Inferring Symbolic Automata

Dana Fisman
Ben-Gurion University, Be'er Sheva, Israel

Hadar Frenkel
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Sandra Zilles
University of Regina, Regina, Canada

Abstract

We study the learnability of *symbolic finite state automata*, a model shown useful in many applications in software verification. The state-of-the-art literature on this topic follows the *query learning* paradigm, and so far all obtained results are positive. We provide a necessary condition for efficient learnability of SFAs in this paradigm, from which we obtain the first negative result. The main focus of our work lies in the learnability of SFAs under the paradigm of *identification in the limit using polynomial time and data*. We provide a necessary condition and a sufficient condition for efficient learnability of SFAs in this paradigm, from which we derive a positive and a negative result.

2012 ACM Subject Classification Theory of computation → Regular languages; Theory of computation → Formal languages and automata theory; Theory of computation → Models of computation; Computing Methodologies → Machine Learning

Keywords and phrases Symbolic Finite State Automata, Query Learning, Characteristic Sets

Digital Object Identifier 10.4230/LIPIcs.CSL.2022.27

Funding *Dana Fisman*: This research was partially supported by the United States - Israel Binational Science Foundation (BSF) grant 2016239.

1 Introduction

Symbolic finite state automata, SFAs for short, are an automata model in which transitions between states correspond to predicates over a domain of concrete alphabet letters. Their purpose is to cope with situations where the domain of concrete alphabet letters is large or infinite. As an example for automata over finite large alphabets consider automata over the alphabet 2^{AP} where AP is a set of atomic propositions; these are used in model checking [21]. Another example, used in string sanitizer algorithms [32], are automata over predicates on the Unicode alphabet which consists of over a million symbols. An infinite alphabet is used for example in *event recording automata*, a determinizable class of timed automata [2] in which an alphabet letter consists of both a symbol from a finite alphabet, and a non-negative real number. Formally, the transition predicates in an SFA are defined wrt. an effective Boolean algebra as defined in §2.

SFAs have proven useful in many applications [23, 44, 10, 34, 45, 39] and consequently have been studied as a theoretical model of automata. Many algorithms for natural operations and decision problems regarding these automata already exist in the literature, in particular, Boolean operations, determinization, and emptiness [49]; minimization [22]; and language inclusion [35]. Recently the subject of learning automata in verification has also attracted attention, as it has been shown useful in many applications, see Vaandrager's survey [48].

There already exists substantial literature on learning restricted forms of SFAs [31, 36, 11, 37, 19], as well as general SFAs [25, 9], and even non-deterministic residual SFAs [20]. For other types of automata over infinite alphabets, [33] suggests learning abstractions, and [47] presents a learning algorithm for deterministic variable automata. All these works consider the query learning paradigm, and provide extensions to Angluin's $\mathbf{L}^*$ algorithm for learning DFAs using membership and equivalence queries [4]. Unique to these works is the work [9] which studies the learnability of SFAs taking as a parameter the learnability of the underlying algebras, providing positive results regarding specific Boolean algebras.

While Argyros and D'Antoni's work [9] is a major advancement towards a systematic way for obtaining results on learnability of SFAs, as it examines the learnability of the underlying algebra, the obtained result allows inferring only positive results, as it relies on a specific query learning algorithm, and does not provide means for obtaining a negative result regarding query learning of SFAs over certain algebras. We provide a necessary condition for efficient learnability of SFAs in the query learning paradigm. From this result we obtain a negative result regarding query learning of SFAs over the propositional algebra. This is, to the best of our knowledge, the first negative result on learning SFAs with membership and equivalence queries and thus gives useful insights into the limitations of the $\mathbf{L}^*$ framework in this context.

The main focus of our work lies on the learning paradigm of *identification in the limit using polynomial time and data*, or its strengthened version *efficient identifiability*. We provide a necessary condition a class of SFAs $\mathbb{M}$ should meet in order to be identified in the limit using polynomial time and data, and a sufficient condition a class of SFAs $\mathbb{M}$ should meet in order to be efficiently identifiable. These conditions are expressed in terms of the existence of certain efficiently computable functions, which we call $\mathsf{Generalize}_{\mathbb{M}}$, $\mathsf{Concretize}_{\mathbb{M}}$, and $\mathsf{Decontaminate}_{\mathbb{M}}$. We then provide positive and negative results regarding the learnability of specific classes of SFAs in this paradigm. In particular, we show that the class of SFAs over *any* monotonic algebras is efficiently identifiable.

2 Preliminaries

2.1 Effective Boolean Algebra

A Boolean Algebra $\mathcal{A}$ can be represented as a tuple $(\mathbb{D}, \mathbb{P}, [\![\cdot]\!], \bot, \top, \vee, \wedge, \neg)$ where $\mathbb{D}$ is a set of domain elements; $\mathbb{P}$ is a set of predicates closed under the Boolean connectives, where $\bot, \top \in \mathbb{P}$; the component $[\![\cdot]\!] : \mathbb{P} \to 2^{\mathbb{D}}$ is the so-called *semantics function*. It satisfies the following three requirements: (i) $[\![\bot]\!] = \emptyset$, (ii) $[\![\top]\!] = \mathbb{D}$, and (iii) for all $\varphi, \psi \in \mathbb{P}$, $[\![\varphi \vee \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$, $[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!]$, and $[\![\neg\varphi]\!] = \mathbb{D} \setminus [\![\psi]\!]$. A Boolean Algebra is *effective* if all the operations above, as well as satisfiability, are decidable. Henceforth, we implicitly assume Boolean algebras to be effective.

One way to define a Boolean algebra is by defining a set $\mathbb{P}_0$ of *atomic formulas* that includes $\top$ and $\bot$ and obtaining $\mathbb{P}$ by closing $\mathbb{P}_0$ for conjunction, disjunction and negation. For a predicate $\psi \in \mathbb{P}$ we say that ψ is *atomic* if $\psi \in \mathbb{P}_0$. We say that ψ is *basic* if ψ is a conjunction of atomic formulas.

We now introduce two Boolean algebras that are discussed extensively in the paper.

The Interval Algebra is the Boolean algebra in which the domain $\mathbb{D}$ is the set $\mathbb{Z} \cup \{-\infty, \infty\}$ of integers augmented with two special symbols with their standard semantics, and the set of atomic formulas $\mathbb{P}_0$ consists of intervals of the form $[a, b)$ where $a, b \in \mathbb{D}$ and $a \leq b$. The semantics associated with intervals is the natural one: $[\![[a, b)]\!] = \{z \in \mathbb{D} \mid a \leq z \text{ and } z < b\}$.

The Propositional Algebra is defined wrt. a set $AP = \{p_1, p_2, \ldots, p_k\}$ of atomic propositions. The set of *atomic predicates* $\mathbb{P}_0$ consists of the atomic propositions and their negations as well as $\top$ and $\bot$. The domain $\mathbb{D}$ consists of all the possible valuations for these propositions,

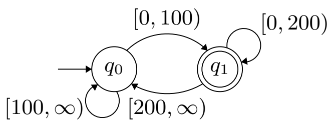

Figure 1 The SFA $\mathcal{M}$ over $\mathcal{A}_{\mathbb{N}}$

thus it is $\mathbb{B}^k$ where $\mathbb{B}=\{0,1\}$. The semantics of an atomic predicate p is given by $[\![p_i]\!] = \{v \in \mathbb{B}^k \mid v[i]=1\}$, and similarly $[\![\neg p_i]\!] = \{v \in \mathbb{B}^k \mid v[i]=0\}$.[1]

2.2 Symbolic Automata

A *symbolic finite automaton* (SFA) is a tuple $\mathcal{M} = (\mathcal{A}, Q, q_\iota, F, \Delta)$ where $\mathcal{A}$ is a Boolean algebra, Q is a finite set of states, $q_\iota \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\Delta \subseteq Q \times \mathbb{P}_{\mathcal{A}} \times Q$ is a finite set of transitions, where $\mathbb{P}_{\mathcal{A}}$ is the set of predicates of $\mathcal{A}$.

We use the term *letters* for elements of $\mathbb{D}$ where $\mathbb{D}$ is the domain of $\mathcal{A}$ and the term *words* for elements of $\mathbb{D}^*$. A run of $\mathcal{M}$ on a word $a_1a_2\dots a_n$ is a sequence of transitions $\langle q_0,\psi_1,q_1\rangle\langle q_1,\psi_2,q_2\rangle\dots\langle q_{n-1},\psi_n,q_n\rangle$ satisfying that $a_i \in [\![\psi_i]\!]$, that $\langle q_i,\psi_{i+1},q_{i+1}\rangle \in \Delta$ and that $q_0 = q_\iota$. Such a run is said to be *accepting* if $q_n \in F$. A word $w = a_1a_2\dots a_n$ is said to be *accepted* by $\mathcal{M}$ if there exists an accepting run of $\mathcal{M}$ on w. The set of words accepted by an SFA $\mathcal{M}$ is denoted $\mathcal{L}(\mathcal{M})$. We use $\hat{\mathcal{L}}(\mathcal{M})$ for the set $\{\langle w,1\rangle \mid w \in \mathcal{L}(\mathcal{M})\} \cup \{\langle w,0\rangle \mid w \notin \mathcal{L}(\mathcal{M})\}$.

An SFA is said to be *deterministic* if for every state $q \in Q$ and every letter $a \in \mathbb{D}$ we have that $|\{\langle q,\psi,q'\rangle \in \Delta \mid a \in [\![\psi]\!]\}| \leq 1$, namely from every state and every concrete letter there exists at most one transition. It is said to be *complete* if $|\{\langle q,\psi,q'\rangle \in \Delta \mid a \in [\![\psi]\!]\}| \geq 1$ for every $q \in Q$ and $a \in \mathbb{D}$, namely from every state and every concrete letter there exists at least one transition. It is not hard to see that, as is the case for finite automata (over concrete alphabets), non-determinism does not add expressive power but does add succinctness. When $\mathcal{A}$ is deterministic we use $\Delta(q,w)$ to denote the state $\mathcal{A}$ reaches on reading word w from state q. If $\Delta(q_\iota,w) = q$ then w is termed an *access word to state* q.

▶ **Example 1.** Consider the SFA $\mathcal{M}$ given in Fig.1. It is defined over the algebra $\mathcal{A}_{\mathbb{N}}$ which is the interval algebra restricted to the domain $\mathbb{D} = \mathbb{N} \cup \{\infty\}$. The language of $\mathcal{M}$ is the set of all words over $\mathbb{D}$ of the form $w_1 \cdot d \cdot w_2$ where w_1 is some word over the domain $\mathbb{D}$, the letter d satisfies $0 \leq d < 100$ and all letters of the word w_2 are numbers smaller than 200.

3 Learning SFAs

In grammatical inference, loosely speaking, we are interested in learning a class of languages $\mathbb{L}$ over an alphabet Σ, from examples which are words over Σ. Examples for classes of languages can be the set of regular languages, the set of context-free languages, etc. A learning algorithm, aka a *learner*, is expected to output some concise representation of the language from a class of representations $\mathbb{R}$ for the class $\mathbb{C}$. For instance, in learning the class $\mathbb{L}_{reg}$ of regular languages one might consider the class $\mathbb{R}_{\text{DFA}}$ of DFAs, or the class $\mathbb{R}_{\text{LIN}}$ of right linear grammars, since both are capable of expressing all regular languages.[2] We often say that a class of representations $\mathbb{R}$ is learnable (or not) when we mean that a class of languages $\mathbb{L}$ is learneable (or not) via the class of representations $\mathbb{R}$. Complexity of learning an unknown language $L \in \mathbb{L}$ via $\mathbb{R}$ is typically measured wrt. the size of the smallest

[1] In this case a basic formula is a *monomial*.
[2] The class of regular languages was shown learnable via various representations including DFAs [4], NFAs [16], and AFAs (alternating finite automata) [7].

representation $R_L \in \mathbb{R}$ for L. For instance, when learning $\mathbb{L}_{reg}$ via $\mathbb{R}_{\text{DFA}}$ a learner is expected to output a DFA for an unknown language in time that is polynomial in the number of states of the minimal DFA for L.

In our setting we are interested in learning regular languages using as a representation classes of SFAs over a certain algebra. To measure complexity we must agree on how to measure the size of an SFA. For DFAs, the number of states is a common measure of size, since the DFA can be fully described by a representation of size polynomial in the number of states. In the case of SFA the situation is different, as the size of the predicates labeling the transitions can vary greatly. In fact, if we measure the size of a predicate by the number of nodes in its parse DAG, then the size of a formula can grow unboundedly. The size and structure of the predicates influence the complexity of their satisfiability check, and thus the complexity of the corresponding algorithms. Another thing to note is that there might be a trade-off between the size of the transition predicates and the number of transitions; e.g. a predicate of the form $\psi_1 \vee \psi_2 \ldots \vee \psi_k$ can be replaced by k transitions, each one labeled by one ψ_i for $1 \leq i \leq k$.

The literature defines an SFA as *normalized* if for every two states q and q' there exists at most one transition from q to q'. This definition prefers fewer transitions over potentially complicated predicates. By contrast, preferring simple transitions at the cost of increasing the number of transitions, leads to *neat* SFAs. An SFA is termed *neat* if all transition predicates are basic predicates. In [27] we proposed to measure the size of an SFA by three parameters: the number of states (n), the maximal out-degree of a state (m) and the size of the most complex predicate (l); we then analyzed the complexity of the standard operations on SFAs, with particular attention to the mentioned special forms. Another important factor regarding size and canonical forms of SFAs, is the underlying algebra, specifically, whether it is monotonic or not.

Monotonicity A Boolean algebra $\mathcal{A}$ over domain $\mathbb{D}$ is said to be *monotonic* if there exists a total order $<$ on the elements of $\mathbb{D}$, there exist two elements $d_{-\infty}, d_{\infty}$ such that $d_{-\infty} \leq d$ and $d \leq d_{\infty}$ for all $d \in \mathbb{D}$, and an atomic predicate $\psi \in \mathbb{P}_0$ can be associated with two concrete values a and b such that $[\![\psi]\!] = \{d \in \mathbb{D} \mid a \leq d < b\}$. The interval algebra (given in §2.1) is clearly monotonic, as is the similar algebra obtained using $\mathbb{R}$ (the real numbers) instead of $\mathbb{Z}$ (the integers). On the other hand, the propositional algebra is clearly non-monotonic.

Learning Paradigms The exact definition regarding learnability of a class depends on the *learning paradigm*. In this work we consider two widely studied paradigms: *learning with membership and equivalence queries*, and *identification in the limit using polynomial time and data*. Their definitions are provided in the respective sections.

Non-Trivial Classes of SFAs In the sequel we would like to prove results regarding non-trvial classes of SFAs, which are defined as follows.

▶ **Definition 2.** *A class of SFAs $\mathbb{M}$ over a Boolean Algebra $\mathcal{A}$ with a set of predicates $\mathbb{P}$ is termed* non-trivial *if for every predicate $\varphi \in \mathbb{P}$ the SFA $\mathcal{M}_\varphi = (\mathcal{A}, \{q_\iota, q_{ac}, q_{rj}\}, q_\iota, \{q_{ac}\}, \Delta)$ where $\Delta = \{\langle q_\iota, \varphi, q_{ac}\rangle, \langle q_\iota, \neg\varphi, q_{rj}\rangle, \langle q_{rj}, \top, q_{rj}\rangle, \langle q_{ac}, \top, q_{rj}\rangle\}$ is in $\mathbb{M}$. Note that $\mathcal{M}_\varphi$ accepts only words of length one consisting of a concrete letter satisfying φ, and it is minimal among all complete deterministic SFAs accepting this language (minimal in both number of states and number of transitions).*

4 Efficient Identifiability

While in *active learning* (e.g. query learning) the learner can select any word and query about its membership in the unknown language, in *passive learning* the learner is given a set of words, and for each word w in the set, a label b_w indicating whether w is in the unknown language or not. Formally, a *sample* for a language L is a finite set $\mathcal{S}$ consisting of labeled examples, that is, pairs of the form $\langle w, b_w \rangle$ where w is a word and $b_w \in \{0, 1\}$ is its label, satisfying that $b_w = 1$ if and only if $w \in L$. The words that are labeled 1 are termed *positive* words, and those that are labeled 0 are termed *negative* words. Note that if L is recognized by $\mathcal{M}$, we have that $\mathcal{S} \subseteq \hat{\mathcal{L}}(\mathcal{M})$ (as defined in §.2.2). If $\mathcal{S}$ is a sample for L we often say that $\mathcal{S}$ *agrees with* L. Given two words w, w', we say that w and w' are not equivalent wrt. $\mathcal{S}$, denoted $w \not\sim_{\mathcal{S}} w'$, iff there exists z such that $\langle wz, b \rangle, \langle w'z, b' \rangle \in \mathcal{S}$ and $b \neq b'$. Otherwise we say that w and w' are equivalent wrt. $\mathcal{S}$, and write $w \sim_{\mathcal{S}} w'$.

Given a sample $\mathcal{S}$ for a language L over a concrete domain $\mathbb{D}$, it is possible to construct a DFA that agrees with $\mathcal{S}$ in polynomial time. Indeed one can create the *prefix-tree automaton*, a simple automaton that accepts all and only the positively labeled words in the sample. Clearly the constructed automaton may not be the minimal automaton that agrees with $\mathcal{S}$. There are several algorithms, in particular the popular RPNI [42], that minimize the prefix-tree automaton, and due to state merging may accept an infinite language. Obviously though, this procedure is not guaranteed to return an automaton for the unknown language, as the sample may not provide sufficient information. For instance if $L = aL_1 \cup bL_2$ and the sample contains only words starting with a, there is no way for the learner to infer L_2 and hence also L correctly. One may thus ask, given a language L, what should a sample contain in order for a passive learning algorithm to infer L correctly, and can such sample be of polynomial size with respect to a minimal representation (e.g., a DFA) for the language.

One approach to answer these questions is captured in the paradigm of *identification in the limit using polynomial time and data*. This model was proposed by Gold [28], who also showed that it admits learning of regular languages represented by DFAs. We follow de la Higuera's more general definition [24].[3] This definition requires that for any language L in a class of languages $\mathbb{L}$ represented by $\mathbb{R}$, there exists a sample $\mathcal{S}_L$ of size polynomial in the size of the smallest representation $R \in \mathbb{R}$ of L (e.g., the smallest DFA for L), such that a valid learner can infer the unknown language L from the information contained in $\mathcal{S}_L$. The set $\mathcal{S}_L$ is then termed a *characteristic sample*.[4] Here, a valid learner is an algorithm that learns the target language exactly and efficiently. In particular, a valid learner produces in polynomial time a representation that agrees with the provided sample. The learner also has to correctly learn the unknown language L when given the characteristic sample $\mathcal{S}_L$ as input. Moreover, if the input sample $\mathcal{S}$ subsumes $\mathcal{S}_L$ yet is still consistent with L, the additional information in the sample should not "confuse" the learner; the latter still has to output a correct representation for L. (Intuitively, this requirement precludes situations in which the sample consists of some smart encoding of the representation that the learner simply deciphers. In particular, the learner will not be confused if an adversary "contaminates" the

[3] This paradigm may seem related to conformance testing. The relation between conformance testing for Mealy machines and automata learning of DFAs has been explored in [14].

[4] De la Higuera's notion of characteristic sample is a core concept in grammatical inference, for various reasons. Firstly, it addresses shortcomings of several other attempts to formulate polynomial-time learning in the limit [5, 43]. Secondly, this notion has inspired the design of popular algorithms for learning formal languages such as, for example, the RPNI algorithm [42]. Thirdly, it was shown to bear strong relations to a classical notion of machine teaching [30]; models of the latter kind are currently experiencing increased attention in the machine learning community [50].

characteristic sample by adding labeled examples for the target language.) We provide the formal definition after the following informal example.

► **Example 3.** For the class of DFAs, let us consider the regular language $L = a^*$ over the alphabet $\{a, b\}$. Further, consider a sample set $\mathcal{S} = \{\langle\epsilon, 1\rangle, \langle a, 1\rangle, \langle b, 0\rangle, \langle bb, 0\rangle, \langle ba, 0\rangle\}$ for L. There is a valid learner for the class of all DFAs that uses the sample $\mathcal{S}$ as a characteristic sample for L. By definition, such a learner has to output a DFA for L when fed with $\mathcal{S}$, but also has to output equivalent DFAs whenever given any superset of $\mathcal{S}$ as input, as long as this superset agrees with L. Naturally, the sample $\mathcal{S}$ is also consistent with the regular language $L' = \{\epsilon, a\}$. However, this does not pose any problem, since the same learner can use a characteristic sample for L' that disagrees with L, for example, $\mathcal{S}' = \{\langle\epsilon, 1\rangle, \langle a, 1\rangle, \langle aa, 0\rangle\}$. When defining a system of characteristic samples like that, the core requirement is that the size of a sample be bounded from above by a function that is polynomial in the size of the smallest DFA for the respective target language.

► **Definition 4** (identification in the limit using polynomial time and data). *A class of languages* $\mathbb{L}$ *is said to be* identified in the limit using polynomial time and data *via representations in a class* $\mathbb{R}$ *if there exists a learning algorithm* $\boldsymbol{A}$ *such that the following requirements are met.*

1. *Given a finite sample* $\mathcal{S}$ *of labeled examples,* $\boldsymbol{A}$ *returns a hypothesis* $\mathcal{R} \in \mathbb{R}$ *that agrees with* $\mathcal{S}$ *in polynomial time.*
2. *For every language* $L \in \mathbb{L}$*, there exists a sample* $\mathcal{S}_L$*, termed* a characteristic sample*, of size polynomial in the minimal representation* $\mathcal{R} \in \mathbb{R}$ *for* L *such that the algorithm* $\boldsymbol{A}$ *returns a correct hypothesis when run on any sample* $\mathcal{S}$ *for* L *that subsumes* $\mathcal{S}_L$*.*

Note that the first condition ensures polynomial time and the second polynomial data. However, the latter is not a worst-case measure; the algorithm may fail to return a correct hypothesis on arbitrarily large finite samples (if they do not subsume a characteristic set).

Note also that the definition does not require the existence of an efficient algorithm that constructs a characteristic sample for each language in the underlying class. When such an algorithm is also available we say that the class is *efficiently identifiable.* In the full version of the paper we provide an example of a class of languages that possesses polynomial-size characteristic sets, yet without the ability to construct such sets effectively. Since we are concerned with learning classes of automata we formulate the definition of *efficient identification* directly over classes of automata.

► **Definition 5** (efficient identification). *A class of automata* $\mathbb{M}$ *over an alphabet* Σ *is said to be* efficiently identified *if the following two requirements are met.*

1. *There exists a polynomial time learning algorithm* $\boldsymbol{Infer} : 2^{(\Sigma^* \times \{0,1\})} \to \mathbb{M}$ *such that, for any sample* $\mathcal{S}$*, we have* $\mathcal{S} \subseteq \hat{\mathcal{L}}(\boldsymbol{Infer}(\mathcal{S}))$*.*
2. *There exists a polynomial time algorithm* $\boldsymbol{Char} : \mathbb{M} \to 2^{(\Sigma^* \times \{0,1\})}$ *such that, for every* $\mathcal{M} \in \mathbb{M}$ *and every sample* $\mathcal{S}$ *satisfying* $\boldsymbol{Char}(\mathcal{M}) \subseteq \mathcal{S} \subseteq \hat{\mathcal{L}}(\mathcal{M})$*, the automaton* $\boldsymbol{Infer}(\mathcal{S})$ *recognizes the same language as* $\mathcal{M}$*.*

When we apply this definition for a class of SFAs over a Boolean algebra $\mathcal{A}$ with domain $\mathbb{D}$ and predicates $\mathbb{P}$, the characteristic sample is defined over the concrete set of letters $\mathbb{D}$ rather than the set of predicates $\mathbb{P}$ because this is the alphabet of the words accepted by an SFA (inferring an SFA from a set of words labeled by predicates can be done using the methods for inferring DFAs, by considering the alphabet to be the set of predicates).

Throughout this section we study whether a class of SFAs $\mathbb{M}$ is efficiently identifiable. That is, we are interested in the existence of algorithms $\mathbf{Infer}_{\mathbb{M}}$ and $\mathbf{Char}_{\mathbb{M}}$ satisfying the requirements of Def.5. In §4.1 we provide a necessary condition for a non-trivial class of SFAs to be identified in the limit using polynomial time and data. In §4.2 we provide a sufficient condition for a non-trivial class of SFAs to be efficiently identifiable. On the positive side, we show in §4.3 that the class of SFAs over the interval algebra is efficiently identifiable. On the negative side, we show in §4.4 that SFAs over the general propositional algebra cannot be identified in the limit using polynomial time and data.

Efficient Identification of DFAs

Before investigating efficient identification of SFAs, it is worth noting that DFAs are efficiently identifiable. We state a result that provides more details about the nature of these algorithms, since we need it later, in §.4.3, to provide our positive result. Intuitively, it says that there exists a valid learner such that if $\mathcal{D}$ is a minimal DFA recognizing a certain language L then the learner can infer L from a characteristic sample consisting of access words to each state of $\mathcal{D}$ and their extensions with distinguishing words (words showing each pair of states cannot be merged) as well as one letter extensions of the access words that are required to retrieve the transition relation.

▶ **Theorem 6** ([42]). ***I.*** *The class of DFAs is efficiently identifiable via procedures* ***CharDFA*** *and* ***InferDFA****.* ***II.*** *Furthermore, these procedures satisfy that if $\mathcal{D}$ is a minimal and complete DFA and* ***CharDFA***$(\mathcal{D}) = \mathcal{S}_\mathcal{D}$ *then the following holds:*

1. *$\mathcal{S}_\mathcal{D}$ contains a prefix-closed set A of access words. Moreover, A can be chosen to contain only lex-access words, i.e., only the lexicographically smallest access word for each state.*
2. *For every $u_1, u_2 \in A$ it holds that $u_1 \not\sim_{\mathcal{S}_\mathcal{D}} u_2$.*
3. *For every $u, v \in A$ and $\sigma \in \Sigma$, if $\Delta(q_\iota, u\sigma) \neq \Delta(q_\iota, v)$ then $u\sigma \not\sim_{\mathcal{S}_\mathcal{D}} v$.*

We briefly describe **CharDFA** and **InferDFA**.

The algorithm **CharDFA** works as follows. It first creates a prefix-closed set of access words to states. This can be done by considering the graph of the automaton and running an algorithm for finding a spanning tree from the initial state. Choosing one of the letters on each edge, the access word for a state is obtained by concatenating the labels on the unique path of the obtained tree that reaches that state. If we wish to work with lex-access words, we can use a depth-first search algorithm that spans branches according to the order of letters in Σ, starting from the smallest. The labels on the paths of the spanning tree constructed this way will form the set of lex-access words. Let S be the set of access words (or lex-access words). Next the algorithm turns to find a distinguishing word $v_{i,j}$ for every pair of state $s_i, s_j \in S$ (where $s_i \neq s_j$). It holds that any pair of states of the minimal DFA has a distinguishing word of size quadratic in the size of the DFA. Let E be the set of all such distinguishing words. Then the algorithm returns the set $\mathcal{S}_\mathcal{D} = \{\langle w, \mathcal{D}(w)\rangle \mid w \in (S \cdot E) \cup (S \cdot \Sigma \cdot E)\}$ where $\mathcal{D}(w)$ is the label $\mathcal{D}$ gives w (i.e. 1 if it is accepted, and 0 otherwise). It is easy to see that $\mathcal{S}_\mathcal{D}$ satisfies the properties of Thm.6.

The algorithm **InferDFA**, given a sample of words $\mathcal{S}$, infers from it in polynomial time a DFA that agrees with $\mathcal{S}$. Moreover, if $\mathcal{S}$ subsumes the characteristic set $\mathcal{S}_\mathcal{D}$ of a DFA $\mathcal{D}$ then **InferDFA** returns a DFA that recognizes $\mathcal{D}$. Let W be the set of words in the given sample $\mathcal{S}$ (without their labels). Let R be the set of prefixes of W and C the set of suffixes of W. Note that $\epsilon \in R$ and $\epsilon \in C$. Let $r_0, r_1, \ldots$ be some enumeration of R and $c_0, c_1, \ldots$ some enumeration of C where $r_0 = c_0 = \epsilon$. The algorithm builds a matrix M of size $|R| \times |C|$ whose entries take values in $\{0, 1, ?\}$, and sets the value of entry (i, j) as follows. If $r_i c_j$ is not in W, it is set to ?. Otherwise it is set to 1 iff the word $r_i c_j$ is labeled 1 in $\mathcal{S}$. We get that $r_i \sim_\mathcal{S} r_j$ iff for every k such that both $M(i,k)$ and $M(j,k)$ are different than ? we have that $M(i,k) = M(j,k)$. The algorithm sets $R_0 = \{\epsilon\}$. Once R_i is constructed, the algorithm tries to establish whether for $r \in R_i$ and $\sigma \in \Sigma$, $r\sigma$ is distinguished from all words in R_i. It does so by considering all other words $r' \in R_i$ and checking whether $r \sim_{\mathcal{S}} r'$. If $r\sigma$ is found to be distinct from all words in R_i, then R_{i+1} is set to $R_i \cup \{r\sigma\}$. The algorithm proceeds until no new words are distinguished. Let k be the iteration of convergence. If not all words in R_k are in W (that is $M(i,0) = ?$ for some $r_i \in R_k$), the algorithm returns the prefix-tree automaton. Otherwise, the states of the constructed DFA are set to be the words in R_k. The initial state is ϵ and a state r_i is classified as accepting iff $M(i,0) = 1$ (recall that the entry $M(i,0)$ stands for the value of $r_i \cdot \epsilon$ in $\mathcal{S}$). To determine the transitions, for every $r \in R_k$ and $\sigma \in \Sigma$, recall that there exists at least one state $r' \in R$ that cannot be distinguished from $r\sigma$. The algorithm then adds a transition from r on σ to r'.

4.1 Necessary Condition

We make use of the following definitions. A sequence $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ consisting of sets of concrete letters $\Gamma_i \subseteq \mathbb{D}$ is termed a *concrete partition* of $\mathbb{D}$ if the sets are pairwise disjoint (namely $\Gamma_i \cap \Gamma_j = \emptyset$ for every $i \neq j$). Note that we do not require that in addition $\bigcup_{1 \leq i \leq k} \Gamma_i = \mathbb{D}$. We use $\Pi_{\mathsf{conc}}(\mathbb{D}, m)$ to define the set of all concrete partitions of size m over $\mathbb{D}$. A sequence of predicates $\langle \psi_1, \ldots, \psi_m \rangle$ over a Boolean algebra $\mathcal{A}$ on a domain $\mathbb{D}$ is termed a *predicate partition* if $[\![\psi_i]\!] \cap [\![\psi_j]\!] = \emptyset$ for every $i \neq j$, and in addition $\bigcup_{\leq i \leq k} [\![\psi_i]\!] = \mathbb{D}$. That is, here we do require the assignments to the predicates cover the domain. We use $\Pi_{\mathsf{pred}}(\mathbb{P}, m)$ to define the set of all predicate partitions of size m over $\mathbb{P}$.

▶ **Definition 7.**
- *A function f_g from a concrete partition to a predicate partition is termed* generalizing *if $f_g(\langle \Gamma_1, \ldots, \Gamma_m \rangle) = \langle \psi_1, \ldots, \psi_k \rangle$ implies $k = m$ and $[\![\psi_i]\!] \supseteq \Gamma_i$ for all $1 \leq i \leq m$.*
- *A function f_c from a predicate partition to a concrete partition is termed* concretizing *if $f_c(\langle \psi_1, \ldots, \psi_m \rangle) = \langle \Gamma_1, \ldots, \Gamma_k \rangle$ implies $k = m$ and $\Gamma_i \subseteq [\![\psi_i]\!]$ for all $1 \leq i \leq m$.*

Note that f_g and f_c are *variadic* functions (i.e. can take any number of parameters). We can define their k-adic versions as those that work only on partitions of size k. In particular, their *dyadic* versions work only on partitions of size 2.

We say that f_g (resp. f_c) is *efficient* if it can be computed in polynomial time. Note that if f_c is efficient then the sets Γ_i in the constructed concrete partition are of polynomial size.

We are now ready to provide a necessary condition for identifiability in the limit using polynomial time and data.

▶ **Theorem 8.** *A necessary condition for a non-trivial class of SFAs $\mathbb{M}_{\mathcal{A}}$ over a Boolean algebra $\mathcal{A}$ to be identified in the limit using polynomial time and data is that there exist efficient dyadic concretizing and generalizing functions, $\mathsf{Concretize}_{\mathcal{A}} : \Pi_{\mathit{pred}}(\mathbb{P}, 2) \to \Pi_{\mathit{conc}}(\mathbb{D}, 2)$ and $\mathsf{Generalize}_{\mathcal{A}} : \Pi_{\mathit{conc}}(\mathbb{D}, 2) \to \Pi_{\mathit{pred}}(\mathbb{P}, 2)$, satisfying that*

$$\begin{gathered}
\textit{if } \mathsf{Concretize}_{\mathcal{A}}(\langle \psi_1, \psi_2 \rangle) = \langle \Gamma_1, \Gamma_2 \rangle \\
\textit{and } \mathsf{Generalize}_{\mathcal{A}}(\langle \Gamma'_1, \Gamma'_2 \rangle) = \langle \varphi_1, \varphi_2 \rangle \\
\textit{where } \Gamma_i \subseteq \Gamma'_i \textit{ for every } 1 \leq i \leq 2 \\
\textit{then } [\![\varphi_i]\!] = [\![\psi_i]\!] \textit{ for every } 1 \leq i \leq 2.
\end{gathered}$$

Proof. Assume that $\mathbb{M}_{\mathcal{A}}$ is identified in the limit using polynomial time and data. That is, there exist two algorithms $\mathsf{CharSFA} : \mathbb{M}_{\mathcal{A}} \to 2^{\mathbb{D}^* \times \{0,1\}}$ and $\mathsf{InferSFA} : 2^{\mathbb{D}^* \times \{0,1\}} \to \mathbb{M}_{\mathcal{A}}$ satisfying the requirements of Def.4. We show that efficient dyadic concretizing and generalizing functions do exist.

We start with the definition of $\mathsf{Concretize}_{\mathcal{A}}$. Let $\langle \varphi_1, \varphi_2 \rangle$ be the argument of $\mathsf{Concretize}_{\mathcal{A}}$. Note that $\varphi_2 = \neg\varphi_1$ by the definition of a predicate partition. The implementation of

$\mathsf{Concretize}_{\mathcal{A}}$ invokes $\mathsf{CharSFA}$ on the SFA $\mathcal{M}_{\varphi_1}$ accepting all words of length one consisting of a concrete letter satisfying φ_1, as defined in Def.2. Let $\mathcal{S}$ be the returned sample. Let Γ_1 be the set of positively labeled words in the sample. Note that all such words are of size one, namely they are letters. Let Γ_2 be the set of letters that are first letters in a negative word in the sample. Then $\mathsf{Concretize}_{\mathcal{A}}$ returns $\langle \Gamma_1, \Gamma_2 \rangle$.

We turn to the definition of $\mathsf{Generalize}_{\mathcal{A}}$. Given $\langle \Gamma_1, \Gamma_2 \rangle$ the implementation of $\mathsf{Generalize}_{\mathcal{A}}$ invokes $\mathsf{InferSFA}$ on sample $\mathcal{S} = \{\langle \gamma, 1\rangle \mid \gamma \in \Gamma_1\} \cup \{\langle \gamma, 0\rangle \mid \gamma \in \Gamma_2\} \cup \{\langle \gamma\gamma', 0\rangle \mid \gamma, \gamma' \in \Gamma_1 \cup \Gamma_2\}$. That is, all one-letter words satisfying Γ_1 are positively labeled, all one-letter words satisfying Γ_2 are negatively labeled, and all words of length 2 using some of the given concrete letters, are negatively labeled. Let $\mathcal{M}$ be the returned SFA when given $\mathcal{S}' \supseteq \mathcal{S}$ as an input. Let Ψ_1 be the set of all predicates labeling some edge from the initial state to an accepting state, and let Ψ_2 be the set of all predicates labeling some edge from the initial state to a rejecting state. Let $\varphi = (\bigvee_{\psi \in \Psi_1} \psi) \wedge (\bigwedge_{\psi \in \Psi_2} \neg\psi)$. Then $\mathsf{Generalize}_{\mathcal{A}}$ returns $\langle \varphi, \neg\varphi \rangle$.

It is not hard to verify that the constructed methods $\mathsf{Generalize}_{\mathcal{A}}$ and $\mathsf{Concretize}_{\mathcal{A}}$ satisfy the requirements of the theorem. ◀

The following example shows that for some Boolean algebras, such functions exist, even for a generalization of the requirement for variadic versions of $\mathsf{Concretize}$ and $\mathsf{Generalize}$.

▶ **Example 9.** Consider the class $\mathbb{M}_{\mathcal{A}_{\mathbb{N}}}$ of SFAs over the algebra $\mathcal{A}_{\mathbb{N}}$ of Ex.1 and consider the functions $\mathsf{Concretize}_{\mathcal{A}_{\mathbb{N}}}(\langle [d_1, d_1'), [d_2, d_2'), \ldots, [d_m, d_m')\rangle) = \langle \{d_1\}, \ldots, \{d_m\}\rangle$ and $\mathsf{Generalize}_{\mathcal{A}_{\mathbb{N}}}$ $(\langle \Gamma_1, \ldots, \Gamma_m\rangle) = \langle [\min \Gamma_{\pi(1)}, \min \Gamma_{\pi(2)}), [\min \Gamma_{\pi(2)}, \min \Gamma_{\pi(3)}), \ldots, [\min \Gamma_{\pi(m)}, \infty)\rangle$ where π is the permutation on $(1, \ldots, m)$ satisfying $\max \Gamma_{\pi(i)} < \min \Gamma_{\pi(i+1)}$ for every $1 \leq i < m$. Then, $\mathsf{Concretize}_{\mathcal{A}_{\mathbb{N}}}$ and $\mathsf{Generalize}_{\mathcal{A}_{\mathbb{N}}}$ satisfy the variadic generalization of the conditions of Thm.8.

We would like to relate the necessary condition on the learnability of a class of SFAs over a Boolean algebra $\mathcal{A}$ to the learnability of the Boolean algebra $\mathcal{A}$ itself. For this we need to first define efficient identifiability of a Boolean algebra $\mathcal{A}$. Since to learn an unknown predicate we need to supply two sets: one of negative examples and one of positive examples, it makes sense to say that a Boolean algebra $\mathcal{A}$ with predicates $\mathbb{P}$ over domain $\mathbb{D}$ is *efficiently identifiable* if there exist efficient dyadic concretizing and generalizing functions, $\mathsf{Concretize}_{\mathcal{A}} : \Pi_{\mathsf{pred}}(\mathbb{P}, 2) \to \Pi_{\mathsf{conc}}(\mathbb{D}, 2)$ and $\mathsf{Generalize}_{\mathcal{A}} : \Pi_{\mathsf{conc}}(\mathbb{D}, 2) \to \Pi_{\mathsf{pred}}(\mathbb{P}, 2)$ satisfying the criteria of Theorem 8. Using this terminology we can state the following corollary.

▶ **Corollary 10.** *Efficient identifiability of the Boolean algebra $\mathcal{A}$ is a necessary condition for identification in the limit using polynomial time and data of any non-trivial class of SFAs over $\mathcal{A}$.*

4.2 Sufficient Condition

We turn to discuss a sufficient condition for the efficient identifiability of a class of SFAs $\mathbb{M}_{\mathcal{A}}$ over a Boolean algebra $\mathcal{A}$. To prove that $\mathbb{M}_{\mathcal{A}}$ is efficiently identifiable, we need to supply two algorithms $\mathbf{CharSFA}_{\mathbb{M}_{\mathcal{A}}}$ and $\mathbf{InferSFA}_{\mathbb{M}_{\mathcal{A}}}$ as required in Def.5. The idea is to reduce the problem to efficient identifiablity of DFAs, namely to use the algorithms **CharDFA** and **InferDFA** provided in Thm.6. The implementation of **CharSFA**, given an SFA $\mathcal{M}$ will transform it into a DFA $\mathcal{D}_{\mathcal{M}}$ by applying $\mathsf{Concretize}_{\mathcal{A}}$ on the partitions induced by the states of the DFA. The resulting DFA $\mathcal{D}_{\mathcal{M}}$ will not be equivalent to the given SFA $\mathcal{M}$, but it may be used to create a sample of words $\mathcal{S}_{\mathcal{M}}$ that is a characteristic set for $\mathcal{M}$, see Fig.2. To implement **InferSFA** we would like to use **InferDFA** to obtain, as a first step, a DFA from the given sample, then at the second step, apply $\mathsf{Generalize}_{\mathcal{A}}$ on the concrete-partitions

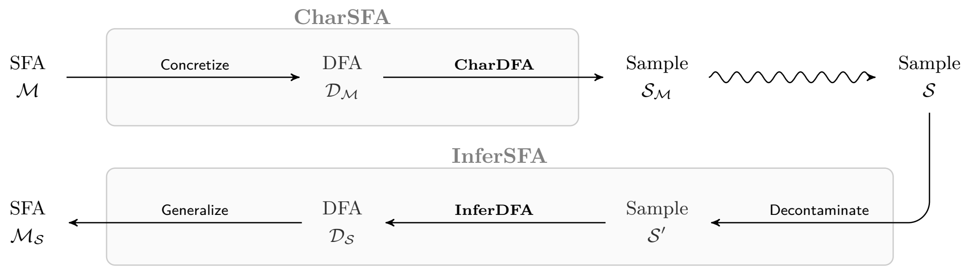

Figure 2 A schematic description of algorithms **CharSFA** and **InferSFA**

Input: An SFA $\mathcal{M}$, function $\mathsf{Concretize}_{\mathcal{A}}$
Output: a DFA $\mathcal{D}_{\mathcal{M}}$

function $\textsc{Concretize}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{M} = \langle \mathcal{A}, Q, q_\iota, F, \Delta \rangle)$
  $\Gamma_{\mathcal{M}} := \bigcup_{q \in Q} \mathsf{Concretize}_{\mathcal{A}}(\pi_q)$
  $\Delta_{\mathcal{D}} := \emptyset$
  for all $q, q' \in Q, \psi \in \pi_q, d \in \Gamma_{\mathcal{M}}$ **do**
    if $\langle q, \psi, q' \rangle \in \Delta$ and $d \in [\![\psi]\!]$ **then**
      $\Delta_{\mathcal{D}} := \Delta_{\mathcal{D}} \cup \langle q, d, q' \rangle$
  return $\mathcal{D}_{\mathcal{M}} := \langle \Gamma_{\mathcal{M}}, Q, q_\iota, F, \Delta_{\mathcal{D}} \rangle$

Algorithm 1 $\mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{M})$

Input: A DFA $\mathcal{D}$, function $\mathsf{Generalize}_{\mathcal{A}}$
Output: An SFA $\mathcal{M}$

function $\textsc{Generalize}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{D} = \langle \Sigma, Q, q_\iota, F, \Delta_{\mathcal{D}} \rangle)$
  $\Delta_{\mathcal{M}} := \emptyset$
  for all $q \in Q$ **do**
    for all $q_i \in Q$ **do** $\Gamma_i := \{\gamma \mid \langle q, \gamma, q_i \rangle \in \Delta_{\mathcal{D}}\}$
    $\langle \psi_1, \dots, \psi_n \rangle := \mathsf{Generalize}_{\mathcal{A}}(\langle \Gamma_1, \dots, \Gamma_n \rangle)$
    for all $q_i \in Q$ **do** $\Delta_{\mathcal{M}} := \Delta_{\mathcal{M}} \cup \langle q, \psi_i, q_i \rangle$
  return $\mathcal{M} := \langle \mathcal{A}, Q, q_\iota, F, \Delta_{\mathcal{M}} \rangle$

Algorithm 2 $\mathsf{Generalize}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{M})$

induced by the DFA states. A subtle issue that we need to cope with is that inference should succeed also on samples subsuming the characteristic sample. The fact that this holds for inference of the DFA does not suffice, since we are guaranteed that the inference of the DFA will not be confused if the sample contains more labeled words, as long as the new words are over the same alphabet. In our case the alphabet of the sample can be a strict subset of the concrete letters $\mathbb{D}$ (and if $\mathbb{D}$ is infinite, this surely will be the case).[5] So we need an additional step to remove words from the given sample if they are not over the alphabet of the characteristic sample. We call a method implementing this $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$.

Formally, we first define the extension of $\mathsf{Concretize}_{\mathcal{A}}$ and $\mathsf{Generalize}_{\mathcal{A}}$ to automata instead of partitions, which we term $\mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}}}$ and $\mathsf{Generalize}_{\mathbb{M}_{\mathcal{A}}}$ (with $\mathbb{M}$ in the subscript).

- The formal definition of $\mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}}}$ is given in Alg.1. Let $\mathcal{M} = (\mathcal{A}, Q, q_\iota, F, \Delta)$ be an SFA. Then $\mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{M})$ is the DFA $\mathcal{D}_{\mathcal{M}} = (\Sigma, Q, q_\iota, F, \Delta_{\mathcal{D}})$ where $\Delta_{\mathcal{D}}$ is defined as follows. For each state $q \in Q$ let $\pi_q = \langle \psi_1, \dots, \psi_m \rangle$ be the predicate partition consisting of all predicates labeling a transition exiting q in $\mathcal{M}$. Intuitively, in $\mathcal{D}$, the outgoing transitions of each state q correspond to $\mathsf{Concretize}_{\mathcal{A}}(\pi_q)$. That is, let $\mathsf{Concretize}_{\mathcal{A}}(\pi_q) = \langle \Gamma_1, \dots, \Gamma_m \rangle$. Then, if $\langle q, \psi_i, q' \rangle \in \Delta$, then $\langle q, \gamma, q' \rangle \in \Delta_{\mathcal{D}}$ for every $\gamma \in \Gamma_i$.
- The formal definition of $\mathsf{Generalize}_{\mathbb{M}_{\mathcal{A}}}$ is given in Alg.2. Let $\mathcal{D} = (\Sigma, Q, q_\iota, F, \Delta_{\mathcal{D}})$ be a DFA. We define $\mathsf{Generalize}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{D})$ wrt. an algebra $\mathcal{A}$ as follows. Let $\mathcal{M} = (\mathcal{A}, Q, q_\iota, F, \Delta_{\mathcal{M}})$ where $\Delta_{\mathcal{M}}$ is defined as follows. For each state $q \in Q$ let $\langle \Gamma_1, \dots, \Gamma_m \rangle$ be the concrete partition consisting of letters labeling outgoing transitions from q. Note that $\langle \Gamma_1, \dots, \Gamma_m \rangle$ is a concrete partition, since $\mathcal{D}$ is a DFA. Let $\mathsf{Generalize}_{\mathcal{A}}(\langle \Gamma_1, \dots, \Gamma_m \rangle) = \langle \psi_1, \dots, \psi_m \rangle$. Then, $\langle q, \psi_i, q' \rangle \in \Delta_{\mathcal{M}}$ if Γ_i is the set of letters labeling transitions from q to q' in $\mathcal{D}$.

We are now ready to define the conditions the *decontaminating* function has to satisfy.

[5] In the full version of this paper we provide an example illustrating this problem for the class of SFAs over a monotonic algebra $\mathcal{A}_m$, for which respective methods $\mathsf{Concretize}_{\mathcal{A}_m}$ and $\mathsf{Generalize}_{\mathcal{A}_m}$ exist.

▶ **Definition 11.** *A function $f_d : 2^{(\mathbb{D}^* \times \{0,1\})} \to 2^{(\mathbb{D}^* \times \{0,1\})}$ is called* decontaminating *for a class of SFAs $\mathbb{M}$ and a respective $\mathsf{Concretize}_{\mathbb{M}}$ function if the following holds. Let $\mathcal{M} \in \mathbb{M}$ be an SFA, and $\mathcal{D} = \mathsf{Concretize}_{\mathbb{M}}(\mathcal{M})$. Let $\mathcal{S}_{\mathcal{D}} = \textbf{CharDFA}(\mathcal{D})$. Then, for every $\mathcal{S}' \supseteq \mathcal{S}_{\mathcal{D}}$ s.t. $\mathcal{S}'$ agrees with $\mathcal{M}$, it holds that $\mathcal{S}_{\mathcal{D}} \subseteq f_d(\mathcal{S}') \subseteq (\mathcal{S}' \cap \Gamma_{\mathcal{D}})$, where $\Gamma_{\mathcal{D}}$ is the alphabet of $\mathcal{S}_{\mathcal{D}}$.*

As before we say that f_d is *efficient* if it can be computed in polynomial time. We can now provide the sufficient condition.

▶ **Theorem 12.** *Let $\mathbb{M}_{\mathcal{A}}$ be a class of SFAs over a Boolean algebra $\mathcal{A}$. If there exist efficient functions $\mathsf{Concretize}_{\mathcal{A}}$ and $\mathsf{Generalize}_{\mathcal{A}}$ satisfying that*

$$\begin{array}{c}
\text{if } \mathsf{Concretize}_{\mathcal{A}}(\langle \psi_1, \ldots, \psi_m \rangle) = \langle \Gamma_1, \ldots, \Gamma_m \rangle \\
\text{and } \mathsf{Generalize}_{\mathcal{A}}(\langle \Gamma'_1, \ldots, \Gamma'_m \rangle) = \langle \varphi_1, \ldots, \varphi_m \rangle \\
\text{where } \Gamma_i \subseteq \Gamma'_i \text{ for every } 1 \leq i \leq m \\
\text{then } [\![\varphi_i]\!] = [\![\psi_i]\!] \text{ for every } 1 \leq i \leq m
\end{array}$$

and in addition there exists an efficient decontaminating function $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$, then the class $\mathbb{M}_{\mathcal{A}}$ is efficiently identifiable.

Given functions $\mathsf{Concretize}_{\mathcal{A}}$, $\mathsf{Generalize}_{\mathcal{A}}$ and $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$ for a class $\mathbb{M}_{\mathcal{A}}$ of SFAs over a Boolean algebra $\mathcal{A}$ meeting the criteria of Thm.12, we show that $\mathbb{M}_{\mathcal{A}}$ can be efficiently identified by providing two algorithms **CharSFA** and **InferSFA**, described bellow. These algorithms make use of the respective algorithms **CharDFA** and **InferDFA** guaranteed in Thm.6.I., as well as the methods provided by the theorem.

We briefly describe these two algorithms, and then turn to prove Thm.12. The algorithm **CharSFA** receives an SFA $\mathcal{M} \in \mathbb{M}$, and returns a characteristic sample for it. It does so by applying $\mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{M})$ (Alg.1) to construct a DFA $\mathcal{D}_{\mathcal{M}}$ and generating the sample $\mathcal{S}_{\mathcal{M}}$ using the algorithm **CharDFA** applied on the DFA $\mathcal{D}_{\mathcal{M}}$.

Algorithm **InferSFA**, given a sample set $\mathcal{S}$, if $\mathcal{S}$ subsumes a characteristic set of an SFA $\mathcal{M}$, returns an equivalent SFA. Otherwise it suffices with returning an SFA that agrees with the sample. First, it applies $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$ to find a subset $\mathcal{S}' \subseteq \mathcal{S}$ over the alphabet of the subsumed characteristic sample, if such a subsumed sample exists. Then it uses $\mathcal{S}'$ to construct a DFA by applying the inference algorithm **InferDFA** on $\mathcal{S}'$. From this DFA it constructs an SFA, $\mathcal{M}_{\mathcal{S}}$, by applying $\mathsf{Generalize}_{\mathbb{M}_{\mathcal{A}}}$ (Alg.2). If the resulting automaton disagrees with the given sample it resorts to returning the prefix-tree automaton. In brief,

- $\textbf{CharSFA}(\mathcal{M}) = \textbf{CharDFA}(\mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{M}))$
- $\textbf{InferSFA}(\mathcal{S}) = \begin{cases} \mathcal{M}_{\mathcal{S}} := \mathsf{Generalize}_{\mathbb{M}_{\mathcal{A}}}(\textbf{InferDFA}(\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{S}))) & \text{if } \mathcal{S} \subseteq \hat{\mathcal{L}}(\mathcal{M}_{\mathcal{S}}) \\ \text{The prefix-tree automaton of } \mathcal{S} & \text{otherwise} \end{cases}$

In §4.3 we provide methods $\mathsf{Concretize}_{\mathcal{A}}$, $\mathsf{Generalize}_{\mathcal{A}}$ and $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$ for SFAs over monotonic algebras, deriving their identification in the limit result. We now prove Thm.12.

Proof of Thm.12. Given functions $\mathsf{Concretize}_{\mathcal{A}}$, $\mathsf{Generalize}_{\mathcal{A}}$, and $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$, we show that the algorithms **CharSFA** and **InferSFA** satisfy the requirements of Def.5.

For the first condition, given that **CharDFA**, $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$ and $\mathsf{Generalize}_{\mathcal{A}}$ run in polynomial time, and that the prefix-tree automaton can be constructed in polynomial time, it is clear that so does **InferSFA**. In addition, the test performed in the definition of **InferSFA** ensures the output agrees with the sample.

For the second condition, note that the sample generated by **CharSFA** is polynomial in the size of $\mathcal{D}_{\mathcal{M}}$, from the correctness of **CharDFA**. In addition, since $\mathsf{Concretize}_{\mathcal{A}}$ is efficient, $\mathcal{D}_{\mathcal{M}}$ is polynomial in the size of $\mathcal{M}$, and thus $\mathcal{S}_{\mathcal{M}}$ generated by **CharSFA** is polynomial in

$\mathcal{M}$ as well. It is left to show that given $\mathcal{S}_{\mathcal{M}}$ is the concrete sample produced by **CharSFA** when running on an SFA $\mathcal{M}$, then when **InferSFA** runs on any sample $\mathcal{S} \supseteq \mathcal{S}_{\mathcal{M}}$ it returns an SFA for $\mathcal{L}(\mathcal{M})$. Since $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$ is a decontaminating function, and $\mathcal{S} \supseteq \mathcal{S}_{\mathcal{M}}$, the set $\mathcal{S}' = \mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}(\mathcal{S})$ satisfies $\mathcal{S}' \supseteq \mathcal{S}_{\mathcal{M}}$ and is only over the alphabet $\Gamma_{\mathcal{M}}$, which is the alphabet of the DFA $\mathcal{D}_{\mathcal{M}}$ generated in Alg.1.

From the correctness of **InferDFA**, given $\mathcal{S}' \supseteq \mathcal{S}_{\mathcal{M}}$, applying **InferDFA** on the output $\mathcal{S}'$ of $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}}}$ results in a DFA $\mathcal{D}$ that is equivalent to $\mathcal{D}_{\mathcal{M}}$ constructed in Alg.1. Since $\mathcal{D}_{\mathcal{M}}$ is complete wrt. its alphabet $\Gamma_{\mathcal{M}}$, for state q of $\mathcal{D}$, the concrete partition $\langle \Gamma_1, \ldots, \Gamma_n \rangle$ generated in Alg.2 line 4, covers $\Gamma_{\mathcal{M}}$ and subsumes the output of $\mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}}}$ on π_q (Alg.1, line 2). Thus, since $\mathsf{Generalize}_{\mathcal{A}}$ and $\mathsf{Concretize}_{\mathcal{A}}$ satisfy the criteria of Thm.12, it holds that the constructed predicates agree with the original predicates. In addition, since $\mathcal{S}$, and therefore $\mathcal{S}'$, agrees with $\mathcal{M}$, the test performed in the definition of **InferSFA** fails and the returned SFA is equivalent to $\mathcal{M}$. ◀

4.3 Positive Result

We present the following positive result regarding monotonic algebras.

▶ **Theorem 13.** *Let $\mathbb{M}_{\mathcal{A}_m}$ be the set of SFAs over a monotonic Boolean algebra $\mathcal{A}_m$. Then $\mathbb{M}_{\mathcal{A}_m}$ is efficiently identifiable.*

In order to prove Thm.13, we show that the sufficient condition holds for the case of monotonic algebras. In the full version we provide an example that demonstrates how to apply **CharSFA** and **InferSFA** in order to learn an SFA over the algebra $\mathcal{A}_{\mathbb{N}}$.

▶ **Proposition 14.** *There exist functions $\mathsf{Concretize}_{\mathcal{A}_m}$ and $\mathsf{Generalize}_{\mathcal{A}_m}$ for a monotonic Boolean algebra $\mathcal{A}_m$, satisfying the criteria of Thm.12.*

Proof. Let $\mathbb{D}$ be the domain of $\mathcal{A}_m$. We provide the functions $\mathsf{Concretize}_{\mathcal{A}_m}$ and $\mathsf{Generalize}_{\mathcal{A}_m}$ and prove that the criteria of Thm.12 hold for them. For ease of presentation, for the function $\mathsf{Concretize}$ we consider basic predicates. Note that for monotonic algebras, basic predicates are in fact intervals, as a conjunction of intervals is an interval. We can assume all predicates are basic since, as we show in [27, Lemma 3], for monotonic algebras the transformation from a general formula to a DNF formula of basic predicates is linear. Then, each basic predicate in the formula corresponds to a different predicate in the predicate partition. The definitions of $\mathsf{Concretize}_{\mathcal{A}_m}$ and $\mathsf{Generalize}_{\mathcal{A}_m}$ are generalizations of the functions $\mathsf{Concretize}_{\mathcal{A}_{\mathbb{N}}}$ and $\mathsf{Generalize}_{\mathcal{A}_{\mathbb{N}}}$ given in Ex.9. We define $\mathsf{Concretize}_{\mathcal{A}_m}(\langle \psi_1, \ldots \psi_m \rangle) = \langle \Gamma_1, \ldots, \Gamma_m \rangle$ where we set $\Gamma_i = \{\ \min\{d \in \mathbb{D} \mid d \in [\![\psi_i]\!]\}\}$ for $1 \leq i \leq m$. Since $\mathcal{A}_m$ is monotonic, Γ_i is well defined and contains a single element, thus $\mathsf{Concretize}_{\mathcal{A}_m}$ is an efficient concretizing function.

We define $\mathsf{Generalize}_{\mathcal{A}_m}(\langle \Gamma_1, \ldots, \Gamma_m \rangle) = \langle \psi_1, \ldots, \psi_m \rangle$, where ψ_i is defined as follows. Let $\Gamma = \bigcup_{1 \leq i \leq m} \Gamma_i$. First, for all $1 \leq i \leq m$ we set $\psi_i = \bot$. Then, we iteratively look for the minimal element $\gamma \in \Gamma$. Let i be such that $\gamma \in \Gamma_i$, and let γ' be the minimal element in Γ satisfying $\gamma' \notin \Gamma_i$. We then set $\psi_i = \psi_i \vee [\gamma, \gamma')$, and remove all elements $\gamma \leq \gamma'' < \gamma'$ from Γ. We repeat the process until for the found $\gamma \in \Gamma_j$, there is no $\gamma' > \gamma$ such that $\gamma' \notin \Gamma_j$. In that case, we define $\psi_j = \psi_j \vee [\gamma, d_\infty)$. Then, $\Gamma_i \subseteq [\![\psi_i]\!]$ and the predicates are disjoint, thus $\mathsf{Generalize}_{\mathcal{A}_m}$ is an efficient generalizing function.

Now, let $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ be the concrete partition obtained from $\mathsf{Concretize}_{\mathcal{A}_m}$ when applied on the predicate partition $\langle \psi_1, \ldots, \psi_m \rangle$. Assume further that the predicate partition $\langle \Gamma'_1, \ldots, \Gamma'_m \rangle$ satisfies $\Gamma_i \subseteq \Gamma'_i \subseteq [\![\psi_i]\!]$ for $1 \leq i \leq m$. In particular, $\min(\Gamma'_i) = \min(\Gamma_i)$, since Γ_i contains the minimal elements in $[\![\psi_i]\!]$, and $\Gamma_i \subseteq \Gamma'_i \subseteq [\![\psi_i]\!]$. Thus applying $\mathsf{Generalize}_{\mathcal{A}_m}$ will result in the same interval, satisfying the criterion of Thm.12. ◀

Input: set $\mathcal{S}$ over alphabet Σ
Output: set $\mathcal{S}'$ over alphabet Σ'

1 **function** $\textsc{Decontaminate}_{\mathbb{M}_{\mathcal{A}_m}}(\mathcal{S})$
2 $\quad A_w := \{\epsilon\}$, $\Sigma' := \{d_{inf}\}$, $\sigma_{max} := d_{inf}$
3 $\quad$ **repeat**
4 $\quad\quad$ **for all** $u \in A_w$, by lexicographic order **do**
5 $\quad\quad\quad$ **for all** $\sigma \in \Sigma$, by lexicographic order **do**
6 $\quad\quad\quad\quad$ **if** $\sigma > \sigma_{max}$ and $u\sigma \not\sim_{\mathcal{S}} u\sigma_{max}$ **then**
7 $\quad\quad\quad\quad\quad$ **if** $\forall \sigma'.\ \sigma_{max} < \sigma' < \sigma :\ u\sigma' \sim_{\mathcal{S}} u\sigma_{max}$ **then**
8 $\quad\quad\quad\quad\quad\quad \Sigma' := \Sigma' \cup \{\sigma\}$
9 $\quad\quad\quad\quad\quad\quad$ **if** $\forall u' \in A_w.\ u\sigma \not\sim_{\mathcal{S}} u'$ **then** $A_w := A_w \cup \{u\sigma\}$
10 $\quad\quad\quad\quad \sigma_{max} := \sigma$
11 $\quad\quad \sigma_{max} := d_{inf}$
12 $\quad$ **until** Σ' is remained unchanged
13 $\quad$ **return** $\mathcal{S}' := \mathcal{S} \cap \Sigma'^*$

Algorithm 3 $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}_m}}$ – finding the necessary letters for a characteristic sample

▶ **Example 15.** Let $\Gamma_1 = \{0, 100, 400, 500\}$ and $\Gamma_2 = \{150, 200\}$ over the algebra $\mathcal{A}_{\mathbb{N}}$ with domain $\mathbb{N} \cup \{\infty\}$. Then, $\mathsf{Generalize}_{\mathcal{A}_{\mathbb{N}}}$ sets $\Gamma = \{0, 100, 150, 200, 400, 500\}$, and finds the minimal element in γ which is 0. Since $0 \in \Gamma_1$, it then looks for the minimal element $\gamma \in \Gamma$ such that $\gamma \notin \Gamma_1$, and finds $150 \in \Gamma_2$. Therefore $\psi_1 = [0, 150)$ and Γ is updated to be $\Gamma = \{150, 200, 400, 500\}$. Next, it finds the minimal element, which is 150 and is in Γ_2, and the minimal element that is not in Γ_2 is 400. Then, ψ_2 is set to be $\psi_2 = [150, 400)$ and $\Gamma = \{400, 500\}$. Last, $\psi_1 = [0, 150) \vee [400, \infty)$ since $400 \in \Gamma_1$ and there is no greater element that is not in Γ_1.

To show that any class of SFAs $\mathbb{M}_{\mathcal{A}_m}$ over a monotonic algebra $\mathcal{A}_m$ is efficiently identifiable, we define in Alg.3 an algorithm that implements a decontaminating function $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}_m}}$, fulfilling the requirements of Thm.12. Loosely speaking, the idea of the algorithm is to simultaneously collect elements into two sets A_w and Σ' s.t. A_w will consist of the minimal representative according to the lexicographic order of each equivalence class in $\sim_{\mathcal{S}}$ and Σ' will consist of minimal letters aiding to distinguishing these words. When this process terminates the algorithm returns the subset of words in the sample that consist of only letters in Σ'.

▶ **Lemma 16.** *Assume the input to $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}_m}}$ is $\mathcal{S}$ with $\mathcal{S} \supseteq \mathcal{S}_{\mathcal{M}}$ for some $\mathcal{M} \in \mathbb{M}_{\mathcal{A}_m}$ s.t. $\mathcal{S}_{\mathcal{M}} = \textbf{CharDFA}(\mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}_m}}(\mathcal{M}))$, and $\mathcal{D}_{\mathcal{M}} = \mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}_m}}(\mathcal{M})$ is over alphabet $\Gamma_{\mathcal{M}}$. Then for Σ' constructed by $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}_m}}$ (Alg.3) it holds that $\Sigma' = \Gamma_{\mathcal{M}}$.*

Proof sketch. Let $\mathcal{M} = (\mathcal{A}, Q, q_\iota, F, \Delta_{\mathcal{M}})$, $\mathcal{D}_{\mathcal{M}} = \mathsf{Concretize}_{\mathbb{M}_{\mathcal{A}_m}}(\mathcal{M})$ where $\mathcal{D}_{\mathcal{M}} = (\Gamma_{\mathcal{M}}, Q, q_\iota, F, \Delta_{\mathcal{D}})$, and $\mathcal{S}_{\mathcal{M}} = \textbf{CharDFA}(\mathcal{D}_{\mathcal{M}})$. We inductively show that for $\mathsf{Decontaminate}_{\mathbb{M}_{\mathcal{A}_m}}$ given in Alg.3, if its input $\mathcal{S}$ satisfies $\mathcal{S} \supseteq \mathcal{S}_{\mathcal{M}}$ then the set A_w is exactly the set of all lex-access words of states in $\mathcal{D}_{\mathcal{M}}$ and that $\Sigma' = \Gamma_{\mathcal{M}}$ (where $\Gamma_{\mathcal{M}}$ is the alphabet of $\mathcal{D}_{\mathcal{M}}$).

First, we show that every $u \in A_w$ is a lex-access word and that $\Sigma' \subseteq \Gamma_{\mathcal{M}}$. For the base case, we have $A_w = \{\epsilon\}$ and $\Sigma' = \{d_{-\infty}\}$. Since ϵ is the minimal element in the lexicographic order, it holds that $\epsilon \in A_w$ is indeed a lex-access word (of the state q_ι). For $d_{-\infty} \in \Sigma'$, since $\mathsf{Concretize}_{\mathcal{A}_m}$ returns the minimal element of each interval, it holds that $d_{-\infty} \in \Gamma_{\mathcal{M}}$.

For the induction step, assume that A_w contains only lex-access words and that the current Σ' is a subset of $\Gamma_{\mathcal{M}}$. Then, when considering $u \in A_w$ in line 4, it holds that u is a lex-access word of some state q. Then, if σ is added to Σ' it must be a minimal element of some interval labeling an outgoing transition from q, thus it is in $\Gamma_{\mathcal{M}}$, and hence $\Sigma' \subseteq \Gamma_{\mathcal{M}}$. Let $u\sigma$ be a word added to A_w in line 9. Thus, for all $u' \in A_w$ it holds that $u\sigma \not\sim_{\mathcal{S}} u'$.

Claim. In this setting, $u\sigma \not\sim_{\mathcal{S}} u'$ implies $u\sigma \not\sim_{\mathcal{S}_{\mathcal{M}}} u'$.
See the full version for a detailed proof of the lemma, and in particular, a proof of this claim.

Then, for all $u' \in A_w$ we have $\Delta_{\mathcal{D}}(q_\iota, u\sigma) \neq \Delta_{\mathcal{D}}(q_\iota, u')$ where $\Delta_{\mathcal{D}}$ is the transition relation of $\mathcal{D}_{\mathcal{M}}$. Since u is a lex-access word and σ is minimal, $u\sigma$ is a lex-access word for $\Delta_{\mathcal{D}}(q_\iota, u\sigma)$. This concludes the first direction.

For the second direction, we show that every lex-access word is in A_w and that $\Gamma_{\mathcal{M}} \subseteq \Sigma'$. The lex-access word ϵ is in A_w. Let $u\sigma$ be a lex-access word. For all lex-access words u' found in previous iterations it holds that $u\sigma \not\sim_{\mathcal{S}_{\mathcal{M}}} u'$ from item 2 of Thm.6.II, and thus $u\sigma \not\sim_{\mathcal{S}} u'$ since $\mathcal{S}_{\mathcal{M}} \subseteq \mathcal{S}$. Thus, $u\sigma$ satisfies the condition of line 9 in Alg.3 and is added to A_w. For $\Gamma_{\mathcal{M}} \subseteq \Sigma'$, let $\sigma \in \Gamma_{\mathcal{M}}$. From the construction of $\mathsf{Concretize}_{\mathscr{A}_m}$ it holds that σ is the left endpoint of some interval that is an outgoing transition from q_ι. Then, indeed σ is found in the first iteration of line 4. Inductively, since A_w contains all lex-access words, for every state q, the outgoing transitions of q will be considered in some following iteration. Thus, all minimal letters indicating new intervals are added to Σ' and we have that $\Gamma_{\mathcal{M}} \subseteq \Sigma'$. ◀

▶ **Proposition 17.** *The sufficient condition of Thm.12 holds for the class $\mathbb{M}_{\mathscr{A}_m}$ of SFAs over a monotonic Boolean algebra $\mathscr{A}_m$.*

Proof. In Prop.14 we have shown that there exist functions $\mathsf{Concretize}_{\mathscr{A}_m}$ and $\mathsf{Generalize}_{\mathscr{A}_m}$ for a monotonic Boolean algebra $\mathscr{A}_m$, satisfying the criteria of Thm.12. It is left to show that $\mathsf{Decontaminate}_{\mathbb{M}_{\mathscr{A}_m}}$ is an efficient decontaminating function. Assume that $\mathcal{S} \supseteq \mathcal{S}_{\mathcal{M}}$ where $\mathcal{S}_{\mathcal{M}} = \mathbf{CharDFA}(\mathsf{Concretize}_{\mathbb{M}_{\mathscr{A}_m}}(\mathcal{M}))$, and $\mathsf{Concretize}_{\mathbb{M}_{\mathscr{A}_m}}(\mathcal{M})$ is over alphabet $\Gamma_{\mathcal{M}}$. In Lemma 16 we showed that under these assumptions it holds that the alphabet Σ' of the returned sample $\mathcal{S}'$ is $\Gamma_{\mathcal{M}}$. Then, for the set $\mathcal{S}'$ returned in line 13 (Alg.3) it holds that $\mathcal{S}' = \mathcal{S} \cap \Gamma^*_{\mathcal{M}}$. Since $\mathcal{S} \supseteq \mathcal{S}_{\mathcal{M}}$ and $\Gamma^*_{\mathcal{M}} \supseteq \mathcal{S}_{\mathcal{M}}$, it holds that $\mathcal{S}' \supseteq \mathcal{S}_{\mathcal{M}}$ and $\mathcal{S}'$ is defined over the alphabet $\Gamma_{\mathcal{M}}$. Therefore, $\mathsf{Decontaminate}_{\mathbb{M}_{\mathscr{A}_m}}$ is a decontaminating function. In addition, it runs in time polynomial in the size of $\mathcal{S}$, thus the conditions of Thm.12 are met. ◀

4.4 Negative Result

The result of Thm.13 does not extend to the non-monotonic case, as stated in Thm.18 regarding SFAs over the general propositional algebra. Let $\mathbb{D}_{\mathbb{B}} = \{\mathbb{B}^k\}_{k\in\mathbb{N}}$. Let $\mathbb{P}_{\mathbb{B}} = \{\mathbb{P}_{\mathbb{B}_k}\}_{k\in\mathbb{N}}$ where $\mathbb{P}_{\mathbb{B}_k}$ is the set of predicates over at most k variables. Let $\mathscr{A}_{\mathbb{B}}$ be the Boolean algebra defined over the discrete domain $\mathbb{D}_{\mathbb{B}}$ and the set of predicates $\mathbb{P}_{\mathbb{B}}$, and the usual operators $\vee$, $\wedge$ and $\neg$. Let $\mathbb{M}_{\mathscr{A}_{\mathbb{B}}}$ be the class of SFAs over the Boolean algebra $\mathscr{A}_{\mathbb{B}}$. We show that unless $P = NP$, this class of SFAs is not efficiently identifiable.

▶ **Theorem 18.** *The class $\mathbb{M}_{\mathscr{A}_{\mathbb{B}}}$ is not efficiently identifiable unless $P = NP$.*

Proof. We show that there is no pair of efficient concretizing and generalizing functions $f_c : \Pi_{\mathsf{pred}}(\mathbb{P}_{\mathbb{B}}, 2) \to \Pi_{\mathsf{conc}}(\mathbb{D}_{\mathbb{B}}, 2)$ and $f_g : \Pi_{\mathsf{conc}}(\mathbb{D}_{\mathbb{B}}, 2) \to \Pi_{\mathsf{pred}}(\mathbb{P}_{\mathbb{B}}, 2)$ unless $P = NP$. From Thm.8 it follows that $\mathbb{M}_{\mathbb{B}}$ is not efficiently identifiable unless $P = NP$.

Assume towards contradiction that such a pair of functions exist. We provide a polynomial time algorithm $\mathsf{A}_{\mathsf{SAT}}$ for SAT. On predicate φ, the algorithm $\mathsf{A}_{\mathsf{SAT}}$ invokes $f_c(\langle\varphi, \neg\varphi\rangle)$. Suppose the returned concrete partition is $\langle\Gamma_1, \Gamma_2\rangle$. Then $\mathsf{A}_{\mathsf{SAT}}$ returns "true" if and only if $\Gamma_1 \neq \emptyset$. Correctness follows from the fact that if there exists a system of characteristic samples for $\mathbb{P}_{\mathbb{B}}$ then the set of positive examples associated with a satisfiable predicate φ must be non-empty, as otherwise f_g cannot distinguish φ from $\bot$. ◀

5 Query Learning

The paradigm of *query learning* stipulates that the *learner* can interact with an *oracle* (*teacher*) by asking it several types of allowed queries. Angluin showed, on the negative side, that regular languages cannot be learned (in the exact model) from only *membership queries* (MQ) [3] or only *equivalence queries* (EQ) [6]. On the positive side, she showed that regular languages, represented as DFAs, can be learned using both MQ and EQ [4]. The celebrated algorithm, termed $\mathbf{L}^*$, was extended to learning many other classes of languages and representations, e.g. [46, 15, 1, 16, 7, 38, 8, 41], see the survey [26] for more references.

In particular, an extension of $\mathbf{L}^*$, termed $\mathbf{MAT}^*$, to learn SFAs was provided in [9] which proved that SFAs over an algebra $\mathcal{A}$ can be efficiently learned using $\mathbf{MAT}^*$ if and only if the underlying algebra is efficiently learnable, and the size of disjunctions of k predicates doesn't grow exponentially in k.[6] From this it was concluded that SFAs over the following underlying algebras are efficiently learnable: Boolean algebras over finite domains, equality algebra, tree automata algebra, and SFAs algebra. Efficient learning of SFAs over a monotonic algebra using MQ and EQ was established in [19], which improved the results of [36, 37] by using a binary search instead of a helpful teacher.

The result of [9] provides means to establish new positive results on learning classes of SFAs using MQ and EQ, but it does not provide means for obtaining negative results for query learning of SFAs using MQ and EQ. We strengthen this result by providing a learnability result that is independent of the use of a specific learning algorithm. In particular, we show that efficient learnability of a Boolean algebra $\mathcal{A}$ using MQ and EQ is a necessary condition for the learnability of the class of SFAs over $\mathcal{A}$, as we state in Thm. 19.

▶ **Theorem 19.** *A non-trivial class of SFAs $\mathbb{M}$ over a Boolean algebra $\mathcal{A}$ is polynomially learnable using* MQ *and* EQ*, only if $\mathcal{A}$ is polynomially learnable using* MQ *and* EQ*.*

Proof. Assume that $\mathbb{M}$ is polynomially learnable using MQ and EQ, using an algorithm $\mathbf{Q}_{\mathbb{M}}$. We show that there exists a polynomial learning algorithm $\mathbf{Q}_{\mathcal{A}}$ for the algebra $\mathcal{A}$ using MQ and EQ. The algorithm $\mathbf{Q}_{\mathcal{A}}$ uses $\mathbf{Q}_{\mathbb{M}}$ as a subroutine, and behaves as a teacher for $\mathbf{Q}_{\mathbb{M}}$. Whenever $\mathbf{Q}_{\mathbb{M}}$ asks a $\mathbb{M}$-MQ on word $\gamma_1 \dots \gamma_k$, if $k > 1$ then $\mathbf{Q}_{\mathcal{A}}$ answers "no". If $k=1$ then the $\mathbb{M}$-MQ is essentially an $\mathcal{A}$-MQ, thus $\mathbf{Q}_{\mathcal{A}}$ issues this query and passes the answer to $\mathbf{Q}_{\mathbb{M}}$. Whenever $\mathbf{Q}_{\mathbb{M}}$ asks a $\mathbb{M}$-EQ on SFA $\mathcal{M}$, if $\mathcal{M}$ is of the form $\mathcal{M}_\psi$ for some ψ (as defined in Def.2) then $\mathbf{Q}_{\mathcal{A}}$ answers "no" to the $\mathbb{M}$-EQ and returns some word $w \in \mathcal{L}(\mathcal{M})$ s.t. $|w| > 1$ and w was not provided before, as a counterexample. Otherwise (if the SFA is of the form $\mathcal{M}_\psi$ for some ψ) $\mathbf{Q}_{\mathcal{A}}$ asks an $\mathcal{A}$-EQ on ψ. If the answer is "yes" then $\mathbf{Q}_{\mathcal{A}}$ terminates and returns ψ as the result of the learning algorithm; if the answer to the $\mathcal{A}$-EQ on ψ is "no", then the provided counterexample $\langle \gamma, b_\gamma \rangle$ is passed back to $\mathbf{Q}_{\mathbb{M}}$ together with the answer "no" to the $\mathbb{M}$-EQ. It is easy to verify that $\mathbf{Q}_{\mathcal{A}}$ terminates correctly in polynomial time. ◀

From Thm. 19 we derive what we believe to be the first negative result on learning SFAs from MQ and EQ, as we show that SFAs over the propositional algebra over k variables $\mathcal{A}_{\mathbb{B}_k}$ are not polynomially learnable using MQ and EQ. Polynomiality is measured with respect to the parameters $\langle n, m, l \rangle$ representing the size of the SFA and the number k of atomic propositions. Note that the algebra $\mathcal{A}_{\mathbb{B}_k}$ is a restriction of the algebra $\mathcal{A}_{\mathbb{B}}$ considered in §.4.4 and therefore implies a negative result also with regard to the algebra $\mathcal{A}_{\mathbb{B}}$ considered there.

[6] As is the case, for instance, in the OBDD (Ordered Binary Decisions Diagrams) algebra [17].

We achieve this by showing that no learning algorithm **A** for the propositional algebra using MQ and EQ can do better than asking 2^k MQ/EQ, where k is the number of atomic propositions.[7] We assume the learning algorithm is *sound*, that is, if S_i^+ and S_i^- are the sets of positive and negative examples observed by the algorithm up to stage i, then at stage $i+1$ the algorithm will not ask a MQ for a word in $S_i^+ \cup S_i^-$ or an EQ for an automaton that rejects a word in S_i^+ or accepts a word in S_i^-.

▶ **Proposition 20.** *Let **A** be a sound learning algorithm for the propositional algebra over $\mathbb{B}^k$. There exists a target predicate ψ of size k, for which **A** will be forced to ask at least $2^k - 1$ queries (either MQ or EQ).*

Proof. Since **A** is sound, at stage $i+1$ we have $S_{i+1}^+ \supseteq S_i^+$ and $S_{i+1}^- \supseteq S_i^-$ and at least one inclusion is strict. Since the size of the concrete alphabet is 2^k, for every round $i < 2^k$, an adversarial teacher can answer both MQ and EQ negatively. In the case of EQ there must be an element in $\mathbb{B}^k \setminus (S_i^- \cup S_i^+)$ with which the provided automaton disagrees. The adversary will return one such element as a counterexample. This forces **A** to ask at least 2^k-1 queries. Note that for any element v in $\mathbb{B}^k$ there exists a predicate φ_v of size k such that $[\![\varphi_v]\!] = \{v\}$. ◀

▶ **Corollary 21.** *SFAs over the propositional algebra $\mathcal{A}_{\mathbb{B}_k}$ with k propositions cannot be learned in poly(k) time using MQ and EQ.*

The propositional algebra $\mathcal{A}_{\mathbb{B}_k}$ is a special case of the n-dimensional boxes algebra. Learning n-dimensional boxes was studied using MQ and EQ [29, 18, 12], as well as in the PAC setting [13]. The algorithms presented in [29, 18, 12, 13] are mostly exponential in n. Alternatively, [29, 18] suggest algorithms that are exponential in the number of boxes in the union. In [12] a linear query learning algorithm for unions of disjoint boxes is presented. Since n-dimensional boxes subsume the propositional algebra, Corollary 21 implies the following.

▶ **Corollary 22.** *The class of SFAs over the n-dimensional boxes algebra cannot be learned in poly(n) time using MQ and EQ.*

6 Discussion

We examine the question of learnability of a class of SFAs over certain algebras where the main focus of our study is on passive learning. We provide a necessary condition for identification of SFAs in the limit using polynomial time and data, as well as a necessary condition for efficient learning of SFAs using MQ and EQ. We note that a positive result on learning SFAs using MQ and EQ implies a positive result for identification of SFAs in the limit using polynomial time and data. The latter follows because a systematic set of characteristic samples $\{S_L\}_{L\in\mathbb{L}}$ for a class of languages $\mathbb{L}$ may be obtained by collecting the words observed by the query learner when learning L. However, it does not imply a positive result regarding the stronger notion of *efficient identifiability*, as the latter requires the set to be also constructed efficiently. We thus provide a sufficient condition for *efficient identification* of a class of SFAs, and show that the class of SFAs over any monotonic algebra satisfies these conditions.

We hope that these sufficient or necessary conditions will help to obtain more positive and negative results for learning of SFAs, and spark an interest in investigating characteristic samples in other automata models used in verification.

[7] In [40] Boolean formulas represented using OBDDs are claimed to be polynomially learnable with MQ and EQ. However, [40] measures the size of an OBDD by its number of nodes, which can be exponential in the number of propositions.

References

1. F. Aarts and F. Vaandrager. Learning I/O automata. In *Concurrency Theory, 21th Int. Conf., CONCUR 2010*, pages 71–85, 2010.
2. R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theor. Comput. Sci.*, 211(1-2):253–273, 1999.
3. D. Angluin. A note on the number of queries needed to identify regular languages. *Inf. Control.*, 51(1):76–87, 1981.
4. D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.
5. D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
6. D. Angluin. Negative results for equivalence queries. *Mach. Learn.*, 5:121–150, 1990.
7. D. Angluin, S. Eisenstat, and D. Fisman. Learning regular languages via alternating automata. In *Proc. of the Twenty-Fourth Int. Joint Conf. on Artificial Intelligence, IJCAI*, pages 3308–3314, 2015.
8. D. Angluin and D. Fisman. Learning regular omega languages. *Theor. Comput. Sci.*, 650:57–72, 2016.
9. G. Argyros and L. D'Antoni. The learnability of symbolic automata. In *Computer Aided Verification - 30th Int. Conf., CAV*, volume 10981 of *LNCS*, pages 427–445. Springer, 2018.
10. G. Argyros, I. Stais, S. Jana, A. D. Keromytis, and A. Kiayias. Sfadiff: Automated evasion attacks and fingerprinting using black-box differential automata learning. In *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security*, pages 1690–1701. ACM, 2016.
11. G. Argyros, I. Stais, A. Kiayias, and A. D. Keromytis. Back in black: Towards formal, black box analysis of sanitizers and filters. In *IEEE Symposium on Security and Privacy, SP*, pages 91–109, 2016.
12. A. Beimel and E. Kushilevitz. Learning boxes in high dimension. *Algorithmica*, 22(1/2):76–90, 1998.
13. A. Beimel and E. Kushilevitz. Learning unions of high-dimensional boxes over the reals. *Inf. Process. Lett.*, 73(5-6):213–220, 2000.
14. T. Berg, O. Grinchtein, B. Jonsson, M. Leucker, H. Raffelt, and B. Steffen. On the correspondence between conformance testing and regular inference. In *Fundamental Approaches to Software Engineering, 8th Int. Conf., FASE*, volume 3442, pages 175–189. Springer, 2005.
15. F. Bergadano and S. Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM J. Comput.*, 25(6):1268–1280, 1996.
16. B. Bollig, P. Habermehl, C. Kern, and M. Leucker. Angluin-style learning of NFA. In *IJCAI*, pages 1004–1009, 2009.
17. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.
18. N. H. Bshouty, P. W. Goldberg, S. A. Goldman, and H. D. Mathias. Exact learning of discretized geometric concepts. *SIAM J. Comput.*, 28(2):674–699, 1998.
19. K. Chubachi, Diptarama, R. Yoshinaka, and A. Shinohara. Query learning of regular languages over large ordered alphabets. In *1st Workshop on Learning and Automata (LearnAut)*, 2017.
20. K. Chubachi, D. Hendrian, R. Yoshinaka, and A. Shinohara. Query learning algorithm for residual symbolic finite automata. In *Proc. Tenth Int. Symposium on Games, Automata, Logics, and Formal Verification, GandALF*, pages 140–153, 2019.
21. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 2001.
22. L. D'Antoni and M. Veanes. Minimization of symbolic tree automata. In *Proc. of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, pages 873–882. ACM, 2016.
23. L. D'Antoni, M. Veanes, B. Livshits, and D. Molnar. Fast: a transducer-based language for tree manipulation. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation, PLDI*, pages 384–394. ACM, 2014.

24 C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138, 1997.

25 S. Drews and L. D'Antoni. Learning symbolic automata. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd Int. Conf., TACAS*, pages 173–189, 2017.

26 D. Fisman. Inferring regular languages and ω-languages. *J. Log. Algebraic Methods Program.*, 98:27–49, 2018.

27 D. Fisman, H. Frenkel, and S. Zilles. On the complexity of symbolic finite-state automata, 2021. `arXiv:2011.05389`.

28 E. M. Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978.

29 P. W. Goldberg, S. A. Goldman, and H. D. Mathias. Learning unions of boxes with membership and equivalence queries. In *Proc. of the Seventh Annual ACM Conf. on Computational Learning Theory, COLT*, pages 198–207. ACM, 1994.

30 S. A. Goldman and H. D. Mathias. Teaching a smarter learner. *J. Comput. Syst. Sci.*, 52(2):255–267, 1996.

31 O. Grinchtein, B. Jonsson, and M. Leucker. Learning of event-recording automata. *Theor. Comput. Sci.*, 411(47):4029–4054, 2010.

32 P. Hooimeijer, B. Livshits, D. Molnar, P. Saxena, and M. Veanes. Fast and precise sanitizer analysis with BEK. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proc.* USENIX Association, 2011.

33 F. Howar, B. Steffen, and M. Merten. Automata learning with automated alphabet abstraction refinement. In *Verification, Model Checking, and Abstract Interpretation - 12th Int. Conf., VMCAI*, volume 6538 of *LNCS*, pages 263–277. Springer, 2011.

34 Q. Hu and L. D'Antoni. Automatic program inversion using symbolic transducers. In *Proc. of the 38th ACM SIGPLAN Conf. on Programming Language Design and Implementation, PLDI*, pages 376–389. ACM, 2017.

35 M. Keil and P. Thiemann. Symbolic solving of extended regular expression inequalities. In *34th Int. Conf. on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, pages 175–186, 2014.

36 O. Maler and I. Mens. Learning regular languages over large alphabets. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th Int. Conf., TACAS*, volume 8413 of *LNCS*, pages 485–499. Springer, 2014.

37 O. Maler and I. Mens. A generic algorithm for learning symbolic automata from membership queries. In *Models, Algorithms, Logics and Tools - Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, pages 146–169, 2017.

38 O. Maler and A. Pnueli. On the learnability of infinitary regular sets. *Inf. Comput.*, 118(2):316–326, 1995.

39 K. Mamouras, M. Raghothaman, R. Alur, Z. G. Ives, and S. Khanna. StreamQRE: modular specification and efficient evaluation of quantitative queries over streaming data. In *Proc. of the 38th ACM SIGPLAN Conf. on Prog. Lang. Design and Impl., PLDI*, pages 693–708, 2017.

40 A. Nakamura. Query learning of bounded-width obdds. *Theor. Comput. Sci.*, 241(1-2):83–114, 2000.

41 D. Nitay, D. Fisman, and M. Ziv-Ukelson. Learning of structurally unambiguous probabilistic grammars. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 9170–9178, 2021. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/17107`.

42 J. Oncina and P. García. Inferring regular languages in polynomial update time. *World Scientific*, 01 1992. `doi:10.1142/9789812797902\_0004`.

43 L. Pitt. Inductive inference, DFAs, and computational complexity. In *Proc. Int. Workshop on Analogical and Inductive Inference*, pages 18–44, 1989.

44 M. D. Preda, R. Giacobazzi, A. Lakhotia, and I. Mastroeni. Abstract symbolic automata: Mixed syntactic/semantic similarity analysis of executables. In *Proc. of the 42nd Annual ACM SIGPLAN-SIGACT Symp. on Princ. of Prog. Lang., POPL*, pages 329–341, 2015.

45 O. Saarikivi and M. Veanes. Translating c# to branching symbolic transducers. In *IWIL@LPAR 2017 Workshop and LPAR-21 Short Presentations*, volume 1 of *Kalpa Publications in Computing*. EasyChair, 2017.

46 Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theor. Comput. Sci.*, 76(2-3):223–242, 1990.

47 S. Sheinvald. Learning deterministic variable automata over infinite alphabets. In *Formal Methods - The Next 30 Years - Third World Congress, FM*, volume 11800 of *LNCS*, pages 633–650. Springer, 2019.

48 Frits W. Vaandrager. Model learning. *Commun. ACM*, 60(2):86–95, 2017. doi:10.1145/2967606.

49 M. Veanes, P. de Halleux, and N. Tillmann. Rex: Symbolic regular expression explorer. In *Third Int. Comf, on Software Testing, Verification and Validation, ICST*, pages 498–507. IEEE Computer Society, 2010.

50 X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty. An overview of machine teaching. *CoRR ArXiv*, abs/1801.05927, 2018.