Program Equivalence: An Interactive Relational Separation Logic Prover Implemented in Maude

Andrei Alin Corodescu

Scientific Coordinator:
Conf. Dr. Ciobaca Stefan

July 2, 2018

Outline

The Problem

Problem Description

Reducing the gap between theoretical and practical aspects of formal program equivalence verification to increase software quality by making robust methods of verification accessible and easy to use.

The Problem

Problem Description

Reducing the gap between theoretical and practical aspects of formal program equivalence verification to increase software quality by making robust methods of verification accessible and easy to use.

Difficulties

- Representing the theoretical concepts
- Computationally-hard problems
- User experience

Outline

Relational Separation Logic

- Helps reason about how two programs are related
- Hoare Quadruples : $\{R\} \begin{matrix} C \\ C\prime \end{matrix} \{S\}$
- Proof rules : $$\frac{R \Rightarrow R_1 \quad \{R_1\} \begin{matrix} C \\ C\prime \end{matrix} \{S_1\} \quad S_1 \Rightarrow S}{\{R\} \begin{matrix} C \\ C\prime \end{matrix} \{S\}}$$
- Separation Logic axioms : $\{E \mapsto -\}[E] := F\{E \mapsto F\}$.

Outline

Maude

- Based on Rewriting logic
- Natural Representation of logics through sorts and operators
- Powerful meta language applications

Example

```
rl [Consequence] : { R } C1 -- C2 { S } => ((R => R1) <> ({
    R1} C1 -- C2 {S1})) <> (S1 => S) [nonexec] .
```

Outline

Goals

- The central concept of the prover is Goal, which represents something to be proven.
- A Goal is consumed if it is an matched with an *axiom*, is *manually admitted* by the user, is *automatically proven* (in case of implications) or it is *replaced* by the goals generated by applying a proof rule.
- Goals are stored in a GoalStack structure

Outline

Axiom Recognition

- Simple goals that match axioms are automatically admitted by the prover.
- Equality between variables is interpreted before matching axioms
- Axioms are represented as terms with variables

Outline

Automatic Proof of Implications

- Uses the search functionality of Maude
- Searches for a series of rewrites from `R => S` to `true`
- Rewrite rules denoting relation equivalences and implications

Demo

Conclusions

- The prover showcases:
  - A promising executable environment for Relational Separation Logic which can be improved upon in the future
  - The features of Maude that make it fit for this purpose
- Personal conclusions:
  - Learning by modelling and applying logics
  - Shortcomings of Maude because of it not being widely adopted