"ALEXANDRU IOAN CUZA" UNIVERSITY OF IAȘI
FACULTY OF COMPUTER SCIENCE

Program Equivalence : Relational Separation Logic interactive prover implemented in Maude

ANDREI-ALIN CORODESCU

Session: *July, 2018*

Scientific Coordinator
Conf. Dr. Ciobâcă Ștefan

Contents

1 Introduction

The present paper describes the development of an interactive tool for reasoning how to programs are related, based on studied and previously used theoretical concepts and technologies which facilitate the implementation.

The tool represents an implementation of Hoare Logic - which allows formal reasoning about a program - , along with 2 of its extensions, namely the Separation Logic (named Separation Logic from now on) and Relational Separation Logic [1] (named Relational Logic from now on). The 2 extensions simplify the Hoare Logic proofs, mainly using the "*" connector, allowing for local reasoning of effects of statements in a program . The tool has been implemented in Maude, a high performance logical framework with powerful metalanguage applications which facilitate the implementations of executable environments for logics.

The tool is built as a CLI which helps [2] [3] [4] [5] [6] [7] [8] [9] with reasoning how two programs are related using Relational Separation Logic specifications. As a consequence of the dependency of Relational Separation Logic on Separation Logic, proofs about single programs using the latter are also supported by the tool. The tool has been developed with extensibility in mind, the main desired extensions being concurrent programs support and automatic proofs.

The rest of the paper is organized as follows:

2 Contributions

Personal contributions to the realization of the project :

- Modelled the Relational Logic and Separation Logic using Maude equational and rewriting logic specifications .

- Developed an interactive tool for reasoning about program behaviour using the aforementioned logics.

- Automation of some tasks which makes the tool more convenient to use .

- Examples of formal proofs done using the tool

3 Description of the problem

The problem this project is aiming to solve is related to formal reasoning about the execution of code, mainly focusing on how two programs are related to each other (most often the relation to be proven is equivalence)

Comparing programs or code fragments and studying their equivalence is part of every software engineer's activities when they are testing an alternative implementation for an existing solution, fixing bugs, launching new product versions, etc . Naturally, for every process completed manually there are efforts being made in order to make it more efficient, less error-prone and, in the end, automate the process all together. Once such a task is automated in software engineering, it can be included in the flow of any research or development phase of a product. An example benefiting from a formal proof of program equivalence is compiler optimization, where the optimized code needs to be equivalent to the input one .

This project aims to lay the foundations of a tool which facilitates formal reasoning about program equivalence with a focus on extensibility and automation of tasks.

4 Previous work

Previous work related to the topic has been done mostly in terms of Separation Logic based tools, with Relational Separation Logic not being treated as much.

A notable example which also uses the same framework as this project, namely Maude, is the Java+ITP[4] tool, which enables analysis of Java programs using Separation Logic. The implementation relies heavily on Maude's ITP (iterative theorem prover).

5 Relational Separation Logic Interactive Prover

This chapter will include details related to the implementation and usage of the Relational Separation Logic prover and highlight a few of Maude's features which made it the a language fit for the purpose of this project.

Aici va veni figura care explica input -> rewriting maxim -> output

5.1 Modelling the Separation Logics

Thanks to Maude's ability to naturally represent logics, the modelling of the Separation Logic and Relational Separation Logic is mostly reduced to translating the logical formulae into appropriate syntax for Maude, making the representational distance [6] almost non existent.

Each proof rule in the Separation Logic and Relational Separation Logic has been translated into a Maude rewrite rule. The resulting rules have been marked as non-executable because the goal is to control when they are applied.

5.2 Interaction by Maude LOOP-MODE

Maude's LOOP-MODE module is the entry point for any interactive Maude application, being the only way to store state between commands. In this chapter we will discuss how we handle input and output , the structure of a state and the role every element plays in the execution of the prover.

We will make a distinction between **Loop state**, which specific to Maude's LOOP-MODE module and the **Prover state** which is specific to our use-case, and holds the necessary data related to the state of the proof we are currently working on. The **Prover state** is encapsulated within **Loop state**.

5.3 Input and output handling

To understand how our prover handles input and output, it is important to note that the flows of the LOOP-MODE module is as follows:

5.4 State of the program

Following the examples provided together with the Maude Manual [7], our state is composed multiple sections modified accordingly during program execution .

5.5 Prover execution flow

In this section we will describe the flow of a demonstration made using our prover.

5.6   Automated processes

This section will describe the functional parts of the prover which have been automated to make it more user-friendly and even prove sub-goals of the main demonstration by itself. Since the automation of the prover is part of the future goals of this project, we will also include a few notes on how we might approach problems related to this goal, even though they have not been implemented yet.

5.6.1   Automatic matching of axioms and previously proven goals

5.6.2   Automatic demonstration of implications

5.7   User Interface

Since our project aims to simplify the demonstrations of Separation Logic / Relational Separation Logic formulae, using an interactive prover, the user interface plays a major role in the success of the project - the easier it is to use and understand, the more chances of success it has. This idea has been presented in [2], when discussing key elements of tools related to program verification.

Maude provides a simple, yet powerful system for controlling the output formatting of the CLI through which we aim to make the demonstrations more readable and user-friendly.

For a more advanced user interface or to expose the prover's functionalities through a programming interface, a wrapper written in more feature-rich language through which to control the Maude environment is probably the best choice. The concept is exemplified the Maude plugin for Eclipse, **citation needed** which enables writing of Maude code inside the Eclipse IDE, and the plugin communicates with the underlying Maude environment.

5.8   Future directions

6   Conclusions

References

[1] Hongseok Yang. Relational separation logic. *Theor. Comput. Sci.*, 375(1-3):308–334, April 2007.

[2] Peter W. O'Hearn. A primer on separation logic (and automatic program verification and analysis). In *Software Safety and Security*, 2012.

[3] J. C. Reynolds. Separation logic: a logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, pages 55–74, 2002.

[4] Ralf Sasse and José Meseguer. Java+itp: A verification tool based on hoare logic and algebraic semantics. *Electronic Notes in Theoretical Computer Science*, 176(4):29 – 46, 2007. Proceedings of the 6th International Workshop on Rewriting Logic and its Applications (WRLA 2006).

[5] Traian Florin Şerbănuţă, Grigore Roşu, and José Meseguer. A rewriting logic approach to operational semantics. *Information and Computation*, 207(2):305 – 340, 2009. Special issue on Structural Operational Semantics (SOS).

[6] Theodore McCombs. Maude 2.0 primer, August 2003.

[7] Manuel Clavel, Francisco Duran, Steven Eker, Santiago Escobar, Narciso Patrick Lincoln, Marti-Olietand Jose Meseguer, and Carolyn Talcott. *Maude Manual (Version 2.7.1)*, July 2016.

[8] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73 – 155, 1992.

[9] James Brotherston, Nikos Gorogiannis, and Rasmus L Petersen. A generic cyclic theorem prover. In *Asian Symposium on Programming Languages and Systems*, pages 350–367. Springer, 2012.

[10] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All About Maude - a High-performance Logical Framework: How to Specify, Program and Verify Systems in Rewriting Logic*. Springer-Verlag, Berlin, Heidelberg, 2007.