Learning DFAs from Positive Examples Only via Word Counting

Benjamin Bordais[1,2,3] and Daniel Neider[1,2]

[1]TU Dortmund University, Dortmund, Germany
[2]Center for Trustworthy Data Science and Security, University Alliance Ruhr, Dortmund, Germany
[3]Université Libre de Bruxelles, Belgium

Abstract

Learning finite automata from positive examples has recently gained attention as a powerful approach for understanding, explaining, analyzing, and verifying black-box systems. The motivation for focusing solely on positive examples arises from the practical limitation that we can only observe what a system is capable of (positive examples) but not what it cannot do (negative examples). Unlike the classical problem of passive DFA learning with both positive and negative examples, which has been known to be NP-complete since the 1970s, the topic of learning DFAs exclusively from positive examples remains poorly understood. This paper introduces a novel perspective on this problem by leveraging the concept of counting the number of accepted words up to a carefully determined length. Our contributions are twofold. First, we prove that computing the minimal number of words up to this length accepted by DFAs of a given size that accept all positive examples is NP-complete, establishing that learning from positive examples alone is computationally demanding. Second, we propose a new learning algorithm with a better asymptotic runtime that the best-known bound for existing algorithms. While our experimental evaluation reveals that this algorithm under-performs state-of-the-art methods, it demonstrates significant potential as a preprocessing step to enhance existing approaches.

1 Introduction

In recent years, the rapid advancement and proliferation of Artificial Intelligence systems have transformed numerous fields, from healthcare [ZWL22] to transportation [Bha23] or finance [CMR+23]. This surge in AI capabilities has greatly increased the need of interpretable models able to provide insights into the decision-making processes of complex black-box systems.

Interpretable models are synthesized from temporal data. The goal is to obtain a model that is coherent with the observations gathered, which in turn can be used to decipher the system's past behavior. Different kinds of (interpretable) models are used in the literature, often of one of two types: finite-state machines [WGY24] and (temporal) logic formulas [CM19]. In this paper, we focus on Deterministic Finite Automata (DFAs) [RS59], that enjoy both an accessible formalism, making them suitable e.g. classifier of sequential data [SLIM21], and many desirable properties, such as tractable model checking.

In the literature, synthesizing a DFA from observations is usually formalized as a passive learning task where we are given a positive set of words that a prospective DFA should accept, a negative set of words that it should reject, and an integer bound on its number of states [Gol78, GLP06, HV10]. However, a significant obstacle on the usefulness of the passive learning task from positive and negative examples (PLPN) is the difficulty of coming up with negative

examples: they correspond to observations of a system's undesirable behaviors, which can be intrinsically hard (e.g., with black-box systems) or unrealistic (e.g., with self-driving cars) to gather.

To circumvent that issue, the focus has recently shifted to the passive learning task of synthesizing DFAs from positive examples only (PLP). Without any additional restriction, the DFA accepting exactly the (positive) words of the input set $\mathcal{P}$—on the one end of the spectrum—and the trivial accepting DFA (the one-state DFA accepting every word)—on the other end of the spectrum—both meet the requirement of this PLP task while providing no insight whatsoever w.r.t. the input set $\mathcal{P}$. To forbid these two extreme solutions, constraints on the prospective DFA are added. On the one hand, as in the classical PLPN setting, we consider a bound $n \in \mathbb{N}$ on the number of states of the prospective DFA. The benefit is twofold: first, it prevents synthesizing the DFA accepting exactly the words in $\mathcal{P}$ (assuming n is small enough); second, for explainability purposes, the smaller the DFA, the better. On the other hand, we look for a DFA that is language-minimal, i.e., whose language is minimal (for inclusion) among those DFAs with at most n states that accept all words in $\mathcal{P}$. As argued by Avellaneda and Petrenko (2018), this constraint follows naturally from a parsimony standpoint: we look for a simplest solution. With these restrictions, we obtain a PLP task $\mathcal{T}$.

A counterexample (CE) guided algorithm solving the task $\mathcal{T}$ was first developed by Avellaneda and Petrenko (2018), then a few years later it was improved into a symbolic algorithm by Roy et al. (2023), which now constitutes the state-of-the-art. The CE algorithm relies on iteratively synthesizing DFAs by solving instances of the PLPN task, where $\mathcal{P}$ always constitutes the positive set, while the negative set is initially empty and iteratively grown by adding words witnessing the non language-minimality of candidate DFAs (i.e., counterexamples). On the other hand, the symbolic algorithm starts from the trivial accepting DFA, and iteratively looks for DFAs accepting all words in $\mathcal{P}$ with a smaller language. Both the CE and symbolic algorithms iteratively call SAT solvers at each step to find new candidate DFAs.

By construction, the symbolic algorithm avoids redundancies that could occur in the CE algorithm. However, even with the symbolic algorithm, a significant drawback remains: the best bound on the number of steps of the algorithm, and thus on the number of calls to SAT solvers, is exponential in n. This is due to the fact that there are exponentially many words of length at most (polynomial in) n. Furthermore, while the complexity of the PLPN task (which can be stated as a decision problem) is well understood—it is known to be NP-complete since the 70s [Gol78]—the PLP task is poorly understood complexity-wise. In particular, [AP18, RGB+23] do not provide a complexity lower bound on any closely-related decision problem. As such, it is not clear that solving the PLP task $\mathcal{T}$ even requires the use of SAT solvers.

Our contributions. In this paper, we tackle the PLP task $\mathcal{T}$ from a new perspective which consists in assigning to each DFA a word-counting metric that greatly enhances our ability to compare two DFAs (as opposed to checking language inclusion). Specifically, this word-counting metric is the number of words accepted by each DFA up to length $2n - 2$. To demonstrate the usefulness of this new perspective, we show that a word-counting minimal DFA is necessarily language-minimal. Thus, we focus on a new PLP task $\mathcal{T}'$—that requires a prospective DFA to be word-counting minimal instead of only language-minimal—and on the associated decision problem about the minimal word-counting metric of DFAs with at most n states accepting all words in $\mathcal{P}$. Our main contribution is that this problem is NP-complete, which constitutes the first complexity result directly related to a PLP task. This NP-completeness result has two main consequences: first, because it is in NP, we immediately obtain that the task $\mathcal{T}'$, and thus the task $\mathcal{T}$ as well, can be solved with polynomially many calls to a SAT solver, via a binary search; second, because the problem is NP-hard, it is unlikely that we can do better than polynomially

many calls to a SAT solver to solve task $\mathcal{T}'$ (though task $\mathcal{T}$ could be intrinsically easier to solve than task $\mathcal{T}'$).

We have implemented an algorithm solving task $\mathcal{T}'$ (and thus also task $\mathcal{T}$) via an Integer Linear Programming (ILP) encoding. Our experiments show that this algorithm under-performs the symbolic state-of-the-art algorithm solving task $\mathcal{T}$. However, building on the word-counting ideas presented above, we have developed a method that could be used to improve the performance of the symbolic algorithm for solving $\mathcal{T}$. Specifically, we have designed a heuristic algorithm that looks for a DFA with at most n states, accepting all words in $\mathcal{P}$, with a word-counting metric as small as possible. This algorithm is intended to be used as a pre-processing step outputting a DFA that can then be used as a starting point by the symbolic algorithm. Our experiments show promising results of this method.

Missing technical details can be found in the extended version [BN25].

Related work. As already mentioned, there is a large body of work focusing on the classical DFA passive learning. Another widely studied DFA learning setting is active learning, where a learner successively queries a teacher. Angluin's seminal work [Ang87] essentially started the research on this topic, and is still very influential nowadays [SG09, BHKL09].

DFA learning from positive examples only was introduced by Gold (1967), where it is shown that DFAs cannot be identified at the limit. This partly explains why there was no further study of this topic until recently [AP18, RGB+23]. More popular subjects are the passive learning from positive examples only of better-behaved models, such as pattern languages [Ang80], hidden Markov chains [SO92], or stochastic machines [CO99].

Finally, an ILP encoding a learning task (of a logical (LTLf) formula) was first introduced in [ILF+23].

2 Definitions and notations

We let $\mathbb{N}$ denote the set of non-negative integers. For all $i \leq j \in \mathbb{N}$, we let $[\![i, j]\!]$ denote the set $\{i, i+1, \ldots, j\} \subseteq \mathbb{N}$.

Alphabet and automata. An alphabet Σ refers to a non-empty finite set of letters. We let Σ^* (resp. Σ^+) denote the set of (resp. non-empty) finite words with letters in Σ. We let $\epsilon \in \Sigma^*$ denote the empty word. For all words $u \in \Sigma^*$, we let $|u| \in \mathbb{N}$ denote the length of the word u, i.e., its number of letters. For all $m \in \mathbb{N}$, we let Σ^m (resp. $\Sigma^{\leq m}$) denote the set of words of length m (resp. at most m). For all $L \subseteq \Sigma^*$, we let $\mathsf{Pref}(L)$ denote the set of (non-strict) prefixes of words in L. Let us now recall the definition of DFAs.

Definition 1. *Consider an alphabet Σ. A* deterministic finite automaton (DFA) *on Σ is a tuple $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, \delta, F)$ where Q is a finite non-empty set of states, $q_{\mathsf{init}} \in Q$ is the initial state, $\delta \colon Q \times \Sigma \to Q$ is the transition function, and $F \subseteq Q$ is the set of final states. The transition function δ is naturally extended to finite words into a function $\delta^* \colon Q \times \Sigma^* \to Q$ defined as follows: for all $q \in Q$, $\delta^*(q, \epsilon) := q$, and for all $u \in \Sigma^*$ and $\alpha \in \Sigma$: $\delta^*(q, u \cdot \alpha) := \delta(\delta^*(q, u), \alpha)$.*

A word $u \in \Sigma^*$ is accepted *by the DFA $\mathcal{A}$ if $\delta^*(q_{\mathsf{init}}, u) \in F$. We let $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ denote the language of the DFA $\mathcal{A}$, i.e., the set of words that it accepts: $\mathcal{L}(\mathcal{A}) := \{u \in \Sigma^* \mid \delta^*(q_{\mathsf{init}}, u) \in F\}$. We let $|\mathcal{A}|$ denote the size of the DFA $\mathcal{A}$, i.e., its number of states $|\mathcal{A}| := |Q|$. Furthermore, let $\mathsf{DFA}(\Sigma)$ denote the set of DFAs on the alphabet Σ.*

In the following, unless otherwise stated, a DFA $\mathcal{A}$ on an alphabet Σ will refer to the tuple $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, \delta, F)$.

Given an integer $n \in \mathbb{N}$ and a set of words $\mathcal{P}$, we let $\mathsf{Rec}(\mathcal{P}, n) := \{\mathcal{A} \in \mathsf{DFA}(\Sigma) \mid |\mathcal{A}| \leq n,\ \mathcal{P} \subseteq \mathcal{L}(\mathcal{A})\}$ denote the set of DFAs with at most n states that accept all words in $\mathcal{P}$.

Strict partial orders. A relation $\prec \subseteq \mathsf{DFA}(\Sigma) \times \mathsf{DFA}(\Sigma)$ is a strict partial order on $\mathsf{DFA}(\Sigma)$ if it is irreflexive, antisymmetric, and transitive. For $S \subseteq \mathsf{DFA}(\Sigma)$ and $\mathcal{A} \in S$, $\mathcal{A}$ is said to be S-minimal w.r.t. $\prec$ if, for all $\mathcal{A}' \in S$, we do *not* have $\mathcal{A}' \prec \mathcal{A}$.

Given two DFAs $\mathcal{A}, \mathcal{A}' \in \mathsf{DFA}$, we write $\mathcal{A} \prec_{\mathcal{L}} \mathcal{A}'$ when $\mathcal{L}(\mathcal{A}) \subsetneq \mathcal{L}(\mathcal{A}')$. Clearly, $\prec_{\mathcal{L}}$ is a strict partial order on $\mathsf{DFA}(\Sigma)$.

Learning from positive examples only. In this paper, we focus on the task of synthesizing a DFA with at most $n \geq 1$ states accepting all words in a given set $\mathcal{P}$, i.e., we look for DFAs in $\mathsf{Rec}(\mathcal{P}, n)$. Furthermore, among those DFAs, we look for a language-minimal one (following the parsimony principle), i.e., one that is $\mathsf{Rec}(\mathcal{P}, n)$-minimal w.r.t. the order $\prec_{\mathcal{L}}$. Formally, the task that we consider is defined below.

Task 2. *Given as input an alphabet Σ, a set $\mathcal{P} \subseteq \Sigma^*$, and $n \geq 1$, find a DFA $\mathcal{A}$ such that: a) $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$; and b) the DFA $\mathcal{A}$ is $\mathsf{Rec}(\mathcal{P}, n)$-minimal w.r.t. $\prec_{\mathcal{L}}$.*

The size of this task is equal to $n + |\mathsf{Pref}(\mathcal{P})|$.

3 Learning DFAs via word counting

Before we detail our approach for solving Task 2, let us describe the state-of-the-art (symbolic) algorithm.

The state-of-the-art algorithm. It was developed by Roy et al. (2023) and proceeds as follows. Initially, one starts with the trivial accepting DFA $\mathcal{A}_0 \in \mathsf{Rec}(\mathcal{P}, n)$. Then, at step $i \geq 1$, one computes a DFA $\mathcal{A}_i \in \mathsf{Rec}(\mathcal{P}, n)$ satisfying $\mathcal{A}_i \prec_{\mathcal{L}} \mathcal{A}_{i-1}$ using a SAT solver. The process stops when no such DFA is found. Overall, this algorithm iteratively computes a decreasing chain of DFAs $\mathcal{A}_0 \succ_{\mathcal{L}} \ldots \succ_{\mathcal{L}} \mathcal{A}_k \succ_{\mathcal{L}} \ldots$ with $\mathcal{A}_i \in \mathsf{Rec}(\mathcal{P}, n)$, for all $i \in \mathbb{N}$. Since the languages of DFAs may be infinite, it may not be clear that this chain is necessarily finite. However, it is indeed the case, as argued by Roy et al. (2023): two DFAs with at most n states have the same language if they accept exactly the same words of length at most n^2 (a tighter bound exists, as discussed below). Thus, since there are finitely many words of length at most n^2, the above decreasing chain of DFAs is necessarily finite. However, the best bound on the number of iterations that we can extract from this argument is exponential in n (since there are exponentially many words of length at most n^2), with each iteration involving a call to a SAT solver.

From a language-inclusion to a numerical order. To design a method that does significantly less calls to SAT solvers, we shift perspective from focusing on language inclusion, via the strict partial order $\prec_{\mathcal{L}}$, to focusing on a numerical order induced by values that we assign to DFAs. As we will see in the remainder of this paper, this has two main benefits: first, numerical orders are compatible with binary searches; second, the values assigned to DFAs can be seen as a score that we aim at optimizing (in this case, minimize). However, it is crucial that this numerical order relates to the language-inclusion order $\prec_{\mathcal{L}}$ previously defined, as this is the one involved in Task 2. In fact, taking as values assigned to DFAs the number of words, up to a certain length, that they accept does the job. This is formally defined below.

Definition 3. *Consider some $h \in \mathbb{N}$. For all DFAs $\mathcal{A}, \mathcal{A}'$, we write $\mathcal{A} \prec_h \mathcal{A}'$ when $|\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq h}| < |\mathcal{L}(\mathcal{A}') \cap \Sigma^{\leq h}|$.*

In other words, we have $\mathcal{A} \prec_h \mathcal{A}'$ when $\mathcal{A}$ accepts fewer words of length at most h than $\mathcal{A}'$. Clearly, for all $h \in \mathbb{N}$, $\prec_h$ is a strict partial order on DFAs. Furthermore, this order satisfies the following property, thus showing that minimizing the number of accepted words also minimizes the language.

Proposition 4. *For an alphabet Σ, a set $\mathcal{P} \subseteq \Sigma^*$, $n \geq 1$, and $h := 2n-2$, a DFA $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$ that is $\mathsf{Rec}(\mathcal{P}, n)$-minimal w.r.t. $\prec_h$ is $\mathsf{Rec}(\mathcal{P}, n)$-minimal w.r.t. $\prec_{\mathcal{L}}$.*

This proposition relies on the folk theorem recalled below.

Theorem 5 (Folk result)**.** *Consider an alphabet Σ, and $\mathcal{A}, \mathcal{A}' \in \mathsf{DFA}(\Sigma)$. If $\mathcal{L}(\mathcal{A}) \neq \mathcal{L}(\mathcal{A}')$, then $\Sigma^{\leq |\mathcal{A}|+|\mathcal{A}'|-2} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{L}(\mathcal{A}') \cup \mathcal{L}(\mathcal{A}') \setminus \mathcal{L}(\mathcal{A})) \neq \emptyset$.*

We provide a proof of this theorem in [BN25]. It essentially relies on the iterative computation of the Myhill-Nerode equivalence classes of DFAs. We also show that the bound $h = |\mathcal{A}| + |\mathcal{A}'| - 2$ is tight.

The proof of Proposition 4 is now direct.

Proof. Consider a DFA $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$ that is $\mathsf{Rec}(\mathcal{P}, n)$-minimal w.r.t. $\prec_h$, for $h = 2n - 2$. Consider any DFA $\mathcal{A}' \in \mathsf{Rec}(\mathcal{P}, n)$ and assume towards a contradiction that we have $\mathcal{A}' \prec_{\mathcal{L}} \mathcal{A}$, i.e., $\mathcal{L}(\mathcal{A}') \subsetneq \mathcal{L}(\mathcal{A})$. By Theorem 5, there is some $u \in \Sigma^{\leq h} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{L}(\mathcal{A}'))$. Since we do not have $\mathcal{A}' \prec_h \mathcal{A}$, it follows that $|\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq h}| \leq |\mathcal{L}(\mathcal{A}') \cap \Sigma^{\leq h}|$. Hence, there must be some $u' \in \Sigma^{\leq h} \cap (\mathcal{L}(\mathcal{A}') \setminus \mathcal{L}(\mathcal{A}))$. Thus, we do not have $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$. Hence, the contradiction. □

Following Proposition 4, we consider a new task.

Task 6. *Given as input an alphabet Σ, a set $\mathcal{P} \subseteq \Sigma^*$, and $n \geq 1$, find a DFA $\mathcal{A}$ such that: a) $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$; and b) the DFA $\mathcal{A}$ is $\mathsf{Rec}(\mathcal{P}, n)$-minimal w.r.t. $\prec_{2n-2}$.*

Proposition 4 gives that a DFA satisfying the requirements of Task 6 also satisfies the requirements of Task 2. Therefore, let us focus on solving this Task 6.

4 Computing the minimal number of accepted words of length at most $2n - 2$.

To study how we can solve Task 6, we investigate the associated decision problem, formally defined below.

Decision Problem 7. *Given as input an alphabet Σ, a set $\mathcal{P} \subseteq \Sigma^*$, $n \geq 1$, and $k \in \mathbb{N}$, decide if there exists a DFA $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$ such that: $|\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq 2n-2}| \leq k$.*

The size of the input of this task is $n + |\mathsf{Pref}(\mathcal{P})| + \log k$.

As a first step in the study of this problem, let us consider how we can compute the number of words, up to a certain length, accepted by a DFA. Actually, this can be done in a direct (iterative) manner, as described in Lemma 8 below.

Lemma 8. *Consider an alphabet Σ and a DFA $\mathcal{A} \in \mathsf{DFA}(\Sigma)$. For all $q \in Q$ and $m \in \mathbb{N}$, we let $\mathsf{N}(q, m) := \{w \in \Sigma^m \mid \delta^*(q_0, w) = q\}$. Then, $\mathsf{N}(q, 0) = 1$ if $q = q_0$, and $\mathsf{N}(q, 0) = 0$ otherwise. Furthermore, for all $(m, q) \in \mathbb{N} \times Q$: $\mathsf{N}(q, m+1) = \sum_{q' \in Q} \mathsf{N}(q', m) \cdot |\{\alpha \in \Sigma \mid \delta(q', \alpha) = q\}|$.*

In addition, $|\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq m}| = \sum_{i=0}^{m} \sum_{q \in F} \mathsf{N}(q, i)$.

Note that this lemma would not hold with automata that are not deterministic.

Let us now state the main result of this paper.

Theorem 9. *The decision problem 7 is* NP*-complete.*

Proof. We argue the NP-hardness in a dedicated section below. As for Problem 7 being in NP, this problem can be solved as follows: we guess a DFA $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$—we can check that it is indeed the case in time polynomial in $|\mathsf{Pref}(\mathcal{P})|$ by simulating the runs of $\mathcal{A}$ on words in $\mathcal{P}$—, we compute $|\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq 2n-2}| \in \mathbb{N}$ and compare it with k. Although $|\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq 2n-2}|$ may be exponential, we can compute it in polynomial time with the help of Lemma 8. □

Before we present the NP-hardness proof, let us discuss the two major implications of Theorem 9. First, the fact that the decision problem 7 is in NP suggests an algorithm solving Task 6, and thus Task 2 as well, with seemingly better asymptotic guarantees than the state-of-the-art algorithm.

Proposition 10. *Given an alphabet Σ, a set $\mathcal{P} \subseteq \Sigma^*$, and an integer $n \in \mathbb{N}$, we can solve Task 6 with $\mathcal{O}(poly(|\Sigma|, n))$-calls to a SAT solver on inputs of size $\mathcal{O}(poly(|\Sigma|, |\mathsf{Pref}(\mathcal{P})|, n))$.*

Proof. We have $|\Sigma^{\leq 2n-2}| = \frac{m^{2n-1}-1}{m-1}$, for $m := |\Sigma| \geq 2$. Hence, a binary search querying, at each step, a SAT solver on an encoding of Problem 7 with $1 \leq k \leq |\Sigma^{\leq 2n-2}|$ would take at most $\log(|\Sigma^{\leq 2n-2}|) \leq (2n-1) \cdot \log m$ steps to find a DFA $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$ satisfying $|\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq 2n-2}| = \min_{\mathcal{A}' \in \mathsf{Rec}(\mathcal{P},n)} |\mathcal{L}(\mathcal{A}') \cap \Sigma^{\leq 2n-2}|$. By definition, such a DFA $\mathcal{A}$ would be $\mathsf{Rec}(\mathcal{P}, n)$-minimal w.r.t. $\prec_{2n-2}$. □

Second, unless P = NP, the NP-hardness of Decision Problem 7 makes it unlikely that it is possible to solve Task 6 with an asymptotic bound significantly better than that of Proposition 10. This only applies to Task 6, and there might be an efficient way to solve Task 2 without solving Task 6.

NP-hardness proof. The NP-hardness proof of Theorem 9 is quite technical, see [BN25] for a detailed proof. Here, we only give a bird's eye view of the steps that we take to prove the result.

The NP-hardness proof relies on a reduction from the All-Pos-Neg SAT (APN-SAT) problem that was already used by Lingg, de Oliveira Oliveira, and Wolf (2024) to establish the NP-hardness of the classical passive learning problem with positive and negative words (with a binary alphabet). The APN-SAT problem is a variant of the classical SAT problem where we are given a set of clauses to satisfy, with each clause being constituted of only positive literals (i.e., without negations) or only negative literals (i.e., with negations). This problem is defined as follows.

Decision Problem 11. *Consider as input a set $X := \{x_i \mid i \in [\![1, r]\!]\}$ of variables, a set of clauses $\mathcal{C} = \{C_j \mid j \in [\![1, s]\!]\}$ that is described as the disjoint union of sets of positive and negative clauses: $\mathcal{C} = \mathcal{C}_+ \uplus \mathcal{C}_-$, where for all $j \in [\![1, s]\!]$, we have $C_j \subseteq X$. We have to decide if there is a valuation $\nu \colon X \to \{\top, \bot\}$ of the variables satisfying $(X, \mathcal{C})$, i.e., such that, for all $j \in [\![1, s]\!]$:*

$$\exists x \in C_j : \ \nu(x) = \begin{cases} \top & \text{if } C_j \in \mathcal{C}_+ \\ \bot & \text{if } C_j \in \mathcal{C}_- \end{cases}$$

For the remainder of this section, we consider an instance $(X, \mathcal{C})$ of Problem 11 with $X = \{x_i \mid i \in [\![1, r]\!]\}$ the set of variables, and $\mathcal{C} = \{C_j \mid j \in [\![1, s]\!]\} = \mathcal{C}_+ \uplus \mathcal{C}_-$ the set of clauses. The alphabet that we consider is $\Sigma := \{a, b\}$. Our goal is to define a set of positive words $\mathcal{P}$ and two

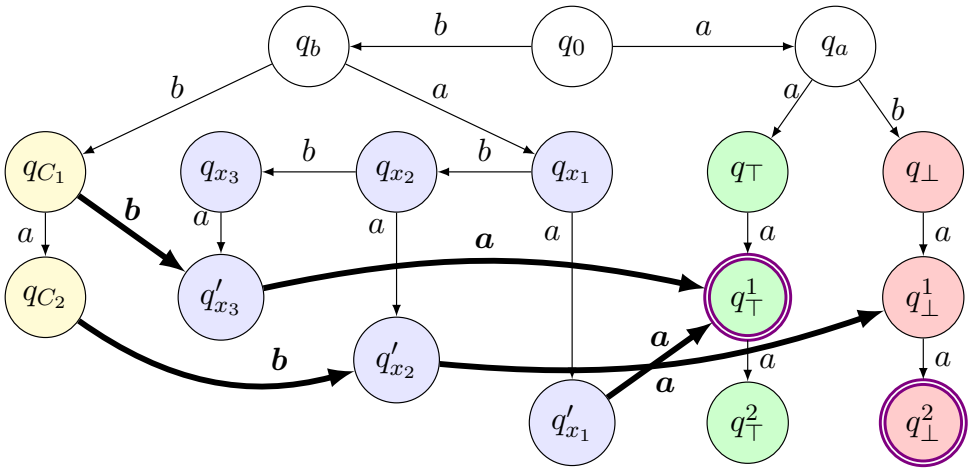


Figure 1: The shape of a DFA $\mathcal{A}_{\mathsf{ex}}$.

integers n and k such that $(X, \mathcal{C})$ is a positive instance of Problem 11 if and only if there is a $(\mathcal{P}, n, k)$-suitable DFA $\mathcal{A}$, i.e., such that 1) $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$; and 2) $|\Sigma^{\leq 2n-2} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P})| \leq k$[1].

Let us first discuss what a DFA satisfying conditions 1) and 2) could look like on a simple input of Problem 11. That way, we will be able to illustrate (some of) the kinds of words that we include in $\mathcal{P}$, which the DFA has to accept.

Example 12. *Consider the set of variables $X = \{x_1, x_2, x_3\}$, a single positive clause $C_1 = \{x_1, x_3\} \in \mathcal{C}_+$ and single a negative clause $C_2 = \{x_2, x_3\} \in \mathcal{C}_-$. In Figure 1, we scheme the shape of a DFA $\mathcal{A}_{\mathsf{ex}}$ that would be $(\mathcal{P}, n, k)$-suitable, with $(\mathcal{P}, n, k)$—which we partly describe here—being the result of the reduction from the instance $(X, \mathcal{C})$. The double-circled states are accepting.*

In $\mathcal{P}$, we define a "$\top$-word" $a \cdot a \cdot a$ and a "$\bot$-word" $a \cdot b \cdot a \cdot a$. One can see that when the DFA $\mathcal{A}_{\mathsf{ex}}$ runs on those words, it visits the right-most (green and red) states.

We also define "variables words" in $\mathcal{P}$ that the DFA $\mathcal{A}_{\mathsf{ex}}$ processes by visiting the central (blue) states. In particular, these words impose to $\mathcal{A}_{\mathsf{ex}}$ that we reach either $q^1_\top$ or $q^1_\bot$ upon reading the letter a from the states q'_{x_i}, for $i \in \{1, 2, 3\}$. This corresponds to the bold transitions from the central states to the right-most states. These transitions uniquely define a valuation $\nu \colon X \to \{\top, \bot\}$, specifically: $\nu(x_1) = \top$, $\nu(x_2) = \bot$, $\nu(x_3) = \top$.

In addition, we define "clause words" in $\mathcal{P}$ that $\mathcal{A}_{\mathsf{ex}}$ processes by visiting the left-most (yellow) states. Some of these words ensure that upon reading the letter b from q_{C_j}, for $j \in \{1, 2\}$, we reach a state q'_{x_i}, for $i \in \{1, 2, 3\}$, such that $x_i \in C_j$. The other "clause words" ensure that a positive (resp. negative) clause-state eventually leads to a $\top$-state (resp. $\bot$-state). Namely, the word $b \cdot b \cdot b \cdot a \in \mathcal{P}$ encodes that C_1 is a positive clause, and the word $b \cdot b \cdot a \cdot b \cdot a \cdot a \in \mathcal{P}$ encodes that C_2 is a negative clause.

Overall, the valuation $\nu \colon X \to \{\top, \bot\}$ uniquely defined by $\mathcal{A}_{\mathsf{ex}}$ satisfies $(X, \mathcal{C})$, e.g., for the negative clause C_2, there is a bold edge from q_{C_2} to q'_{x_2}, thus we must have $x_2 \in C_2$ and $\nu(x_2) = \bot$, which is indeed the case.

Let us now describe a little more formally how the words that we define in $\mathcal{P}$ ensure that the existence of a $(\mathcal{P}, n, k)$-suitable DFA $\mathcal{A}$ implies that $(X, \mathcal{C})$ is a positive instance of Problem 11. Below, we consider a DFA $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, \delta, F)$ satisfying conditions 1) and 2).

[1]This condition constitutes a slight change compared to Problem 7. This is however harmless, see the extended version.

1. We define two words $w_\top, w_\bot \in \mathcal{P}$ and we ensure that $\delta^*(q_0, w_\top) \neq \delta^*(q_0, w_\bot)$. This fact will allow us to distinguish when a variable is mapped to true ($\top$) and to false ($\bot$). In the DFA $\mathcal{A}_{\mathsf{ex}}$ of Example 12, we have $\delta^*(q_0, w_\top) = q^1_\top \neq q^1_\bot = \delta^*(q_0, w_\bot)$.

2. For each variable $x \in X$, we define a variable word $w_x \in \mathcal{P}$ and we ensure that, for all $x \neq x' \in X$, $\delta^*(q_0, w_x) \neq \delta^*(q_0, w_{x'})$. In the DFA $\mathcal{A}_{\mathsf{ex}}$, we have $\delta^*(q_0, w_{x_1}) = q'_{x_1}$, $\delta^*(q_0, w_{x_2}) = q'_{x_2}$, and $\delta^*(q_0, w_{x_3}) = q'_{x_3}$.

3. We add words in $\mathcal{P}$ to ensure that there is some $w_{\mathsf{Var}} \in \Sigma^*$, such that, for all $x \in X$, we have $\delta^*(q_0, w_x \cdot w_{\mathsf{Var}}) \in \{\delta^*(q_0, w_\top), \delta^*(q_0, w_\bot)\}$. In Example 12, $w_{\mathsf{Var}} = a$. When this property is ensured, a valuation $\nu\colon X \to \{\top, \bot\}$ is uniquely defined by, for all $x \in X$: $\delta^*(q_0, w_x \cdot w_{\mathsf{Var}}) = \delta^*(q_0, w_{\nu(x)})$. (Note that the valuation $\nu\colon X \to \{\top, \bot\}$ is unique because $\delta^*(q_0, w_\top) \neq \delta^*(q_0, w_\bot)$.)

4. For each clause $C \in \mathcal{C}$, we define a clause word $w_C \in \mathcal{P}$ and we ensure that, for all $C, C' \in \mathcal{C}$, $\delta^*(q_0, w_C) \neq \delta^*(q_0, w_{C'})$. In the DFA $\mathcal{A}_{\mathsf{ex}}$, we have $\delta^*(q_0, w_{C_1}) = q_{C_1}$, and $\delta^*(q_0, w_{C_2}) = q_{C_2}$.

5. We add words in $\mathcal{P}$ to ensure that there is some $w_{\mathsf{Cl}} \in \Sigma^*$, such that, for all $C \in \mathcal{C}$, there is some variable $x_C \in C$ such that we have $\delta^*(q_0, w_C \cdot w_{\mathsf{Cl}}) = \delta^*(q_0, w_{x_C})$. In Example 12, we have $w_{\mathsf{Cl}} = b$.

6. Finally, we add words in $\mathcal{P}$ to ensure that, for all clauses $C \in \mathcal{C}$, we have: $\delta^*(q_0, w_C \cdot w_{\mathsf{Cl}} \cdot w_{\mathsf{Var}}) = \delta^*(q_0, w_\top)$ if $C \in \mathcal{C}_+$, and $\delta^*(q_0, w_C \cdot w_{\mathsf{Cl}} \cdot w_{\mathsf{Var}}) = \delta^*(q_0, w_\bot)$ if $C \in \mathcal{C}_-$. In Example 12, this corresponds to the words $b \cdot b \cdot b \cdot a, b \cdot b \cdot a \cdot b \cdot a \cdot a \in \mathcal{P}$.

With Items 5. and 6., we have that the valuation defined in Item 3. satisfies $(X, \mathcal{C})$. Indeed, consider e.g. a positive clause $C \in \mathcal{C}$. By Fact 5, there is variable $x_C \in C$ such that $\delta^*(q_0, w_C \cdot w_{\mathsf{Cl}}) = \delta^*(q_0, w_{x_C})$. By Fact 6, we have $\delta^*(q_0, w_{x_C} \cdot w_{\mathsf{Var}}) = \delta^*(q_0, w_C \cdot w_{\mathsf{Cl}} \cdot w_{\mathsf{Var}}) = \delta^*(q_0, w_\top)$. Thus, $\nu(x_C) = \top$ and the valuation ν satisfies the clause C. Note that, in the actual reduction described in the extended version [BN25], there are many words in $\mathcal{P}$ associated with $\top$, $\bot$, variables and clauses.

Now that we have described the general idea of the construction, we would like to sketch how we prove that a DFA satisfying conditions 1) and 2) necessarily ensures the various properties laid out in Items 1., 2., 3., 4., 5., and 6. above. We use two different kinds of arguments that leverage the two opposing constraints on DFAs satisfying conditions 1) on 2): on the one hand, such DFAs must not make too many errors (i.e., accepting too many words not in $\mathcal{P}$), this is leveraged by TME-arguments ("Too Many Errors"); on the other hand, such DFAs must not have too many states, this is leveraged by TMS-arguments ("Too Many States").

- TME-arguments are used to show that two words $w, w' \in \Sigma^*$ are mapped to two different states by the function $\delta^*(q_{\mathsf{init}}, \cdot)$. Typically, the structure of a TME-argument is as follows. Assume that there are $k+1$ words $(u_\sigma)_{\sigma \in [\![1, k+1]\!]}$ such that we have $w \cdot u_\sigma \in \mathcal{P}$ and $w' \cdot u_\sigma \notin \mathcal{P}$ for all $\sigma \in [\![1, k+1]\!]$. In that case, it cannot be that $\delta^*(q_{\mathsf{init}}, w) = \delta^*(q_{\mathsf{init}}, w')$, since otherwise we would have $w' \cdot u_\sigma \in \mathcal{L}(\mathcal{A}) \setminus \mathcal{P}$, for all $\sigma \in [\![1, k+1]\!]$, which is a contradiction with condition 2) (assuming that $2n - 2$ is large enough). This type of argument allows to establish the properties described in Facts 1., 2. and 4.

- TMS-arguments are used to show that in two sets of words $W, W' \subseteq \Sigma^*$ there must be some $w \in W, w' \in W'$ such that $\delta^*(q_{\mathsf{init}}, w) = \delta^*(q_{\mathsf{init}}, w')$. To do so, it suffices to show that we have $|W| + |W'| > n$. TMS-arguments allow us to establish the properties described in Facts 3. and 5.

As for the properties of Fact 6, they are direct consequences of the properties of other facts (specifically, Fact 3 and Fact 5), and of the words that we define in $\mathcal{P}$.

5 Experiments

Our goal is now to use our new word-counting perspective to improve the state-of-the-art algorithm solving Task 2. All of our code and data is available on GitHub (cf. the top page). Before we discuss our algorithms, let us describe the experimental setup that we have used.

Experimental setup. We ran all of our experiments on Ubuntu 24.04.2 LTS, using 30 GiB of RAM, 12 CPU cores, and a clock speed of 5.2 GHz.

In order to compare our algorithms with the state-of-the-art symbolic algorithm developed by Roy et al. (2023), we have used the inputs used in that paper to compare the performance of the symbolic algorithm and the counterexample guided algorithm developed by Avellaneda and Petrenko (2018). These inputs are constituted of 28 samples of around 1000 words of lengths from 1 to 10. These words are obtained by sampling the languages of randomly generated small DFAs (of at most 10 states), with an alphabet of size 3. As in [RGB+23], we consider a timeout (TO) of 1000s to solve instances of Task 2 and Task 6. All experiments are done with a number of states n between 1 and 8. For all samples, we have run the algorithms 10 times, and we use the mean runtime (with the standard deviation).

5.1 ILP algorithm solving Task 6

To solve Task 6, we encode it as an Integer Linear Programming (ILP) minimization problem. In that setting, the goal is to minimize a linear combination of integer variables, subject to a system of linear inequalities (with integer coefficients). When encoding Task 6 in the ILP setting, the quantity to minimize corresponds to the number of words up to length $2n-2$ accepted by a prospective DFA.

Let us describe on a high level how our ILP encoding works—the details can be found in the extended version [BN25]. Our encoding relies on two types of variables: binary variables, encoding the DFA and checking that it is in $\mathsf{Rec}(\mathcal{P}, n)$; and integer variables which are used to count the number of accepted words. The inequalities involving the binary variables are actually very close to the logical constraints used in the SAT-encoding developed by [RGB+23] (2023). As for the integer variables, our encoding relies on Lemma 8: the inductive relation described in this lemma allows us to store in an integer variable the number of words of length at most $2n-2$ accepted by the prospective DFA, which we can set as a minimization goal, subject to the other inequalities. Note that, to avoid having variable multiplication, we had to use the classical Big-M method [GNS08].

We implemented our algorithm in Gurobi [Gur24] which we have used with a Python3 API. In Figure 2, we have plotted in blue dots the computation time of our ILP algorithm (y-axis) and the computation time of the symbolic algorithm (x-axis) on each 28 samples with $n = 4$. Clearly, the ILP algorithm largely under-performs the symbolic algorithm. The situation is not significantly better for the case $n = 6$ plotted as red stars[2]. Overall, our ILP algorithm does not stand the comparison with the symbolic algorithm. We discuss below three possible reasons for this significant runtime difference.

First, from a theoretical standpoint, the symbolic algorithm could asymptotically take exponentially many steps to terminate. However, in practice, the symbolic algorithm does not

[2]We only plotted the samples for which the ILP algorithm did not reach a full timeout (i.e., all 10 attempts timeout) before $n = 6$.

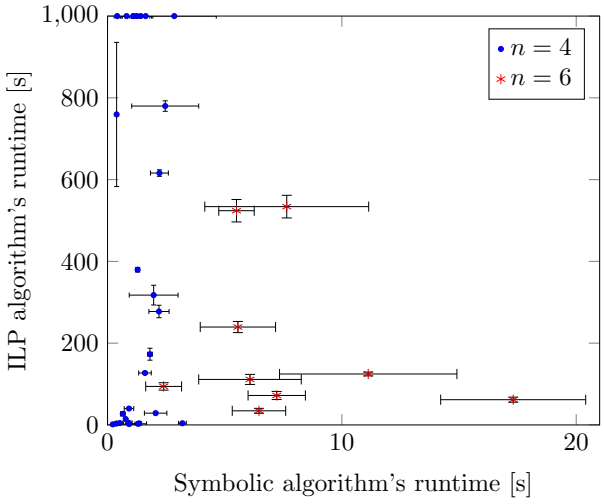


Figure 2: The comparison on the runtime of the symbolic and ILP algorithms.

take many steps to terminate (around 20, in the worst case), and thus greatly outperforms our ILP algorithm. There could be some kinds of samples for which the symbolic algorithm takes much more steps to conclude, in which case the runtime difference could be much smaller.

Furthermore, ILP solvers and SAT solvers internally use really different mechanisms: branch and bound for ILP solvers and DPLL for SAT solvers. It could be that in the case of these samples, DPLL works much better than branch and bound. This could be tested by implementing a binary search algorithm solving Task 6 by querying SAT solvers.

Finally, the ILP algorithm solves Task 6, while the symbolic algorithm solves Task 2. We have shown in Proposition 4 that Task 6 is at least as hard to solve as Task 2, it could very well be that it is simply much harder to solve.

5.2 A pre-processing algorithm

The symbolic algorithm starts from the trivial accepting DFA and then computes a $\prec_{\mathcal{L}}$-decreasing chain of DFAs until it finds a language-minimal one. To speed-up this process, we have designed a heuristic algorithm that computes a DFA that can then be used as a starting point for the symbolic algorithm. We first describe this heuristic algorithm, and then we discuss its experimental evaluation.

Description of the heuristic algorithm. The goal of this heuristic algorithm is to output a DFA $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$ that accepts as few words of length at most $2n-2$ as possible. The general idea of the algorithm is to randomly search among DFAs, and pick one that minimizes the number of accepted words. However, it is not clear how to randomly search among DFAs in $\mathsf{Rec}(\mathcal{P}, n)$. Therefore, our heuristic algorithm focuses on transition systems instead. A transition system can be seen as a DFA without starting or final states, i.e., it is described by a tuple $T = (Q, \Sigma, \delta)$ with $\delta \colon Q \times \Sigma \to Q$. Consider then a transition system $T = (Q, \Sigma, \delta)$ with $|Q| \leq n$ and a starting state $q \in Q$. It is clear that choosing $F_{T,q} := \{\delta^*(q, w) \mid w \in \mathcal{P}\}$ minimizes the number of words accepted by the DFA $\mathcal{A}_{T,q} := (Q, \Sigma, \delta, q, F_{T,q})$ while ensuring that $\mathcal{A}_{T,q} \in \mathsf{Rec}(\mathcal{P}, n)$. Following, given a transition system T, we define the *score* of the transition system T as follows: $\min_{q \in Q} |\mathcal{L}(\mathcal{A}_{T,q}) \cap \Sigma^{\leq 2n-2}|$. We naturally extend the notion of score to DFAs. Our heuristic algorithm then proceeds as follows:

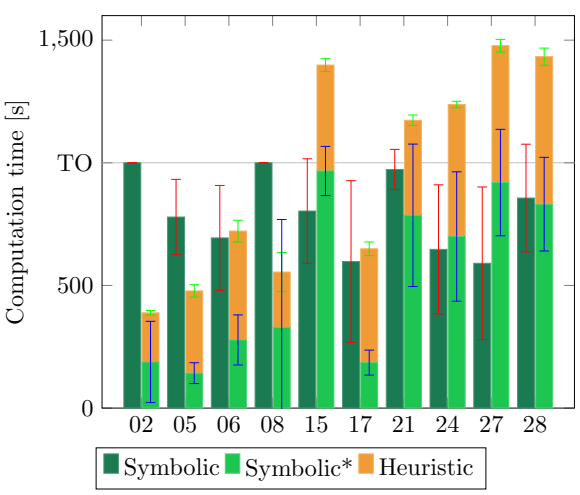


Figure 3: The runtime comparison of the symbolic and the heuristic+symbolic* algorithms.

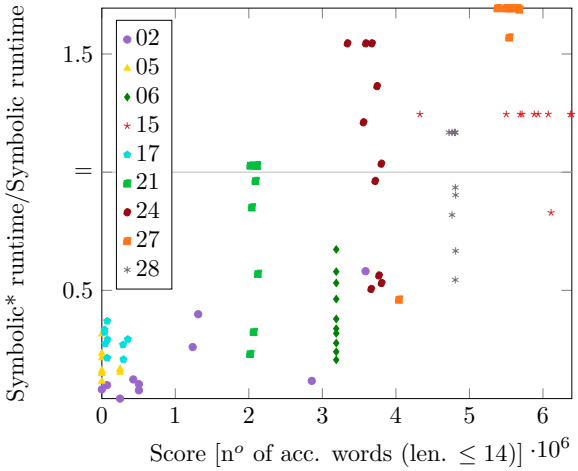


Figure 4: The quotient of the runtime of the symbolic(*) algorithms as a function of the starting score.

1. Randomly search (InitRand attempts) n-states transition systems and pick one with the best (i.e., lower) score.
2. From the current transition system, look through all possible changes of a single transition. Make the change that improves the score the most, if one exists. Otherwise, stop. (Theoretically, this could take exponentially many steps to terminate, in practice, around 30 steps suffice.)
3. Do the two above steps NbRun times, pick the transition system T with the best score, extract a DFA from T with the best score (over all possible starting states).

This heuristic algorithm has two hyper-parameters: InitRand and NbRun. From several experiments conducted on sample 02 (with InitRand $\in \{10, 100, 1000\}$ and NbRun $\in \{10, 20, 50, 100\}$), we have chosen InitRand $= 100$ and NbRun $= 50$ as a compromise between running time and efficiency, though it is plausible that better values could be chosen (even for sample 02).

Experimental evaluation. We have implemented this heuristic algorithm in C++ and compared the performance of the symbolic algorithm starting from the trivial accepting DFA and of the symbolic algorithm starting from a DFA computed by the heuristic algorithm, referred to as the symbolic* algorithm. Since the runtime of the heuristic algorithm is not negligible, the experiments are performed on the samples (and number of states n) for which the symbolic algorithm takes at least 500s. This covers 10 samples, all with $n = 8$ except for sample 08, where $n = 7$. The results are summarized in Figure 3.

There are two main positive takeaways from these results. First, there are three samples (02, 05, 08) where our algorithm (heuristic+symbolic*) significantly outperforms the symbolic algorithm. This is a promising result with a portfolio approach in mind where both the heuristic+symbolic* and symbolic algorithms would run in parallel. Second, there are five samples (02, 05, 06, 08, 17) where the computation time of the symbolic* algorithm is much lower than that of the symbolic algorithm. This result is promising for harder instances where the runtime of the heuristic algorithm becomes more-or-less negligible compared to the runtime of the symbolic algorithm.

On the negative side, on the five other samples, the symbolic* algorithm does not outperform the symbolic algorithm, and even under-performs it for sample 27. This shows that a lower starting score (i.e., score of a starting DFA) does not always mean a better performance of the symbolic algorithm. To better identify how the starting score impacts the runtime of the symbolic algorithm, we have represented relevant data in Figure 4.

In this Figure, each data point corresponds to a run of the symbolic* algorithm[3]: in the x-axis, the starting score is depicted, in the y-axis is depicted the quotient of the runtime of the symbolic* algorithm and the mean of the runtime of the symbolic algorithm on that sample.

From this figure, we can extract the following facts: when starting from a very low score, the runtime ratio is small (as exemplified for samples 02, 05, 17 and also possibly 27); and when starting from higher scores, the impact of the score on the runtime is hard to see. These facts may suggest that the significant differences on the various samples may mostly come from the fact that for some samples the heuristic algorithm has found a starting DFA with a (very) low score, while it is not the case for the other samples. Following, improving the capacity of our heuristic algorithm at finding DFAs with (very) low score (when possible) may have the potential to greatly lower the runtime of the heuristic+symbolic* algorithm. Nonetheless, how the starting score influences the runtime of the SAT solver remains rather cryptic despite our experiments, and further investigating this relationship seems warranted to improve the efficiency of the minimizing-score heuristic approach.

6 Conclusion

In this paper, we have studied the task of learning DFAs from positive examples only with a novel word-counting perspective. The core of our studies is theoretical: we have shown the first complexity result on a decision problem directly related to this task, and we have exhibited a new algorithm solving this task with better theoretical guarantees than the state-of-the-art. On the experimental side, our ILP algorithm performs poorly, but our heuristic algorithm shows promising results as a pre-processing tool to use prior to running the symbolic algorithm. Notably, this algorithm shows that transition systems are particularly well-suited for handling this learning from positive examples task.

Our main takeaway is that a better understanding (both theoretically and experimentally) of the relationship between the language minimality of DFAs and their number of accepted words would constitute a valuable insight in the design of efficient algorithms learning DFAs from positive examples only.

Acknowledgements. This work has been financially supported by Deutsche Forschungsgemeinschaft, DFG Project number 434592664.

References

[Ang80] Dana Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21(1):46–62, 1980.

[Ang87] Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[3]To compare the scores, we have only depicted the samples where the computation of the heuristic+symbolic* algorithm is done with $n = 8$, thus we excluded sample 08.

[AP18] Florent Avellaneda and Alexandre Petrenko. Inferring DFA without negative examples. In Olgierd Unold, Witold Dyrka, and Wojciech Wieczorek, editors, *Proceedings of the 14th International Conference on Grammatical Inference, ICGI 2018, Wrocław, Poland, September 5-7, 2018*, volume 93 of *Proceedings of Machine Learning Research*, pages 17–29. PMLR, 2018.

[Bha23] Jasmin Praful Bharadiya. Artificial intelligence in transportation systems a critical review. *American Journal of Computing and Engineering*, 6(1):35–45, 2023.

[BHKL09] Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of NFA. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1004–1009, 2009.

[BN25] Benjamin Bordais and Daniel Neider. Learning dfas from positive examples only via word counting, 2025.

[CM19] Alberto Camacho and Sheila A. McIlraith. Learning interpretable models expressed in linear temporal logic. In J. Benton, Nir Lipovetzky, Eva Onaindia, David E. Smith, and Siddharth Srivastava, editors, *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2019, Berkeley, CA, USA, July 11-15, 2019*, pages 621–630. AAAI Press, 2019.

[CMR+23] Xun-Qi Chen, Chao-Qun Ma, Yi-Shuai Ren, Yu-Tian Lei, Ngoc Quang Anh Huynh, and Seema Narayan. Explainable artificial intelligence in finance: A bibliometric review. *Finance Research Letters*, 56:104145, 2023.

[CO99] Rafael C. Carrasco and José Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO Theor. Informatics Appl.*, 33(1):1–20, 1999.

[Esp21] Javier Esparza. Automata theory, an algorithmic approach. lecture notes. page 558, 2021.

[GLP06] Olga Grinchtein, Martin Leucker, and Nir Piterman. Inferring network invariants automatically. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4130 of *Lecture Notes in Computer Science*, pages 483–497. Springer, 2006.

[GNS08] Igor Griva, Stephen G Nash, and Ariela Sofer. *Linear and nonlinear optimization 2nd edition*. SIAM, 2008.

[Gol78] E. Mark Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978.

[Gur24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[HV10] Marijn Heule and Sicco Verwer. Exact DFA identification using SAT solvers. In José M. Sempere and Pedro García, editors, *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings*, volume 6339 of *Lecture Notes in Computer Science*, pages 66–79. Springer, 2010.

[ILF+23] Antonio Ielo, Mark Law, Valeria Fionda, Francesco Ricca, Giuseppe De Giacomo, and Alessandra Russo. Towards ilp-based LTL f passive learning. In Elena Bellodi, Francesca Alessandra Lisi, and Riccardo Zese, editors, *Inductive Logic Programming - 32nd International Conference, ILP 2023, Bari, Italy, November 13-15, 2023, Proceedings*, volume 14363 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 2023.

[RGB+23] Rajarshi Roy, Jean-Raphaël Gaglione, Nasim Baharisangari, Daniel Neider, Zhe Xu, and Ufuk Topcu. Learning interpretable temporal properties from positive examples only. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 6507–6515. AAAI Press, 2023.

[RS59] Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959.

[SG09] Muzammil Shahbaz and Roland Groz. Inferring mealy machines. In Ana Cavalcanti and Dennis Dams, editors, *FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands, November 2-6, 2009. Proceedings*, volume 5850 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2009.

[SLIM21] Maayan Shvo, Andrew C. Li, Rodrigo Toro Icarte, and Sheila A. McIlraith. Interpretable sequence classification via discrete optimization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9647–9656. AAAI Press, 2021.

[SO92] Andreas Stolcke and Stephen Omohundro. Hidden markov model induction by bayesian model merging. *Advances in neural information processing systems*, 5, 1992.

[WGY24] Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural networks using queries and counterexamples (extended version). *Machine Learning*, 113(5):2877–2919, 2024.

[ZWL22] Yiming Zhang, Ying Weng, and Jonathan Lund. Applications of explainable artificial intelligence in diagnosis and surgery. *Diagnostics*, 12(2):237, 2022.

A Word-counting and language minimality

Here, we establish the folk result Theorem 1. (One can alternatively find a proof and statement in and proof in [Esp21, Exercice 37].) The proof of this theorem relies on the lemma* below.

Lemma 13. *Consider a DFA $\mathcal{A} \in \mathsf{DFA}(\Sigma)$ and let $n := |\mathcal{A}|$. For all pairs of states $(q, q') \in Q^2$, we say that a word $w \in \Sigma^*$ is* (q, q')-distinguishing *if $(\delta(q, w), \delta(q', w)) \in F \times (Q \setminus F) \cup (Q \setminus F) \times F$. Then, for all pairs of states $(q, q') \in Q^2$, if there exists a (q, q')-distinguishing word, there is one of length at most $n - 2$.*

Proof. For all $i \in \mathbb{N}$, we consider the relation $\sim_i \subseteq Q^2$ defined by, for all pairs of states $(q, q') \in Q^2$, $q \sim_i q'$ if and only if there are no (q, q')-distinguishing word of length at most i. This relation $\sim_i$ is trivially reflexive and symmetric, it is also transitive: if $q \sim_i q' \sim_i q''$, for some $q, q', q'' \in Q$, then for all words $w \in \Sigma^{\leq i}$, we have $\delta(q'', w) \in F \Leftrightarrow \delta(q', w) \in F \Leftrightarrow \delta(q, w) \in F$. Thus, we have $q \sim_i q''$. In fact, the relation $\sim_i$ is an equivalence relation Q. We let $C_i^1, \ldots, C_i^{k_i} \subseteq Q$ denote the sets of $\sim_i$-equivalence classes, with $k_i \geq 1$ denoting the number of $\sim_i$-equivalence classes.

Now, we make three straightforward observations:

- $k_0 = 2$, with $\{C_0^1, C_0^2\} = \{F, Q \setminus F\}$.
- For all $i \in \mathbb{N}$, $k_i \leq n$;
- For all $i \in \mathbb{N}$, $\sim_{i+1} \subseteq \sim_i$.

We have in addition the two, slightly less straightforward, facts:

- For all $i \in \mathbb{N}$, if $k_i = k_{i+1}$, then $\sim_i = \sim_{i+1}$. Indeed, letting $k := k_i = k_{i+1}$, since $\sim_{i+1} \subseteq \sim_i$, there is a function $f : [\![1, k]\!] \to [\![1, k]\!]$ such that, for all $j \in [\![1, k]\!]$, we have $C_{i+1}^j \subseteq C_i^{f(j)}$. Since $\cup_{j \in [\![1,k]\!]} C_i^j = Q = \cup_{j \in [\![1,k]\!]} C_{i+1}^j$, the function f is necessarily surjective, and thus bijective, and for all $j \in [\![1, k]\!]$, we have $C_{i+1}^j = C_i^{f(j)}$. That is, the relations $\sim_i$ and $\sim_{i+1}$ have the same equivalence classes, and are thus equal.
- For all $i \in \mathbb{N}$, if $\sim_i = \sim_{i+1}$, then for all $k \geq i$, we have $\sim_i = \sim_k$. Indeed, assuming that $\sim_i = \sim_{i+1}$, let us show by induction on $k \geq i$ that $\sim_k = \sim_{k+1}$. This directly holds for $k = i$. Assume now that this holds for some $k \geq i$. Let us show that $\sim_{k+1} = \sim_{k+2}$. We have $\sim_{k+2} \subseteq \sim_{k+1}$. Furthermore, consider a pair of states $(q, q') \in Q^2$ such that $q \sim_{k+1} q'$. Assume towards a contradiction that there is a (q, q')-distinguishing word $w \in \Sigma^{k+2}$. Let us write $w = \alpha \cdot v$, for some $\alpha \in \Sigma$ and $v \in \Sigma^{k+1}$. Hence, the word $v \in \Sigma^{k+1}$ is $(\delta(q, \alpha), \delta(q', \alpha))$-distinguishing. Hence, $\delta(q, \alpha) \not\sim_{k+1} \delta(q', \alpha)$ and $\delta(q, \alpha) \sim_k \delta(q', \alpha)$. Thus, there is a $(\delta(q, \alpha), \delta(q', \alpha))$-distinguishing $v' \in \Sigma^k$. Then, the word $\alpha \cdot v' \in \Sigma^{k+1}$ is (q, q')-distinguishing, thus $q \not\sim_{k+1} q'$. Hence the contradiction. In fact, there is no (q, q')-distinguishing word of length $k + 2$, therefore $q \sim_{k+2} q'$.

Now, consider the least $m \in \mathbb{N}$ such that $\sim_m = \sim_{m+1}$ (which exists since $\sim_0 \supseteq \sim_1 \supseteq \ldots$). We have $2 = k_0 < k_1 < \ldots < k_m \leq n$, thus $m \leq n - 2$. (In fact, the relation $\sim_m$ corresponds to the well-known Myhill-Nerode equivalence relation.)

Consider now any pair of states $(q, q') \in Q^2$. Assume that there exists a (q, q')-distinguishing word. Therefore, there is some $k \in \mathbb{N}$ such that $q \not\sim_k q'$. Therefore, since $\sim_0 \supseteq \sim_1 \supseteq \ldots \supseteq \sim_m = \sim_{m+1} = \ldots$, we have $\sim_m = \cap_{i \in \mathbb{N}} \sim_i$, and thus $q \not\sim_m q'$. Hence, by definition $\sim_m$, there exists a (q, q')-distinguishing word of length at most $m \leq n - 2$. □

The proof of Theorem 1 is now straightforward.

Proof. Let $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, \delta, F)$ and $\mathcal{A}' = (Q', \Sigma, q'_{\mathsf{init}}, \delta', F')$. Without loos of generality, we may assume that $Q \cap Q' = \emptyset$. Then, let $\mathcal{A}'' = (Q'', \Sigma, q''_{\mathsf{init}}, \delta'', F'') \in \mathsf{DFA}(\Sigma)$ be the DFA obtained as the (disjoint) union of $\mathcal{A}$ and $\mathcal{A}'$, that is:

- $Q'' = Q \cup Q'$;
- $q''_{\mathsf{init}} \in Q''$ is defined arbitrarily;
- for all $q \in Q$ (resp. Q') and $\alpha \in \Sigma$, we have $\delta''(q, \alpha) := \delta(q, \alpha) \in Q$ (resp. $\delta''(q, \alpha) := \delta'(q, \alpha) \in Q'$);

- $F'' = F \cup F'$.

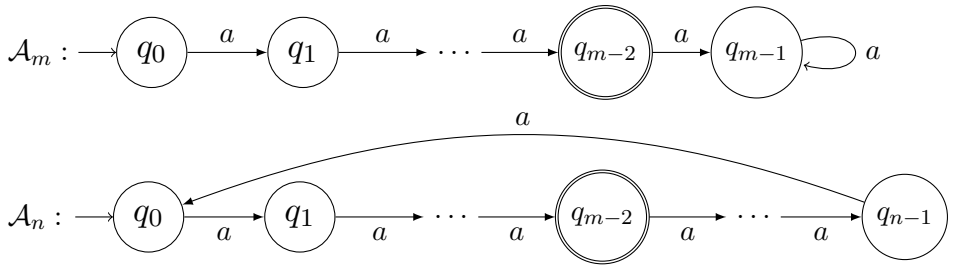
Then, we have $|\mathcal{A}''| = |\mathcal{A}|+|\mathcal{A}'|$, and since $\mathcal{L}(\mathcal{A}) \neq \mathcal{L}(\mathcal{A}')$, then there are $(q_{\mathsf{init}}, q'_{\mathsf{init}})$-distinguishing words $u \in \Sigma^*$, of length at most $|\mathcal{A}''| - 2$, by Lemma* 13. By definition, such distinguishing words $u \in \Sigma^{\leq |\mathcal{A}|+|\mathcal{A}'|-2}$ satisfy $u \in (\mathcal{L}(\mathcal{A}) \setminus \mathcal{L}(\mathcal{A}')) \cup (\mathcal{L}(\mathcal{A}') \setminus \mathcal{L}(\mathcal{A}))$. □

A.1 Two tightness results

We would like to consider two tightness results: one about Theorem 1, and one about Proposition 1. Specifically, we show the proposition* below. Note that his is a also folk result, see e.g., this math overflow thread.

Proposition 14. *Consider a unary alphabet $\Sigma = \{a\}$. For all $m, n \geq 1$, there are two DFAs $\mathcal{A}_m, \mathcal{A}_n \in \mathsf{DFA}(\Sigma)$ such that $|\mathcal{A}_m| = m$, $|\mathcal{A}_n| = n$, and the length of the smallest word in $(\mathcal{L}(\mathcal{A}_n) \setminus \mathcal{L}(\mathcal{A}_m)) \cup (\mathcal{L}(\mathcal{A}_m) \setminus \mathcal{L}(\mathcal{A}_n))$ is $m + n - 2$.*

Proof. The result is direct when $m = n = 1$ and $\{m, n\} = \{1, 2\}$. Assume now that $m, n \geq 2$. Let us assume without loss of generality that $m \leq n$. We consider the DFAs $\mathcal{A}_m, \mathcal{A}_n$ described below, where the double circled states are accepting:



Clearly, we have $|\mathcal{A}_m| = m$ and $|\mathcal{A}_n| = n$. Furthermore, we have:

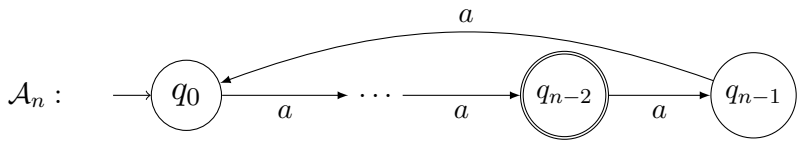
$$\mathcal{L}(\mathcal{A}_m) = \{a^{m-2}\} \text{ and } \mathcal{L}(\mathcal{A}_n) = \{a^x \mid x = m - 2 \mod n\}$$

Thus, $(\mathcal{L}(\mathcal{A}_n) \setminus \mathcal{L}(\mathcal{A}_m)) \cup (\mathcal{L}(\mathcal{A}_m) \setminus \mathcal{L}(\mathcal{A}_n)) = \{a^x \mid x \neq m-2,\ x = m-2 \mod n\}$; the smallest word in that set is a^{m-2+n}, of length $m + n - 2$. □

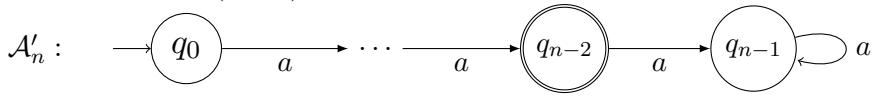
We also establish the proposition* below.

Proposition 15. *Consider a unary alphabet $\Sigma = \{a\}$. For all $n \geq 2$, there is a DFA $\mathcal{A}_n \in \mathsf{DFA}(\Sigma)$ and a language $L_n \subseteq \Sigma^*$ such that $\mathcal{A}_n \in \mathsf{Rec}(L_n, n)$, A_n is $\mathsf{Rec}(L_n, n)$-minimal w.r.t. $\prec_{2n-3}$, but A_n is not $\mathsf{Rec}(L_n, n)$-minimal w.r.t. $\prec_{\mathcal{L}}$.*

Proof. Consider the language $L_n := \{a^{n-2}\} \subseteq \Sigma^*$ and the DFA $\mathcal{A}_n$ depicted below:



We have $|\mathcal{A}_n| = n$, $L_n \subseteq \mathcal{L}(\mathcal{A}_n) = \{a^x \mid x = n - 2 \mod n\}$, and $\mathcal{L}(\mathcal{A}_n) \cap \Sigma^{\leq 2n-3} = \{a^{n-2}\}$. Therefore, A_n is $\mathsf{Rec}(L_n, n)$-minimal w.r.t. $\prec_{2n-3}$. Consider now the DFA $\mathcal{A}'_n$ depicted below:



We have $|\mathcal{A}'_n| = n$ and $\mathcal{L}(\mathcal{A}_n) = L_n$. Therefore, A_n is not $\mathsf{Rec}(L_n, n)$-minimal w.r.t. $\prec_{\mathcal{L}}$. □

B The NP-hardness proof from Theorem 2

Before we go on to the NP-hardness proof, let us first consider the (simple) proof of Lemma 1.

Proof. For all $k \in \mathbb{N}$, we have:

$$\sum_{q' \in Q} \mathsf{N}(q', k) \cdot |\{\alpha \in \Sigma \mid \delta(q', \alpha) = q\}| = \sum_{q' \in Q} |\{w \in \Sigma^k \mid \delta^*(q_{\mathsf{init}}, w) = q'\}| \cdot |\{\alpha \in \Sigma \mid \delta^*(q', w) = q\}|$$
$$= |\{w \cdot \alpha \in \Sigma^{k+1} \mid \delta^*(q_{\mathsf{init}}, w) = q\}| = \mathsf{N}(q, k+1)$$

Furthermore, we have:

$$\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq m} = \{w \in \Sigma^{\leq m} \mid \delta^*(q_0, w) \in F\} = \bigcup_{i=0}^{m} \{w \in \Sigma^i \mid \delta^*(q_0, w) \in F\}$$
$$= \bigcup_{i=0}^{m} \bigcup_{q \in F} \{w \in \Sigma^i \mid \delta^*(q_0, w) = q\}$$

Lemma 1 follows. □

Now, let us turn to the NP-hardness proof. Let us first introduce and recall a few notations. For each $u, v \in \Sigma^*$, we write $u \cdot v \in \Sigma^*$ the concatenation of the word u followed by the word v. For all $u \in \Sigma^*$ and $V \subseteq \Sigma^*$, we let $u \cdot V$ and $V \cdot u$ denote the sets $u \cdot V := \{u \cdot v \mid v \in V\} \subseteq \Sigma^*$ and $V \cdot u := \{v \cdot u \mid v \in V\} \subseteq \Sigma^*$. A word $u \in \Sigma^*$ is a prefix of a word $v \in \Sigma^*$ if there is some $w \in \Sigma^*$ such that $u \cdot w = v$. This is denoted $u \sqsubseteq v$.

Our goal is to establish the lemma* below.,

Lemma 16. *The decision problem 1 is* NP*-hard, even with k given in unary, and an alphabet Σ of size 2.*

For the remainder of this section, we consider an instance $(X, \mathcal{C})$ of Problem 2 (the All-Pos-Neg SAT problem) with $X = \{x_i \mid i \in [\![1, r]\!]\}$ the set of variables, and $\mathcal{C} = \{C_j \mid j \in [\![1, s]\!]\} = \mathcal{C}_+ \uplus \mathcal{C}_-$ the set of clauses. The alphabet that we consider is $\Sigma := \{a, b\}$.

Our goal is to define a set of positive words $\mathcal{P}$ and three integers n, k and m such that $(X, \mathcal{C})$ is a positive instance of Problem 2 if and only if there is a $(\mathcal{P}, n, k, m)$-suitable DFA $\mathcal{A}$, i.e., such that 1) $|\mathcal{A}| \leq n$; 2) $\mathcal{P} \subseteq \mathcal{L}(\mathcal{A})$; and 3) $|\Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P})| \leq k$. Note that this third condition is little different from the one considered in the paper: the maximal length of the words consider is an integer m. We will then ensure that our reduction works also for the case $m = 2n - 2$.

Let us now formally define the positive set of words $\mathcal{P}$ that we consider. This set of words is parameterized not only by an instance $(X, \mathcal{C})$ of Problem 2, but also by several integers that we do not fix yet. Later (namely, in Lemmas* 18 and 22), we will show that, assuming that these integers satisfy some (in)equalities, then the set of words that we define below is suitable for a reduction. As there are many words, these may not be easy to parse. In the next subsection, we will illustrate the DFA construction on an example, which should make it easier to follow how these sets are defined.

Definition 17. *Consider an instance $(X, \mathcal{C})$ of the Problem 2. Let $k, d, M, T \in \mathbb{N}$. We define the following sets of words:*

- *First of all, let us introduce a notation for a set of words: for $\alpha \neq \beta \in \{a, b\}$ and $i \in \mathbb{N}$, we let $U_i(\alpha) := \{\beta^x \cdot \alpha \mid 1 \leq x \leq i + 1\}$;*

- *We let*
$$\mathcal{P}_\top(k,d) := a \cdot a \cdot U_{k+1}(a) \cdot \{a^d \quad , a^{d+1} \cdot u_a \mid u_a \in U_{k+1}(a)\}$$
and
$$\mathcal{P}_\bot(k,d) := a \cdot b \cdot U_{k+1}(a) \cdot \{a^{d+1}, a^{d+1} \cdot u_a \mid u_a \in U_{k+1}(a)\}$$
These are the $\top$- and $\bot$-sets of words. As in the example of the paper, $\top$-words start with $a \cdot a$, and $\bot$-words start with $a \cdot b$.

- *For all $i \in [\![1, r]\!]$, we let* $\mathsf{App}(x_i) := \{1 \leq j \leq s \mid x_i \in C_j\}$ *and* $\mathsf{NotApp}(x_i) := \{1 \leq j \leq s \mid x_i \notin C_{j'}\}$ *and* $\mathsf{Ind}(x_i) := \mathsf{App}(x_i) \cup \{s + \sigma \mid \sigma \in \mathsf{NotApp}(x_i)\} \cup \{s + s + i + t \cdot r \mid t \in [\![0, T-1]\!]\})\}$. *Then, we let:*
$$\mathcal{P}_{\mathsf{Var}}(X, \mathcal{C}, k, d, T, M, i) := b \cdot a \cdot U_M(b) \cdot b^i \cdot a \cdot b^d \cdot \{a^{d+1} \cdot u_a,\ b^\sigma,\ b^{s+s+T\cdot r} \cdot u_b \mid u_a \in U_{k+1}(a),\ \sigma \in \mathsf{Ind}(x_i),\ u_b \in U_{k+1}(b)\}$$
These are the variable words, which all start with $b \cdot a$.

- *For all $j \in [\![1, s]\!]$, we let:*
$$\mathcal{P}_{\mathsf{Cl}}(X, \mathcal{C}, k, d, T, j) := b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \{b^j, b^{s+s+T\cdot r} \cdot u_b \mid u_b \in U_{k+1}(b)\}$$
and
$$\mathcal{P}^{\mathsf{acc}}_{\mathsf{Cl}}(k, d, j) := b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \begin{cases} \{a^d \quad , a^{d+1} \cdot u_a \mid u_a \in U_{k+1}(a)\} & \text{if } C_j \in \mathcal{C}_+ \\ \{a^{d+1}, a^{d+1} \cdot u_a \mid u_a \in U_{k+1}(a)\} & \text{if } C_j \in \mathcal{C}_- \end{cases}$$
These are the clause words, which all start with $b \cdot b$. Note that clause words in $\mathcal{P}^{\mathsf{acc}}_{\mathsf{Cl}}(k, d, j)$ correspond to those defined in the paper that encode that some clause is positive while some other clause is negative.

We can now define the set of words that we consider:
$$\mathcal{P}(X, \mathcal{C}, k, d, T, M) := \left(\bigcup_{i \in [\![1,r]\!]} \mathcal{P}_{\mathsf{Var}}(X, \mathcal{C}, k, d, T, M, i) \right) \cup \left(\bigcup_{j \in [\![1,s]\!]} \mathcal{P}_{\mathsf{Cl}}(X, \mathcal{C}, k, d, T, j) \cup \mathcal{P}^{\mathsf{acc}}_{\mathsf{Cl}}(k, d, j) \right) \cup \mathcal{P}_\top(k, d) \cup \mathcal{P}_\bot(k, d)$$

We also define two integers that will be particularly useful when discussing the size of DFAs:
$$\omega_1(X, d) := d \cdot (2 + r)$$
and
$$\omega_2(X, \mathcal{C}, k, T, M) := 18 + s + M + 4 \cdot k + r + 2 \cdot r \cdot s + r^2 \cdot T$$

B.1 If there is a satisfying valuation, then there is a suitable DFA

Let us first show that, assuming that n and k are large enough, if there exists a valuation satisfying $(X, \mathcal{C})$, then there is a DFA satisfying conditions a), b), c). This is stated in the lemma* below.

Lemma 18. *Consider a positive instance $(X, \mathcal{C})$ of Problem 2. Let $n, k, d, M, T \in \mathbb{N}$ and assume that $n \geq \omega_1(r, d) + \omega_2(X, \mathcal{C}, k, T, M)$ and $k \geq M \cdot r + s \cdot (s + T - 1)$. Then, for all $m \in \mathbb{N}$, there is a $(\mathcal{P}, n, k, m)$-suitable DFA.*

To prove this lemma* we define a DFA parameterized by the above integers and a satisfying valuation of the variables. This definition is illustrated on an example afterwards. We invite the reader to glimpse at Figure 5 to get an idea on the shape of the DFA that we define.

Definition 19. *Consider an instance $(X, \mathcal{C})$ of Problem 2 and a valuation $\nu : X \to \{\top, \bot\}$ satisfying $(X, \mathcal{C})$. We define the DFA $\mathcal{A}(X, \mathcal{C}, k, d, T, M, \nu) = (Q, \Sigma, q_{\mathsf{init}}, \delta, F)$ where:*

$$Q := Q_1 \uplus Q_{\mathsf{Cl}} \uplus Q_{\mathsf{Var}} \uplus Q_\top \uplus Q_\bot \uplus Q_{u_a} \uplus Q_{u_b}$$

with

$$\begin{aligned}
Q_1 &:= \{q_{\mathsf{init}}, q_a, q_b, q_{\mathsf{acc}}, q_{\mathsf{rej}}\} \\
Q_{\mathsf{Cl}} &:= \{q_{\mathsf{Cl}}, Q_{C_j} \mid j \in [\![1, s]\!]\} \\
Q_{\mathsf{Var}} &:= \{q_{\mathsf{Var}}, q^\sigma_{\mathsf{Var},u}, q_X \mid \sigma \in [\![1, M]\!]\} \uplus \cup_{i \in [\![1,r]\!]} Q_{x_i} \\
Q_{x_i} &:= \{q_{x_i}, q^\sigma_{x_i}, (x_i \in C_j), (x_i \notin C_j), (x_i, l, t) \mid \sigma \in [\![1, d]\!], j \in [\![1, s]\!], l \in [\![1, r]\!], t \in [\![0, T-1]\!]\} \\
Q_\top &:= \{q_\top, q^\sigma_{\top,u}, q^{\sigma'}_\top \mid \sigma \in [\![1, k+1]\!], \sigma' \in [\![0, d+1]\!]\} \\
Q_\bot &:= \{q_\bot, q^\sigma_{\bot,u}, q^{\sigma'}_\bot \mid \sigma \in [\![1, k+1]\!], \sigma' \in [\![0, d+1]\!]\} \\
Q_{u_a} &:= \{q^\sigma_{u_a} \mid \sigma \in [\![1, k+1]\!]\} \\
Q_{u_b} &:= \{q^\sigma_{u_b} \mid \sigma \in [\![1, k+1]\!]\}
\end{aligned}$$

Before we finish the definition of this DFA, let us count its number of states. We have:

$$\begin{aligned}
|Q| &= \underbrace{5}_{|Q_1|} + \underbrace{s+1}_{|Q_{\mathsf{Cl}}|} + \underbrace{2 + M + r \cdot (1 + d + 2 \cdot s + r \cdot T)}_{|Q_{\mathsf{Var}}|} + \underbrace{d+k+4}_{|Q_\top|} + \underbrace{d+k+4}_{|Q_\bot|} + \underbrace{k+1}_{|Q_{u_a}|} + \underbrace{k+1}_{|Q_{u_b}|} \\
&= d \cdot (2 + r) + 18 + s + M + 4 \cdot k + r + 2 \cdot r \cdot s + r^2 \cdot T \\
&= \omega_1(X, d) + \omega_2(X, \mathcal{C}, k, T, M)
\end{aligned}$$

The set of final states $F \subseteq Q$ is equal to:

$$F := \{q^d_\top, q^{d+1}_\bot, q_{\mathsf{acc}}, \} \uplus \bigcup_{i \in [\![1,r]\!]} \left(\bigcup_{j \in \mathsf{App}(x_i)} (x_i \in C_j) \cup \bigcup_{j \in \mathsf{NotApp}(x_i)} (x_i \notin C_j) \cup \bigcup_{t \in [\![0, T-1]\!]} (x_i, i, t) \right)$$

Let us now define the transition function δ. We start by defining those transitions that do not depend on the valuation $\nu : X \to \{\top, \bot\}$. First, we define the initial transitions leading from q_{init} to $q_\top, q_\bot, q_{\mathsf{Var}}, q_{\mathsf{Cl}}$. Specifically, we have:

$$\delta(q_{\mathsf{init}}, a) := q_a,\ \delta(q_{\mathsf{init}}, b) := q_b,\ \delta(q_a, a) := q_\top,\ \delta(q_a, b) := q_\bot,\ \delta(q_b, a) := q_{\mathsf{Var}},\ \delta(q_b, b) := q_{\mathsf{Cl}}$$

Let us now define the other transitions (that still do not depend on the valuation ν). To improve the readability of the definition of these other transitions, we describe them in a somewhat graphical manner. Specifically:

- *for all $q, q' \in Q$ and $\alpha \in \Sigma$, we write $q \xrightarrow{\alpha} q'$ to express that $\delta(q, \alpha) := q'$;*
- *for all $q_0, q_1, \ldots, q_\theta \in Q$ and $\alpha \neq \beta \in \Sigma$, we write $q_0 \xrightarrow{\alpha} q_1 \xrightarrow{\alpha} \ldots \xrightarrow{\alpha} q_\theta \overset{\beta}{\Longrightarrow} q$ to express that, for all $0 \leq \sigma \leq \theta - 1$, we have $\delta(q_\sigma, \alpha) := q_{\sigma+1}$ and $\delta(q_\sigma, \beta) := q$, while $\delta(q_\theta, \beta) := q$ (but we do not have $\delta(q_\theta, \alpha) := q$).*

Furthermore, for almost all transitions $q \xrightarrow{\alpha} q'$ that we describe below, there is some $\bullet \in \{\mathsf{Cl}, \mathsf{Var}, \top, \bot, u_a, u_b\}$ such that $q, q' \in Q_\bullet$. We therefore group these transitions by the set of states $Q_\bullet$ to which all states, but the ones we depict in boxes, belong.

$$
\begin{aligned}
Q_{\mathsf{Cl}} &: q_{\mathsf{Cl}} \xrightarrow{a} q_{C_1} \xrightarrow{a} q_{C_2} \xrightarrow{a} \dots \xrightarrow{a} q_{C_s} \\
Q_{\mathsf{Var}} &: q_{\mathsf{Var}} \xrightarrow{a} q^1_{\mathsf{Var},u} \\
&\quad q^1_{\mathsf{Var},u} \xrightarrow{a} q^2_{\mathsf{Var},u} \xrightarrow{a} \dots \xrightarrow{a} q^M_{\mathsf{Var},u} \xRightarrow{b} q_X \xrightarrow{b} q_{x_1} \xrightarrow{b} q_{x_2} \xrightarrow{b} \dots \xrightarrow{b} q_{x_r} \\
&\quad \forall i \in [\![1, r]\!] : \; q_{x_i} \xrightarrow{a} q_{x_i^0} \xrightarrow{b} q_{x_1^1} \xrightarrow{b} \dots \xrightarrow{b} q_{x_i^d} \xrightarrow{b} \\
&\qquad\qquad (x_i \in C_1) \xrightarrow{b} (x_i \in C_2) \xrightarrow{b} \dots \xrightarrow{b} (x_i \in C_s) \xrightarrow{b} \\
&\qquad\qquad (x_i \notin C_1) \xrightarrow{b} (x_i \notin C_2) \xrightarrow{b} \dots \xrightarrow{b} (x_i \notin C_s) \xrightarrow{b} \\
&\qquad\qquad (x_i, 1, 0) \xrightarrow{b} (x_i, 2, 0) \xrightarrow{b} \dots \xrightarrow{b} (x_i, r, 0) \xrightarrow{b} \\
&\qquad\qquad (x_i, 1, 1) \xrightarrow{b} (x_i, 2, 1) \xrightarrow{b} \dots \xrightarrow{b} (x_i, r, 1) \xrightarrow{b} \\
&\qquad\qquad \dots \xrightarrow{b} \\
&\qquad\qquad (x_i, 1, T-1) \xrightarrow{b} (x_i, 2, T-1) \xrightarrow{b} \dots \xrightarrow{b} (x_i, r, T-1) \xrightarrow{a} \boxed{q^1_{u_b}} \\
Q_\top &: q_\top \xrightarrow{a} q^1_{\top,u} \\
&\quad q^1_{\top,u} \xrightarrow{b} q^2_{\top,u} \xrightarrow{b} \dots \xrightarrow{b} q^{k+1}_{\top,u} \xRightarrow{a} q^1_\top \xrightarrow{a} q^2_\top \xrightarrow{a} \dots \xrightarrow{a} q^{d+1}_\top \xrightarrow{a} \boxed{q^1_{u_a}} \\
Q_\bot &: q_\bot \xrightarrow{a} q^1_{\bot,u} \\
&\quad q^1_{\bot,u} \xrightarrow{b} q^2_{\bot,u} \xrightarrow{b} \dots \xrightarrow{b} q^{k+1}_{\bot,u} \xRightarrow{a} q^1_\bot \xrightarrow{a} q^2_\bot \xrightarrow{a} \dots \xrightarrow{a} q^{d+1}_\bot \xrightarrow{a} \boxed{q^1_{u_a}} \\
Q_{u_a} &: q^1_{u_a} \xrightarrow{b} q^2_{u_a} \xrightarrow{b} \dots \xrightarrow{b} q^{k+1}_{u_a} \xRightarrow{a} \boxed{q_{\mathsf{acc}}} \\
Q_{u_b} &: q^1_{u_b} \xrightarrow{a} q^2_{u_b} \xrightarrow{a} \dots \xrightarrow{a} q^{k+1}_{u_b} \xRightarrow{b} \boxed{q_{\mathsf{acc}}}
\end{aligned}
$$

Let us now describe the transitions that depend on the valuation ν. Since ν satisfies $(X, \mathcal{C})$, for all $j \in [\![1, s]\!]$, there is some $i_j \in [\![1, r]\!]$ such that $x_{i_j} \in C_j$ and, if $C_j \in \mathcal{C}_+$ (resp. $C_j \in \mathcal{C}_-$), then $v(x_{i_j}) = \top$ (resp. $v(x_{i_j}) = \bot$). Then, we have transitions from states in Q_{Cl} to states in Q_{Var}, and from states in Q_{Var} to states in $Q_\top \cup Q_\bot$. Specifically:

- *From Q_{Cl} to Q_{Var}: for all $j \in [\![1, s]\!]$, $\delta(q_{C_j}, b) := q^0_{x_{i_j}}$.*
- *From Q_{Var} to $Q_\top \cup Q_\bot$: for all $i \in [\![1, r]\!]$, $\delta(q^d_{x_i}, a) := q^1_{\nu(x_i)}$.*

Finally, for all $q \in Q$ and $\alpha \in \Sigma$, if $\delta(q, \alpha)$ is not defined above, then we have $\delta(q, \alpha) := q_{\mathsf{rej}}$. In particular, this implies that $\delta(q_{\mathsf{rej}}, a) = \delta(q_{\mathsf{rej}}, b) = \delta(q_{\mathsf{acc}}, a) = \delta(q_{\mathsf{acc}}, b) := q_{\mathsf{rej}}$, thus q_{rej} is a self-looping sink state from which all words are rejected.

We illustrate the above definition in Example* 20.

Let us now proceed to the proof of Lemma* 18. This proof is not hard, as it essentially consists in computing the language accepted by the DFA of Definition 19, however it is quite long since we compute this language step by step.

Proof. Consider a positive instance $(X, \mathcal{C})$ of Problem 2 and let $n, k, d, M, T \in \mathbb{N}$ be such that $n \geq \omega_1(r, d) + \omega_2(X, \mathcal{C}, k, T, M)$ and $k \geq M \cdot r + s \cdot (s + T - 1)$. Let $m \in \mathbb{N}$.

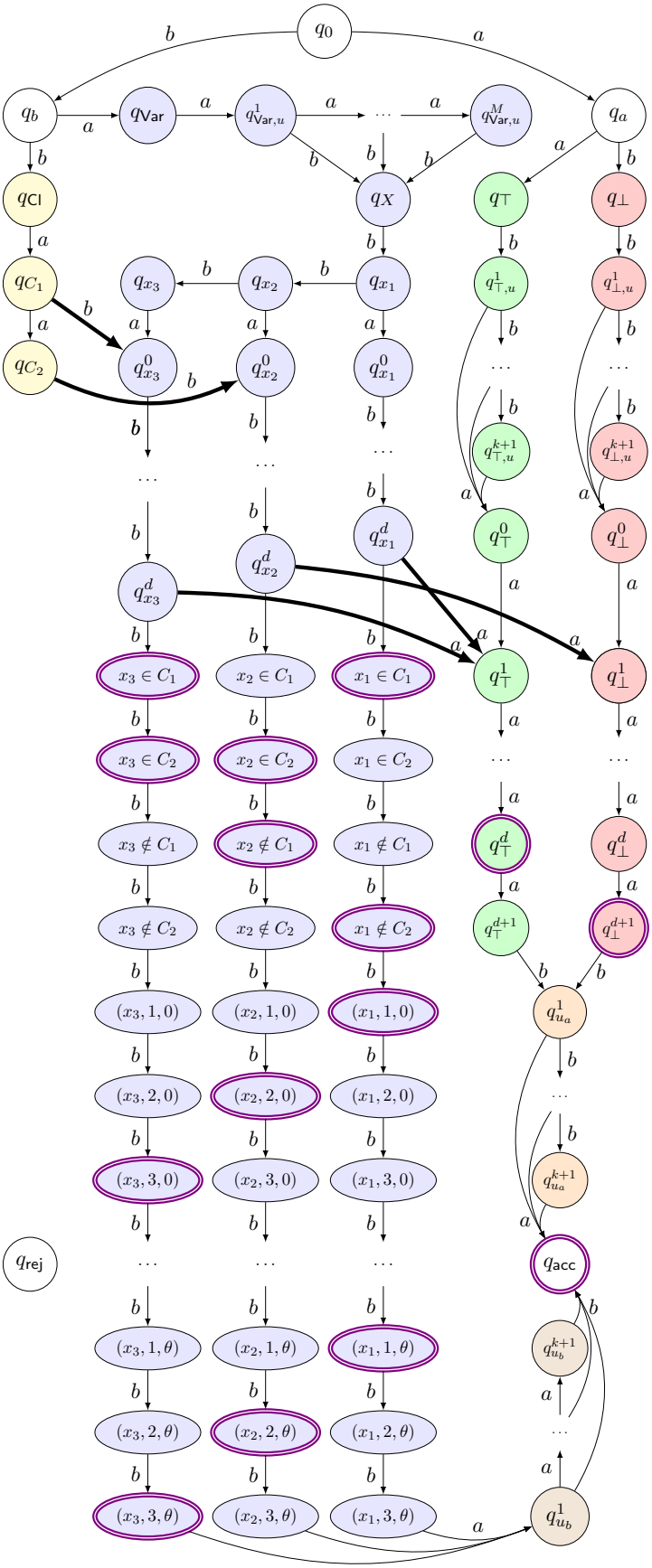


Figure 5: A DFA $\mathcal{A}(X, \mathcal{C}, k, d, T, M, \nu)$.

Example 20. *Consider the set of variables $X = \{x_1, x_2, x_3\}$, and a positive clause $C_1 = \{x_1, x_3\} \in \mathcal{C}_+$ and a negative clause $C_2 = \{x_2, x_3\} \in \mathcal{C}_-$. Consider the valuation $\nu : X \to \{\top, \bot\}$ such that $\nu(x_1) = \nu(x_3) := \top$, and $\nu(x_2) := \bot$. This valuation satisfies $(X, \mathcal{C})$ with $i_1 := 3$ (since $x_3 \in C_1$, $C_1 \in \mathcal{C}_+$, and $\nu(x_3) = \top$), and $i_2 := 2$ (since $x_2 \in C_2$, $C_2 \in \mathcal{C}_-$, and $\nu(x_2) = \bot$). Given some integers k, d, T, M, the DFA $\mathcal{A} := \mathcal{A}(X, \mathcal{C}, k, d, T, M, \nu)$ of Definition 19 is depicted in Figure 5 (with $\theta := T - 1$). We let $\mathcal{P} := \mathcal{P}(X, \mathcal{C}, k, d, T, M)$.*

In this DFA, states are colored in white, yellow, blue, green, red, orange, and brown according to the set of states in $\{Q_1, Q_{\mathsf{Cl}}, Q_{\mathsf{Var}}, Q_\top, Q_\bot, Q_{u_a}, Q_{u_b}\}$ to which they belong. We did not depict any transition to the self-looping sink state q_{rej}.

Accepting states are depicted with a double-circled violet border. For instance: $(x_1 \in C_1) \in Q$ is accepting (and thus $(x_1 \notin C_1) \in Q$ is not accepting) since $x_1 \in C_1$, while $(x_2 \in C_1) \in Q$ is not accepting (and thus $(x_2 \notin C_1) \in Q$ is accepting) since $x_2 \notin C_2$. We also have $q^d_\top$ accepting and $q^{d+1}_\top$ not accepting, while $q^d_\bot$ is not accepting and $q^{d+1}_\bot$ is accepting.

The transitions that depend on the valuation ν are depicted in bold, with the transitions from Q_{Cl} to Q_{Var} defined by the indices $i_j \in [\![1, 3]\!]$, for $j \in [\![1, 2]\!]$, while with the transitions from Q_{Var} to $Q_\top \cup Q_\bot$ are given directly by the valuation ν. Since this valuation ν satisfies $(X, \mathcal{C})$, we have that:

- *the word $b \cdot b \cdot a^1 \cdot b \cdot b^d \cdot a^d \in \mathcal{P}$ (since $C_1 \in \mathcal{C}_+$) is accepted by $\mathcal{A}$ since $\delta(q_{C_1}, b) = q^0_{x_3}$, and $\delta(q^d_{x_3}, a) = q^1_\top$;*
- *the word $b \cdot b \cdot a^2 \cdot b \cdot b^d \cdot a^{d+1} \in \mathcal{P}$ (since $C_2 \in \mathcal{C}_-$) is accepted by the DFA $\mathcal{A}$ since $\delta(q_{C_2}, b) = q^0_{x_2}$, and $\delta(q^d_{x_2}, a) = q^1_\bot$.*

By definition, there is some valuation $\nu : X \to \{\top, \bot\}$ satisfying $(X, \mathcal{C})$. Consider the DFA $\mathcal{A} := \mathcal{A}(X, \mathcal{C}, k, d, T, M, \nu) = (Q, \Sigma, q_{\mathsf{init}}, \delta, F)$ from Definition 19. We have $|Q| \leq n$ by assumption. Let us now compute the language $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ and relate it to the set of positive words $\mathcal{P}(X, \mathcal{C}, k, d, T, M)$ from Definition 17. To compute this language, we will, as intermediary steps, compute sets of words accepted from various states $q \in Q$, i.e. the language:

$$\mathcal{L}(\mathcal{A}, q) := \{w \in \Sigma^* \mid \delta^*(q, w) \in F\}$$

Let us start by computing the language $\mathcal{L}(\mathcal{A}, q^1_{u_a})$ and $\mathcal{L}(\mathcal{A}, q^1_{u_b})$. The state q_{acc} is the single accepting state reachable from the state $q^1_{u_a}$. From the definition of the transition function δ for states in Q_{u_a}, it is clear that we have:

$$\mathcal{L}(\mathcal{A}, q^{k+1}_{u_a}) = \{a\}$$
$$\forall \sigma \in [\![2, k+1]\!],\ \mathcal{L}(\mathcal{A}, q^{\sigma-1}_{u_a}) = \{a\} \cup b \cdot \mathcal{L}(\mathcal{A}, q^{\sigma}_{u_a})$$

Hence, we have:

$$\mathcal{L}(\mathcal{A}, q^1_{u_a}) = \{b^\sigma \cdot a \mid \sigma \in [\![0, k]\!]\} \text{ and } b \cdot \mathcal{L}(\mathcal{A}, q^1_{u_a}) = U_{k+1}(a)$$

Similarly, we have:

$$\mathcal{L}(\mathcal{A}, q^1_{u_b}) = \{a^\sigma \cdot b \mid \sigma \in [\![0, k]\!]\} \text{ and } a \cdot \mathcal{L}(\mathcal{A}, q^1_{u_b}) = U_{k+1}(b)$$

In a similar manner, letting $\mathcal{L}(\mathcal{A}, q, q') := \{w \in \Sigma^* \mid \delta^*(q, w) = q'\}$ for all $q, q' \in Q$, we have $\mathcal{L}(\mathcal{A}, q_\top, q^0_\top) = \mathcal{L}(\mathcal{A}, q_\bot, q^0_\bot) = U_{k+1}(a)$, and $\mathcal{L}(\mathcal{A}, q_{\mathsf{Var}}, q_X) = U_M(b)$.

Let us now compute the languages $\mathcal{L}(\mathcal{A}, q^1_\top)$ and $\mathcal{L}(\mathcal{A}, q^1_\bot)$. A word in Σ^* is accepted from the state $q^1_\top$ if, upon reading it from $q^1_\top$, we either stop at the accepting state $q^d_\top$, or we visit the state $q^1_{u_a}$ and the corresponding suffix is in the language $\mathcal{L}(\mathcal{A}, q^1_{u_a})$. Therefore, we have:

$$\mathcal{L}(\mathcal{A}, q^1_\top) = \{a^{d-1}\} \cup a^d \cdot b \cdot \mathcal{L}(\mathcal{A}, q^1_{u_a}) = \{a^{d-1}\} \cup a^d \cdot U_{k+1}(a)$$

Similarly, we have:

$$\mathcal{L}(\mathcal{A}, q^1_\bot) = \{a^d\} \cup a^d \cdot b \cdot \mathcal{L}(\mathcal{A}, q^1_{u_a}) = \{a^d\} \cup a^d \cdot U_{k+1}(a)$$

Now, let us compute the languages $\mathcal{L}(\mathcal{A}, q_\top)$ and $\mathcal{L}(\mathcal{A}, q_\bot)$. We start with $\mathcal{L}(\mathcal{A}, q_\top)$:

$$\mathcal{L}(\mathcal{A}, q_\top) = \mathcal{L}(\mathcal{A}, q_\top, q^0_\top) \cdot a \cdot \mathcal{L}(\mathcal{A}, q^1_\top) = U_{k+1}(a) \cdot \Big(\{a^d\} \cup a^{d+1} \cdot U_{k+1}(a)\Big)$$

Hence, we have:

$$\mathcal{L}(\mathcal{A}) \cap a \cdot a \cdot \Sigma^* = a \cdot a \cdot \mathcal{L}(\mathcal{A}, q_\top) = a \cdot a \cdot U_{k+1}(a) \cdot \Big(\{a^d\} \cup a^{d+1} \cdot U_{k+1}(a)\Big) = \mathcal{P}_\top(k, d) \quad (1)$$

Similarly, we have:

$$\mathcal{L}(\mathcal{A}, q_\bot) = U_{k+1}(a) \cdot \Big(\{a^{d+1}\} \cup a^{d+1} \cdot U_{k+1}(a)\Big)$$

and

$$\mathcal{L}(\mathcal{A}) \cap a \cdot b \cdot \Sigma^* = a \cdot b \cdot U_{k+1}(a) \cdot \Big(\{a^{d+1}\} \cup a^{d+1} \cdot U_{k+1}(a)\Big) = \mathcal{P}_\bot(k, d) \quad (2)$$

Let us now compute the language $\mathcal{L}(\mathcal{A}, q_{\mathsf{Var}})$. To do so, we let $i \in [\![1, r]\!]$ and we compute the language $b \cdot \mathcal{L}(\mathcal{A}, (x_i \in C_1))$. We have:

$$b \cdot \mathcal{L}(\mathcal{A}, (x_i \in C_1)) = \{b^\sigma \mid \sigma \in \mathsf{Ind}(x_i)\} \cup b^{s+s+T \cdot r} \cdot a \cdot \mathcal{L}(\mathcal{A}, q^1_{u_b})$$

with $a \cdot \mathcal{L}(\mathcal{A}, q^1_{u_b}) = U_{k+1}(b)$ (as established above). Furthermore, we have:

$$\mathcal{L}(\mathcal{A}, q^0_{x_i}) = b^d \cdot \Big(a \cdot \mathcal{L}(\mathcal{A}, q^1_{\nu(x_i)}) \cup b \cdot \mathcal{L}(\mathcal{A}, (x_i \in C_1))\Big)$$

and

$$\mathcal{L}(\mathcal{A}, q_{\mathsf{Var}}) = \mathcal{L}(\mathcal{A}, q_{\mathsf{Var}}, q_X) \cdot \left(\bigcup_{i \in [\![1,r]\!]} \mathcal{L}(\mathcal{A}, q_X, q^0_{x_i}) \cdot \mathcal{L}(\mathcal{A}, q^0_{x_i}) \right) = U_M(b) \cdot \left(\bigcup_{i \in [\![1,r]\!]} b^i \cdot a \cdot \mathcal{L}(\mathcal{A}, q^0_{x_i}) \right)$$

Therefore, letting $d(\top) := d$ and $d(\bot) := d+1$, and $L := \mathcal{L}(\mathcal{A}) \cap b \cdot a \cdot \Sigma^*$, we have:

$$\begin{aligned}
L = b \cdot a \cdot \mathcal{L}(\mathcal{A}, q_{\mathsf{Var}}) &= b \cdot a \cdot U_M(b) \cdot \left(\bigcup_{i \in [\![1,r]\!]} b^i \cdot a \cdot \mathcal{L}(\mathcal{A}, q^0_{x_i}) \right) \\
&= \bigcup_{i \in [\![1,r]\!]} b \cdot a \cdot U_M(b) \cdot b^i \cdot a \cdot \mathcal{L}(\mathcal{A}, q^0_{x_i}) \\
&= \bigcup_{i \in [\![1,r]\!]} b \cdot a \cdot U_M(b) \cdot b^i \cdot a \cdot b^d \cdot \left(\underbrace{a \cdot \mathcal{L}(\mathcal{A}, q^1_{\nu(x_i)})}_{= \{a^{d(\nu(x_i))}\} \cup a^{d+1} \cdot U_{k+1}(a)} \cup \{b^\sigma \mid \sigma \in \mathsf{Ind}(x_i)\} \cup d^{s+s+T \cdot r} \cdot U_{k+1}(b) \right) \\
&= \bigcup_{i \in [\![1,r]\!]} b \cdot a \cdot U_M(b) \cdot b^i \cdot a \cdot b^d \cdot \Big(a^{d+1} \cdot U_{k+1}(a) \cup \{b^\sigma \mid \sigma \in \mathsf{Ind}(x_i)\} \cup d^{s+s+T \cdot r} \cdot U_{k+1}(b)\Big) \\
&\cup \bigcup_{i \in [\![1,r]\!]} b \cdot a \cdot U_M(b) \cdot b^i \cdot a \cdot b^d \cdot a^{d(\nu(x_i))}
\end{aligned}$$

Therefore, letting $\mathsf{ErrVar} := \bigcup_{i \in [\![1,r]\!]} b \cdot a \cdot U_M(b) \cdot b^i \cdot a \cdot b^d \cdot a^{d(\nu(x_i))}$, we obtain:

$$\mathcal{L}(\mathcal{A}) \cap b \cdot a \cdot \Sigma^* = \bigcup_{i \in [\![1,r]\!]} \mathcal{P}_{\mathsf{Var}}(X, \mathcal{C}, k, d, T, M, i) \cup \mathsf{ErrVar} \tag{3}$$

Let us now finally turn to the language $\mathcal{L}(\mathcal{A}, q_{\mathsf{Cl}})$. Clearly, we have:

$$\begin{aligned}
\mathcal{L}(\mathcal{A}, q_{\mathsf{Cl}}) &= \bigcup_{j \in [\![1,s]\!]} \mathcal{L}(\mathcal{A}, q_{\mathsf{Cl}}, q_{C_j}) \cdot b \cdot \mathcal{L}(\mathcal{A}, q^0_{x_{i_j}}) = \bigcup_{j \in [\![1,s]\!]} a^j \cdot b \cdot \mathcal{L}(\mathcal{A}, q^0_{x_{i_j}}) \\
&= \bigcup_{j \in [\![1,s]\!]} a^j \cdot b \cdot \Big(b^d \cdot a \cdot \mathcal{L}(\mathcal{A}, q^1_{\nu(x_{i_j})}) \cup b^d \cdot b \cdot \mathcal{L}(\mathcal{A}, (x_{i_j} \in C_1))\Big) \\
&= \bigcup_{j \in [\![1,s]\!]} a^j \cdot b \cdot \Big(\{b^d \cdot a^{d(\nu(x_{i_j}))}\} \ \cup \ b^d \cdot a^{d+1} \cdot U_{k+1}(a) \\
&\qquad\qquad \cup b^d \cdot \{b^\sigma \mid \sigma \in \mathsf{Ind}(x_{i_j})\} \cup b^{s+s+T \cdot r} \cdot U_{k+1}(b)\Big)
\end{aligned}$$

For all $j \in [\![1,s]\!]$, since $x_{i_j} \in C_j$, we have $j \in \mathsf{App}(x_{i_j}) \subseteq \mathsf{Ind}(x_{i_j})$. Hence, we have:

$$\begin{aligned}
b \cdot b \cdot \mathcal{L}(\mathcal{A}, q_{\mathsf{Cl}}) &= \bigcup_{j \in [\![1,s]\!]} b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \Big(\{a^{d(\nu(x_{i_j}))}\} \ \cup \ a^{d+1} \cdot U_{k+1}(a)\Big) \\
&\cup \bigcup_{j \in [\![1,s]\!]} b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \Big(\{b^j\} \cup b^{s+s+T \cdot r} \cdot U_{k+1}(b)\Big) \\
&\cup \bigcup_{j \in [\![1,s]\!]} b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \{b^\sigma \mid \sigma \in \mathsf{Ind}(x_{i_j}) \setminus \{j\}\}
\end{aligned}$$

Furthermore, for all $j \in [\![1, s]\!]$, we have:

$$b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \left(\{b^j\} \cup b^{s+s+T \cdot r} \cdot U_{k+1}(b)\right) = \mathcal{P}_{\mathsf{Cl}}(X, \mathcal{C}, k, d, T, j)$$

In addition, since ν satisfies $(X, \mathcal{C})$, if $C_j \in \mathcal{C}_+$, then $\nu(x_{i_j}) = \top$, and if $C_j \in \mathcal{C}_-$, then $\nu(x_{i_j}) = \bot$. Thus, by definition of $d(\nu(x_{i_j}))$ and $\mathcal{P}^{\mathsf{acc}}_{\mathsf{Cl}}(k, d, j)$, we have $b \cdot b \cdot a^j \cdot b \cdot b^d \cdot a^{d(\nu(x_{i_j}))} \in \mathcal{P}^{\mathsf{acc}}_{\mathsf{Cl}}(k, d, j)$. Therefore:

$$b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \left(\{a^{d(\nu(x_{i_j}))}\} \ \cup \ a^{d+1} \cdot U_{k+1}(a)\right) = \mathcal{P}^{\mathsf{acc}}_{\mathsf{Cl}}(k, d, j)$$

Hence, letting $\mathsf{ErrCl} := \bigcup_{j \in [\![1,s]\!]} b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \{b^\sigma \mid \sigma \in \mathsf{Ind}(x_{i_j}) \setminus \{j\}\}$, we have:

$$\mathcal{L}(\mathcal{A}) \cap b \cdot b \cdot \Sigma^* = b \cdot b \cdot \mathcal{L}(\mathcal{A}, q_{\mathsf{Cl}}) = \bigcup_{j \in [\![1,s]\!]} (\mathcal{P}^{\mathsf{acc}}_{\mathsf{Cl}}(k, d, j) \cup \mathcal{P}_{\mathsf{Cl}}(X, \mathcal{C}, k, d, T, j)) \cup \mathsf{ErrCl} \tag{4}$$

Overall, with Equations (1), (2), (3), and (4), we obtain:

$$\begin{aligned}\mathcal{L}(\mathcal{A}) &= (\mathcal{L}(\mathcal{A}) \cap a \cdot a \cdot \Sigma^*) \cup (\mathcal{L}(\mathcal{A}) \cap a \cdot b \cdot \Sigma^*) \cup (\mathcal{L}(\mathcal{A}) \cap b \cdot a \cdot \Sigma^*) \cup (\mathcal{L}(\mathcal{A}) \cap b \cdot b \cdot \Sigma^*) \\ &= \mathcal{P}(X, \mathcal{C}, k, d, T, M) \cup \mathsf{ErrVar} \cup \mathsf{ErrCl}\end{aligned}$$

with

$$|\mathsf{ErrVar}| = |\bigcup_{i \in [\![1,r]\!]} b \cdot a \cdot U_M(b) \cdot b^i \cdot a \cdot b^d \cdot a^{d(\nu(x_i))}| = M \cdot r$$

and, since $\mathsf{App}(x_{i_j}) \cup \mathsf{NotApp}(x_{i_j}) = [\![1, s]\!]$:

$$\begin{aligned}|\mathsf{ErrCl}| &= |\bigcup_{j \in [\![1,s]\!]} b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \{b^\sigma \mid \sigma \in \mathsf{Ind}(x_{i_j}) \setminus \{j\}\}| = |\mathsf{Ind}(x_{i_j})| - 1 \\ &= \sum_{j \in [\![1,s]\!]} (|\mathsf{App}(x_{i_j}) \cup \{s + \sigma \mid \sigma \in \mathsf{NotApp}(x_{i_j})\} \cup \{s + s + i_j + t \cdot r \mid t \in [\![0, T-1]\!]\})| - 1) \\ &= \sum_{j \in [\![1,s]\!]} (|\mathsf{App}(x_{i_j})| + |\mathsf{NotApp}(x_{i_j})| + |[\![0, T-1]\!]| - 1) \\ &= s \cdot (s + T - 1)\end{aligned}$$

Therefore, we have:

- $|Q| \leq n$;
- $\mathcal{P}(X, \mathcal{C}, k, d, T, M) \subseteq \mathcal{L}(\mathcal{A})$;
- $|\mathcal{L}(\mathcal{A}) \setminus \mathcal{P}(X, \mathcal{C}, k, d, T, M)| = |\mathsf{ErrCl}| + |\mathsf{ErrVar}| = M \cdot r + s \cdot (s + T - 1) \leq k$

Thus, for all $m \in \mathbb{N}$, there is a $(\mathcal{P}(X, \mathcal{C}, k, d, T, M), n, k, m)$-suitable DFA. □

B.2 If there is a suitable DFA, then there is a satisfying valuation

In all of this subsection, we fix a positive instance $(X, \mathcal{C})$ of Problem 2, and $n, k, m, d, M, T \in \mathbb{N}$. We also let $\mathcal{P} := \mathcal{P}(X, \mathcal{C}, d, k, T, M)$. The goal of this subsection is to establish that, under some conditions on the integers $n, k, m, d, M, T \in \mathbb{N}$, if there is a $(\mathcal{P}, n, k, m)$-suitable DFA, then $(X, \mathcal{C})$ is a positive instance of Problem 2. Let us introduce below the inequalities between the integers that we will consider.

Definition 21. *We will consider four different inequalities between the above integers:*

- *Assumption A:* $m \geq 2\max_{u\in\mathcal{P}}|u| + \max(r,d)$;
- *Assumption B:* $k < M \cdot T$;
- *Assumption C:* $n < d\cdot(2+r)+d$;
- *Assumption D:* $k \leq s\cdot(T+s-1)+M\cdot r$.

Let us now state the result that we establish in this subsection.

Lemma 22. *If Assumptions A, B, C, and D hold, and there is a $(\mathcal{P},n,k,m)$-suitable DFA, then $(X,\mathcal{C})$ is a positive instance of Problem 2.*

To establish this lemma, we introduce several notations below.

Definition 23. *We define two sets of prefixes of words in $\mathcal{P}$. Specifically, we let:*

$$\begin{aligned}\mathsf{Cont}_{k+1}(a) := \{\ & a\cdot a\cdot u_a, a\cdot b\cdot u_a, b\cdot a\cdot u\cdot b^i\cdot a\cdot b^d \\ & \mid u_a\in U_{k+1}(a), u\in U_M(b), i\in[\![1,r]\!]\}\end{aligned}$$

We consider these prefixes because, by definition, for all $v\in\mathsf{Cont}_{k+1}(a)$ and $u'_a\in U_{k+1}(a)$, we have: $v\cdot a^{d+1}\cdot u'_a\in\mathcal{P}$. Similarly, we also let:

$$\begin{aligned}\mathsf{Cont}_{k+1}(b) := \{\ & b\cdot a\cdot u\cdot b^i\cdot a, b\cdot b\cdot a^j\cdot b \\ & \mid u\in U_M(b), i\in[\![1,r]\!], j\in[\![1,s]\!]\}\end{aligned}$$

We consider these prefixes because, by definition, for all $v\in\mathsf{Cont}_{k+1}(b)$ and $u'_b\in U_{k+1}(b)$, we have: $v\cdot b^{d+s+s+T\cdot r}\cdot u'_b\in\mathcal{P}$.

Now, we state a lemma* divided into four consecutive steps (which we prove separately afterwards) from which we will be able to straightforwardly deduce Lemma* 22.

Lemma 24. *Assume that there is DFA $\mathcal{A}=(Q,\Sigma,q_{\mathsf{init}},\delta,F)$ such that: 1) $|\mathcal{A}|\leq n$; 2) $\mathcal{P}\subseteq\mathcal{L}(\mathcal{A})$; and 3) $|\Sigma^{\leq m}\cap(\mathcal{L}(\mathcal{A})\setminus\mathcal{P})|\leq k$. For all words $u,v\in\Sigma^*$, we let:*

$$\Delta(u,v) := \{\delta^*(q_{\mathsf{init}},u\cdot w)\mid w\sqsubseteq v, w\neq\epsilon\}\subseteq Q$$

Step 1: Assume that A holds. *For all $l,l'\in[\![1,d]\!]$, we have:*

- $\mathsf{Prop}(a,a)$*: For all $v_a,v'_a\in\mathsf{Cont}_{k+1}(a)$, if $l\neq l'$ then: $\delta^*(q_{\mathsf{init}},v_a\cdot a^l)\neq\delta^*(q_{\mathsf{init}},v'_a\cdot a^{l'})$*
- $\mathsf{Prop}(b,b)$*: For all $v_b,v'_b\in\mathsf{Cont}_{k+1}(b)$, if $l\neq l'$ then: $\delta^*(q_{\mathsf{init}},v_b\cdot b^l)\neq\delta^*(q_{\mathsf{init}},v'_b\cdot b^{l'})$*
- $\mathsf{Prop}(a,b)$*: For all $v_a\in\mathsf{Cont}_{k+1}(a)$, $v_b\in\mathsf{Cont}_{k+1}(b)$: $\delta^*(q_{\mathsf{init}},v_a\cdot a^l)\neq\delta^*(q_{\mathsf{init}},v_b\cdot b^{l'})$*

Furthermore, there is $u_a^\top,u_a^\bot\in U_{k+1}(a)$ such that $\Delta(a\cdot a\cdot u_a^\top,a^d)\cap\Delta(a\cdot b\cdot u_a^\bot,a^d)=\emptyset$ and:

$$a\cdot a\cdot u_a^\top\cdot a^d, a\cdot b\cdot u_a^\bot\cdot a^{d+1}\in\mathcal{L}(\mathcal{A})$$
$$a\cdot b\cdot u_a^\bot\cdot a^d, a\cdot a\cdot u_a^\top\cdot a^{d+1}\notin\mathcal{L}(\mathcal{A})$$

Step 2: Additionally assume that B holds. *For all $i\in[\![1,r]\!]$, there is some $u_i\in U_M(b)$ such that the set $\bigcup_{1\leq i\leq r}\Delta(b\cdot a\cdot u_i\cdot b^i\cdot a,b^d)$ is of cardinality $r\cdot d$.*
Therefore, the set V defined below is of cardinality $(2+r)\cdot d$:

$$V := \Delta(a\cdot a\cdot u_a^\top,a^d)\cup\Delta(a\cdot b\cdot u_a^\bot,a^d)\cup\bigcup_{1\leq i\leq r}\Delta(b\cdot a\cdot u_i\cdot b^i\cdot a,b^d)$$

Step 3: Additionally assume that C holds. *Let $t_\top := \delta^*(q_{\mathsf{init}},a\cdot a\cdot u_a^\top\cdot a^d)\in Q$ and $t_\bot := \delta^*(q_{\mathsf{init}},a\cdot b\cdot u_a^\bot\cdot a^d)$.*

- *Let $i \in [\![1, r]\!]$. For all $u \in U_M(b)$, we have: $\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d \cdot a^d) \in \{t_\top, t_\bot\}$. Then, letting $t_{x_i} := \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_i \cdot b^i \cdot a \cdot b^d)$, we have: $\delta^*(t_{x_i}, a^d) \in \{t_\top, t_\bot\}$;*
- *For all $j \in [\![1, s]\!]$, there is some $i_j \in [\![1, r]\!]$ such that: $\delta^*(q_{\mathsf{init}}, b \cdot b \cdot a^j \cdot b \cdot b^d) = t_{x_{i_j}}$.*

Step 4: Additionally assume that D holds. *For all $j \in [\![1, s]\!]$, we have $x_{i_j} \in C_j$. Thus, the valuation $\nu : X \to \{\top, \bot\}$ such that, for all $x \in X$, $\delta^*(t_x, a^d) = t_{\nu(x)}$, satisfies $(X, \mathcal{C})$.*

We prove the four steps of this lemma* in four different proofs. Recalling the bird's eye view that we have presented in the paper, the proofs of Steps 1 and 2 rely on TME-arguments, the proof of Step 3 relies on TMS-arguments, and the proof of Step 4 relies on a precise TME-argument.

Proof of Step 1 of Lemma* 24, assuming A

Proof. Let $l, l' \in [\![1, d]\!]$. The cases $\mathsf{Prop}(a, a)$ and $\mathsf{Prop}(b, b)$ are almost identical, and are both very close to the case $\mathsf{Prop}(a, b)$.

- $\mathsf{Prop}(a, a)$: Consider some $v_a, v'_a \in \mathsf{Cont}_{k+1}(a)$. Assume towards a contradiction that $j := l' - l > 0$ and $\delta^*(q_{\mathsf{init}}, v_a \cdot a^l) = \delta^*(q_{\mathsf{init}}, v'_a \cdot a^{l'})$. Then, for all $u_a \in U_{k+1}(a)$, we have $\delta^*(q_{\mathsf{init}}, v_a \cdot a^{d+1-j} \cdot u_a) = \delta^*(q_{\mathsf{init}}, v'_a \cdot a^{d+1} \cdot u_a) \in F$ since $v'_a \cdot a^{d+1} \cdot u_a \in \mathcal{P} \subseteq \mathcal{L}(\mathcal{A})$. In addition, $|v_a \cdot a^{d+1-j} \cdot u_a| \leq |v'_a \cdot a^{d+1} \cdot u_a| \leq m$. Thus: $v_a \cdot a^{d+1-j} \cdot u_a \in \Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P})$. Hence the contradiction since this holds for all $u_a \in U_{k+1}(a)$ and $|U_{k+1}(a)| = k + 1$.
- $\mathsf{Prop}(b, b)$: Consider some $v_b, v'_b \in \mathsf{Cont}_{k+1}(b)$. Assume towards a contradiction that $j := l' - l > 0$ and $\delta^*(q_{\mathsf{init}}, v_b \cdot b^l) = \delta^*(q_{\mathsf{init}}, v'_b \cdot b^{l'})$. Then, for all $u_b \in U_{k+1}(b)$, we have $\delta^*(q_{\mathsf{init}}, v_b \cdot b^{d-j+s+s+T\cdot r} \cdot u_b) = \delta^*(q_{\mathsf{init}}, v'_b \cdot b^{d+s+s+T\cdot r} \cdot u_b) \in F$ since $v_b \cdot b^{d-j+s+s+T\cdot r} \cdot u_b \in \mathcal{P} \subseteq \mathcal{L}(\mathcal{A})$. In addition, $|v_b \cdot b^{d-j+s+s+T\cdot r} \cdot u_b| \leq |v'_b \cdot b^{d+s+s+T\cdot r} \cdot u_b| \leq m$. Thus: $v_b \cdot b^{d-j+s+s+T\cdot r} \cdot u_b \in \Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P})$. Hence the contradiction since this holds for all $u_b \in U_{k+1}(b)$ and $|U_{k+1}(b)| = k + 1$.
- $\mathsf{Prop}(a, b)$: Let $v_a \in \mathsf{Cont}_{k+1}(a)$ and $v_b \in \mathsf{Cont}_{k+1}(b)$. Assume towards a contradiction that: $\delta^*(q_{\mathsf{init}}, v_a \cdot a^l) = \delta^*(q_{\mathsf{init}}, v_b \cdot b^{l'})$. Then, for all $u_b \in U_{k+1}(b)$, we have: $\delta^*(q_{\mathsf{init}}, v_a \cdot a^l \cdot b^{d-l'+s+s+T\cdot r} \cdot u_b) = \delta^*(q_{\mathsf{init}}, v_b \cdot b^{d+s+s+T\cdot r} \cdot u_b) \in F$. Furthermore, we have $v_a \cdot a^l \cdot b^{d-l'+s+s+r\cdot T} \cdot u_b \notin \mathcal{P}$, since $l \geq 1$, and $|v_a \cdot a^l \cdot b^{d-l'+s+s+r\cdot T} \cdot u_b| \leq |v_b \cdot b^{d+s+s+T\cdot r} \cdot u_b| + |v_a| - |v_b| + l \leq m$. Thus: $v_a \cdot a^l \cdot b^{d-l'+s+s+r\cdot T} \cdot u_b \in \Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P})$. Hence, the contradiction since this holds for all $u_b \in U_{k+1}(b)|$ and $|U_{k+1}(b)| = k + 1$.

Furthermore, for all $u_a \in U_{k+1}(a)$, we have $a \cdot a \cdot u_a \cdot a^{d+1}, a \cdot b \cdot u_a \cdot a^d \in \Sigma^{\leq m} \setminus \mathcal{P}$. Hence, since $|U_{k+1}(a)| = k + 1$, it follows that there is some $u_a^\top, u_a^\bot \in U_{k+1}(a)$ such that $a \cdot a \cdot u_a^\top \cdot a^{d+1}, a \cdot b \cdot u_a^\bot \cdot a^d \notin \mathcal{L}(\mathcal{A})$. They are such that $a \cdot a \cdot u_a^\top \cdot a^d, a \cdot b \cdot u_a^\bot \cdot a^{d+1} \in \mathcal{P} \subseteq \mathcal{L}(\mathcal{A})$. □

Proof of Step 2 of Lemma* 24, assuming A, B

Proof. Let $i \in [\![1, r]\!]$. Assume towards a contradiction that, for all $u \in U_M(b)$, there is some $v_u \in U_M(b)$ and $i_u \neq i \in [\![1, r]\!]$ such that $\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d) = \delta^*(q_{\mathsf{init}}, b \cdot a \cdot v_u \cdot b^{i_u} \cdot a \cdot b^d)$. Consider some $t \in [\![0, T-1]\!]$ and $u \in U_M(b)$. Since:

$$b \cdot a \cdot u \cdot b^i \cdot a \cdot b^{d+s+s+i+t\cdot r} \in \mathcal{P} \subseteq \mathcal{L}(\mathcal{A})$$

and $|b \cdot a \cdot v_u \cdot b^{i_u} \cdot a \cdot b^{d+s+s+i+t\cdot r}| \leq |b \cdot a \cdot u \cdot b^i \cdot a \cdot b^{d+s+s+i+t\cdot r}| + |v_u| + i_u \leq m$, we have:

$$b \cdot a \cdot v_u \cdot b^{i_u} \cdot a \cdot b^{d+s+s+i+t\cdot r} \in \Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P})$$

Hence: $\{b \cdot a \cdot v_u \cdot b^{i_u} \cdot a \cdot b^{d+s+s+i+t\cdot r} \mid u \in U_M(b), t \in [\![0, T-1]\!]\} \subseteq (\Sigma^{\leq m} \cap \mathcal{L}(\mathcal{A}) \setminus \mathcal{P})$. Since $|U_M(b)| \cdot |[\![0, T-1]\!]| = M \cdot T > k$, this is a contradiction. Therefore, there is some $u_i \in U_M(b)$, such that, for all $i' \neq i \in [\![1, r]\!]$ and $u \in U_M(b)$, we have $\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_i \cdot b^i \cdot a \cdot b^d) \neq \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^{i'} \cdot a \cdot b^d)$.

Then, by definition, we have:

$$\bigcup_{i \in [\![1,r]\!]} \Delta(b \cdot a \cdot u_i \cdot b^i \cdot a, b^d) = \{\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_i \cdot b^i \cdot a \cdot b^l) \mid i \in [\![1, r]\!],\ l \in [\![1, d]\!]\}$$

Consider any $i, i' \in [\![1, r]\!], l, l' \in [\![1, d]\!]$. If $l \neq l'$, then we have $\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_i \cdot b^i \cdot a \cdot b^l) \neq \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_{i'} \cdot b^{i'} \cdot a \cdot b^{l'})$ by $\mathsf{Prop}(b, b)$. Furthermore, if $l = l'$ and $\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_i \cdot b^i \cdot a \cdot b^l) = \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_{i'} \cdot b^{i'} \cdot a \cdot b^{l'})$, then it follows that $\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_i \cdot b^i \cdot a \cdot b^d) = \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_{i'} \cdot b^{i'} \cdot a \cdot b^d)$, and thus $i = i'$, by choice of $u_i, u_{i'}$. Overall, we have that if $\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_i \cdot b^i \cdot a \cdot b^l) = \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_{i'} \cdot b^{i'} \cdot a \cdot b^{l'})$, then $i = i'$ and $l = l'$. Therefore, the set $\bigcup_{i \in [\![1,r]\!]} \Delta(b \cdot a \cdot u_i \cdot b^i \cdot a, b^d)$ is of cardinality $r \cdot d$.

Then, the set V is by definition equal to:

$$V = \Delta(a \cdot a \cdot u_a^\top, a^d) \cup \Delta(a \cdot b \cdot u_a^\bot, a^d) \cup \bigcup_{i \in [\![1,r]\!]} \Delta(b \cdot a \cdot u_i \cdot b^i \cdot a, b^d) \subseteq Q$$

By $\mathsf{Prop}(a, a)$, for all $v, v' \in \{a \cdot a \cdot u_a^\top, a \cdot b \cdot u_a^\bot\}$ and $l, l' \in [\![1, d]\!]$, we have $\delta^*(q_{\mathsf{init}}, v \cdot a^l) = \delta^*(q_{\mathsf{init}}, v' \cdot a^{l'})$ implies $l = l'$, which implies $\delta^*(q_{\mathsf{init}}, v \cdot a^d) = \delta^*(q_{\mathsf{init}}, v' \cdot a^d)$. Since, by choice of $u_a^\top, u_a^\bot$, we have $a \cdot a \cdot u_a^\top \cdot a^d \in \mathcal{L}(\mathcal{A})$ and $a \cdot b \cdot u_a^\bot \cdot a^d \notin \mathcal{L}(\mathcal{A})$, it follows that $\delta^*(q_{\mathsf{init}}, v \cdot a^l) = \delta^*(q_{\mathsf{init}}, v' \cdot a^{l'})$ implies $l = l'$ and $v = v'$. Therefore, the set $\Delta(a \cdot a \cdot u_a^\top, a^d) \cup \Delta(a \cdot b \cdot u_a^\bot, a^d)$ is of cardinality $2 \cdot d$. Furthermore, by $\mathsf{Prop}(a, b)$, we have:

$$\Big(\Delta(a \cdot a \cdot u_a^\top, a^d) \cup \Delta(a \cdot b \cdot u_a^\bot, a^d)\Big) \cap \left(\bigcup_{i \in [\![1,r]\!]} \Delta(b \cdot a \cdot u_i \cdot b^i \cdot a, b^d)\right) = \emptyset$$

Therefore, the cardinal of the set V is equal to $(2 + r) \cdot d$. □

Proof of Step 3 of Lemma* 24, assuming A, B, C

Proof. The proofs of both items are very similar, and rather straightforward with the help of $\mathsf{Prop}(a, a)$, $\mathsf{Prop}(b, b)$, and $\mathsf{Prop}(a, b)$.

Consider first some i $\in [\![1, r]\!]$ and $u \in U_M(b)$. Let $W := \Delta(b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d, a^d) \subseteq Q$. By $\mathsf{Prop}(a, a)$, W is of cardinal d. Thus, we have $|Q| \leq n < (2 + r) \cdot d + d = |V| + |W|$. Hence, it must be that $V \cap W \neq \emptyset$. That is, there exist $(v, c) \in \{(a \cdot a \cdot u_a^\top, a), (a \cdot b \cdot u_a^\bot, a), (b \cdot a \cdot u_{i'} \cdot b^{i'} \cdot a, b) \mid i' \in [\![1, r]\!]\}$ and $l, l' \in [\![1, d]\!]$ such that $\delta^*(q_{\mathsf{init}}, v \cdot c^l) = \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d \cdot a^{l'})$. By $\mathsf{Prop}(a, b)$, it must be that $v \in \{a \cdot a \cdot u_a^\top, a \cdot b \cdot u_a^\bot\}$. By $\mathsf{Prop}(a, a)$, it must be that $l = l'$. Hence, we have:

$$\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d \cdot a^l) \in \{\delta^*(q_{\mathsf{init}}, a \cdot a \cdot u_a^\top \cdot a^l), \delta^*(q_{\mathsf{init}}, a \cdot b \cdot u_a^\bot \cdot a^l)\}$$

Thus, we have:

$$\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d \cdot a^d) \in \{\delta^*(q_{\mathsf{init}}, a \cdot a \cdot u_a^\top \cdot a^d), \delta^*(q_{\mathsf{init}}, a \cdot b \cdot u_a^\bot \cdot a^d)\} = \{t_\top, t_\bot\}$$

Now, consider some $j \in [\![1, s]\!]$ and let $W := \Delta(b \cdot b \cdot a^j \cdot b, b^d) \subseteq Q$. By $\mathsf{Prop}(b, b)$, W is of cardinal d. Thus, we have $|Q| \leq n < (2 + r) \cdot d + d = |V| + |W|$. Hence, it must be that $V \cap W \neq \emptyset$. That is, there exist $(v, c) \in \{(a \cdot a \cdot u_a^\top, a), (a \cdot b \cdot u_a^\bot, a), (b \cdot a \cdot u_i \cdot b^i \cdot a, b) \mid i \in [\![1, r]\!]\}$ and $l, l' \in [\![1, d]\!]$ such that $\delta^*(q_{\mathsf{init}}, v \cdot c^l) = \delta^*(q_{\mathsf{init}}, b \cdot b \cdot a^j \cdot b \cdot b^{l'})$. By $\mathsf{Prop}(a, b)$, it must be that

$v \in \{b \cdot a \cdot u_i \cdot b^i \cdot a \mid i \in [\![1, r]\!]\}$. By $\mathsf{Prop}(b, b)$, it must be that $l = l'$. It follows that there is some $i_j \in [\![1, r]\!]$ such that:

$$\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_{i_j} \cdot b^{i_j} \cdot a \cdot b^l) = \delta^*(q_{\mathsf{init}}, b \cdot b \cdot a^j \cdot b \cdot b^l)$$

Thus:

$$\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_{i_j} \cdot b^{i_j} \cdot a \cdot b^d) = \delta^*(q_{\mathsf{init}}, b \cdot b \cdot a^j \cdot b \cdot b^d)$$

Since $t_{x_{i_j}} = \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_{i_j} \cdot b^{i_j} \cdot a \cdot b^d)$, we obtain $t_{x_{i_j}} = \delta^*(q_{\mathsf{init}}, b \cdot b \cdot a^j \cdot b \cdot b^d)$. □

Proof of Step 4 of Lemma* 24, assuming A, B, C, D

Proof. First of all, note that we have:

$$t_\top, \delta(t_\bot, a) \in F \text{ since } a \cdot a \cdot u_a^\top \cdot a^d, a \cdot b \cdot u_a^\bot \cdot a^{d+1} \in \mathcal{L}(\mathcal{A})$$
$$t_\bot, \delta(t_\top, a) \notin F \text{ since } a \cdot b \cdot u_a^\bot \cdot a^d, a \cdot a \cdot u_a^\top \cdot a^{d+1} \notin \mathcal{L}(\mathcal{A})$$

Therefore, we have $t_\top \neq t_\bot$.

Now, let $i \in [\![1, r]\!]$ and $u \in U_M(b)$. We have shown that:

$$\delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d \cdot a^d) \in \{t_\top, t_\bot\}$$

We let $\mathsf{U}(i, \top) := \{u \in U_M(b) \mid \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d \cdot a^d) = t_\top\}$ and $\mathsf{U}(i, \bot) := \{u \in U_M(b) \mid \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u \cdot b^i \cdot a \cdot b^d \cdot a^d) = t_\bot\} = U_M(b) \setminus \mathsf{U}(i, \top)$. Then, we let:

$$\mathsf{Err}^i_{\top,\bot} := \{\, b \cdot a \cdot u_\top \cdot b^i \cdot a \cdot b^d \cdot a^d, b \cdot a \cdot u_\bot \cdot b^i \cdot a \cdot b^d \cdot a^{d+1} \mid\ u_\top \in \mathsf{U}(i, \top), u_\bot \in \mathsf{U}(i, \bot)\}$$

and we have:

$$\mathsf{Err}^i_{\top,\bot} \subseteq \Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P}) \text{ with } |\mathsf{Err}^i_{\top,\bot}| = M$$

Consider now any $j \in [\![1, s]\!]$. We have shown that:

$$\delta^*(q_{\mathsf{init}}, b \cdot b \cdot a^j \cdot b \cdot b^d) = \delta^*(q_{\mathsf{init}}, b \cdot a \cdot u_{i_j} \cdot b^{i_j} \cdot a \cdot b^d) = t_{x_{i_j}}$$

Furthermore, we have that for all $j' \in \mathsf{App}(x_{i_j})$:

$$b \cdot a \cdot u_{i_j} \cdot b^{i_j} \cdot a \cdot b^{d+j'} \in \mathcal{P}$$

Thus, $b \cdot b \cdot a^j \cdot b \cdot b^{d+j'} \in \mathcal{L}(\mathcal{A})$ with $b \cdot b \cdot a^j \cdot b \cdot b^{d+j'} \in \mathcal{P}$ if and only if $j' = j$. Similarly, we have that for all $j' \in \mathsf{NotApp}(x_{i_j})$:

$$b \cdot a \cdot u_{i_j} \cdot b^{i_j} \cdot a \cdot b^{d+s+j'} \in \mathcal{P} \text{ thus } b \cdot b \cdot a^j \cdot b \cdot b^{d+s+j'} \in \mathcal{L}(\mathcal{A}) \setminus \mathcal{P}$$

In addition, for all $t \in [\![0, T-1]\!]$, we have:

$$b \cdot a \cdot u_{i_j} \cdot b^{i_j} \cdot a \cdot b^{d+s+s+i_j+t \cdot r} \in \mathcal{P} \text{ thus } b \cdot b \cdot a^j \cdot b \cdot b^{d+s+s+i_j+t \cdot r} \in \mathcal{L}(\mathcal{A}) \setminus \mathcal{P}$$

Therefore, we let:

$$\mathsf{ErrApp}^j := b \cdot b \cdot a^j \cdot b \cdot b^d \cdot \{\, b^{j_1}, b^{s+j_2}, b^{s+s+i_j+t \cdot r} \mid j_1 \in \mathsf{App}(x_{i_j}) \setminus \{j\}, j_2 \in \mathsf{NotApp}(x_{i_j}), t \in [\![0, T-1]\!]\}$$

Since $\mathsf{App}(x_{i_j}) \cup \mathsf{NotApp}(x_{i_j}) = [\![1, s]\!]$, we have:

$$\mathsf{ErrApp}^j \subseteq \Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P}) \text{ with } |\mathsf{ErrApp}^j| \geq T + s - 1$$

and the above inequality is an equality if and only if $j \in \mathsf{App}(x_{i_j})$. Overall, we have:

$$\left(\bigcup_{i \in [\![1,r]\!]} \mathsf{Err}^i_{\top,\bot} \cup \bigcup_{j \in [\![1,s]\!]} \mathsf{ErrApp}^j \right) \subseteq \Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P})$$

Hence, we have:

$$\begin{aligned} M \cdot r + s \cdot (T + s - 1) &\geq |\Sigma^{\leq m} \cap (\mathcal{L}(\mathcal{A}) \setminus \mathcal{P})| \\ &\geq |\bigcup_{i \in [\![1,r]\!]} \mathsf{Err}^i_{\top,\bot} \cup \bigcup_{j \in [\![1,s]\!]} \mathsf{ErrApp}^j| = \sum_{i \in [\![1,r]\!]} |\mathsf{Err}^i_{\top,\bot}| + \sum_{j \in [\![1,s]\!]} |\mathsf{ErrApp}^j| \\ &\geq \sum_{i \in [\![1,r]\!]} M + \sum_{j \in [\![1,s]\!]} (T + s - 1) = M \cdot r + s \cdot (T + s - 1) \end{aligned}$$

Therefore, all of the above inequalities are in fact equalities. Hence, for all $j \in [\![1, s]\!]$, we have $|\mathsf{ErrApp}^j| = T + s - 1$, that is $j \in \mathsf{App}(x_{i_j})$.

We can now conclude. Consider the valuation $\nu : X \to \{\top, \bot\}$ such that, for all $x \in X$, $\delta^*(t_x, a^d) = t_{\nu(x)}$. Consider any $j \in [\![1, s]\!]$.

- Assume that $C_j \in \mathcal{C}_+$. We have $b \cdot b \cdot a^j \cdot b \cdot b^d \cdot a^d \in \mathcal{P}$. Therefore, $\delta^*(t_{x_{i_j}}, a^d) = \delta^*(q_{\mathsf{init}}, b \cdot b \cdot a^j \cdot b \cdot b^d \cdot a^d) \in F$. That is $\delta^*(t_{x_{i_j}}, a^d) = t_\top$. Hence, $v(x_{i_j}) = \top$ and $x_{i_j} \in C_j$ since $j \in \mathsf{App}(x_{i_j})$.
- Assume that $C_j \in \mathcal{C}_-$. We have $b \cdot b \cdot a^j \cdot b \cdot b^d \cdot a^{d+1} \in \mathcal{P}$. Therefore, $\delta^*(t_{x_{i_j}}, a^{d+1}) = \delta^*(q_{\mathsf{init}}, b \cdot b \cdot a^j \cdot b \cdot b^d \cdot a^{d+1}) \in F$. That is $\delta^*(t_{x_{i_j}}, a^d) = t_\bot$. Hence, $v(x_{i_j}) = \bot$ and $x_{i_j} \in C_j$ since $j \in \mathsf{App}(x_{i_j})$.

In fact, the valuation ν does satisfy $(X, \mathcal{C})$. □

The proof of Lemma* 22 is now direct.

Proof. Assuming that A, B, C, and D hold, by Lemma* 24, we can exhibit a valuation $\nu : X \to \{\top, \bot\}$ satisfying $(X, \mathcal{C})$. Thus, $(X, \mathcal{C})$ is a positive instance of Problem 2. □

We can now proceed to the proof of Lemma* 16.

Proof. Consider an instance $(X, \mathcal{C})$ of Problem 2. Let $n, k, m, d, M, T \in \mathbb{N}$ and $\mathcal{P} := \mathcal{P}(X, \mathcal{C}, d, k, T, M)$. Assume that we have the following inequalities:

1. $n \geq \omega_1(X, d) + \omega_2(X, \mathcal{C}, k, T, M)$;
2. $k \geq M \cdot r + s \cdot (s + T - 1)$;
3. $m \geq 2 \max_{u \in \mathcal{P}} |u| + \max(r, d)$;
4. $k < M \cdot T$;
5. $n < d \cdot (2 + r) + d = \omega_1(X, d) + d$;
6. $k \leq s \cdot (T + s - 1) + M \cdot r$.

Then, by Lemmas* 18 and 22, $(X, \mathcal{C})$ is a positive instance of Problem 2 if and only if there is a $(\mathcal{P}, n, k, m)$-suitable DFA. Let us choose these integers $n, k, m, d, M, T \in \mathbb{N}$ such that all of the above inequalities are met. First, we choose:

$$M := 3(s + r) \text{ and } T = 2s + 3r$$

and (due to inequalities 2. and 6.):

$$k = s \cdot (T + s - 1) + M \cdot r$$

That way, we have $k = s \cdot (3(s + r) - 1) + 3(s + r) \cdot r \leq 3(s + r)^2 < M \cdot T$. Thus, inequality 4. is also satisfied. Furthermore, we let:

$$d := \omega_2(X, \mathcal{C}, k, T, M) + 1$$

and

$$n := \omega_1(X, d) + \omega_2(X, \mathcal{C}, k, T, M)$$

With these choices, inequalities 1. and 5. are satisfied.

Let us consider the quantity $\max_{u \in \mathcal{P}} |u|$. Assuming that $r \geq 2$—which is harmless, up to adding a dummy variable which does not appear in any clause in the input $(X, \mathcal{C})$—one can see that, for any $u \in \mathcal{P}$, the set $\Delta(q_0, \epsilon, u)$ constitutes less than half of all the states in the DFA. (We use the assumption $r \geq 2$ for words starting by b.) Therefore, we have $\max_{u \in \mathcal{P}} |u| \leq |Q|/2 \leq n/2$. Hence, any $m \geq n + \max(r, d) = n + d$ satisfies inequalities 3. This is in particular the case for $m = 2n - 2$, since $n \geq 2d$.

Therefore, considering $k' := k + |\Sigma^{\leq 2n-2} \cap \mathcal{P}(X, \mathcal{C}, d, k, T, M)| = k + |\mathcal{P}(X, \mathcal{C}, d, k, T, M)|$, we have that $(X, \mathcal{C})$ is a positive instance of Problem 2 if and only if there is a $(\mathcal{P}, n, k, 2n - 2)$-suitable DFA if and only if $(\mathcal{P}, n, k')$ is a positive instance of Problem 2 Therefore, Problem 1 is NP-hard, even with k written in unary, and Σ of cardinality 2. □

C Our algorithms

C.1 Our ILP encoding

Let us first recall the formal setting of Integer Linear Programming problems.

Minimization problem 1. *Consider as input a set I and*

- *for all $i \in I$, a lower bound $m_i \in \mathbb{Z}$ and an upper bound $M_i \in \mathbb{Z}$ on the value of the integer variable x_i; and*
- *an I-indexed integer vector $(c_i)_{i \in I} \in \mathbb{Z}^I$; and*
- *a set Ω indexing I-indexed integer vectors $(p_{\omega,i})_{\omega \in \Omega, i \in I}$ and, for each $\omega \in \Omega$, an integer $p_\omega \in \mathbb{Z}$.*

The goal is to find, over all I-indexed integer vectors $(x_i)_{i \in I} \in \Pi_{i \in I} [\![m_i, M_i]\!]$, the minimal value of $\sum_{i \in I} c_i \cdot x_i \in \mathbb{Z}$, subject to the inequalities: for all $\omega \in \Omega$, $\sum_{i \in I} p_{\omega,i} \cdot x_i \geq p_\omega$.

The size of the input of this problem is equal to $|I| \cdot |\Omega| \cdot \log N$, where N is the maximum (absolute) value of all the input integers.

Our goal is, given an alphabet Σ, a set of word $\mathcal{P} \subseteq \Sigma^*$, and an integer n, to define a set I, an I-indexed integer vector $(c_i)_{i \in I} \in \mathbb{Z}^I$, a set Ω and, for all $\omega \in \Omega$, an I-indexed integer vector $(p_{\omega,i})_{i \in I} \in \mathbb{Z}^I$, and an integer $p_\omega \in \mathbb{Z}$ such that:

$\mathcal{A} \to x$: From all DFAs $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$, we can derive an integer vector $x \in \Pi_{i \in I} [\![m_i, M_i]\!]$ satisfying $\sum_{i \in I} p_{\omega,i} \cdot x_i = p_\omega$, for all $\omega \in \Omega$, and such that $\sum_{i \in I} c_i \cdot x_i = |\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq 2n-2}|$.

$x \to \mathcal{A}$: Reciprocally, from all integer vectors $x \in \Pi_{i \in I} [\![m_i, M_i]\!]$ satisfying $\sum_{i \in I} p_{\omega,i} \cdot x_i = p_\omega$, for all $\omega \in \Omega$, we can (easily) derive a DFA $\mathcal{A} \in \mathsf{Rec}(\mathcal{P}, n)$ such that $\sum_{i \in I} c_i \cdot x_i = |\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq 2n-2}|$.

To do so, we encode the prospective DFA with integer variables $(x_i)_{i \in I}$. To this end, we assume that the set of states of the DFA is $Q = \{q_0, \ldots, q_{n-1}\}$. Then, we use the following binary variables (i.e. integer variables for which the lower bound is 0 and the upper bound is 1) and integer variables. Let us first introduce the binary variables, their informal meaning, and the inequalities that we define to ensure their meaning.

Transition variables. For each potential transition $\delta(p, \alpha) = q$ (for $p, q \in Q$, $\alpha \in \Sigma$), we introduce a binary variable $x_{p,\alpha,q}$. The meaning of $x_{p,\alpha,q} = 1$ is $\delta(p, \alpha) = q$ in the prospective DFA. We let $I_{\mathsf{Trans}} := Q \times \Sigma \times Q$.

State variables. For each state $q \in Q$, we introduce a binary variable x_q. The meaning of $x_q = 1$ is that $q \in F$. We let $I_{\mathsf{State}} := Q$.

Word variables. For each state $q \in Q$, and prefixes $u \in \Sigma^*$ of words in $\mathcal{P}$, we introduce a binary variable $x_{u,q}$. The meaning of $x_{u,q} = 1$ is that $\delta^*(q_0, u) = q$. We let $I_{\mathsf{Word}} := \mathsf{Pref}(\mathcal{P}) \times Q$.

Now, we impose constraints on these variables $(x_i)_{i \in I}$ to enforce their meaning by defining the various (in)equalities, indexed by the set Ω, that must be satisfied. Each one of these (in)equalities, which corresponds to a different element of $\omega \in \Omega$, implicitly defines the integer constant $p_\omega \in \mathbb{Z}$ and integer vector $(p_{\omega,i})_{i \in I} \in \mathbb{Z}^I$.

We first impose that these variables encode a DFA, that is: for all $p \in Q$ and $\alpha \in \Sigma$, there is unique $q \in Q$ such that $\delta(p, \alpha) = q$. This is ensured by the following equalities:

$$\forall p \in Q,\ \forall \alpha \in \Sigma : \sum_{q \in Q} x_{p,\alpha,q} = 1 \qquad (\omega^{p,\alpha}_{\mathsf{DFA}})$$

We let $\Omega_{\mathsf{DFA}} := \{\omega^{p,\alpha}_{\mathsf{DFA}} \mid p \in Q,\ \alpha \in \Sigma\}$.

Next, we need to enforce that the prospective DFA does accept all the words in $\mathcal{P}$. To this end, we first enforce the meaning of the state variables by defining (in)equalities corresponding to the following facts: $\delta^*(q_0, \epsilon) = q_0$; for all $u \in \mathsf{Pref}(\mathcal{P})$, there is a unique $q \in Q$ such that $\delta^*(q_0, u) = q$; and for all $u \cdot \alpha \in \mathsf{Pref}(\mathcal{P})$, $\delta^*(q, u \cdot \alpha) = \delta(\delta^*(q, u), \alpha)$:

$$x_{\epsilon,q_0} = 1 \qquad (\omega^{\mathsf{init}}_{\mathsf{run}})$$

$$\forall u \in \mathsf{Pref}(\mathcal{P}) : \sum_{q \in Q} x_{u,q} = 1 \qquad (\omega^{u}_{\mathsf{run}})$$

$$\forall u \cdot \alpha \in \mathsf{Pref}(\mathcal{P}),\ \forall p, q \in Q : x_{u,p} + x_{p,\alpha,q} \leq 1 + x_{u \cdot \alpha, q} \qquad (\omega^{u \cdot \alpha, p, q}_{\mathsf{run}})$$

The last inequality amounts to the logical implication: ($x_{u,p} = 1$ and $x_{p,\alpha,q} = 1$) implies $x_{u \cdot \alpha, q} = 1$. We let $\Omega_{\mathsf{run}} := \{\omega^{\mathsf{init}}_{\mathsf{run}}, \omega^{u}_{\mathsf{run}}, \omega^{v \cdot \alpha, p, q}_{\mathsf{run}} \mid u, v \cdot \alpha \in \mathsf{Pref}(\mathcal{P}), p, q \in Q\}$.

We can now enforce that the prospective DFA is consistent with $\mathcal{P}$, i.e. for all $u \in \mathcal{P}$, we have $\delta^*(q, u) \in F$. This is ensured with the following inequalities:

$$\forall u \in \mathcal{P},\ \forall q \in Q : x_{u,q} \leq x_q \qquad (\omega^{u,q}_{\mathsf{consistent}})$$

We let $\Omega_{\mathsf{consistent}} := \{\omega^{u,q}_{\mathsf{consistent}} \mid u \in \mathcal{P}, q \in Q\}$.

Let us consider counting variables.

Counting variables. For each state $q \in Q$ and $k \in [\![1, 2n-2]\!]$, we introduce an integer variable $x_{q,k}$ which counts the number of paths from q_0 to q of size exactly k (i.e. $\mathsf{N}(q,k)$). Thus, we set $m_{q,k} := 0$ and $M_{q,k} := |\Sigma|^k$.

Since we consider only linear (in)equalities, and thus we cannot multiply two variables, we also consider, for each state $p \in Q$, letter $\alpha \in \Sigma$, state $q \in Q$ and $k \in [\![0, 2n-3]\!]$, the integer variable $x_{p,\alpha,q,k}$, that stands for the product $x_{p,\alpha,q} \cdot x_{p,k}$. We set $m_{p,\alpha,q,k} := 0$ and $M_{p,\alpha,q,k} := |\Sigma|^k$. We let $I_{\mathsf{Count}} := Q \times [\![0, 2n-2]\!] \cup Q \times \Sigma \times Q \times [\![0, 2n-3]\!]$.

Similarly, since we are interested in the accepted words, i.e., those that lead from the initial state to an accepting state, we also introduce, for each state $q \in Q$ and $k \in [\![0, 2n-2]\!]$, the variable $x_{q,k,F}$ which stands for the product $x_{q,k} \cdot x_q$ (recall that $x_q = 1$ means that $q \in F$). We set $m_{q,k,F} = 0$ and $M_{q,k,F} = |\Sigma|^k$.

Finally, we introduce an integer variable x_F counting the number of accepted words of size at most $2n-2$ (i.e. $|\mathcal{L}(\mathcal{A}) \cap \Sigma^{\leq 2n-2}|$). Thus, we set $m_F = 0$ and $M_F = \sum_{k=0}^{2n-2} |\Sigma|^k$. We let $I_{\mathsf{FinalCount}} := \{F\} \cup Q \times [\![0, 2n-2]\!] \times \{F\}$.

Overall, the set of variable indices I that we consider is equal to $I := I_{\mathsf{Trans}} \cup I_{\mathsf{State}} \cup I_{\mathsf{Word}} \cup I_{\mathsf{Count}} \cup I_{\mathsf{FinalCount}}$.

The quantity to minimize is the number of accepted words of size at most $2n-2$, i.e., given the meaning of the above variables: x_F. Thus, we define $c_F := 1$ and for all $i \in I \setminus \{F\}$, $c_i := 0$.

Let us now enforce the meaning of the counting variables:

$$x_{q_0,0} = 1 \qquad (\omega_{\mathsf{count}}^{\mathsf{init},q_0})$$

$$\forall q \in Q \setminus \{q_0\}, : x_{q,0} = 0 \qquad (\omega_{\mathsf{count}}^{\mathsf{init},q})$$

$$\forall (p,\alpha,q) \in Q \times \Sigma \times Q,\ \forall k \in [\![0, 2n-3]\!], : x_{p,k} \leq x_{p,\alpha,q,k} + (1 - x_{p,\alpha,q}) \cdot M \qquad (\omega_{\mathsf{count},1}^{\mathsf{step},p,\alpha,q,k})$$

$$\forall (p,\alpha,q) \in Q \times \Sigma \times Q,\ \forall k \in [\![0, 2n-3]\!], : x_{p,\alpha,q,k} \leq x_{p,\alpha,q} \cdot M \qquad (\omega_{\mathsf{count},2}^{\mathsf{step},p,\alpha,q,k})$$

$$\forall q \in Q \times \Sigma \times Q,\ \forall k \in [\![1, 2n-2]\!], : x_{q,k} = \sum_{p \in Q} \sum_{\alpha \in \Sigma} x_{p,\alpha,q,k-1} \qquad (\omega_{\mathsf{count}}^{\mathsf{step},q,k})$$

$$\forall q \in Q,\ \forall k \in [\![0, 2n-2]\!], : x_{q,k} \leq x_{q,k,F} + (1 - x_q) \cdot M \qquad (\omega_{\mathsf{count},1}^{\mathsf{final},q,k})$$

$$\forall q \in Q,\ \forall k \in [\![0, 2n-2]\!], : x_{q,k,F} \leq x_q \cdot M \qquad (\omega_{\mathsf{count},2}^{\mathsf{final},q,k})$$

$$x_F = \sum_{q \in Q} \sum_{k \in [\![0,2n-2]\!]} x_{q,k,F} \qquad (\omega_{\mathsf{count}}^{\mathsf{final}})$$

The constraints $\omega_{\mathsf{count},1}^{\mathsf{step},p,\alpha,q,k}, \omega_{\mathsf{count},2}^{\mathsf{step},p,\alpha,q,k}, \omega_{\mathsf{count},1}^{\mathsf{final},q,k}, \omega_{\mathsf{count},2}^{\mathsf{final},q,k}$ use the classical big-M method to encode a product (between a binary and an integer variable) in ILP, which works as soon as M is bigger than the value of any of the variables, e.g. $M = M_F + 1$ works. Furthermore, the constraints $\omega_{\mathsf{count}}^{\mathsf{init},q}, \omega_{\mathsf{count}}^{\mathsf{step},q,k}$ exactly follow the iterative computation described in Lemma 1, ensuring that we do have $x_{q,k} = \mathsf{N}(q,k)$.

We let $\Omega_{\mathsf{count}} := \{\omega_{\mathsf{count}}^{\mathsf{init},q}, \omega_{\mathsf{count},1}^{\mathsf{step},p,\alpha,q,k_1}, \omega_{\mathsf{count},2}^{\mathsf{step},p,\alpha,q,k_1}, \omega_{\mathsf{count}}^{\mathsf{step},q,k_2}, \omega_{\mathsf{count},1}^{\mathsf{final},q,k_3}, \omega_{\mathsf{count},2}^{\mathsf{final},q,k_3}, \omega_{\mathsf{count}}^{\mathsf{final}} \mid p, q \in Q,\ \alpha \in \Sigma,\ k_1 \in [\![0, 2n-3]\!],\ k_2 \in [\![1, 2n-2]\!],\ k_3 \in [\![0, 2n-2]\!]\}$.

The set Ω that we consider is then constituted of all the (in)equality indexes defined above:

$$\Omega := \Omega_{\mathsf{DFA}} \cup \Omega_{\mathsf{run}} \cup \Omega_{\mathsf{consistent}} \cup \Omega_{\mathsf{count}}$$

With these constraints, the informal meanings of all these integers variables are ensured, and thus conditions $\mathcal{A} \to x$ and $x \to \mathcal{A}$ are met.

Let us count the number of variables and (in)equalities that we have defined, letting $p := |\mathsf{Pref}(\mathcal{P})|$ and $s := |\Sigma|$:

$$\begin{aligned}|I| &\leq \underbrace{n^2 \cdot s}_{=|I_{\mathsf{Trans}}|} + \underbrace{n}_{=|I_{\mathsf{State}}|} + \underbrace{p \cdot n}_{=|I_{\mathsf{Word}}|} + \underbrace{n \cdot 2n + n^2 \cdot s \cdot 2n}_{\leq |I_{\mathsf{Count}}|} + \underbrace{1 + n \cdot 2n}_{=|I_{\mathsf{FinalCount}}|} \\ &= O(n^3 \cdot s + p \cdot n)\end{aligned}$$

and

$$\begin{aligned}|\Omega| &\leq \underbrace{n \cdot s}_{=|\Omega_{\mathsf{DFA}}|} + \underbrace{1 + p + p \cdot n^2}_{\leq |\Omega_{\mathsf{run}}|} + \underbrace{p \cdot n}_{=|\Omega_{\mathsf{consistent}}|} + \underbrace{n + 2 \cdot n^2 \cdot s \cdot 2n + n \cdot 2n + 2 \cdot n \cdot 2n + 1}_{\leq |\Omega_{\mathsf{count}}|} \\ &= O(n^3 \cdot s + n^2 \cdot p)\end{aligned}$$

Furthermore, the larger integer defined is $M_F + 1$, with $\log(M_F) \leq \log(2n) + k \cdot \log(|\Sigma|)$. Therefore, the size of the input $((c_i)_{i \in I}, (p_{\omega,i})_{\omega \in \Omega, i \in I})$ is polynomial in the size of $(\mathcal{P}, n)$.

C.2 Our heuristic algorithm

In this subsection, we formally describe our heuristic algorithm. Let us first formally introduce the notion of transition system in the definition* below.

Definition 25. *A* transition system T *is a tuple* $T = (Q, \Sigma, \delta)$ *where* Q *is a non-empty set of states,* Σ *is an alphabet, and* $\delta \colon Q \times \Sigma \to Q$ *is the* transition function.

Consider any transition system $T = (Q, \Sigma, \delta)$. *For all* $\mathcal{P} \subseteq \Sigma^*$ *and* $q \in Q$, *we consider the DFA* $\mathcal{A}_{T,q,\mathcal{P}} := (Q, \Sigma, \delta, q, \{\delta^*(q, u) \mid u \in \mathcal{P}\})$ *and, for all* $n \geq 1$, *we let:*

$$\mathsf{score}(T, \mathcal{P}, n) := \max_{q \in Q} |\mathcal{L}(\mathcal{A}_{T,q,\mathcal{P}}) \cap \Sigma^{\leq 2n-2}|$$

Furthermore, for all $(q, \alpha, q') \in Q \times \Sigma \times Q$, *we let* $T[q, \alpha \to q']$ *denote the transition system* $T[q, \alpha \to q'] = (Q, \Sigma, \delta')$, *with* $\delta' \colon Q \times \Sigma \to Q$ *is such that, for all* $(\tilde{q}, \beta) \in Q \times \Sigma$, *we have:*

$$\delta'(\tilde{q}, \beta) := \begin{cases} q' & \text{if } (\tilde{q}, \beta) = (q, \alpha) \\ \delta(q, \alpha) & \text{otherwise} \end{cases}$$

Our heuristic algorithm is formally described as Algorithm 1, where Q_n refers to any set of cardinal n. Let us explain step by step this algorithm:

- First, in Line 1, we initialize the best total score. We will attempt to improve it with NbRun runs, as can be witnessed by the "For" loop from Line 2 to Line 24.
- Then, in Lines 3-12, we randomly pick a transition system with the best score (as given in Definition* 25) over InitRand attempts. The Rand function used in Line 6 picks uniformly at random in the set Q_n an α-successor to the state q.
- Following, in Lines 13-22, we iteratively improve the score of the current transition system by changing a single transition at a time. With the "For" loop of Line 17, we range over all possible transition changes (one at a time), and we pick the transition change that most improves the score. If no change improves the score, the Boolean variable HasImproved stays False, and we exit the "While" loop.
- In Lines 23-24, we update, if relevant, the best score over all the transition systems computed so far on all runs.

Algorithm 1 MinScore$(\mathcal{P}, \Sigma, n)$: Computes a DFA in Rec$(\mathcal{P}, n)$ with a score a small as possible.

Input: An alphabet Σ, a positive set of words $\mathcal{P} \subseteq \Sigma^*$, a bound $n \in \mathbb{N}$ on the number of states

```
1: TotalBestScore = ∞
2: for i ∈ [[1, NbRun]] do
3:     BestScore = ∞
4:     for j ∈ [[1, InitRand]] do
5:         for q ∈ Q_n, α ∈ Σ do
6:             δ(q, α) ← Rand(Q_n)
7:         T ← (Q_n, Σ, δ)
8:         s ← Score(T, P, n)
9:         if s < BestScore then
10:            BestScore ← s, BestTS ← T
11:    CurrentScore ← BestScore
12:    CurrentTS ← BestTS
13:    HasImproved ← True
14:    while HasImproved do
15:        HasImproved ← False
16:        for (q, α, q') ∈ Q_n × Σ × Q_n do
17:            T ← CurrentTS[q, α → q']
18:            s ← Score(T, P, n)
19:            if s < CurrentScore then
20:                CurrentScore ← s, CandidateTS ← T, HasImproved ← True
21:        if HasImproved then
22:            CurrentTS ← CandidateTS
23:    if CurrentScore < TotalBestScore then
24:        TotalBestScore ← CurrentScore, TotalBestTS ← CurrentTS
25: for q ∈ Q_n do
26:    A ← A_{TotalBestTS,q,P}
27:    if |L(A) ∩ Σ^{≤2n−2}| = TotalBestScore then return A
```

- Finally, in Lines 25-27, once we have chosen the transition system with the best score over NbRun runs, we extract a DFA from that transition system whose number of accepted words (of length at most $2n-2$) is equal to the score of the transition system.

Note that the scores in Lines 8 and 18 are computed via matrix multiplication (with a fast exponentiation method).