15-745 Project Proposal: Shape Analysis

Kunming Jiang (kunmingj), Andrew Haberlandt (ahaberla)

Mar 19, 2023

Project URL

The project can be accessed at `https://ndrewh.github.io/15-745-project/`.

Project Overview

Pointer analysis is crucial for optimizing languages like C and C++, and is a foundational analysis in the compilers for these langauges. A well-established pointer analysis is foundational for reasoning on dependencies within memory, identifying indirect control-flow targets, and bug detection, as well as enabling a variety of optimizations [1]. Despite its numerous benefits, applying pointer analysis is also resource and time intensive, and often difficult to achieve precision [1]. Conventional pointer analysis involves constructions of either an inclusion-based relationship [2], or a point-to graph to demonstrate the relationship between pointers within a program [3]. These methods are costly and often ignore higher-level semantics of these pointers, i.e. the underlying data structure these pointers represent. A direct acyclic graph (DAG), for instance, can rule out the possibility of a cyclic point-to relationship, and disinclude edges in the relation graph that are otherwise difficult to eliminate.

Shape Analysis [4] proposes a framework to infer potential data structures allocated in the heap. The basic construct involves a dataflow analysis that generates a direction and inference matrix between all pointers, and separates the shape of the data structure each pointer points to into three categories: tree, DAG, or cycle. Using these information, the compiler can efficiently deduce the updates to the point-to information of each pointer operation.

While the original frame work for shape analysis was proposed in 1996, the idea is seldomly implemented in modern compiler machines. In this project, we plan to implement in LLVM15 and compare its improvement, if any, to newer pointer analysis methods. Our 75% goal is to finish a functioning shape analysis pass in LLVM, as well as framework to support inference of additional shapes. Our 100% goal involves a detailed report comparing shape analysis with other forms of pointer analysis, identifying potential advantages and drawbacks of applying shape analysis. Finally, the 125% goal aims to investigate adding additional types of data structures to the shape analysis.

Logistics

Plan of Attack and Schedule

- **Week 1 (March 23rd):** Review foundational shape analysis papers, identify C-dependent components of the paper's technique + check if/how they can be applied to LLIR (i.e. decide if we have to modify the frontend).

- **Week 2 (March 30th):** Begin implementing prototype, ensuring we become comfortable with LLVM's points-to analysis, which is critical for implementing the shape analysis.
- **Week 3 (April 6th):** Continue implementing prototype; the gen and kill sets for all of the common instructions should be properly implemented, and some cases of shape identification (i.e. trees) should work.
- **Week 4 (April 13th):** Wrap up the implementation. The prototype should now successfully identify trees, DAGs, and cycles.
- **Week 5 (April 20th):** Design and evaluate matrices to determine the effectiveness of the implemented shape analysis.
- **Week 6 (April 26th):** Propose additional improvements to shape analysis. Analyze and estimate the benefits of these proposals.

Milestone

By Thursday, April 13th, we plan to finish the majority of the coding section, as per the goal of week 4 in plan of attack and schedule.

Resources Needed

We are planning to implement the project in LLVM. There are no other tools required for this project.

Literature Search

Our project mainly focuses on the work of Ghiya et al. [4], which will be our primary point of reference. We have also gathered other helpful literatures in the reference section below.

References

[1] V. Kanvar and U. P. Khedker, "Heap abstractions for static analysis," *ACM Comput. Surv.*, vol. 49, jun 2016.

[2] P. Liu, Y. Li, B. Swain, and J. Huang, "Pus: A fast and highly efficient solver for inclusion-based pointer analysis," in *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, (New York, NY, USA), p. 1781–1792, Association for Computing Machinery, 2022.

[3] P. M. Gharat, U. P. Khedker, and A. Mycroft, "Generalized points-to graphs: A precise and scalable abstraction for points-to analysis," *ACM Trans. Program. Lang. Syst.*, vol. 42, may 2020.

[4] R. Ghiya and L. J. Hendren, "Is it a tree, a dag, or a cyclic graph? a shape analysis for heap-directed pointers in c," in *Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '96, (New York, NY, USA), p. 1–15, Association for Computing Machinery, 1996.