The Complexity of Satisfiability Checking for Symbolic Finite Automata

Rodrigo Raya

School of Computer and Communication Sciences, EPFL, Switzerland

July 3, 2023

Abstract

We study the satisfiability problem of symbolic finite automata and decompose it into the satisfiability problem of the theory of the input characters and the monadic second-order theory of the indices of accepted words. We use our decomposition to obtain tight computational complexity bounds on the decision problem for this automata class and an extension that considers linear arithmetic constraints on the underlying effective Boolean algebra.

1 Introduction

Symbolic finite automata (SFAs) are an extension of finite automata that allow transitions to be labelled with monadic predicates over some universe rather than symbols from a finite alphabet. They were first mentioned in [29], but they attracted renewed interest starting in [28]. SFAs have been used in a variety of applications including the analysis of regular expressions [4,28], string encoders, sanitizers [9,14,16], functional programs [7], code generation, parallelization [21] and symbolic matching [22].

A series of theoretical investigations has been carried out on this automata model, including [2, 4, 25]. In particular, the authors of [27] observed that such an automata model had been studied previously by Bès in [3]. In his paper, Bès introduced a class of multi-tape synchronous finite automata whose transitions are labelled by first-order formulas. He then proved various properties of the languages accepted by such automata including closure under Boolean, rational, and the projection operations, logical characterizations in terms of MSO logic and the Eilenberg-Elgot-Shepherdson formalism as well as decidability properties. Remarkably [27], the paper showed that recognizability for such automata coincides with definability for certain generalized weak powers, first-studied by Feferman and Vaught in [11].

The techniques of Feferman and Vaught allow decomposing the decision problem for the first-order theory of a product of structures, $Th(\prod_i \mathcal{M}_i)$ into the first-order theory of the structures $\mathcal{M}_i$, $Th(\mathcal{M}_i)$, and the monadic second-order theory of the index set I, $Th^{mon}(\langle I, \ldots \rangle)$, where the structure $\langle I, \ldots \rangle$ may contain further relations such as a finiteness predicate, a cardinality operator, etc. If the theory of the components $Th(\mathcal{M}_i)$ is decidable, then the decision problem reduces to that of the theory $Th^{mon}(\langle I, \ldots \rangle)$. To analyse these structures, Feferman and Vaught extend results that go back to Skolem [24]. Technically, the decomposition is expressed in terms of so-called reduction sequences.

It is known [8] that many model-theoretic constructions incur in non-elementary blow-ups in the formula size. This includes the case of the size of the Feferman-Vaught reduction sequences in the case of disjoint unions. Perhaps for this reason, no computational complexity results have been obtained for the theory of symbolic automata and related models. Instead, the results in the literature [5, 6, 13] refer to the decidability of the satisfiability problem of the monadic predicates or provide asymptotic run-times rather than a refined computational complexity classification.

As a **main contribution**, we show how to reduce the satisfiability problem for finite symbolic automata to the satisfiability problem of the existential first-order theory of the theory of the elements and the existential monadic second-order theory of the indices. This decomposition allows us to derive tight complexity bounds for the decision problem of the automaton in the precise sense of Corollary 1. We then study an extension of the formalism of symbolic finite automata which also imposes linear arithmetic constraints on the cardinalities of the Venn regions of the underlying effective Boolean algebra. In particular, this extension allows expressing the number of occurrences of a particular kind of letter in a word. We show in Corollary 2 that the computational complexity of the corresponding satisfiability problem is the same as the one for the simpler model without cardinalities. Similar extensions for related models of automata are considered in the literature [12].

Organisation of the paper. Section 2 introduces symbolic finite automata. Section 3 gives the Feferman-Vaught decomposition of symbolic finite automata in terms of the theory of the elements and the theory of the indices. Section 4 describes the decision procedure with which, in Section 5, after presenting the quantifier-free theory of Boolean algebra with Presburger arithmetic, we obtain the tight complexity bounds announced. Section 6 describes the extension of symbolic finite automata that uses linear arithmetic constraints over the cardinalities of the automaton's underlying effective Boolean algebra and proves the corresponding upper bounds for the associated satisfiability problem. Section 7 concludes the paper.

2 Symbolic Finite Automata (SFA)

Symbolic automata are run over Boolean algebras of interpreted sets. The family of monadic predicates used for these interpretations needs to be closed

under Boolean operations and contain formulae denoting the empty set and the universe. Furthermore, in the original formulation, checking non-emptiness of these interpreted sets needs to be decidable. In Section 4, we will refine this assumption with a complexity-theoretic bound.

Definition 1 ([6]). An effective Boolean algebra $\mathcal{A}$ is a tuple

$$(\mathfrak{D}, \Psi, [\![\cdot]\!], \bot, \top, \vee, \wedge, \neg)$$

where $\mathfrak{D}$ is a set of domain elements, Ψ is a set of unary predicates over $\mathcal{D}$ that are closed under the Boolean connectives, with $\bot, \top \in \Psi$ and $[\![\cdot]\!] : \Psi \to 2^{\mathcal{D}}$ is a function such that 1. $[\![\bot]\!] = \emptyset$, 2. $[\![\top]\!] = \mathfrak{D}$, and 3. For all $\psi, \psi_1, \psi_2 \in \Psi$, we have that (a) $[\![\psi_1 \vee \psi_2]\!] = [\![\psi_1]\!] \cup [\![\psi_2]\!]$ (b) $[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!]$ (c) $[\![\neg\psi]\!] = \mathfrak{D} \backslash [\![\psi]\!]$. 4. Checking $[\![\psi]\!] \neq \emptyset$ is decidable. A predicate $\psi \in \Psi$ is atomic if it is not a Boolean combination of predicates in Ψ.

Our initial motivation was to generalise the complexity results obtained for array theories in [1, 20]. The notion of SMT algebra [6, Example 2.3] precisely corresponds to the language introduced in [20, Definition 5] without cardinality constraints. We take this as a first example of effective Boolean algebra.

Example 1. The SMT algebra for a type τ is the tuple $(\mathcal{D}, \Psi, [\![\cdot]\!], \bot, \top, \vee, \wedge, \neg)$ where $\mathcal{D}$ is the domain of τ, Ψ is the set of all quantifier-free formulas with one fixed free variable of type τ, $[\![\cdot]\!]$ maps each monadic predicate to the set of its satisfying assignments, $\bot$ denotes the empty set, $\top$ denotes the universe $\mathcal{D}$ and $\vee, \wedge, \neg$ denote the Boolean algebra operations of union, intersection, and complement respectively.

This example should be contrasted with other representations of the predicates that take into account implementation details. An example of the latter is the k-bit bitvector effective Boolean algebra described in [26].

Example 2. The powerset algebra $2^{bv(k)}$ is the tuple $(D, \Psi, [\![\cdot]\!], \bot, \top, \vee, \wedge, \neg)$ where $\mathcal{D}$ is the set $bv(k)$ of all non-negative integers less than 2^k or equivalently, all k-bit bit-vectors for some $k > 0$, Ψ is the set of BDDs of depth k, $[\![\cdot]\!]$ maps each BDD β to the set of all integers n such that the binary representation of n is a solution of β, $\bot$ denotes the BDD representing the empty set, $\top$ denotes the BDD representing the universal set and $\vee, \wedge, \neg$ denote the Boolean algebra operation of union, intersection, and complement as they are implemented for BDDs.

We now introduce the automata model we will investigate in the paper.

Definition 2 ([6]). A symbolic finite automaton (s-FA) is a tuple

$$M = (\mathcal{A}, Q, q_0, F, \Delta)$$

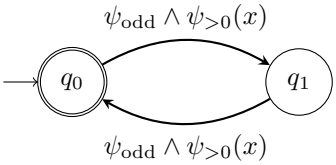
where 1. $\mathcal{A}$ is an effective Boolean algebra. 2. Q is a finite set of states. 3. $q_0 \in Q$ is the initial state. 4. $F \subseteq Q$ is the set of final states. 5. $\Delta \subseteq Q \times \Psi_{\mathcal{A}} \times Q$ is a finite set of transitions.

A symbolic transition $\rho = (q_1, \psi, q_2) \in \Delta$, also denoted $q_1 \xrightarrow{\psi} q_2$, has source state q_1, target state q_2, and guard ψ. For $d \in \mathfrak{D}$, the concrete transition $q_1 \xrightarrow{d} q_2$ denotes that $q_1 \xrightarrow{\psi} q_2$ and $d \in [\![\psi]\!]$ for some ψ.

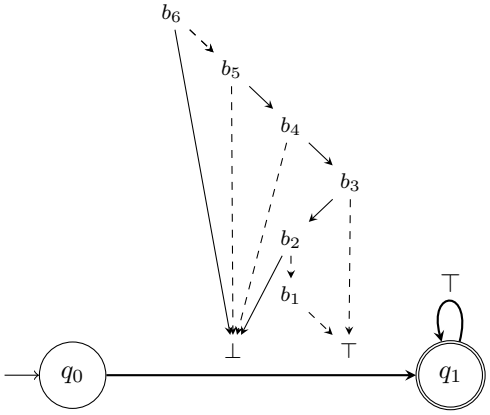
A string $w = d_1 d_2 \dots d_k$ is accepted at state q if and only if for $1 \leq i \leq k$, there exist transitions $q_{i-1} \xrightarrow{d_i} q_i$ such that $q_0 = q$ and $q_k \in F$. The set of strings accepted at q is denoted by $\mathcal{L}_q(M)$ and the language of M is $\mathcal{L}(M) = \mathcal{L}_{q_0}(M)$.

We now give examples of automata running over the algebras of Examples 1 and 2. We use the traditional graphical representation used in automata theory textbooks [15].

Example 3 ([6])**.** We consider the language of linear arithmetic over the integers. We set two formulae $\psi_{>0}(x) \equiv x > 0$ satisfied by all positive integers and $\psi_{\text{odd}}(x) \equiv x \mod 2 = 1$ satisfied by all odd integers. The following symbolic finite automaton accepts all strings of even length consisting only of positive odd numbers.



Example 4 ([26])**.** We consider the language of BDDs over bit-vectors of length six. The following symbolic finite automaton accepts all strings that start by a bit-vector representing either of the numbers $6, 14, 22, 38$ or 54 followed by an arbitrary number of bit-vectors.



3 Feferman-Vaught Decomposition for SFAs

Let $M = (\mathcal{A}, Q, q_0, F, \Delta)$ be a symbolic finite automaton and let $\psi_1, \dots, \psi_k, \dots$ be the atomic predicates in $\mathcal{A}$. The definition of symbolic finite automaton allows assuming that the set of these predicates is finite.

Lemma 1. There exists a symbolic finite automaton $M' = (\mathcal{A}', Q, q_0, F, \Delta)$ such that $\mathcal{L}(M) = \mathcal{L}(M')$ and the cardinality of $\Psi_{\mathcal{A}'}$ is finite.

Proof. The automaton has a finite number of transitions. We take $\Psi_{\mathcal{A}'}$ to be the Boolean closure of the predicates occurring in these transitions. It follows that $\Psi_{\mathcal{A}'}$ is a finite set. Otherwise, we define the components of $\mathcal{A}'$ as those in $\mathcal{A}$. Since the automaton is unchanged, $\mathcal{L}(M) = \mathcal{L}(M')$. □

Since $\Psi_{\mathcal{A}}$ can be assumed to be finite, it follows that the set of atomic predicates is finite too. In the remaining of the paper, we let $\phi_1, \ldots, \phi_k$ be the generators of the effective Boolean algebra used by the symbolic finite automaton M. Similarly, we let $\psi_1, \ldots, \psi_m$ denote the actual predicates used in the transitions of M. We will decompose the study of $\mathcal{L}(M)$ into the study of the properties of the elements in $\mathcal{D}$ and the ordering properties induced by the transition structure of the automaton. Both kinds of properties will refer to sets of indices to stay in sync with each other [30].

To specify the properties of the elements in $\mathcal{D}$, we use set interpretations of the form

$$S = \{\, n \in \mathbb{N} \mid \psi(d(n)) \,\} = [\![\psi]\!] \tag{1}$$

where $d(n)$ is the n-th element occurring in $d \in \mathcal{D}^*$. These sets can be pictured via a Venn diagram of interpreted sets, such as the one in Figure 1. Each formula in $\Psi_{\mathcal{A}}$ corresponds to a particular Venn region in this diagram and can be referred to using a Boolean algebra expression on the variables $S_1, \ldots, S_k$, thanks to the set interpretation (1).

A concrete transition $q_1 \xrightarrow{d} q_2$ requires a value $d \in \mathcal{D}$. This value will lie in some elementary Venn region of the diagram in Figure 1, i.e. in a set of the form $S_1^{\beta_1} \cap \ldots \cap S_k^{\beta_k}$ where $\beta = (\beta_1, \ldots, \beta_k) \in \{0,1\}^k$, $S^0 := S^c$ and $S^1 := S$. We will denote such Venn region with the bit-string β. To specify the transition structure of the automaton, what is relevant to us is the region of the Venn diagram, not the specific value that it takes there. It follows that a run of the automaton can be encoded as a sequence of bit-strings $\bar{t} = (t_1, \ldots, t_k) \in (\{0,1\}^{|\bar{t}|})^k$ and that these bit-strings only need to satisfy the propositional formulae corresponding to the predicates labelling the transitions of the automaton. Figure 2 represents one such run over an uninterpreted Venn diagram.

Example 5. If in Example 3 we take as atomic formulae the predicates ψ_{odd} and $\psi_{>0}$ then the formula $\psi_{\text{odd}} \wedge \psi_{>0}(x)$, which labels the automaton transitions, corresponds to the propositional formula $S_1 \wedge S_2$.

We denote by $L_1, \ldots, L_m$ such propositional formulae and by $M(L_1, \ldots, L_m)$ the set of bit-string runs accepted by M, which we call *tables* [17].

Lemma 2.

$$\mathcal{L}(M) = \Big\{ d \in \mathcal{D}^* \Big| \exists \bar{t} \in M(L_1, \ldots, L_m). \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N}; \phi_i(d(n)) \,\} = \{\, n \in \mathbb{N}; t_i(n) \,\} \Big\}$$

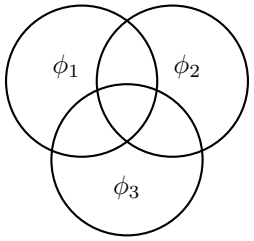


Figure 1: A Venn diagram representing a finitely generated effective Boolean algebra with atomic predicates ψ_1, ψ_2 and ψ_3.

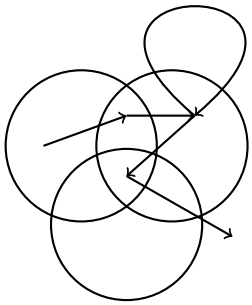


Figure 2: A table accepted by a symbolic automaton represented over an uninterpreted Venn diagram.

Proof. The proof uses the definition of $\mathcal{L}(M)$ and $M(L_1, \ldots, L_m)$. In one direction, one defines $\bar{t}$ from the membership of the values $d(i)$ in elementary Venn regions β_i. In the other direction, the definition of $M(L_1, \ldots, L_m)$ ensures that there is an accepting run corresponding to these values and any witness of the formula in the associated elementary Venn regions can be taken to conform the word d. □

In the next sections, we make use of this decomposition to devise a decision procedure for symbolic finite automata, which, will refine the existing computational complexity results for the corresponding satisfiability problem.

4 Decision Procedure for Satisfiability of SFAs

Definition 3. The satisfiability problem for a symbolic finite automaton M is the problem of determining whether $\mathcal{L}(M) \neq \emptyset$.

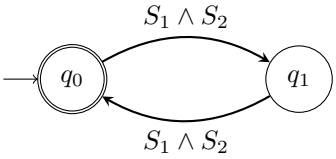
By Lemma 2, checking non-emptiness of the language of a symbolic finite

automaton reduces to checking whether the following formula is true:

$$\begin{aligned} &\exists S_1, \ldots, S_k. \exists d. \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N}; \phi_i(d(n)) \,\} \wedge \\ &\exists \overline{t} \in M(L_1, \ldots, L_m). \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N}; t_i(n) \,\} \end{aligned} \tag{2}$$

To establish the complexity of deciding formulae of the form (2), we will have to analyse further the set $M(L_1, \ldots, L_m)$. Each table $\overline{t}$ in $M(L_1, \ldots, L_m)$ corresponds to a *symbolic table* $\overline{s}$ whose entries are the propositional formulae that the bit-strings of $\overline{t}$ satisfy. More generally, these symbolic tables are generated by the symbolic automaton obtained by replacing the predicates of the symbolic automaton by propositional formulae. The set of symbolic tables accepted by the automaton M is a regular set and will be denoted by $M_S(L_1, \ldots, L_m)$.

Example 6. The automaton in Example 3 corresponds, according to Example 5, to the symbolic automaton shown.



The symbolic tables generated by this automaton are of the form $((S_1 \wedge S_2)(S_1 \wedge S_2))^*$. The corresponding tables would be of the form $((1,1)(1,1))^*$.

Consider first the case where the propositional formulae $L_1, \ldots, L_m$ for the automaton M denote disjoint Venn regions. In this case, all we need to do to check the satisfiability of formula (2) is whether there exists a symbolic table $\overline{s}$ such that whenever the number of times a certain propositional letter occurs is non-zero, then the corresponding Venn region interpreted according to (1) has a satisfiable defining formula. From this, it follows that our decision procedure will need to compute the so-called Parikh image of the regular language $M_S(L_1, \ldots, L_m)$.

Definition 4 (Parikh Image)**.**
The Parikh image of $M_S(L_1, \ldots, L_m)$ is the set

$$\mathsf{Parikh}(M_S(L_1, \ldots, L_m)) = \{(|\overline{s}|_{L_1}, \ldots, |\overline{s}|_{L_m}) | \overline{s} \in M_S(L_1, \ldots, L_m)\}$$

where $|\overline{s}|_{L_i}$ denotes the number of occurrences of the propositional formula L_i in the symbolic table $\overline{s}$.

We will use a description of the Parikh image in terms of linear-size existential Presburger arithmetic formulae first obtained by Seidl, Schwentick, Muscholl and Habermehl.

Lemma 3 ([23])**.** The set $\mathsf{Parikh}(M_S(L_1,\ldots,L_n))$ is definable by an existential Presburger formula ρ of size $O(|M|)$ where $|M|$ is the number of symbols used to describe the automaton M.

When propositional letters denote overlapping Venn regions, a partitioning argument is required. This is formalised in Theorem 1. First, we fix some notation. We set $p_\beta := \bigcap_{i=1}^{k} S_i^{\beta_i}$ where $\beta \in \{0,1\}^k$, $p_L := \bigcup_{\beta \models L} p_\beta$ where L is a propositional formula and $\varphi^\beta(d) := \bigwedge_{i=1}^{k} \varphi_i^{\beta(i)}(d)$. We write $S_1 \dot\cup S_2$ to denote the set $S_1 \cup S_2$ where it is known that $S_1 \cap S_2 = \emptyset$. Finally, we write $[n] := \{\, 1,\ldots,n \,\}$.

Theorem 1. Formula (2) is equivalent to the formula

$$
\begin{array}{l}
\exists s \in [m].\sigma : [s] \hookrightarrow [m].\exists \beta_1,\ldots,\beta_s \in \{0,1\}^k.\ \displaystyle\bigwedge_{j=1}^{s} \exists d.\phi^{\beta_j}(d) \wedge \\
\exists k_1,\ldots,k_m.\exists S_1,\ldots,S_k,P_1,\ldots,P_s. \\
\qquad \rho(k_1,\ldots,k_m) \wedge \displaystyle\bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{m} p_{L_i} = \dot\cup_{i=1}^{s} P_i \wedge \\
\qquad \displaystyle\bigwedge_{i=1}^{s} |P_i| = k_{\sigma(i)} \wedge \bigwedge_{i=1}^{s} p_{\beta_i} \cap P_i \neq \emptyset
\end{array}
\tag{3}
$$

where σ is an injection from $\{\, 1,\ldots,s \,\}$ to $\{\, 1,\ldots,m \,\}$ and ρ is the arithmetic expression in Lemma 3.

Formula (3) has two parts. The first part corresponds to the subterm $\bigwedge_{j=1}^{s} \exists d.\varphi^{\beta_j}(d)$ and falls within the theory of the elements in $\mathcal{D}$, $Th_{\exists^*}(\mathcal{D})$. The second part corresponds to the remaining subterm and falls within the quantifier-free first-order theory of Boolean Algebra with Presburger arithmetic (QFBAPA) [19], which can be viewed as the monadic second order theory $Th^{mon}_{\exists^*}(\langle \mathbb{N}, \subseteq, \sim \rangle)$ where $\sim$ is the equicardinality relation between two sets.

Formula (2) is distilled from a non-deterministic decision procedure for the formulae of the shape (2). The existentially quantified variables $s, \sigma, \beta_1,\ldots,\beta_s$ are guessed by the procedure. These guessed values are then used by specialised procedures for $Th_{\exists^*}(\mathcal{D})$ and $Th^{mon}_{\exists^*}(\langle \mathbb{N}, \subseteq, \sim \rangle)$. For the convenience of the reader, we describe here what these values mean. The value of s represents the number of Venn regions associated to the formulae $L_1,\ldots,L_m$ that will be non-empty. σ indexes these non-empty regions. $\beta_1,\ldots,\beta_s$ are elementary Venn regions contained in the non-empty ones.

The reason to introduce the partition variables $P_1,\ldots,P_s$ is that the Venn regions may overlap.

Example 7. Consider the situation where $S_1 \wedge S_2$ and $S_2 \wedge S_3$ are two propositional formulae labelling the transitions of the symbolic automaton. These formulae correspond to the Venn regions $S_1 \cap S_2$ and $S_2 \cap S_3$, which share the

region $S_1 \cap S_2 \cap S_3$. Given a model of S_1, S_2 and S_3, how do we guarantee that the indices in the region $S_1 \cap S_2 \cap S_3$ are consistent with a run of the automaton? For instance, the automaton may require one element in $S_1 \cap S_2$ and another in $S_2 \cap S_3$. Placing a single index in $S_1 \cap S_2 \cap S_3$ would satisfy the overall cardinality constraints, but not the fact that overall we need to have two elements. Trying to specify this in the general case would reduce to specifying an exponential number of cardinalities.

We proceed next to the proof of the theorem.

Proof of Theorem 1. $\Rightarrow$) If formula (2) is satisfiable, then there are sets $S_1, \ldots, S_k$, a word d and a table $\overline{t}$ satisfying

$$\bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N}; \phi_i(d) \,\} \wedge \overline{t} \in M(L_1, \ldots, L_s) \wedge \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N}; t_i(n) \,\}$$

Let $\overline{s} \in M(L_1, \ldots, L_s)$ be the symbolic table corresponding to $\overline{t}$. We define $k_i := |\overline{s}|_{L_i}, s = |\{\, i \mid k_i \neq 0 \,\}|$, σ mapping the indices in $[s]$ to the indices of the terms for which k_i is non-zero and $P_i = \{\, n \in \mathbb{N}; \overline{s}(n) = L_{\sigma(i)} \,\}$. It will be convenient to work out the following equalities:

$$\begin{aligned} p_{L_i} &= \bigcup_{\beta \models L_i} \bigcap_{j=1}^{k} S_j^{\beta_j} = \bigcup_{\beta \models L_i} \left\{ n \in \mathbb{N} \;\middle|\; \bigwedge_{j=1}^{k} t_j^{\beta_j}(n) \right\} = \left\{\, n \in \mathbb{N} \;\middle|\; \overline{t}(n) \models L_i \,\right\} \\ p_{L_i} &= \bigcup_{\beta \models L_i} \bigcap_{j=1}^{k} S_j^{\beta_j} = \bigcup_{\beta \models L_i} \left\{ n \in \mathbb{N} \;\middle|\; \bigwedge_{j=1}^{k} \phi_j^{\beta_j}(d) \right\} = \left\{\, d \in \mathcal{D} \;\middle|\; L_i(\overline{\phi}(d)) \,\right\} \end{aligned} \tag{4}$$

where $L_i(\overline{\phi}(d(n)))$ is the propositional formula obtained by substituting set variables by the formulae $\phi_i(d(n))$. We now deduce formula (3):

- $\rho(k_1, \ldots, k_m)$: from $\overline{s} \in P(L_1, \ldots, L_m)$, we have that

  $$(k_1, \ldots, k_m) \in \mathsf{Parikh}(M_S(L_1, \ldots, L_m))$$

  and therefore $\rho(k_1, \ldots, k_m)$.

- $P_i \subseteq p_{L_{\sigma(i)}}$: since $\overline{s}$ corresponds to $\overline{t}$, for all $n \in \mathbb{N}$ we have $\overline{t}(n) \models \overline{s}(n)$ and the inclusion follows from the definition of P_i and equation 4.

- $|P_i| = k_{\sigma(i)}$: since $|P_i| = \left| \left\{\, n \in \mathbb{N} \;\middle|\; \overline{s}(n) = L_{\sigma(i)} \,\right\} \right| = |\overline{s}|_{L_{\sigma(i)}} = k_{\sigma(i)}$.

- Each pair of sets P_i, P_j with $i < j$ is disjoint:

  $$\begin{aligned} P_i \cap P_j &= \left\{\, n \in \mathbb{N} \;\middle|\; \overline{s}(n) = L_{\sigma(i)} \,\right\} \cap \left\{\, n \in \mathbb{N} \;\middle|\; \overline{s}(n) = L_{\sigma(j)} \,\right\} = \\ &= \left\{\, n \in \mathbb{N} \;\middle|\; \overline{s}(n) = L_{\sigma(i)} = L_{\sigma(j)} \,\right\} = \emptyset \end{aligned}$$

  using that the letters L are chosen to be distinct and that σ is an injection (so $\sigma(i) \neq \sigma(j)$).

- $p_{L_1} \cup \ldots \cup p_{L_m} = P_1 \dot\cup \ldots \dot\cup P_s$: since by definition $P_i = \left\{ n \in \mathbb{N} \;\middle|\; \overline{s}(n) = L_{\sigma(i)} \right\}$, $p_{L_i} = \left\{ n \in \mathbb{N} \;\middle|\; \overline{t}(n) \models L_i \right\}$ and by definition of σ it follows that the only letters that can appear in $\overline{s}$ are $L_{\sigma(1)}, \ldots, L_{\sigma(s)}$. Thus, we have $p_{L_1} \cup \ldots \cup p_{L_m} = [1, |\overline{t}|] = [1, |\overline{s}|] = P_1 \dot\cup \ldots \dot\cup P_s$.

- There exists $\beta_1, \ldots, \beta_s \in \{0,1\}^k$, such that $\bigwedge_{i=1}^{s} P_{\beta_i} \cap P_i \neq \emptyset$: note that $P_i \neq \emptyset$ by definition of σ. Thus, there must exist some β_i such that $p_{\beta_i} \cap P_i \neq \emptyset$. We pick any such β_i.

- $\bigwedge_{j=1}^{s} \exists d. \varphi^{\beta_j}(d)$: follows from $p_{\beta_j} \cap P_j \neq \emptyset$ and formula (4).

$\Leftarrow$) Conversely, if formula (3) is satisfiable, then there is an integer $s \in [n]$, an injection $\sigma : [s] \hookrightarrow [n]$, bit-strings $\beta_1, \ldots, \beta_s \in \{0,1\}^k$, integers $k_1, \ldots, k_m$ and sets $S_1, \ldots, S_k, P_1, \ldots, P_s$ satisfying

$$\begin{array}{l} \bigwedge_{j=1}^{s} \exists d. \varphi^{\beta_j}(d) \wedge \rho(k_1, \ldots, k_m) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{m} p_{L_i} = \dot\cup_{i=1}^{s} P_i \wedge \\ \bigwedge_{i=1}^{s} |P_i| = k_{\sigma(i)} \wedge \bigwedge_{i=1}^{s} p_{\beta_i} \cap P_i \neq \emptyset \end{array} \tag{5}$$

From $\psi(k_1, \ldots, k_n)$ follows that there is a symbolic table $\overline{s} \in M_S(L_1, \ldots, L_m)$ such that $|\overline{s}|_{L_i} = k_i$ for each $L_i \in \{ L_1, \ldots, L_m \}$. From formula (4) and

$$p_{L_1} \cup \ldots \cup p_{L_m} = P_1 \dot\cup \ldots \dot\cup P_s \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \bigwedge_{i=1}^{s} |P_i| = k_{\sigma(i)}$$

follows that we can replace the formulae L_i occurring in the symbolic table $\overline{s}$ by the bit-strings representing the elementary Venn regions to which the indices of the sets P_i belong. Moreover, thanks to the condition $\bigwedge_{i=1}^{s} p_{\beta_i} \cap P_i \neq \emptyset$ follows that we can replace the letters L_i by the bit-strings β_i, defining $\overline{t}$ as $\overline{t}(n) = \begin{cases} \beta_i & \text{if } n \in P_i \end{cases}$. In this way, we obtain a table $\overline{t} \in M(L_1, \ldots, L_s)$. We then define the corresponding word over $\mathcal{D}$, thanks to the property $\bigwedge_{i=1}^{s} \exists d. \phi^{\beta_i}(d)$. Naming the witnesses of these formulae as d_i, we define $d(n) = \begin{cases} d_i & \text{if } n \in P_i \end{cases}$. To conclude, note that:

$$\{ n \in \mathbb{N} \mid t_j(n) \} = \cup_{\{ 1 \leq i \leq k \mid \beta_i(j) = 1 \}} P_i = \{ n \in \mathbb{N} \mid \phi_j(d(n)) \}$$

Thus, we have that formula (2) is satisfied by the set variables

$$S_j := \{ n \in \mathbb{N} \mid t_j(n) \} = \{ n \in \mathbb{N} \mid \phi_j(d(n)) \}$$

□

5 Quantifier-free Boolean Algebra with Presburger Arithmetic

The arguments following the statement of Theorem 1 sketch a non-deterministic procedure for the satisfiability problem of symbolic finite automata, based on

the existence of decision procedures for $Th_{\exists^*}(\mathcal{D})$ and $Th^{mon}_{\exists^*}(\langle \mathbb{N}, \subseteq, \sim \rangle)$. In this section, we recall the non-deterministic polynomial time decision procedure for $Th^{mon}_{\exists^*}(\langle \mathbb{N}, \subseteq, \sim \rangle)$. As a consequence, we obtain Corollary 1 which situates the decision problem of symbolic finite automata in the classical complexity hierarchy. This section should also prepare the reader for the extension of these results, where the automaton can require linear arithmetic constraints on the cardinalities of the effective Boolean algebra. This extension is carried out in Section 6.

Instead of working with $Th^{mon}_{\exists^*}(\langle \mathbb{N}, \subseteq, \sim \rangle)$ directly, we use the logic QFBAPA [19] which has the same expressive power [18, Section 2]. The syntax of QFBAPA is given in Figure 3. The meaning of the syntax is as follows. F presents the Boolean structure of the formula, A stands for the top-level constraints, B gives the Boolean restrictions and T the Presburger arithmetic terms. The operator dvd stands for the divisibility relation and $\mathcal{U}$ represents the universal set. The remaining interpretations are standard.

$$
\begin{aligned}
F &::= A \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \\
A &::= B_1 = B_2 \mid B_1 \subseteq B_2 \mid T_1 = T_2 \mid T_1 \leq T_2 \mid K \text{ dvd } T \\
B &::= x \mid \emptyset \mid \mathcal{U} \mid B_1 \cup B_2 \mid B_1 \cap B_2 \mid B^c \\
T &::= k \mid K \mid T_1 + T_2 \mid K \cdot T \mid |B| \\
K &::= \ldots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \ldots
\end{aligned}
$$

Figure 3: QFBAPA's syntax

The satisfiability problem of this logic is reducible to propositional satisfiability in polynomial time. Our proofs will rely on the method of [19], which we sketch briefly here. The basic argument to establish a NP complexity bound on the satisfiability problem of QFBAPA is based on a theorem by Eisenbrand and Shmonin [10], which in our context says that any element of an integer cone can be expressed in terms of a polynomial number of generators. Figure 4 gives a verifier for this basic version of the algorithm. The algorithm uses an auxiliary verifier V_{PA} for the quantifier-free fragment of Presburger arithmetic. The key step is showing equisatisfiability between 2.(b) and 2.(c). If $x_1, \ldots, x_k$ are the variables occurring in $b_0, \ldots, b_p$ then we write $p_\beta = \bigcap_{i=1}^{k} x_i^{e_i}$ for $\beta = (e_1, \ldots, e_k) \in \{0,1\}^k$ where we define $x^1 := x$ and $x^0 := \mathcal{U} \setminus x$ as before. If we define $[\![b_i]\!]_{\beta_j}$ as the evaluation of b_i as a propositional formula with the assignment given in β and introduce variables $l_\beta = |p_\beta|$, then $|b_i| = \sum_{j=0}^{2^e-1} [\![b_i]\!]_{\beta_j} l_{\beta_j}$, so the restriction $\bigwedge_{i=0}^{p} |b_i| = k_i$ in 2.(b) becomes $\bigwedge_{i=0}^{p} \sum_{j=0}^{2^e-1} [\![b_i]\!]_{\beta_j} l_{\beta_j} = k_i$ which can be seen as a linear combination in the

set of vectors $\{([\![b_0]\!]_{\beta_j}, \ldots, [\![b_p]\!]_{\beta_j}).j \in \{0, \ldots, 2^e - 1\}\}$. Eisenbrand-Shmonin's result allows then to derive 2.(c) for N polynomial in $|x|$. In the other direction, it is sufficient to set $l_{\beta_j} = 0$ for $j \in \{0, \ldots, 2^e - 1\} \setminus \{i_1, \ldots, i_N\}$. Thus, we have:

On input $\langle x, w \rangle$:

1. Interpret w as:
   (a) a list of indices $i_1, \ldots, i_N \in \{0, \ldots, 2^e - 1\}$ where e is the number of set variables in x.
   (b) a certificate C for V_{PA} on input x' defined below.
2. Transform x into x' by:
   (a) rewriting boolean expressions according to the rules:
   $$b_1 = b_2 \mapsto b_1 \subseteq b_2 \wedge b_2 \subseteq b_1$$
   $$b_1 \subseteq b_2 \mapsto |b_1 \cap b_2^c| = 0$$
   (b) introducing variables k_i for cardinality expressions:
   $$G \wedge \bigwedge_{i=0}^{p} |b_i| = k_i$$
   where G is the resulting quantifier-free Presburger arithmetic formula.
   (c) rewriting into:
   $$G \wedge \bigwedge_{j=i_1,\ldots,i_N} l_{\beta_j} \geq 0 \wedge \bigwedge_{i=0}^{p} \sum_{j=i_1,\ldots,i_N} [\![b_i]\!]_{\beta_j} \cdot l_{\beta_j} = k_i$$
3. Run V_{PA} on $\langle x', C \rangle$.
4. Accept iff V_{PA} accepts.

Figure 4: Verifier for QFBAPA

Theorem 2 ([19])**.** The satisfiability problem of QFBAPA is in NP.

From Theorems 1 and 2, we obtain the following improvement of [6, Theorem 2.8]:

Corollary 1. Let $Th_{\exists^*}(\mathcal{D})$ be the existential first-order theory of the formulae used in the transitions of the symbolic finite automaton M.

- If $Th_{\exists^*}(\mathcal{D}) \in \mathrm{P}$ then $\mathcal{L}(M) \neq \emptyset \in \mathrm{NP}$.
- If $Th_{\exists^*}(\mathcal{D}) \in \mathrm{C}$ for some $\mathrm{C} \supseteq \mathrm{NP}$ then $\mathcal{L}(M) \neq \emptyset \in \mathrm{C}$.

6 Decision Procedure for Satisfiability of SFAs with Cardinalities

We now consider the following generalisation of the language of a finite symbolic automaton from Lemma 2.

Definition 5. A symbolic finite automaton with cardinalities accepts a language of the form:

$$\mathcal{L}(M) = \left\{ d \in \mathcal{D}^* \;\middle|\; \begin{array}{l} F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N} \mid \phi_i(d(n)) \,\} \wedge \\ \exists \bar{t} \in P(L_1, \ldots, L_m).\, \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N} \mid t_i(n) \,\} \end{array} \right\}$$

where F is a formula from QFBAPA.

Thus, checking non-emptiness of the language of a symbolic finite automaton with cardinalities reduces to checking whether the following formula is true:

$$\begin{array}{l} \exists S_1, \ldots, S_k.F(S_1, \ldots, S_k) \wedge \\ \qquad \exists d. \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N} \mid \phi_i(d(n)) \,\} \wedge \\ \qquad \exists \bar{t} \in M(L_1, \ldots, L_m) \wedge \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N} \mid t_i(n) \,\} \end{array} \tag{6}$$

To show that Theorem 1 and Corollary 1 stay true with linear arithmetic constraints on the cardinalities, we need to repeat part of the argument in Theorem 1 since if F denotes the newly introduced QFBAPA formula and G, H are the formulae shown equivalent in Theorem 1, then from:

$$\exists S_1, \ldots, S_k.F(S_1, \ldots, S_k) \wedge G(S_1, \ldots, S_k)$$

and

$$\Big[\exists S_1, \ldots, S_k.G(S_1, \ldots, S_k)\Big] \iff \Big[\exists S_1, \ldots, S_k.H(S_1, \ldots, S_k)\Big]$$

it does not follow that $\exists S_1, \ldots, S_k.F(S_1, \ldots, S_k) \wedge H(S_1, \ldots, S_k)$. Instead, the algorithm derives the cardinality constraints from each theory and then uses the sparsity of solutions *over the satisfiable regions*. In the proof, we set $[\![\beta_j \models b_i]\!]$ to be one, if the bit-string β_j satisfies the Boolean expression b_i as a propositional assignment and zero otherwise. We also write $l_\beta = |p_\beta|$ for $\beta \in \{0,1\}^k$.

Theorem 3. Formula (6) is equivalent to:

$$\begin{aligned}
&\exists N \leq p(|F|), \exists s \in [m].\sigma : [s] \hookrightarrow [m].\exists \beta_1, \ldots, \beta_N \in \{0,1\}^k. \bigwedge_{j=1}^{N} \exists d.\phi^{\beta_j}(d) \wedge \\
&\exists k_1, \ldots, k_m.\exists S_1, \ldots, S_k, P_1, \ldots, P_s. \\
&\qquad \rho(k_1, \ldots, k_m) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i \wedge \\
&\qquad \bigwedge_{i=1}^{s} |P_i| = k_{\sigma(i)} \wedge \cup_{i=1}^{N} p_{\beta_i} = \dot{\cup}_{i=1}^{s} P_i
\end{aligned} \tag{7}$$

where p is a polynomial and $|F|$ is the number of symbols used to write F.

Proof. $\Rightarrow$) If formula (6) is true, then there are sets $S_1, \ldots, S_k$, a finite word d and a table $\bar{t}$ such that:

$$\begin{aligned}
&F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N} \mid \phi_i(d(n)) \,\} \wedge \\
&\bar{t} \in M(L_1, \ldots, L_m) \wedge \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N} \mid t_i(n) \,\}
\end{aligned} \tag{8}$$

Thus, there exists a symbolic table $\bar{s} \in M_S(L_1, \ldots, L_s)$ corresponding to $\bar{t}$. We define $k_i := |\bar{s}|_{L_i}, s = |\{\, i \mid k_i \neq 0 \,\}|$, σ maps the indices in $[s]$ to the indices of the terms for which k_i is non-zero and $P_i = \left\{\, n \in \mathbb{N} \;\middle|\; \bar{s}(n) = L_{\sigma(i)} \,\right\}$. As in Theorem 1, we have the equalities $p_{L_i} = \left\{\, n \in \mathbb{N} \;\middle|\; \bar{t}(n) \models L_i \,\right\}$, $p_{L_i} = \left\{\, n \in \mathbb{N} \;\middle|\; L_i(\overline{\phi}(d)) \,\right\}$ and we can show that the following formula holds:

$$\begin{aligned}
&\rho(k_1, \ldots, k_m) \wedge \bigwedge_{i=1}^{m} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i \wedge \\
&\bigwedge_{i=1}^{s} |P_i| = k_{\sigma(i)} \wedge F(S_1, \ldots, S_k)
\end{aligned} \tag{9}$$

We need to find a sparse model of (9). To achieve this, we follow the methodology in Theorem 2. This leads to a system of equations of the form:

$$\exists c_1, \ldots, c_p.G \wedge \sum_{j=0}^{2^e-1} \begin{pmatrix} [\![\beta_j \models b_0]\!] \\ \cdots \\ [\![\beta_j \models b_p]\!] \end{pmatrix} \cdot l_{\beta_j} = \begin{pmatrix} c_1 \\ \ldots \\ c_p \end{pmatrix}$$

We remove those elementary Venn regions where $l_\beta = 0$. This includes regions whose associated formula in the interpreted Boolean algebra is unsatisfiable, and regions corresponding to table entries not occurring in $\bar{t}$. This transformation gives a reduced set of indices $\mathcal{R}$ participating in the sum.

Using Eisenbrand-Shmonin's theorem, we have a polynomial (in the size of the original formula) family of Venn regions $\beta_1, \ldots, \beta_N$ and corresponding cardinalities $l'_{\beta_1}, \ldots, l'_{\beta_N}$, which we can assume to be non-zero, such that

$$\exists c_1, \ldots, c_p.G \wedge \sum_{\beta \in \{\beta_1, \ldots, \beta_N\} \subseteq \mathcal{R}} \begin{pmatrix} [\![\beta_j \models b_0]\!] \\ \cdots \\ [\![\beta_j \models b_p]\!] \end{pmatrix} \cdot l'_{\beta_j} = \begin{pmatrix} c_1 \\ \ldots \\ c_p \end{pmatrix} \tag{10}$$

The satisfiability of formula (10) implies the existence of sets of indices p'_β satisfying the conditions derived in formula (9). However, it does not imply which explicit indices belong to these sets and which are the contents corresponding to each index. From the condition

$$\psi(k_1, \ldots, k_n) \wedge \bigwedge_{i=1}^{n} P'_i \subseteq p'_{L_{\sigma(i)}} \wedge \cup_{i=1}^{n} p'_{L_i} = \dot{\cup}_{i=1}^{s} P'_i \wedge \bigwedge_{i=1}^{s} |P'_i| = k_{\sigma(i)}$$

follows that there is a symbolic table $\overline{s}'$ satisfying $M_S(L_1, \ldots, L_n)$ with $k_{\sigma(i)}$ letters $L_{\sigma(i)}$ and that these letters are made concrete by entries in P'_i for each $i \in \{1, \ldots, s\}$. We take the Venn regions $\beta \in \{\beta_1, \ldots, \beta_N\}$ such that $P'_i \supseteq p_\beta$ and label the corresponding entries in $\overline{s}'$ with β. In this way, we obtain a corresponding concrete table $\overline{t}'$. This makes the indices in each Venn region concrete. To make the contents of the indices concrete, note that for each $\beta \in \mathcal{R}$, since $l_\beta \neq 0$, the formula $\exists d.\phi^\beta(d)$ is true. In particular, this applies to each $\beta \in \{\beta_1, \ldots, \beta_N\}$. Thus, we obtain witnesses $d_1, \ldots, d_N$. We form a word by replacing each letter β in $\overline{t}'$ by the corresponding value d_β.

$\Leftarrow$) If formula (7) is true, then there is $N \leq p(|F|)$ where p is a polynomial, $s \in [m]$, $\beta_1, \ldots, \beta_N \in \{0,1\}^k, k_1, \ldots, k_m \in \mathbb{N}$ and sets $S_1, \ldots, S_k, P_1, \ldots, P_s$ such that

$$\bigwedge_{j=1}^{N} \exists d.\phi^{\beta_j}(d) \wedge \rho(k_1, \ldots, k_m) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i \wedge$$
$$\bigwedge_{i=1}^{s} |P_i| = k_{\sigma(i)} \wedge \cup_{i=1}^{N} p_{\beta_i} = \dot{\cup}_{i=1}^{s} P_i$$

From $\rho(k_1, \ldots, k_n)$ follows that there is a symbolic table $\overline{s} \in R(L_1, \ldots, L_m)$ such that $|\overline{s}|_{L_i} = k_i$ for each $L_i \in \{L_1, \ldots, L_m\}$. From formula 8 and

$$p_{L_1} \cup \ldots \cup p_{L_m} = P_1 \dot{\cup} \ldots \dot{\cup} P_s \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \bigwedge_{i=1}^{s} |P_i| = k_{\sigma(i)}$$

follows that we can replace the formulae L_i occurring in the symbolic table $\overline{s}$ by the bit-strings representing the elementary Venn regions to which the indices of the sets P_i belong. Moreover, thanks to the condition $\cup_{i=1}^{N} p_{\beta_i} = \dot{\cup}_{i=1}^{s} P_i$, it follows that we can replace the letters L_i by the bit-strings β_i. In this way, we

obtain a table $\bar{t} \in R(L_1, \ldots, L_m)$. We then define the corresponding word over $\mathcal{D}$, thanks to the property $\bigwedge_{i=1}^{N} \exists d.\phi^{\beta_i}(d)$. To conclude, note that:

$$\{\ n \in \mathbb{N} \mid t_j(n)\ \} = \cup_{\{\ i \mid \beta_i(j)=1\ \}} P_i = \{\ n \in \mathbb{N} \mid \phi_j(d(n))\ \}$$

Thus, we have that formula 2 is satisfied by the set variables

$$S_j := \{\ n \in \mathbb{N} \mid t_j(n)\ \} = \{\ n \in \mathbb{N} \mid \phi_j(d(n))\ \}$$

□

We can thus formulate the analogous to Corollary 1 in the case of finite symbolic automata with cardinalities.

Corollary 2. Let $Th_{\exists^*}(\mathcal{D})$ be the theory of the formulae used in the transitions of a symbolic finite automaton with cardinality constraints.

- If $Th_{\exists^*}(\mathcal{D}) \in$ P then $\mathcal{L}(M) \neq \emptyset \in$ NP.
- If $Th_{\exists^*}(\mathcal{D}) \in$ C for some C $\supseteq$ NP then $\mathcal{L}(M) \neq \emptyset \in$ C.

7 Conclusion

We have revisited the model of symbolic finite automata as it was reintroduced in [28]. We have obtained tight complexity bounds on their satisfiability problem. Our methodology follows the Feferman-Vaught decomposition technique in that it reduces the satisfiability problem of the automaton to the satisfiability problem of the existential first-order theory of the characters accepted by the automaton and the satisfiability problem of the existential monadic second-order theory of the indices.

To combine these two distinct theories we use the ideas from the combination method through sets and cardinalities of Wies, Piskac and Kunčak [30] and the computation of an equivalent linear-sized existentially quantified Presburger arithmetic formula from the Parikh image of a regular language by Seidl, Schwentick, Muscholl and Habermehl [23]. A crucial step in the proofs is a partitioning argument for the underlying Venn regions. We profit from the analysis in [19] to extend our arguments to the satisfiability problem of finite symbolic automata that consider linear arithmetic restrictions over the cardinalities of the Boolean algebra associated with the symbolic finite automaton.

In future work, we plan to extend our methods to other variants of symbolic automata to which we believe similar techniques may be applicable. Another interesting research direction would be to consider extensions of the language that allow free variables in set interpretations of the form (1), which seems to have applications to various satisfiability problems.

References

[1] Alberti, F., Ghilardi, S., Pagani, E.: Cardinality constraints for arrays (decidability results and applications). Formal Methods in System Design **51**(3), 545–574 (Dec 2017). https://doi.org/10.1007/s10703-017-0279-6, https://doi.org/10.1007/s10703-017-0279-6

[2] Argyros, G., D'Antoni, L.: The Learnability of Symbolic Automata. In: Computer Aided Verification, vol. 10981, pp. 427–445. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_23, http://link.springer.com/10.1007/978-3-319-96145-3_23, series Title: Lecture Notes in Computer Science

[3] Bès, A.: An Application of the Feferman-Vaught Theorem to Automata and Logics for Words over an Infinite Alphabet. Logical Methods in Computer Science **4**(1), 8 (Mar 2008). https://doi.org/10.2168/LMCS-4(1:8)2008, https://lmcs.episciences.org/1202

[4] D'Antoni, L., Veanes, M.: Minimization of symbolic automata. In: Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 541–553. POPL '14, Association for Computing Machinery, New York, NY, USA (Jan 2014). https://doi.org/10.1145/2535838.2535849, https://dl.acm.org/doi/10.1145/2535838.2535849

[5] D'Antoni, L., Veanes, M.: The power of symbolic automata and transducers. In: Proceedings 29th International Conference on Computer Aided Verification (Jul 2017), https://www.microsoft.com/en-us/research/publication/power-symbolic-automata-transducers-invited-tutorial/

[6] D'Antoni, L., Veanes, M.: Automata modulo theories. Communications of the ACM **64**(5), 86–95 (May 2021). https://doi.org/10.1145/3419404, https://dl.acm.org/doi/10.1145/3419404

[7] D'Antoni, L., Veanes, M., Livshits, B., Molnar, D.: Fast: A Transducer-Based Language for Tree Manipulation. ACM Transactions on Programming Languages and Systems **38**(1), 1:1–1:32 (Oct 2015). https://doi.org/10.1145/2791292, https://dl.acm.org/doi/10.1145/2791292

[8] Dawar, A., Grohe, M., Kreutzer, S., Schweikardt, N.: Model Theory Makes Formulas Large. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) Automata, Languages and Programming. pp. 913–924. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_78

[9] D'Antoni, L., Veanes, M.: Extended symbolic finite automata and transducers. Formal Methods in System Design **47**(1), 93–119 (Aug

2015). https://doi.org/10.1007/s10703-015-0233-4, https://doi.org/10.1007/s10703-015-0233-4

[10] Eisenbrand, F., Shmonin, G.: Carathéodory bounds for integer cones. Operations Research Letters **34**(5), 564–568 (Sep 2006). https://doi.org/10.1016/j.orl.2005.09.008, https://doi.org/10.1016/j.orl.2005.09.008

[11] Feferman, S., Vaught, R.: The first order properties of products of algebraic systems. Fundamenta Mathematicae **47**(1), 57–103 (1959)

[12] Figueira, D., Lin, A.W.: Reasoning on Data Words over Numeric Domains. In: Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 1–13. ACM, Haifa Israel (Aug 2022). https://doi.org/10.1145/3531130.3533354, https://dl.acm.org/doi/10.1145/3531130.3533354

[13] Fisman, D., Frenkel, H., Zilles, S.: On the Complexity of Symbolic Finite-State Automata (Jul 2021), http://arxiv.org/abs/2011.05389, arXiv:2011.05389 [cs]

[14] Hooimeijer, P., Livshits, B., Molnar, D., Saxena, P., Veanes, M.: Fast and Precise Sanitizer Analysis with BEK. In: 20th USENIX Conference on Security. USENIX Association, San Francisco, CA, Berkeley CA, USA (2011)

[15] Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Prentice Hall, Boston (Jun 2006)

[16] Hu, Q., D'Antoni, L.: Automatic program inversion using symbolic transducers. In: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 376–389. PLDI 2017, Association for Computing Machinery, New York, NY, USA (Jun 2017). https://doi.org/10.1145/3062341.3062345, https://dl.acm.org/doi/10.1145/3062341.3062345

[17] Kleene, S.C.: Representation of Events in Nerve Nets and Finite Automata. In: Representation of Events in Nerve Nets and Finite Automata, pp. 3–42. Princeton University Press (1956). https://doi.org/10.1515/9781400882618-002, https://www.degruyter.com/document/doi/10.1515/9781400882618-002/html

[18] Kuncak, V., Nguyen, H.H., Rinard, M.: Deciding Boolean Algebra with Presburger Arithmetic. Journal of Automated Reasoning **36**(3), 213–239 (Apr 2006). https://doi.org/10.1007/s10817-006-9042-1, https://doi.org/10.1007/s10817-006-9042-1

[19] Kuncak, V., Rinard, M.: Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic. In: Automated Deduction

– CADE-21. pp. 215–230. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73595-3_15

[20] Raya, R., Kunčak, V.: NP Satisfiability for Arrays as Powers. In: Verification, Model Checking, and Abstract Interpretation. pp. 301–318. Lecture Notes in Computer Science, Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-94583-1_15

[21] Saarikivi, O., Veanes, M., Mytkowicz, T., Musuvathi, M.: Fusing effectful comprehensions. In: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 17–32. ACM, Barcelona Spain (Jun 2017). https://doi.org/10.1145/3062341.3062362, https://dl.acm.org/doi/10.1145/3062341.3062362

[22] Saarikivi, O., Veanes, M., Wan, T., Xu, E.: Symbolic Regex Matcher. In: Tools and Algorithms for the Construction and Analysis of Systems: 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part I. pp. 372–378. Springer-Verlag, Berlin, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17462-0_24, https://doi.org/10.1007/978-3-030-17462-0_24

[23] Seidl, H., Schwentick, T., Muscholl, A., Habermehl, P.: Counting in Trees for Free. In: Automata, Languages and Programming. pp. 1136–1149. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27836-8_94

[24] Skolem, T.: Untersuchungen über die Axiome des Klassenkalküls und über Produktations- und Summationsprobleme, welche gewisse Klassen von Aussagen betreffen (1919)

[25] Tamm, H., Veanes, M.: Theoretical Aspects of Symbolic Automata. In: SOFSEM 2018: Theory and Practice of Computer Science, vol. 10706, pp. 428–441. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-73117-9_30, http://link.springer.com/10.1007/978-3-319-73117-9_30, series Title: Lecture Notes in Computer Science

[26] Veanes, M.: Applications of Symbolic Finite Automata. In: Konstantinidis, S. (ed.) Implementation and Application of Automata, vol. 7982, pp. 16–23. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39274-0_3, http://link.springer.com/10.1007/978-3-642-39274-0_3, series Title: Lecture Notes in Computer Science

[27] Veanes, M., Bjørner, N., Nachmanson, L., Bereg, S.: Monadic Decomposition. Journal of the ACM **64**(2), 1–28 (Apr 2017).

https://doi.org/10.1145/3040488, https://dl.acm.org/doi/10.1145/3040488

[28] Veanes, M., de Halleux, P., Tillmann, N.: Rex: Symbolic Regular Expression Explorer. In: Verification and Validation 2010 Third International Conference on Software Testing. pp. 498–507 (Apr 2010). https://doi.org/10.1109/ICST.2010.15, iSSN: 2159-4848

[29] Watson, B.W.: Implementing and using finite automata toolkits. Natural Language Engineering **2**(4), 295–302 (Dec 1996). https://doi.org/10.1017/S135132499700154X, publisher: Cambridge University Press

[30] Wies, T., Piskac, R., Kuncak, V.: Combining Theories with Shared Set Operations. In: Frontiers of Combining Systems. pp. 366–382. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04222-5_23