Harrsh: A Tool for Unified Reasoning about Symbolic-Heap Separation Logic

Jens Katelaan[2], Christoph Matheja[1], Thomas Noll[1], and Florian Zuleger[2]

[1] RWTH Aachen University, Germany
[2] TU Wien, Austria

Harrsh [4] is a tool for unified reasoning about the symbolic heap fragment of separation logic with user-defined systems of inductive predicate definitions (SIDs). It supports satisfiability checking, entailment checking, as well as checking various other *robustness properties* of symbolic heaps. For example, Harrsh can prove for arbitrary symbolic heaps (1) the absence of dangling pointers (establishment), (2) definite reachability between pairs of variables in the heap, (3) the absence of unreachable allocated memory locations (garbage freedom), and (4) acyclicity of all paths involving only non-dangling pointers.

Harrsh decides all of these properties using a novel automaton model, so-called *heap automata* [3], which works directly on the structure of symbolic heaps. A heap automaton examines an SID bottom-up, starting from the non-recursive base case. At each stage of this analysis, a heap automaton remembers a fixed amount of information. Heap automata enjoy a variety of closure properties (intersection, union and complementation). We can decide properties of symbolic heaps by means of heap automata as follows. Given a heap automaton (encoding a property such as satisfiability, acyclicity, or entailment w.r.t. a fixed predicate call) and an SID of interest, we compute the intersection of the heap automaton with the SID (a process called *automatic refinement*) and then check for emptiness of the result.

Heap automata thus enable a unified approach for obtaining decision procedures and counterexample generation for symbolic heaps: To obtain a decision procedure for a property of symbolic heaps, simply add a heap automaton for this property to Harrsh.

We formally introduced the heap automaton framework and defined heap automata for satisfiability and various other robustness properties in [3]. In [4] we presented Harrsh, our implementation of the heap automaton framework and of the heap automata for all robustness properties studied in [3]. In recent research,[3] we have shown how to use heap automata for entailment checking in the fragment SLRD_{btw} proposed in [2], an SL fragment for expressing data structures with bounded treewidth. We have implemented this approach in Harrsh, making it the first tool to support entailment checking for the entire SLRD_{btw} fragment.

Harrsh is licensed under the MIT license and available on GitHub at https://github.com/katelaan/harrsh/. Harrsh was implemented in Scala and runs on the JVM.

[3] not published by the solver registration deadline for SL-COMP 2019

Harrsh has its own input format,[4] but also supports both Cyclist's [1] input format and the SL-COMP'18 input format.

Many SL-COMP entailment benchmarks violate the syntactic restrictions of SLRD$_{btw}$. For this reason, Harrsh comes with a preprocessor that is able to transform many (but not all) benchmarks in the SL-COMP `qf_shid_entl` category into equivalent SLRD$_{btw}$ specifications.

At SL-COMP'18, where Harrsh competed in the categories `qf_shls_sat` and `qf_shid_sat`, we saw that Harrsh's satisfiability checker excels at analyzing intricate, possibly mutually recursive inductive data structure definitions. Specifically, it performed very well in the `qf_shid_sat` category, outperforming all other tools that did not produce wrong results. Harrsh was, however, significantly slower than many other tools on list benchmarks and on queries that contain a large number of predicate calls. These weaknesses are not surprising, as (unlike other solvers) Harrsh does not implement any dedicated list reasoning, nor any optimizations for dealing with large formulas. Compared to all other SL-COMP'18 participants, Harrsh has the additional disadvantage that it runs on the JVM: On simple benchmarks, more than 99% percent of the runtime of Harrsh is spent starting and shutting down the JVM. We conjecture that Harrsh's entailment checker will display strengths and weaknesses similar to its satisfiability checker.

In 2019, Harrsh will compete in the categories `qf_shls_entl` and `qf_shid_entl` as well as `qf_shls_sat` and `qf_shid_sat`.

References

1. Brotherston, J., Gorogiannis, N., Petersen, R.L.: A generic cyclic theorem prover. In: APLAS 2012. pp. 350–367. Springer (2012)
2. Iosif, R., Rogalewicz, A., Simacek, J.: The tree width of separation logic with recursive definitions. In: CADE-24. LNCS, vol. 7898, pp. 21–38. Springer (2013)
3. Jansen, C., Katelaan, J., Matheja, C., Noll, T., Zuleger, F.: Unified reasoning about robustness properties of symbolic-heap separation logic. In: Yang, H. (ed.) Proceedings of ESOP, Uppsala, Sweden. pp. 611–638. Springer Berlin Heidelberg, Berlin, Heidelberg (2017)
4. Katelaan, J., Matheja, C., Noll, T., Zuleger, F.: Harrsh: A tool for unied reasoning about symbolic-heap separation logic. In: Barthe, G., Korovin, K., Schulz, S., Suda, M., Sutcliffe, G., Veanes, M. (eds.) LPAR-22 Workshop and Short Paper Proceedings. Kalpa Publications in Computing, vol. 9, pp. 23–36. EasyChair (2018)

[4] Documented in the GitHub repository