International Journal on Software Tools for Technology Transfer
https://doi.org/10.1007/s10009-026-00867-9

GENERAL

Special Issue SOAP

State of the art in program analysis

Kihong Heo[1] · **Luca Negrini**[2]

Accepted: 25 August 2026

Abstract
Static and dynamic program analysis techniques have been studied for several decades to provide foundations for tools that discover software defects or that verify their adherence to semantic properties. Applying these techniques to real-world programs presents several challenges, such as scalability and precision. This special issue of Software Tools for Technology Transfer comprises novel results and practical applications of these techniques. The papers in this special issue are extended versions of selected workshop papers from the proceedings of the 14th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis (SOAP 2025).

Keywords Static analysis · Dynamic analysis · Formal methods · Programming languages

1 The SOAP workshop

The 14th edition of the ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis (SOAP 2025) [2], colocated with the 46th ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI 2025), featured three keynotes by Xavier Rival, Yulei Sui, and Charles Zhang, and seven regular talks. This special issue of the journal Software Tools for Technology Transfer (STTT) contains revised and extended versions of two papers selected from this program.

As software systems continue to grow in scale, complexity, and societal impact, program analysis plays an increasingly important role in ensuring their reliability, security, and maintainability. Advances in static and dynamic analysis have enabled developers to reason about software behavior, uncover defects, optimize performance, and enforce correctness properties across a wide variety of domains. At the same time, emerging paradigms such as cloud computing, mobile platforms, and AI-driven software development present new challenges that demand innovative analysis techniques and tools. SOAP provides a venue for the program analysis community to present recent advances, share practical experiences, and discuss future research directions. The workshop highlights developments in analysis frameworks, novel methodologies, and applications that bring program analysis closer to the needs of modern software systems.

✉ Kihong Heo
kihong.heo@kaist.ac.kr

Luca Negrini
luca.negrini@unive.it

[1] KAIST, Daejeon, Korea

[2] Ca' Foscari University, Venice, Italy

2 This special issue

All papers and keynotes featured in SOAP 2025 were invited to submit extended versions of their work to this special issue. Each journal submission received at least two reviews in the first round. Two papers were accepted after two review rounds after minor or major revisions. In particular, Muravev and Grigorev [3] define an improved algorithm for CFL-reachability, improving the performance of several types of static analyses that can be instantiated as reachability problems over context-free languages. Instead, Bayarmagnai, Mohammadi, and Prébet [1] present a formal technique for synthesizing loops from arbitrary polynomial invariants.

In the following sections, we provide a summary of each paper featured in this special issue.

2.1 FastMatrixCFPQ: an efficient linear-algebra-based approach to CFL-reachability [3]

Context-free language (CFL) reachability is a fundamental computational framework for formulating key static analyses

Published online: 01 September 2026

(e.g., alias analysis, value-flow analysis, and points-to analysis) as well as some other graph analysis problems. Achieving high performance in universal CFL-reachability solvers remains a significant challenge. Specialized tools such as Pearl and Gigascale are optimized for specific CFLs but lack general applicability, whereas existing universal CFL-reachability solvers often do not scale well in important cases. In particular, prior efforts to leverage high-performance linear algebra operations in universal CFL-reachability solvers produced a matrix-based solver, MatrixCFPQ, that excels at performing common navigational queries on RDF graphs (which are unrelated to program analysis) but is inefficient when it comes to modeling static analyses. The authors introduce FastMatrixCFPQ, a universal matrix-based CFL-reachability solver that overcomes the limitations of MatrixCFPQ by leveraging the properties of the CFL-semiring, common patterns in context-free grammars, and the features of the SuiteSparse: GraphBLAS sparse linear algebra library. The authors also prove that the optimized matrix-based algorithm for CFL-reachability has a worst-case running time of $O(n^3)$ in the number of graph vertices. The experimental results demonstrate that FastMatrixCFPQ outperforms the state-of-the-art universal CFL-reachability solvers across five client analyses — often by orders of magnitude — and, in many cases, even surpasses the speed of specialized solvers designed for specific CFLs.

2.2 From affine to polynomial: synthesizing loops with branches via algebraic geometry [1]

Ensuring software correctness remains a fundamental challenge in formal program verification. One promising approach relies on finding polynomial invariants for loops. Polynomial invariants are properties of a program loop that hold before and after each iteration. Generating such invariants is a crucial task in loop analysis, but it is undecidable in the general case. Recently, an alternative approach to this problem has emerged, focusing on synthesizing loops from invariants. However, existing methods only synthesize affine loops without guard conditions from polynomial invariants. In this paper, we address a more general problem, allowing loops to have polynomial update maps with a given structure, inequations in the guard condition, and polynomial invariants of arbitrary form.

The authors use algebraic geometry tools to design and implement an algorithm that computes a finite set of polynomial equations whose solutions correspond to all nondeterministic branching loops satisfying the given invariants. Furthermore, the paper introduces a new class of invariants for which we present a significantly more efficient algorithm. In other words, the problem of synthesizing loops is reduced to finding solutions of multivariate polynomial systems with rational entries. This final step is handled in an accompanying software using an SMT solver.

Acknowledgements We thank all the authors for their contributions to the workshop and to this special issue. We remark the efforts of the members of SOAP 2025 program committee and referees of this special issue, who took valuable time and effort to make the reviewing process possible: Aditya V. Thakur, Elizabeth Dinella, HeuiChan Lim, Ignacio Tiraboschi, Karoliine Holter, Michael Schwarz, Milla Valnet, Minseok Jeon, Mukund Raghothaman, Sunbeom So, Tian Tan. We also thank the SOAP 2024 chairs, Raphaël Monat and Cindy Rubio-González, for their helpful advice during the whole process.

Funding information Open Access funding enabled and organized by KAIST.

References

1. Bayarmagnai, E., Mohammadi, F., Prébet, R.: From affine to polynomial: Synthesizing loops with branches via algebraic geometry. Int. J. Softw. Tools Technol. Transf. (this issue) (2026)
2. Heo, K., Negrini, L.: Welcome from the chairs. In: SOAP 2025 - Proceedings of the 14th ACM SIGPLAN International Workshop on the State of the Art in Program Analysis, Co-Located with: PLDI 2025 (2025). https://doi.org/10.1145/3735544
3. Muravev, I., Grigorev, S.: Fastmatrixcfpq: an efficient linear-algebra-based approach to cfl-reachability. Int. J. Softw. Tools Technol. Transf. (this issue) (2026)