S2S: An Efficient Solver for Cyclic Proofs

Quang Loc Le

Teesside University, Middlesbrough, United Kingdom - Q.Le@tees.ac.uk

Abstract. This paper describes S2S, a proof system for satisfiability and entailment problems in an extended fragment of separation logic. S2S includes a generic framework for cyclic proofs targeting unbounded data structures and three instantiations: a satisfiability solver and an entailment procedure for separation logic, and a satisfiability solver for arithmetic and strings.

Keywords: Separation Logic, Cyclic Proofs, Entailment, Satisfiability.

1 S2S System

This paper presents S2S system for solvers in array separation logic [11] extended with inductive predicates, string and arithmetic. Central to our system is a generic framework for cyclic proofs to support inductive reasoning over the infinite domains. In the following, we describe the syntax of formulas, principles of cyclic proofs and some notes about our implementation. First, the syntax of formulas is defined as follows.

Definition 1. *Non-heap formulas π (for arithmetical and string constraints), spatial formulas κ, symbolic heaps Δ and disjunctions Φ are given by the following grammar.*

$$\Phi ::= \Delta \mid \Phi_1 \vee \Phi_2 \qquad \Delta ::= \exists \bar{v}.\ (\kappa \wedge \pi)$$
$$\kappa ::= \mathtt{emp} \mid x \mapsto c(f_i{:}v_i) \mid \mathtt{P}(\bar{v}) \mid array(a, a) \mid \kappa * \kappa$$

where v ranges over an infinite set Var of variables, $\bar{v}$ over sequences of variables ($\bar{v}_i$ for its i^{th} element), f_i over a finite set Fields, P over a finite set $\mathcal{P}$.

The array predicate only records the bound of continuous memory blocks, not their contents. Each predicate in $\mathcal{P}$ is inductively defined by a disjunction Φ.

Cyclic Proof System Central to our approach is a generic framework to construct a forest of disjoint cyclic reduction trees for an input, either an entailment or a satisfiability problem. This framework has three main procedures for leaf node evaluation, back-link construction, and proof search (with inference rules) [6]. A cyclic reduction tree is a tuple $(V, E, \mathcal{C})$ where V is a finite set of nodes $\mathtt{e}$, E a finite set of edges, and $\mathcal{C}$ is a partial function which captures back-links in the proof tree. A directed edges $(\mathtt{e},\mathtt{PR},\mathtt{e}') \in E$ (where $\mathtt{e}'$ is a child of $\mathtt{e}$) means that $\mathtt{e}'$ is derived from $\mathtt{e}$ via rule PR. If $\mathcal{C}(\mathtt{e}_c \rightarrow \mathtt{e}_b, \sigma)$ is a back-link, $\mathtt{e}_b$ is linked back to an ancestor node $\mathtt{e}_c$ through the substitution σ.

Implementation We have implemented the system in Ocaml, from scratch. It contains three main components: front end with parsers, the proof systems and backend with

SMT solvers. For the front end, we have implemented a parser for the grammar above. Moreover, we support separation logic formulas in SMT-LIB v2.5 by using the parser described in [2]. For the backend, we utilise Z3 [9]. For the proof system, we have implemented the generic framework through abstract classes for data (e.g., node, edge, substitution) and controller (proof search with three procedures). To generate a concrete cyclic proof system, one needs to provide an implementation for these all classes.

Instantiations We have implemented three concrete cyclic proof systems. The first system is a satisfiability solver in separation logic with general inductive predicates and arithmetic. The implementation is an extension of `S2SAT` [12,8]. The second one is an entailment solver in the same fragment of separation logic above. Its implementation is an extension of CYCLIC [1] with lemma synthesis [3,7]. The last system is a satisfiability solver for string logics. The implementation is based on $\texttt{Kepler}_{22}$ [4]. In all these three systems, any input of the leaf node evaluation method could be transformed into Presburger arithmetic and discharged efficiently by Z3.

SL-COMP 2019 `S2S` participated in all divisions except *bsl_sat*.

2 Future Work

One strength of our proof system is the capability of extending to a bi-abduction procedure. Indeed, we plan to extend our proof system to support compositional reasoning for both safety verification [3] and bug finding [5,10].

References

1. J. Brotherston, D. Distefano, and R. L. Petersen. Automated cyclic entailment proofs in separation logic. In *CADE*, pages 131–146, 2011.
2. Radu Iosif, Cristina Serban, Andrew Reynolds, and Mihaela Sighireanu. Encoding separation logic in smt-lib v2.5, 2018. [online; accessed 26-Dec-2018, `https://github.com/sl-comp/SL-COMP18/blob/master/input/Docs/smtlib-sl.pdf`].
3. Quang Loc Le, Cristian Gherghina, Shengchao Qin, and Wei-Ngan Chin. Shape analysis via second-order bi-abduction. In *CAV*, volume 8559, pages 52–68. 2014.
4. Quang Loc Le and Mengda He. A decision procedure for string logic with quadratic equations, regular expressions and length constraints. In *APLAS*, pages 350–372, 2018.
5. Quang Loc Le, Asankhaya Sharma, Florin Craciun, and Wei-Ngan Chin. Towards complete specifications with an error calculus. In *NASA Formal Methods*, pages 291–306, 2013.
6. Quang Loc Le, Jun Sun, and Wei-Ngan Chin. Satisfiability modulo heap-based programs. In *Proceedings of CAV*. 2016.
7. Quang Loc Le, Jun Sun, and Shengchao Qin. Frame inference for inductive entailment proofs in separation logic. In *Proceedings of TACAS*, pages 41–60, 2018.
8. Quang Loc Le, Makoto Tatsuta, Jun Sun, and Wei-Ngan Chin. A decidable fragment in separation logic with inductive predicates and arithmetic. In *CAV*, pages 495–517, 2017.
9. Leonardo M. and Nikolaj B. Z3: An Efficient SMT Solver. In *TACAS*, 2008.
10. Long H. Pham, Quang Loc Le, Quoc-Sang Phan, Jun Sun, and Shengchao Qin. Enhancing Symbolic Execution of Heap-based Programs with Separation Logic for Test Input Generation. *CoRR*, abs/1712.06025, 2017.
11. J. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *IEEE LICS*, pages 55–74, 2002.
12. Makoto Tatsuta, Quang Loc Le, and Wei-Ngan Chin. Decision procedure for separation logic with inductive predicates and presburger arithmetic. In *APLAS*. 2016.