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


