxembourg*

International Conference on Fundamental Approaches to Software Engineering 2024 *April 8–11, 2024*

- Authors: Dré van Oorschot, Marieke Huisman, Ömer Şakar
- Abstract: Over the last years, deductive program verifiers have substantially improved, and their applicability on non-trivial applications has been demonstrated. However, a major bottleneck is that for every new programming language, a new deductive verifier has to be built. This paper describes the first steps in a project that aims to address this problem, by language-agnostic support for deductive verification: Rather than building a deductive program verifier for every programming language, we develop deductive program verification technology for a widely-used intermediate representation language (LLVM IR), such that we eventually get verification support for any language that can be compiled into the LLVM IR format. Concretely, this paper describes the design of VCLLVM, a prototype tool that adds LLVM IR as a supported language to the VerCors verifier. We discuss the challenges that have to be addressed to develop verification support for such a low-level language. Moreover, we also sketch how we envisage to build verification support for any specified source program that can be compiled into LLVM IR on top of VCLLVM.
- DOI: `https://doi.org/10.1007/978-3-031-57259-3_15`10.1007/978-3-031-57259-3_15

Alpinist: an Annotation-Aware GPU Program *Munich, Germany*

International Conference on Tools and Algorithms for the Construction and Analysis of Systems 2022 *April 2–7, 2022*

- Authors: Ömer Şakar, Mohsen Safari, Marieke Huisman, Anton Wijs
- Abstract: Over the last years, deductive program verifiers have substantially improved, and their applicability on non-trivial applications has been demonstrated. However, a major bottleneck is that for every new programming language, a new deductive verifier has to be built. This paper describes the first steps in a project that aims to address this problem, by language-agnostic support for deductive verification: Rather than building a deductive program verifier for every programming language, we develop deductive program verification technology for a widely-used intermediate representation language (LLVM IR), such that we eventually get verification support for any language that can be compiled into the LLVM IR format. Concretely, this paper describes the design of VCLLVM, a prototype tool that adds LLVM IR as a supported language to the VerCors verifier. We discuss the challenges that have to be addressed to develop verification support for such a low-level language. Moreover, we also sketch how we envisage to build verification support for any specified source program that can be compiled into LLVM IR on top of VCLLVM.
- DOI: `https://doi.org/10.1007/978-3-030-99527-0_18`10.1007/978-3-030-99527-0_18