Tools for low level programs

Deliverable D4-3

ANR project VECOLIB

first version September 2016,
last update August 2017

Abstract

This report gives a synopsis of the set of tools developed during the VECOLIB project for the verification of low level programs. This set includes solvers for Separation Logic, a model checker, and abstract domains used in static analysers. The report compares these tools with similar tools developed outside this project.

1 Decision Procedures

For the verification of low level programs, we implemented several decision procedures for (fragments of) Separation Logic. These procedures and the three tools implementing them have been presented in details in the deliverable D1-1. In this part of the report, we only outline the improvements done to these tools during the last years of the project, from October 2015 to August 2017.

SPEN [14, 8] has been updated to identify a particular case of inductive definitions for which the general procedure is complete and has polynomial time complexity. This particular case includes interesting inductive definitions like skip lists and acyclic doubly linked lists. The updated procedure has been published in [7].

The front-ends of SLIDE [13] and CVC4SEPLOG [6] have been updated to read problems written in the new SMT-LIB format [1] for SL [].

The algorithm of CVC4SEPLOG has been improved and published in [12].

2 Model-checkers

In the VECOLIB proposal, we aim to develop a model-checking tool based on symbolic trace exploration using the encoding of heaps in Separation Logic, interpolation, and bi-abduction. This project has been reconsidered for the reasons detailed in the deliverable D1.3, mainly the lack of maturity of the existing solvers. Due to this reason, in the rest of the VECOLIB project, we envisage experimenting with interpolation in high-level logics, by adapting our alternating data automata (ADA) emptiness checker (reported in deliverable D4.5) to work with strings and algebraic data types, in addition to linear arithmetic.

Although concerned only with the model-checking of formulas (and not programs specified) in SL, we mention the work of Brotherston et al [3] published in 2016. It defines a model-checking algorithm for the fragment of SL used in static analysers, i.e., the symbolic heaps fragment. This result may be interesting for the development of testing techniques.

3 Static Analysers

Frama-C [10] includes a new static analysis architecture, called EVA [4], that eases the development of new analyses by providing a clean interface for connecting new abstract domains. Using the existing memory model of FRAMA-C (see deliverable D2.1), several abstract domains have been connected and experimented on existing case studies of programs manipulating pointers. The results are more precise (some false alarms are removes) without much loss of time performance for the analysis. The precise experimental results are given in [4].

For the analysis of programs using pointers and pointer arithmetics, the new abstract domain of garbled-mix intervals has been developed and integrated at FRAMA-C (both VALUE and EVA plugins), as described in deliverable D2.1.

Celia [2] is a plugin of FRAMA-C which assumes a memory model based on a set of disjoints records. It has been used, before the VECOLIB During the project, we developed a new abstract domain, defined in [9], that assumes a hierarchical memory model in order to be able to analyse low level code used by dynamic memory allocators (DMA). This model is composed of two levels: a level where the memory is a sequence of bytes organised in a heap list, and a level where the model is the set of records. As described in [9]

and resumed in D2.1, we were able to analyse several implementations of DMA. Compared with other analysers, e.g., [5, 11], our abstract domain has more expressive power and therefore is able to deal more precisely with a larger set of DMA. This expressive power is balanced by a loss in time performance. We are currently working on improving our implementation in order to obtain better execution times.

References

[1] C. Barrett, A. Stump, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2010.

[2] A. Bouajjani, C. Dragoi, C. Enea, and M. Sighireanu. On inter-procedural analysis of programs with lists and data. In *PLDI*, pages 578–589. ACM, 2011.

[3] J. Brotherston, N. Gorogiannis, M. I. Kanovich, and R. Rowe. Model checking for symbolic-heap separation logic with inductive predicates. In *Proceedings of POPL*, pages 84–96. ACM, 2016.

[4] D. Bühler. *Structuring an Abstract Interpreter through Value and State Abstractions: EVA an Evolved Value Analysis for Frama-C*. PhD thesis, University of Rennes, 2017.

[5] C. Calcagno, D. Distefano, P. W. O'Hearn, and H. Yang. Beyond reachability: Shape abstraction in the presence of pointer arithmetic. In *SAS*, volume 4134 of *LNCS*, pages 182–203. Springer, 2006.

[6] CVC4SepLog. https://github.com/timothy-king/CVC4SepLogic.

[7] C. Enea, O. Lengál, M. Sighireanu, and T. Vojnar. Compositional entailment checking for a fragmentof separation logic. *Form Methods Syst Des*, 2017.

[8] C. Enea, O. Lengál, M. Sighireanu, and T. Vojnar. SPEN: A solver for separation logic. In *Proceedings of NFM*, volume 10227 of *Lecture Notes in Computer Science*, pages 302–309, 2017.

[9] B. Fang and M. Sighireanu. Hierarchical shape abstraction for analysis of free-list memory allocators. In *LOPSTR*, volume 10184 of *Lecture Notes in Computer Science*. Springer, 2016.

[10] F. Kirchner, N. Kosmatov, V. Prevosto, J. Signoles, and B. Yakobowski. Frama-C: A software analysis perspective. *Formal Asp. Comput.*, 27(3):573–609, 2015.

[11] J. Liu and X. Rival. Abstraction of arrays based on non contiguous partitions. In *VMCAI*, volume 8931 of *LNCS*, pages 282–299. Springer, 2015.

[12] A. Reynolds, R. Iosif, C. Serban, and T. King. *A Decision Procedure for Separation Logic in SMT*, pages 244–261. Springer International Publishing, Cham, 2016.

[13] SLIDE. http://www.fit.vutbr.cz/research/groups/verifit/tools/slide/.

[14] SPEN. https://www.github.com/mihasighi/spen.