Symbolic Automata Theory with Applications

Margus Veanes

Microsoft Research, Redmond, WA, USA
margus@microsoft.com

Abstract

Symbolic automata extend classic finite state automata by allowing transitions to carry predicates over rich alphabet theories. The key algorithmic difference to classic automata is the ability to efficiently operate over very large or infinite alphabets. In this talk we give an overview of what is currently known about symbolic automata, what their main applications are, and what challenges arise when reasoning about them. We also discuss some of the open problems and research directions in symbolic automata theory.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.3 Formal Languages

Keywords and phrases automata, transducers, infinite alphabets

Digital Object Identifier 10.4230/LIPIcs.CSL.2017.7

Category Invited Talk

1 Overview

This talk provides an overview of the recent results in the theory and practice of *symbolic automata*, which are models for automata based reasoning about sequences and trees over complex element domains. Classic finite state automata have been used in a wide variety of applications, including lexical analysis [1], software verification [3], text processing [2], and computational linguistics [13]. One major limitation of classic automata is that their alphabets need to be finite and small for the algorithms to scale. To overcome this limitation, there have been proposals to extend classic automata to use *predicates* instead of concrete characters on state transitions [16, 20]. This talk focuses on work done following the definition of symbolic finite automata presented in [17], where predicates are drawn from an effective Boolean algebra. Other, orthogonal, approaches to accommodate infinite alphabets are based on automata with registers [12]. A meaning of symbolic automata that is different from the one used here is sometimes used to refer to classic finite state automata with BDD based representation of state spaces [14].

Despite the support for infinite alphabets, symbolic automata retain many of the good properties of their finite-alphabet counterparts, such as closure under Boolean operations. The theory and algorithms of *symbolic finite automata* (s-FAs) and *symbolic finite transducers* (s-FTs) has recently received considerable attention [18, 5]. Many applications have emerged that make use of s-FAs and s-FTs: verification of string sanitizers [10], analysis of tree-manipulating programs [9], synthesis of string encoders [11], regex support in parameterized unit testing [17], similarity analysis of binaries [4], parallelization of string manipulating code [19], and fusion of streaming computations [15].

There are also some cases when classic properties over finite alphabets, either do not generalize to the symbolic setting [6, 11], or when algorithms have turned out to be difficult to generalize to the symbolic setting [7] due to lack of proper data structure support. The intent of this talk is to explain the difference between the symbolic and the finite-alphabet case, to give an overview about what is known in symbolic automata theory, which applications have been the driving force behind this theory, and to discuss open problems. A recent tutorial on symbolic automata and transducers is given in [8] that also presents some new properties not formally investigated in earlier papers.

References

1 Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison-Wesley, 2006.

2 Rajeev Alur, Loris D'Antoni, and Mukund Raghothaman. Drex: A declarative language for efficiently evaluating regular string transformations. *ACM SIGPLAN Notices – POPL'15*, 50(1):125–137, 2015.

3 Ahmed Bouajjani, Peter Habermehl, and Tomáš Vojnar. Abstract regular model checking. In *CAV'04*, pages 372–386. Springer, 2004.

4 Mila Dalla Preda, Roberto Giacobazzi, Arun Lakhotia, and Isabella Mastroeni. Abstract symbolic automata: Mixed syntactic/semantic similarity analysis of executables. *ACM SIGPLAN Notices – POPL'15*, 50(1):329–341, 2015.

5 Loris D'Antoni and Margus Veanes. Minimization of symbolic automata. *ACM SIGPLAN Notices – POPL'14*, 49(1):541–553, 2014.

6 Loris D'antoni and Margus Veanes. Extended symbolic finite automata and transducers. *Formal Methods System Design*, 47(1):93–119, August 2015.

7 Loris D'Antoni and Margus Veanes. Forward bisimulations for nondeterministic symbolic finite automata. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 23rd International Conference, TACAS 2017*, volume 10205 of *LNCS*, pages 518–534. Springer, 2017.

8 Loris D'Antoni and Margus Veanes. The power of symbolic automata and transducers. In *Computer Aided Verification, 29th International Conference (CAV'17)*. Springer, 2017.

9 Loris D'Antoni, Margus Veanes, Benjamin Livshits, and David Molnar. Fast: A transducer-based language for tree manipulation. *ACM TOPLAS*, 38(1):1–32, 2015.

10 Pieter Hooimeijer, Benjamin Livshits, David Molnar, Prateek Saxena, and Margus Veanes. Fast and precise sanitizer analysis with BEK. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, 2011.

11 Qinheping Hu and Loris D'Antoni. Automatic program inversion using symbolic transducers. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2017, pages 376–389, 2017.

12 Amaldev Manuel and Ramaswamy Ramanujam. Automata over infinite alphabets. In Deepak D'Souza and Priti Shankar, editors, *Modern Applications of Automata Theory*, pages 529–554. World Scientific, 2012.

13 Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

14 Kristin Y. Rozier and Moshe Y. Vardi. A multi-encoding approach for LTL symbolic satisfiability checking. In *FM'11*, pages 417–431, 2011.

15 Olli Saarikivi, Margus Veanes, Todd Mytkowicz, and Madan Musuvathi. Fusing effectful comprehensions. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2017, pages 17–32. ACM, 2017.

16 Gertjan van Noord and Dale Gerdemann. Finite state transducers with predicates and identities. *Grammars*, 4(3):263–286, 2001.

17 Margus Veanes, Peli de Halleux, and Nikolai Tillmann. Rex: Symbolic regular expression explorer. In *ICST'10*, pages 498–507. IEEE, 2010.

18 Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj Bjørner. Symbolic finite state transducers: Algorithms and applications. *ACM SIGPLAN Notices – POPL'12*, 47(1):137–150, 2012.

19 Margus Veanes, Todd Mytkowicz, David Molnar, and Benjamin Livshits. Data-parallel string-manipulating programs. *ACM SIGPLAN Notices – POPL'15*, 50(1):139–152, 2015.

20 Bruce W. Watson. *Extended Finite State Models of Language*, chapter Implementing and using finite automata toolkits, pages 19–36. Cambridge U. Press, 1999.