Adding Magic Wand Support to Carbon

Problem Description

Gaurav Parthasarathy

gauravp@student.ethz.ch

March 2nd, 2015

1 Background

Separation logic is a permission-based verification logic that makes the verification of certain program properties that, for example, involve shared mutable data structures easier. One of the connectives in separation logic is the magic wand connective which can be very useful to specify partial data structures or invariants of loops that traverse data structures. The formal semantics of the magic wand makes it hard to support the connective in an automatic verifier.

In [1] Malte Schwerhoff and Alexander J. Summers present a solution that adds support for magic wands in automatic verifiers. They also provide an implementation of their approach in Silicon, a verifier based on symbolic execution. The goal of this project is to implement their approach in [1] in Carbon, a verifier based on verification condition generation. Both verifiers handle Silver programs. An overview of the complete verification infrastructure that encompasses the two verifiers and the Silver programming language is given in [2].

2 Core Task

Carbon verifies a Silver program via an encoding to Boogie. One of the main difficulties in implementing the approach outlined in [1] in Carbon is that it is not trivial to find an encoding into Boogie such that the magic wands, which are held in a method in the program at any given point, are tracked directly in the Boogie program. This is hard for two reasons. The first reason is that finding a good representation for arbitrary wands isn't easy, since for an arbitrary wand the Boogie program would need to figure out if any of the wands that it holds matches the specified wand (where the matching wand may be syntactically different to the specified wand). The second reason is that when applying a wand the code in the Boogie program must itself traverse the structure of the wand formula to achieve the effect of the application of a wand as described in [1].

In this bachelor thesis we try to tackle the difficulty described above by using the Carbon verifier itself to track an overapproximation of the magic wands that are held at any program point during the translation of the Silver program. The idea for applying a wand is that the Boogie program checks if the wand to be applied matches any of the wands tracked in the overapproximation stored in Carbon for that program point. If there is a match with one of the wands and that particular wand hasn't been applied in the program execution up to that particular point then the wand can be successfully applied.

This concept can be seen generally as an interaction between Carbon which tracks program state statically and the Boogie program which uses the information gathered by Carbon to execute operations without having to track certain parts of program state explicitly.

The specific tasks to be done are:

- Based on the ideas in the above paragraph design a complete strategy for an encoding from Silver to Boogie for packaging and applying a magic wand (the behaviour of these operations is described in [1])
- Implement the interaction described above between Carbon and the Boogie program in a general fashion in Carbon
- Implement the designed strategy for the magic wand support in Carbon using the implementation in the previous point
- Evaluate the implementation using test cases

3 Extensions

Possible extensions for the bachelor thesis are:

- Compare the magic wand support in the Carbon verifier (which is done in this project) to the magic wand support in the Silicon verifier (which is provided in [1])

- Optimize the devised implementation of the magic wand support in the Carbon verifier
- Implement heuristics as outlined in [1] to automatically infer magic wand annotations
- Apply the concept of the interaction between the Carbon verifier which tracks program state statically and the Boogie program which uses this information to another problem

4 References

1. M. Schwerhoff and A. J. Summers: Lightweight Support for Magic Wands in an Automatic Verifier Technical Report, ETH Zurich, 2014
2. U. Juhasz and I. T. Kassios and P. Müller and M. Novacek and M. Schwerhoff and A. J. Summers: Viper: A Verification Infrastructure for Permission-Based Reasoning Technical Report, ETH Zurich, 2014.