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

Andrei-Alin Corodescu

Scientific Coordinator:
Conf. Dr. Ciobâcă Ștefan

July 4, 2018

Outline

Problem Description

- Program Equivalence.
- Absence of an implementation for Relational Separation Logic, theoretical framework supporting formal proofs for program equivalence.

The Problem

Problem Description

- Program Equivalence.
- Absence of an implementation for Relational Separation Logic, theoretical framework supporting formal proofs for program equivalence.

Difficulties

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

Example

Program 1:

```
while (c != nil) do
  x := [y];
  [c] := -x;
  c := [c + 1]
od
```

Program 2:

```
x' := [y'];
while (c' != nil) do
  [c'] := -x';
  c' := [c' + 1]
od
```

Note

If the address y is part of the list starting at c, the two programs are not equivalent.

Outline

Relational Separation Logic

- Helps reason about how two programs are related
- Hoare Quadruples : $\{R\} \begin{array}{c} C \\ C' \end{array} \{S\}$
- Example proof rule (Consequence) :

$$\frac{R \Rightarrow R_1 \qquad \{R_1\} \begin{array}{c} C \\ C' \end{array} \{S_1\} \qquad S_1 \Rightarrow S}{\{R\} \begin{array}{c} C \\ C' \end{array} \{S\}}$$

- Axioms : $\{E \mapsto -\}[E] := F\{E \mapsto F\}$.

Relational Separation Logic - Example

Consider the two programs exemplified earlier to be denoted by C_1 and C_2.

Example

- $\text{List x} \stackrel{def}{=} (\text{x} = \text{nil} \wedge \text{Emp}) \vee \exists na. \begin{pmatrix} x \mapsto a, n \\ x \mapsto a, n \end{pmatrix} * \text{List } n$
- $\left\{ \left(Same * List\ c * \begin{pmatrix} y \mapsto x_0 \\ y\prime \mapsto x_0 \end{pmatrix} \right) \wedge y = y\prime \wedge c = c\prime \right\}$

$$C_1$$

$$C_2$$

$$\{Same \wedge y = y\prime \wedge c = c\prime\}$$

Outline

1. The Problem

2. Technologies and theoretical concepts
   - Relational Separation Logic
   - Maude

3. Demonstration

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] .
```

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