Bachelor's Thesis - Proposal
Implementation and testing of Monniaux's map elimination algorithm

Luca Bruder
University of Freiburg

January 22, 2019

Abstract

Ultimate [2] is a program analysis framework, which among other things transforms programs to prove satisfiabilty.To this end many over- and sometimes under-aproximations can be made one of which being the elimination of maps. In this thesis we aim to implement an alternate way of eliminating such maps according to a paper by David Monniaux [1] and test it against the existing one.

1 Introduction

Ultimate provides usefull tools to verify and test programs. These tools often include the transformation of said programs. And while there already is a transformation in Ultimate, which eliminates maps it has certain limitations and a different approach might be more efficient. To test this a new algorithm will be implemented and tested over the course of this Bachelor's Thesis.

1.1 Elimination of maps

The elimination of arrays and maps in general in a program enables us to be a lot more efficient. Imagine a simple program (Algorithm 1). If we want to verify if the assertion holds we will have to check every cell of the array. Even worse if we want to run multiple test the array has to be generated and tested every single time.
It is easy to see that this becomes a problem on a bigger scale with more and maybe bigger arrays. But if we eliminate those arrays the tests become a lot more efficient.

1.2 Over-Approximation

The elimination of maps is an over-approximation. Thus the resulting program and every statement we can make about it have certain limitations. The transformation should still be satisfiable in case the original program was satisfiable, but if the original program was unsatisfiable the transformed one might still be satisfiable. This holds true for individual variable assignments as well.

Algorithm 1 Simple loop

1: $i \leftarrow 0$
2: **while** $i \leq 5000$ **do**
3: $\quad x[i] \leftarrow i$
4: $\quad i \leftarrow i + 1$
5: *Assume* $0 \leq k \leq i$
6: *Assert* $x[k] = k$

1.3 iCFG's

Since we are working fromt the viewpoint of the Ultimate toll we will represents programs in a graph-like data structure called interprocedural control flow graphs (iCFG's). These iCFG's are represented with transformulas. We use these transformulas as data structures to represent the schemantics of statements of a programming language.
A transformula is a tuple T = (In, Out, Aux, Φ) with

In A mapping between program variables and term variables (i.e., variables occurring in Φ)

Out A mapping between program variables and term variables

Aux A list of auxiliary variables occurring in Φ

Φ A formula in SMT

1.4 Relevance

The old algorithm introduces two new variables for each statement which writes on the array in question. One for the value and one for the index. So if we have a look at Algorithm 1 again the currently implemented map elimination would create two variables and transform the program to something like Algorithm 2.

Algorithm 2 Transformed loop

1: $i \leftarrow 0$
2: **while** $i \leq 5000$ **do**
3: $\quad x_val \leftarrow i$
4: $\quad x_idx \leftarrow i$
5: $\quad i \leftarrow i + 1$
6: *Assume* $0 \leq k \leq i$
7: *Assert* $x[k] = k$

This would make it very easy to prove that the program is satisfiable *should* k be the same as i. In any other case however we can no longer prove satisfiability. The new algorithm would hopefully solve this problem and be more adaptable. It achieves this by locking a certain preset number of cells and their respective indices.

2 Goals

The goal of this Bachelor's Thesis is to implement and test the algorithm that was adapted from David Monniaux's paper [1] by Lisa Kleinlein and Luca Bruder as seen in 1.4. Furthermore, constraints for the variables which replace the eliminated maps should be added. Lastly the new algorithm will be compared to the one that is already implemented in Ultimate.

3 Approach

Task	Description	Man-weeks
Pre-Analysis	The environment will be analyzed. Questions like "which interfaces will be used", "what are the preliminaries" and "how does the new algorithm fit into Ultimate's toolchain" need to be answered.	1
Implementation of the new algorithm	In this step the written algorithm will be implemented in Ultimate in it's first form	4 - 5
Index constraints support	Since the baseline algorithm does not support the use of constraints on the indices this need to be implemented. As a consequence a method of optaining these constraints will have to be found	2
Testing of the new algorithm	The now implemented algorithm will be tested for problems, exceptions and lastly efficiency. To test the efficiency a fitting test suite must be found.	2
Comparison	The same test suite that has been agreed upon earlier will be used to test the old and already implemented algorithm.	1
Conclusion	In this step the results obtained earlier will be compared against each other. This should yield information about efficiency, precision and potential drawbacks of each algorithm.	1
Written thesis	The written thesis will be worked on continuously over the course of the Bachelor's thesis	12

4 Schedule

The working period for this thesis consists of 12 weeks. The order of the tasks with some leeway for the implementation and testing of the algorithm can be seen in the timetable below (Figure 1).

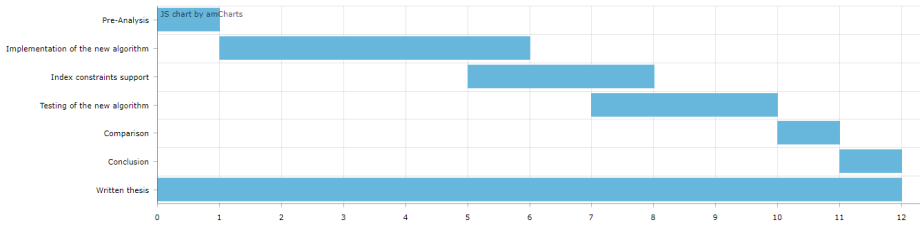

Figure 1: Estimate of the timetable for the thesis.

References

[1] David Monniaux and Francesco Alberti. "A Simple Abstraction of Arrays and Maps by Program Translation". In: *SAS*. Vol. 9291. Lecture Notes in Computer Science. Springer, 2015, pp. 217–234.

[2] *Ultimate*. `https://monteverdi.informatik.uni-freiburg.de/tomcat/Website/`.