

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

Andrei Alin Corodescu

Scientific Coordinator:
Conf. Dr. Ciobaca Stefan

July 1, 2018

- 1 The Problem
- 2 Technologies and theoretical concepts
 - Relational Separation Logic
 - Maude
- 3 Prover Implementation
 - Goals
 - Axiom Recognition
 - Automatic Proof of Implications

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

- 1 The Problem
- 2 Technologies and theoretical concepts
 - Relational Separation Logic
 - Maude
- 3 Prover Implementation
 - Goals
 - Axiom Recognition
 - Automatic Proof of Implications

Relational Separation Logic

- Helps reason about how two programs are related

- Hoare Quadruples : $\{R\} \overset{C}{C'} \{S\}$

- Proof rules :
$$\frac{R \Rightarrow R_1 \quad \{R_1\} \overset{C}{C'} \{S_1\} \quad S_1 \Rightarrow S}{\{R\} \overset{C}{C'} \{S\}}$$
- Separation Logic axioms : $\{E \mapsto -\}[E] := F\{E \mapsto F\}.$

- 1 The Problem
- 2 Technologies and theoretical concepts
 - Relational Separation Logic
 - **Maude**
- 3 Prover Implementation
 - Goals
 - Axiom Recognition
 - Automatic Proof of Implications

- 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] .
```


- 1 The Problem
- 2 Technologies and theoretical concepts
 - Relational Separation Logic
 - Maude
- 3 Prover Implementation
 - Goals
 - Axiom Recognition
 - Automatic Proof of Implications

- The central concept of the prover is **Goal**, which represents something to be proven.
- A **Goal** is consumed if it is 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

- 1 The Problem
- 2 Technologies and theoretical concepts
 - Relational Separation Logic
 - Maude
- 3 Prover Implementation
 - Goals
 - **Axiom Recognition**
 - Automatic Proof of Implications

- 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

- 1 The Problem
- 2 Technologies and theoretical concepts
 - Relational Separation Logic
 - Maude
- 3 Prover Implementation
 - Goals
 - Axiom Recognition
 - Automatic Proof of Implications

Automatic Proof of Implications

- Uses the **search** functionality of Maude
- Searches for a series of **rewrites** from $R \Rightarrow S$ to true
- Rewrite rules denoting relation equivalences and implications

- 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