Modular Collaborative Program Analysis

Dominik Helm, ATHENE, Software Technology Group, TU Darmstadt, Germany

1 Reference

Dominik Helm. *Modular Collaborative Program Analysis*. Dissertation, Technische Universität Darmstadt, 2023, DOI: 10.26083/tuprints-00023220

Abstract

Static analyses are essential to ensure the efficiency and security of software. They face challenges as we use ever more and ever more complex software. We address these challenges by enabling collaborative analyses composed of small, maintainable modules.

In this thesis, we propose the *blackboard analysis architecture* that allows independent modules to collaborate using a central data store. This architecture is framework-independent, applicable to a broad range of static analyses regardless of their implementation paradigm, and allows for modular soundness proofs.

Using four case studies and an extensive evaluation, we show how the blackboard analysis architecture allows improving the soundness, precision, and scalability of static analyses and fosters the exploration of trade-offs between these qualities.

Keywords— Static Analysis, Modularity

2 Introduction

With the world around us increasingly driven by computers, the efficiency and security of software is of paramount importance. Static analysis is an important technique to understand software, improve its efficiency, find bugs, and identify malware. However, it faces challenges in complex programming-language features, such as reflection, in complex analysis problems, and in complex trade-offs between soundness, precision, and scalability. In the past, static analyses were mainly implemented in a monolithic fashion and tailored to a specific task. This approach does not scale to the increasing requirements static analyses face in ever-growing complex software systems.

Complex analysis problems consist of many separate yet interdependent sub-problems (cf. Figure 1 showing sub-problems of a purity analysis). Modular static analyses, thus, decompose analyses into sets of small individual modules. These modules are easier to develop and maintain than monolithic analyses, and modules developed by different domain experts can be composed to face the mentioned challenges.

We thus propose the *blackboard analysis architecture* for modular, collaborative static analyses. In this architecture, modules are fully independent of each other yet collaborate closely to solve complex analysis problems. Compared to prior attempts at modularizing static analyses, our architecture is applicable to a broad range of analyses and does not require adhering to a specific implementation paradigm.

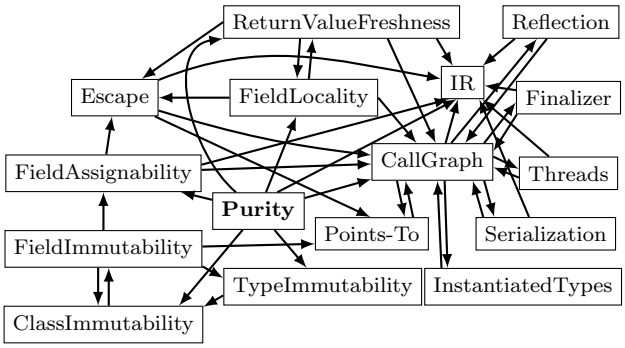

Figure 1: Dependencies Between Sub-problems for a Purity Analysis

In this thesis, we make three central contributions:

1. We propose a novel architecture for modular, collaborative static analyses. We show how this architecture enables modular static analyses in a framework-independent manner and how it enables modular, reusable soundness proofs of composed static analyses.

2. We present four case studies of modular static analyses for intermediate representations, call graphs, immutability analysis, and purity analysis.

3. We perform an extensive evaluation showing that our architecture is applicable to a broad range of static analyses and supports fine-tuning the soundness, precision, and scalability of analyses. We also evaluate how analysis scalability is affected by automatic parallelization, specialized data structures, and different strategies for scheduling analysis tasks, which are all enabled by modularization.

3 Blackboard Analysis Architecture

A framework for modular, collaborative static analyses must fulfill several requirements for the representation of analysis results, composability of analysis, and options to initiate computation. We systematically derive these requirements from case studies from a broad range of dissimilar analyses. Based upon these requirements, we propose a novel architecture for modular static analysis that is reminiscent of the *blackboard architecture* [1]: analyses are split into small, maintainable modules that are independent of each other, yet collaborate closely through a central data store and coordinator, the *blackboard*. This allows modules to be ignorant of each other yet exchange data and work in an interleaved, collaborative manner.

We implement this architecture in the OPAL static analysis framework[1], where analysis modules consist of a declarative specification and an implementation that is often imperative in nature but not restricted by the architecture [4]. In OPAL, the central blackboard provides automated scheduling and parallelization for the analysis modules. In addition, we provide RA2, an alternative implementation of our architecture using reactive programming to achieve semi-implicit parallelization of analysis

[1] https://www.opal-project.de

execution with user-defined scheduling strategies [3].

Furthermore, we formally define the blackboard analysis architecture. Using this formalization, we prove that the architecture allows soundness proofs for analyses to be composed directly from soundness proofs of individual analysis modules [6].

4 Case Studies

We present four case studies, each building upon the previous ones. Each case study advances the state of the art in its respective area, outperforming prior works in terms of soundness, precision, and/or scalability.

TACAI [7] is an intermediate representation based on abstract interpretation. The basic analysis can be extended by optional modules that refine type information for fields or the return values of methods. We show that TACAI is configurable, computed efficiently, and provides refined information that improves the precision of downstream analyses such as call-graph analyses.

Unimocg [5] is a novel architecture for call-graph analyses. By decoupling the computation of type information for local variables from the actual resolution of call targets, Unimocg enables the reuse of call-resolution modules across dissimilar families of call-graph analyses. This enables consistently high sound support for complex programming-language features. As an additional benefit, further analyses, such as immutability analyses, can benefit from the computed type information.

CiFi [8] is a system of four analysis modules for reasoning about the assignability of fields and the immutability of fields, classes, and types. CiFi can infer assignability and immutability information for common programming patterns such as lazy initialization and classes with generic type parameters. This sets it apart from the previous state of the art that could not reason about such patterns and/or only check manually annotated immutability information instead of automatically inferring them.

OPIUM [2] is a family of three analyses that reason about method purity, i.e., whether methods behave in a deterministic manner and free of side effects. They are efficient and precise and provide more fine-grained purity information than the previous state of the art. OPIUM's analyses have different precision/scalability trade-offs and can be combined with various independent analyses to fine-tune the analysis to users' specific needs.

5 Evaluation

We thoroughly evaluated our architecture and case studies, making three core observations:

1. Our modular architecture supports the implementation of a broad range of static analysis kinds, as evidenced by our case studies. The modularity of these analyses benefits both end users and analysis developers by facilitating experimentation with different trade-offs between soundness, precision, and scalability.

2. Modular analyses implemented in OPAL improve upon the respective state of the art w.r.t. precision and soundness. We attribute this to the support for modularity, fostering the implementation of simple modules that improve precision by providing information that would otherwise have to be over-approximated or improve soundness by adding support for complex language features.

3. Our case-study analyses are on par with or outperform the respective state of the art, particularly due to parallelization and specialized data structures. Scheduling strategies can significantly impact execution time and, thus, scalability.

6 Conclusion and Outlook

With this thesis, we laid the foundations for composing arbitrary static analyses. This improves soundness, precision, and scalability and enables the exploration of trade-offs between these qualities. Based on these foundations, future collaborative analyses may incorporate different traditional and novel analysis techniques, combining static analyses with dynamic analyses or machine-learning-based approaches. Modular analyses may also be able to better analyze software implemented using multiple programming languages by composing language-specific analysis modules. Further research is also needed on how to optimally compose and configure static analyses. Our architecture supports this by making analysis modules easy to compose, exchange, and study.

References

[1] D. D. Corkill. "Blackboard Systems". *AI expert* 6.9 (1991), pp. 40–47.

[2] D. Helm, F. Kübler, M. Eichberg, et al. "A Unified Lattice Model and Framework for Purity Analyses". *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ASE'18. Montpellier, France: ACM, 2018, pp. 340–350.

[3] D. Helm, F. Kübler, J. T. Kölzer, et al. "A Programming Model for Semi-implicit Parallelization of Static Analyses". *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ISSTA'20. Virtual Event, USA: ACM, 2020, pp. 428–439.

[4] D. Helm, F. Kübler, M. Reif, et al. "Modular Collaborative Program Analysis in OPAL". *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE'20. Virtual Event, USA: ACM, 2020, pp. 184–196.

[5] D. Helm, T. Roth, S. Keidel, et al. "Unimocg: Modular Call-Graph Algorithms for Consistent Handling of Language Features". *33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. ISSTA'24. Vienna, Austria: ACM, 2024.

[6] S. Keidel, D. Helm, T. Roth, and M. Mezini. "A Modular Soundness Theory for the Blackboard Analysis Architecture". *33rd European Symposium on Programming*. ESOP'24. Luxembourg City, Luxembourg: Springer, 2024, pp. 361–390.

[7] M. Reif, F. Kübler, D. Helm, et al. "TACAI: An Intermediate Representation Based on Abstract Interpretation". *Proceedings of the 9th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis*. SOAP'20. London, UK: ACM, 2020, pp. 2–7.

[8] T. Roth, D. Helm, M. Reif, and M. Mezini. "CiFi: Versatile Analysis of Class and Field Immutability". *2021 36th IEEE/ACM International Conference on Automated Software Engineering*. ASE'21. Virtual Event, Australia: IEEE, 2021, pp. 979–990.