Measuring Shape Analysis Precision

Viktor Pavlu and Markus Schordan (Faculty Mentor)
Institut für Computersprachen
TU Wien
Vienna, Austria
Email: {vpavlu,markus}@complang.tuwien.ac.at

Abstract — *Shape analysis is a static program analysis technique for discovering properties of heap-allocated data structures. It is crucial to finding software bugs or to verify high-level correctness properties. Various analyses have been introduced but their relation in terms of precision often remains unclear as different analyses use different abstractions of the heap.*
The aim of our work is to compare the precision of shape analyses. We propose a novel algorithm based on three-valued logic that extracts alias sets from shape graphs. Smaller sets are more precise and indicate a more precise underlying shape analysis. Using this metric, we experimentally compare – for the first time – the relative quality of the state-of-the-art graph-based shape analyses and make recommendations concerning the combination of analysis parameters.

I. Introduction

Shape analysis is a static program analysis technique that approximates the structure of the runtime heap. It is closely related to pointer alias analysis but has its focus on the heap with dynamically allocated, recursive data structures. Using shape analysis results, many questions about the runtime heap can be answered at compile time. Such information is crucial to finding software bugs or to verify high-level correctness properties.

Various analyses have been introduced but their relative quality in terms of precision often remains unclear as analyses cannot be compared directly due to different representations of analysis results.

The aim of our work is to compare parametrized versions of shape analyses by the size of may-alias sets extracted from shape analysis results. Smaller sets of possible aliases are more precise and indicate a more precise underlying shape analysis. We are thus able to experimentally evaluate precision and analysis runtime tradeoffs for parametrized versions of shape analysis algorithms that use different heap representations.

Parametrized versions of the state-of-the-art graph-based shape analyses, SRW [1] and NNH [2], were implemented for C++ using the SATIrE program analysis framework [3]. Measurements using the aforementioned metric show the relative quality of analyses and the impact analysis parameters have on precision and runtime. We are the first to show that the NNH shape analysis is strictly more precise than SRW. Experimental results allow us to make recommendations concerning the combination of analysis parameters.

II. Contributions

We implemented multiple instances of state-of-the-art graph-based shape analyses for a subset of the C++ programming language and developed an algorithm based on three-valued logic that extracts alias information from shape graphs.

Using this algorithm as a metric indicating the precision of the underlying shape analysis we are able to judge the effects of individual analysis parameters on runtime and precision of the analyses.

In detail, the contributions address theory, practice, and assessment of shape analyses:

A. Theory

We extended both the SRW [1] and the NNH [2] shape analysis algorithms to model information flows from function call sites to matching function bodies (interprocedural). In their original formulation the analyses were intra-procedural.

We adapted the graph-based shape analyses from theoretical programming languages to a subset of C++.

We developed an algorithm for alias extraction from shape graphs based on the technique described in Reps et. al. [4]. Our algorithm improves precision in all cases where objects in the heap are linked with two or more indirections, e. g., linked lists with two or more elements.

B. Practice

We implemented the shape analyses and integrated them into the SATIrE program analysis framework where subsequent program analyses can now use them as powerful alias analyses.

We added automatic annotation of shape- and alias analysis results to SATIrE. The results obtained by the analyses are available as annotations to the SATIrE abstract syntax tree, as annotations to a program's source code, or as graphical visualization of a program's control flow graph.

C. Assessment

We evaluated shape analysis abstractions and discovered that SRW shape analyses cannot contain must-alias information. We found that the SRW algorithm does not always perform a strong update. Both findings were not previously published.

III. FINDINGS

We systematically introduced five parameters to the shape-based alias analyses to thoroughly evaluate their effects on precision and runtime: (1) shape analysis algorithm, (2) inter-procedural context information, (3) retaining temporary variables, (4) "common tails" extension, and (5) lazy/extensive alias testing. This results in 32 variations of shape-based alias analyses for our comparison.

Interpreting the observed results we were able to make recommendations concerning the combination of analysis parameters:

- Ignoring context information produced the worst results *and* took the most time – always use context information for shape analyses.
- We showed that the SRW shape analysis with an extensive alias test cannot be more precise than the lazy test on SRW graphs – always perform the lazy test for the SRW analysis results to save time.
- NNH shape analysis and the lazy alias test produce a poor match: analysis runtimes are always longer than with SRW but only in one case is the result (minimally) more precise; the higher precision gained from NNH (and paid for by the longer analysis runtime) is lost in the lazy alias test as it merges separate execution paths before the test which were deliberately kept apart during analysis.
- Without temporary variables or the "common tail" extension, SRW and NNH have comparable precision but SRW is roughly 5 times faster on our test cases.
- Retaining temporary variables increased precision for both SRW and NNH (23% and 21% smaller alias sets, respectively) but also increased graph sizes and therefore analysis runtime (3 times) – here, the precision is expensive.
- The "common tails" alias test increased precision only in combination with NNH, but then at no measurable increased cost – a cheap precision increase for an already precise analysis.

The number of may-aliases computed from the SRW- or NNH shape analysis results is shown in Figure 1. A more detailed discussion can be found in the first author's thesis [5].

In summary, we can say:

- NNH is strictly more precise than SRW in all observed test cases
- The fastest variant is: SRW, with context information, with lazy alias test, without common tails, without temporary variables
- If more precision is required: NNH, with context information, with extensive alias test, with common tails. Using our proposed metric we observed a 28% gain in precision (reduced alias set sizes) at 36 times increased runtime for our test cases.

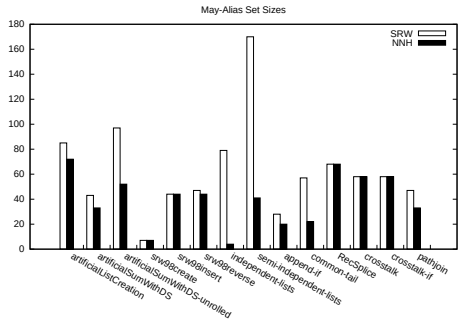

Figure 1: Number of May-Aliases computed from SRW- and NNH shape analysis results (fewer may-aliases are more precise).

IV. CONCLUSIONS

Our work was the first to implement the NNH shape analysis algorithm. We extended state-of-the-art graph-based shape analyses to be used inter-procedurally on a subset of C++ and improved the precision of the established algorithm to extract alias information from shape graphs.

Our work features the first comparison of the relative merits of the two state-of-the-art graph-based shape analyses termed SRW and NNH. We were thus able to answer a theoretically and practically important question, as to which is the more precise shape analysis, which was open since 1999.

REFERENCES

[1] Mooly Sagiv, Thomas W. Reps, and Reinhard Wilhelm. Solving shape-analysis problems in languages with destructive updating. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 20(1):1–50, January 1998.

[2] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*, chapter Shape Analysis, pages 102–129. Springer, 1999.

[3] Markus Schordan. Combining tools and languages for static analysis and optimization of high-level abstractions. In *24. Workshop der GI-Fachgruppe "Programmiersprachen und Rechenkonzepte"*, pages 72–81. Department of Computer Science, Christian-Albrechts-Universitt zu Kiel, 2007.

[4] Thomas W. Reps, Mooly Sagiv, and Reinhard Wilhelm. Shape analysis and applications. In *The Compiler Design Handbook: Optimizations and Machine Code Generation*, pages 175–218. CRC Press, 2002.

[5] Viktor Pavlu. Shape-based alias analysis for object-oriented languages. Master's thesis, TU Wien, Department of Computer Science, Vienna, Austria, 2009.