Formally Connecting an Isorecursive with an Equirecursive Viper Semantics

Practical Work Project Description

Hongyi Ling

Supervised by Prof. Dr. Peter Müller, Thibault Dardinier and Gaurav Parthasarathy

April, 2023

1. Introduction

Program verification techniques enable proving the absence of bugs in programs, which is becoming increasingly important given the widespread use of programs in critical domains. Separation logic is one such program verification technique for heap-manipulating programs. In particular, it is a program logic for proving Hoare triples "$\{P\}\, S\, \{Q\}$" where P and Q are separation logic assertions and S is a heap-manipulating program.

Heap-manipulating programs often involve recursive pointer data structures. For example, the commonly used linked list data structure is defined with each node containing a value field and a pointer field that recursively points to the next node:

```
public class Node {
    int value;
    Node next;
}
```

One natural approach for specifying such recursively-defined structures in separation logic is via recursive predicates that are interpreted via their least fixed point, which informally refers to everything that is implied by any finite unrolling of the predicate. We call this model of predicates the *equirecursive model*. As an example, the least fixed point of the following $List$ predicate

$$List(x) = acc(x.value) * acc(x.next) * (x.next \neq null \longrightarrow List(x.next))$$

is the interpretation where $List(x)$ is true if and only if x is the head of some non-empty list of finite length and *ownership* is held to its fields.

In practice, it is desirable to automate separation logic proofs involving such recursive predicates. There are various static verification tools that follow this goal, Viper [2] being one of them. For such tools, directly implementing the equirecursive model is infeasible, since one cannot statically know when to stop unrolling the recursive definition. Instead, a predicate instance and its body are treated as different objects, and unfold and fold operations are used to explicitly exchange a predicate name for its body (or vice versa). This results in the *isorecursive model* of predicates.

Viper's isorecursive model for predicates has been formalized as part of an operational semantics for a subset of the Viper language via a total heap semantics (THS). The goal of this project is to increase the confidence of THS by formally proving in Isabelle that verification of a Viper program w.r.t. THS implies verification of a Viper program w.r.t. an equirecursive semantics (EquiSem) that uses an equirecursive model for predicates (a more abstract version of EquiSem has been formalized for a subset of Viper).

Summers and Drossopoulou [1] also relate an isorecursive model and an equirecursive model. In particular, they formalize two Hoare logics for the two models w.r.t. a concurrent language and show that if a Hoare triple where predicates are interpreted isorecursively is derivable, then so is the corresponding Hoare triple where predicates are interpreted equirecursively. In our work, we are considering Viper instead of a concurrent language (for example, we must consider inhale and exhale statements, which they do not). Moreover, we aim to directly prove a result between two operational semantics' for Viper programs (THS and EquiSem) instead of relating two Hoare logics. Finally, our definitions of the iso- and equirecursive models do not perfectly match those by Summers and Drossopoulou and their work is not mechanized in an interactive theorem prover.

2. Light Definition of the Models

In this part, we give light definitions of the EquiSem and THS models.

In EquiSem, a program state is described by σ, π, h, where environment σ maps each local variable to a value of its type, permission mask π maps each memory location to a non-negative rational value indicating the fraction of permission held, and h is a partial heap. The semantics of assertions is interpreted as the least fixed point of a set of entailment equations. Specifically, the entailment equation for a predicate P is

$$\sigma, \pi, h \vDash_E P(e) \Leftrightarrow \sigma, \pi, h \vDash_E Body(P(e)),$$

where $Body(P(e))$ is the body of the predicate $P(e)$. Knaster-Tarki's theorem [3,4] allows one to show that the least-fixed point exists for predicates in Viper if recursive occurrences of the predicate only appear in positive positions.

In THS, the permission mask π is extended to Π which also maps each predicate instance to a non-negative rational number, indicating the fractional permission amount to this predicate instance. Moreover, the heap, denoted by H, is total rather than partial as in EquiSem. A program state thus becomes σ, Π, H, with σ defined as in EquiSem. Instead of defining a semantics of possibly recursively-defined predicates as the least fixed point of an entailment equation set, a predicate instance $P(e)$ holds in state σ, Π, H, if there is sufficient permission to the predicate instance in Π.

THS makes use of unfold and fold operations (they are treated as skip statements in EquiSem) to exchange predicate names for their bodies when needed. An "unfold $P(x)$" statement changes the program state by decreasing the permission amount to $P(x)$ by one, and obtaining the assertions specified by the body of $P(x)$ in return. The fold does the reverse of unfold: "fold $P(x)$" changes the program state by replacing the resources specified by the body of $P(x)$ with an increase of permission amount to $P(x)$ itself by one.

3. Project Goals

3.1. Core Goals

- Currently, EquiSem itself has not been formalized. However, a more abstract version of EquiSem (*AbstractInterpSem*) that parametrizes over the predicate

interpretation has been formalized. A first goal is to obtain EquiSem by developing a general fixed point theory in Isabelle for AbstractInterpSem and using this theory to instantiate AbstractInterpSem with the least fixed point interpretation to obtain EquiSem.

- Develop a general strategy for connecting THS and EquiSem.
- Formally define in Isabelle the relation between the EquiSem and THS state models. Using this relation prove a relationship between the assertions in EquiSem and THS (and as a result inhale and exhale statements).

3.2. Extension Goals

- Extend the results obtained in the core goals to formally prove in Isabelle that verification of a Viper program in THS implies verification of a Viper program in EquiSem.

References

[1] Summers, Alexander J., and Sophia Drossopoulou. "A formal semantics for isorecursive and equirecursive state abstractions." *ECOOP 2013–Object-Oriented Programming: 27th European Conference, Montpellier, France, July 1-5, 2013. Proceedings 27*. Springer Berlin Heidelberg, 2013.

[2] Müller, Peter, Malte Schwerhoff, and Alexander J. Summers. "Viper: A verification infrastructure for permission-based reasoning." *Verification, Model Checking, and Abstract Interpretation: 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, January 17-19, 2016. Proceedings 17*. Springer Berlin Heidelberg, 2016.

[3] Cousot, Patrick, and Radhia Cousot. "Constructive versions of Tarski's fixed point theorems." *Pacific journal of Mathematics* 82.1 (1979): 43-57.

[4] Dardinier, Thibault, Peter Müller, and Alexander J. Summers. "Fractional resources in unbounded separation logic." *Proceedings of the ACM on*

Programming Languages 6.OOPSLA2 (2022): 1066-1092.