Journal of Logical and Algebraic Methods in Programming 86 (2017) 391–407

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Logic programming approach to automata-based decision procedures

Gulay Unel [a,*], David Toman [b]

[a] Department of Information Technologies, Işık University, Şile, İstanbul, 34980, Turkey
[b] David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 25 March 2015
Received in revised form 11 February 2016
Accepted 26 February 2016
Available online 2 March 2016

Keywords:
WS1S
WS2S
Automata
Complex-value Datalog (Datalog^{cv})
Magic Sets
SLG-resolution

ABSTRACT

We propose a novel technique that reduces the decision problem of WSnS (weak monadic second-order logic with n successors) to the problem of evaluation of Complex-value Datalog queries. We then show how the use of advanced implementation techniques for Logic Programs, in particular the use of tabling in the XSB system, yields a considerable improvement in performance over more traditional approaches. We also explore various optimizations of the proposed technique based on variants of tabling and goal reordering. Although our primary focus is on WS1S, the logic of single successor, we show that it is straightforward to adapt our approach for other logics with existing automata-theoretic decision procedures, for example WS2S.

1. Introduction

Monadic second-order logics provide means to specify regular properties of systems in a succinct way. In addition, these logics are decidable by virtue of the connection to automata theory [1,2]. However, only recently tools based on these ideas—in particular the MONA system [3]—have been developed and shown to be efficient enough for practical applications [4].

However, for reasoning in large theories consisting of relatively simple constraints, such as theories capturing UML class diagrams or database schemata, the MONA system runs into a serious state-space explosion problem—the size of the automaton capturing the (language of) models for a given formula quickly exceeds the space available in most computers. The problem can be traced to the *automata product* operation that is used to translate conjunctions in the original formulæ rather than to the theoretically more problematic projection/determinization operations needed to handle quantifier alternations.

This paper introduces a technique that combats this problem. Unlike most other approaches that usually attempt to use various compact representation techniques for automata, e.g., based on BDDs [5] or on state space factoring using a *guided* automaton [3], our approach is based on techniques developed for query evaluation in deductive databases, in particular on the *Magic Set transformation* [6] and the *SLG resolution*, a top-down resolution-based approach augmented with memoization [7,8]. We also study the impact of using other optimization techniques developed for Logic Programs, such as goal reordering.

The main contribution of the paper is establishing the connection between the automata-based decision procedures for WS1S (and, analogously, for WS2S) and query evaluation in Complex-value Datalog (Datalog^{cv}) which is equivalent to the complex-value calculus in expressive power [9]. Indeed, the complexity of query evaluation in Datalog^{cv} matches the

* Corresponding author.
E-mail addresses: gulay.unel@isikun.edu.tr (G. Unel), david@cs.uwaterloo.ca (D. Toman).

http://dx.doi.org/10.1016/j.jlamp.2016.02.005
2352-2208/

complexity of the WS1S decision procedure and thus it seems like an appropriate tool for this task. Our approach is based on representing automata using nested relations and on defining the necessary automata-theoretic operations using Datalog^{cv} programs. This reduces the satisfiability of a WSnS formula to posing a closed Datalog^{cv} goal over a Datalog^{cv} program representing implicitly the final automaton. The query checks if there is a path from the starting state to a final state on the automaton using the Datalog^{cv} representation of the automaton and the transitive closure of its transition relation. This observation combined with powerful program execution techniques developed for deductive databases, such as the Magic Set rewriting and SLG resolution, in many cases limits the explored state space to elements needed to show non-emptiness of the automaton and, in turn, satisfiability of the corresponding formula. The use of magic sets and/or SLG resolution automatically transforms the transitive closure query into a reachability query. Hence, the use of these techniques allows us to compute only the relevant parts of the automaton in a goal-driven way.

In addition to showing the connection between the automata-based decision procedures and query evaluation in Datalog^{cv}, we also conduct experiments with the XSB [10] system that demonstrate the benefits of the proposed method over more standard approaches. Note that we use XSB solely as an implementation vehicle for our experiments; any system that is capable of evaluating Datalog^{cv} queries can be used for the implementation of our method.

This method can be used for reasoning on formulas on large theories, such as those corresponding to UML diagrams and database schemata consisting of relatively simple constraints or certain Description Logic formalisms where the constraints or formulas can be mapped to weak monadic second-order logic formulas.

The work we present in this paper is an extension of [11] and [12] with correctness proofs of our method, and extended experiments and heuristics. The remainder of the paper is organized as follows. In Section 2 we formally introduce the weak monadic second-order logic and the connection to finite automata. We also define Datalog^{cv} programs, state their computational properties, and briefly discuss techniques used for program evaluation. Section 3 shows how Datalog^{cv} programs can be used to implicitly represent a finite automaton and to implement automata-theoretic operations on such a representation for the WS1S logic. The results are extended to WS2S in Section 4. Heuristics and optimizations for the proposed methods are presented in Section 5. Related work is discussed in Section 6. Finally, conclusions and future research directions are given in Section 7.

2. Background and definitions

In this section we provide definitions needed for the technical development in the rest of the paper.

2.1. Logics

First, we define the syntax and semantics of the monadic second-order logic of one and two successors.

Definition 1. Let $\mathsf{Var} = \{x, y, z, \ldots\}$ be a (countably infinite) set of variable names. Formulas of second-order logics are defined as follows.

- the expressions $s_i(x, y)$, $x \subseteq y$ for x, y second-order variables are atomic formulas (standing intuitively for the successor relations and the subset relation), and
- given formulas φ and ϕ and a variable x, the expressions $\varphi \wedge \phi$, $\neg\varphi$, and $\exists x : \varphi$ are also formulas.

As variables for individuals (first-order variables) can be simulated using second-order variables bound to singleton sets, a property expressible in WS1S, we allow writing $x \in y$ for $x \subseteq y$ whenever we know that x is a singleton. We also use the standard abbreviations $\varphi \vee \psi$ for $\neg(\neg\varphi \wedge \neg\psi)$, $\varphi \rightarrow \psi$ for $\neg\varphi \vee \psi$, and $\forall x : \varphi$ for $\neg\exists x : \neg\varphi$.

Weak monadic second order logics restrict variables to be interpreted as finite sets. Hence, quantification is allowed only over unary relations (i.e.: sets). The semantics of WS1S is defined w.r.t. the set of natural numbers (successors of 0); second-order variables are interpreted as finite sets of natural numbers in WS1S. The interpretation of the atomic formula $s(x, y)$ is fixed to relating singleton sets $\{n\}$ and $\{n + 1\}$, $n \in \mathbf{N}$.[1] Similarly, the semantics of WS2S is defined over an infinite binary tree $\mathbf{T} = (0 + 1)^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$; first-order variables are interpreted as nodes of the binary tree, and second order variables are interpreted as finite subsets of the nodes in WS2S. Since general trees can be converted to binary trees our results for WS2S can be extended to WSnS.

Definition 2. The definition of truth of a formula is defined over a transition system $\mathcal{T}$ (over $\mathbf{N}$ in WS1S and over $\mathbf{T}$ in WS2S). A valuation is defined as $\mathcal{D} : \mathsf{Var} \rightarrow 2^{\mathcal{Q}}$ where $\mathcal{Q} = \mathbf{N}$ in WS1S, $\mathcal{Q} = \mathbf{T}$ in WS2S, and $2^{\mathcal{Q}}$ is the collection of all finite subsets of Q. Given a valuation $\mathcal{D}$ and a transition system $\mathcal{T}$ we have:

- $\mathcal{T}, \mathcal{D} \models x \subseteq y$ if $\mathcal{D}(x) \subseteq \mathcal{D}(y)$
- $\mathcal{T}, \mathcal{D} \models s_i(x, y)$ if $\mathcal{D}(x)$ and $\mathcal{D}(y)$ are singletons $\{s_x\}$, $\{s_y\}$ and s_y is the ith successor of s_x

[1] The atomic formula $s(x, y)$ is often written as $y = s(x)$ in the literature, emphasizing its nature as a successor *function*.