



HAL
open science

Taking advantage of a very simple property to efficiently infer NFAs

Tomasz Jastrzab, Frédéric Lardeux, Eric Monfroy

► To cite this version:

Tomasz Jastrzab, Frédéric Lardeux, Eric Monfroy. Taking advantage of a very simple property to efficiently infer NFAs. 2022 IEEE 34rd International Conference on Tools with Artificial Intelligence (ICTAI), Oct 2022, Virtual, France. pp.1355-1361, 10.1109/ICTAI56018.2022.00205 . hal-04030407

HAL Id: hal-04030407

<https://univ-angers.hal.science/hal-04030407>

Submitted on 16 Mar 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Taking advantage of a very simple property to efficiently infer NFAs

Tomasz Jastrzab
Silesian University of Technology
Poland

Email: Tomasz.Jastrzab@polsl.pl

Frédéric Lardeux
University of Angers
France

Email: Frederic.Lardeux@univ-angers.fr

Eric Monfroy
University of Angers
France

Email: Eric.Monfroy@univ-angers.fr

Abstract—Grammatical inference consists in learning a formal grammar as a finite state machine or as a set of rewrite rules. In this paper, we are concerned with inferring Nondeterministic Finite Automata (NFA) that must accept some words, and reject some other words from a given sample. This problem can naturally be modeled in SAT. The standard model being enormous, some models based on prefixes, suffixes, and hybrids were designed to generate smaller SAT instances.

There is a very simple and obvious property that says: if there is an NFA of size k for a given sample, there is also an NFA of size $k+1$. We first strengthen this property by adding some characteristics to the NFA of size $k+1$. Hence, we can use this property to tighten the bounds of the size of the minimal NFA for a given sample. We then propose simplified and refined models for NFA of size $k+1$ that are smaller than the initial models for NFA of size k . We also propose a reduction algorithm to build an NFA of size k from a specific NFA of size $k+1$. Finally, we validate our proposition with some experimentation that shows the efficiency of our approach.

Index Terms—grammatical inference, nondeterministic automata, SAT models

I. INTRODUCTION

Grammatical inference [1] consists in studying and designing methods for learning formal grammars (as automata or production rules) from a given sample of words. It is useful in numerous applications, e.g., compiler design, bioinformatics, pattern recognition, machine learning, etc.

Let $S = S^+ \cup S^-$ be a sample of words, made of positive words (the set S^+) and negative words (the set S^-): words from S^+ must be elements of the language, and words from S^- must not. The **problem we tackle** is to learn a finite automaton which accepts words of S^+ and rejects words of S^- . The complexity of such problems is related to the number of states of the automaton, i.e., its size. A deterministic automaton (DFA) for a given language can be (even exponentially) larger than a nondeterministic automaton (NFA) for the same language. Thus, similarly to most of the works on automata inference, we focus here on NFAs. An NFA is represented by a 5-tuple $(Q, \Sigma, \Delta, q_1, F)$ where Q is a finite set of states, the alphabet Σ is a finite set of symbols, the transition function $\Delta : Q \times \Sigma \rightarrow 2^Q$ associates a set of states to a given state and a given symbol, $q_1 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. Let k_NFA denote an NFA with k states. The generic problem consists in minimizing k . However, since most of the techniques are based

on a Boolean model, the problem is simplified to learning a k_NFA , k being given. To minimize k , some lower and upper bounds are determined. For example, an upper bound is given by the size of the prefix tree acceptor (PTA). Then, some algorithms can be used to push up (resp. push down) the lower bound (resp. the upper bound) to find the smallest k (see e.g., [2]).

The problem has already been tackled from several perspectives (see e.g., [3] for a wide panel of NFA inference techniques). Ad-hoc algorithms, such as *DeLeTe2* [4], are based on merging states from the PTA. More recently, a new family of algorithms for regular languages inference was given in [5]. Some approaches are based on metaheuristics, such as in [6] where hill-climbing is applied in the context of regular languages, or [7] which is based on a genetic algorithm. In contrast to metaheuristics, complete solvers are always able to find a solution if there exists one, to prove the unsatisfiability of the problem, and to find the global optimum in case of optimization problems. Generally, the problem is modeled as a Constraint Satisfaction Problem (CSP [8]). For example, in [3], an Integer Non-Linear Programming (INLP) formulation of the problem is given. Parallel solvers for minimizing the inferred NFA size are presented in [9], [10]. The author of [11] proposes two strategies, based on variable ordering, for solving the CSP formulation of the problem. Reference [2] proposes a parallel approach for solving the *optimization* variant of the problem.

In this paper, we do not want to improve a specific solver or to design an ad-hoc solver. On the contrary, we try to **improve the SAT models** starting from the models of [12]. More specifically, we focus on the results that can be obtained using a simple property: for a given sample S , if there is a k_NFA , there is also a $(k+1)_NFA$. This is obvious, for example by adding a new state with no incoming transition. This property can be used to find a new upper-bound and raise the lower bound (in case of UNSAT answer): this is easy to do using previous models, but this is useless since it is more complicated to infer a $(k+1)_NFA$ than a k_NFA . However, we can refine this property by requesting more characteristics of the $(k+1)_NFA$. Assume that $\lambda \notin S$. By design, we can request the $(k+1)_NFA$ to have a single final state without outgoing transitions, and that each incoming transition to the final state is a duplication of a transition of the k_NFA ; we

call such an NFA a $(k+1)^*_\text{NFA}$. Moreover, we **simplify previous models**, which is even more interesting for the suffix and hybrid models that pass from $\mathcal{O}(\sigma \cdot k^3)$ clauses to $\mathcal{O}(\sigma \cdot k^2)$ with $\sigma = \sum_{w \in S} |w|$. Hence, it becomes less complicated to compute a $(k+1)^*_\text{NFA}$ than a k_NFA . The $(k+1)^*_\text{NFA}$ can thus be used to reduce the bounds on k_NFA . Finally, we **propose a linear-time algorithm** to reduce a $(k+1)^*_\text{NFA}$ to a k_NFA . The experimental results for different models are promising and show the advantage of our approach.

The paper is organized as follows. In Sect. II we present how current models are obtained. In Sect. III, we first present the property leading from a k_NFA to a $(k+1)_\text{NFA}$. We then show how to improve the models to infer $(k+1)^*_\text{NFA}$. Section IV presents our reduction algorithm. In Sect. V we report the experimental results and we conclude in Sect. VI.

II. MODELS FOR k_NFA INFERENCE

The models we now present can be used directly to infer k_NFA or $(k+1)_\text{NFA}$. Let $\Sigma = \{a_1, \dots, a_n\}$ be an alphabet of n symbols. A sample $S = S^+ \cup S^-$ is given by a set S^+ of words from Σ^* that must be accepted, and a set S^- of words that must be rejected. Let K be a set of integers, $K = \{1, \dots, k\}$. We will consider the following variables in our models:

- k , an integer, the size of the NFA we want to learn,
- a set of k Boolean variables $F = \{f_1, \dots, f_k\}$ determining whether state q_i is final or not,
- and $\Delta = \{\delta_{a, \overrightarrow{q_i q_j}} \mid a \in \Sigma \text{ and } (i, j) \in K^2\}$, a set of nk^2 Boolean variables representing the transitions from state q_i to state q_j with the symbol $a \in \Sigma$, for each i, j , and a .

The path i_1, i_2, \dots, i_{m+1} for a word $w = a_1 \dots a_m$ exists iff $d = \delta_{a_1, \overrightarrow{q_{i_1} q_{i_2}}} \wedge \dots \wedge \delta_{a_m, \overrightarrow{q_{i_m} q_{i_{m+1}}}}$ is true. We call d a c_path . Let $Pref(S) = \cup_{w \in S} Pref(w)$, with $Pref(w)$ the set of non-empty prefixes of word w . Similarly, $Suf(S)$ is the set of all non-empty suffixes of the words in S .

We skip the direct model (see [2], [13]) which has a bad complexity and does not behave well in practice: the space complexity is in $\mathcal{O}(|S^+| \cdot (|\omega_+| + 1) \cdot k^{|\omega_+|})$ clauses, and $\mathcal{O}(|S^+| \cdot k^{|\omega_+|})$ variables with ω_+ the longest word of S^+ .

Prefix Model (P_k): For each $w \in Pref(S)$, we consider $p_{w, \overrightarrow{q_1 q_i}}$, a Boolean variable, determining the existence of a c_path for w from q_1 to q_i . The constraints are:

$$(\lambda \in S^+ \longrightarrow f_1) \wedge (\lambda \in S^- \longrightarrow \neg f_1) \quad (1)$$

$$\bigvee_{i \in K} \delta_{a, \overrightarrow{q_1 q_i}} \leftrightarrow p_{a, \overrightarrow{q_1 q_i}} \quad (2)$$

$$\bigwedge_{i \in K} (p_{w, \overrightarrow{q_1 q_i}} \leftrightarrow (\bigvee_{j \in K} p_{v, \overrightarrow{q_1 q_j}} \wedge \delta_{a, \overrightarrow{q_j q_i}})) \quad (3)$$

$$\bigvee_{i \in K} p_{w, \overrightarrow{q_1 q_i}} \wedge f_i \quad (4)$$

$$\bigwedge_{i \in K} (\neg p_{w, \overrightarrow{q_1 q_i}} \vee \neg f_i) \quad (5)$$

where Constraint (1) considers λ ; Constraint (2) specifies a c_path of size 1 for each $a \in Pref(S) \cap \Sigma$; for each $w = va$

with $w, v \in Pref(S)$, $a \in \Sigma$, Constraint (3) completes a path; for each word $w \in S^+ \setminus \{\lambda\}$, Constraint (4) specifies that positive words must end in a final state; and for each word $w \in S^- \setminus \{\lambda\}$, Constraint (5) specifies that if there is a path for a negative word it must end in a non-final state.

The prefix model is defined by:

$$P_k = (1) \bigwedge_{w \in Pref(S)} ((2) \wedge (3)) \bigwedge_{w \in S^+} (4) \bigwedge_{w \in S^-} (5)$$

After transformations, P_k is converted into CNF, and its space complexity is in $\mathcal{O}(\sigma \cdot k^2)$ variables, and $\mathcal{O}(\sigma \cdot k^2)$ clauses with $\sigma = \sum_{w \in S} |w|$. See [13] for details.

Suffix Model (S_k): The construction starts from any state and terminates in state q_1 . For each $w \in Suf(S)$, a Boolean variable $p_{w, \overrightarrow{q_i q_j}}$ determines the existence of a c_path for w :

$$\bigvee_{(i, j) \in K^2} \delta_{a, \overrightarrow{q_i q_j}} \leftrightarrow p_{a, \overrightarrow{q_i q_j}} \quad (6)$$

$$\bigwedge_{(i, j) \in K^2} (p_{w, \overrightarrow{q_i q_j}} \leftrightarrow (\bigvee_{l \in K} \delta_{a, \overrightarrow{q_i q_l}} \wedge p_{v, \overrightarrow{q_l q_j}})) \quad (7)$$

where Constraint (6) is used for suffixes of size 1, and Constraint (7) for longer ones. The model is:

$$S_k = (1) \bigwedge_{w \in Suf(S)} ((6) \wedge (7)) \bigwedge_{w \in S^+} (4) \bigwedge_{w \in S^-} (5)$$

Although similar to P_k , the S_k models are in $\mathcal{O}(\sigma \cdot k^3)$ variables, and in $\mathcal{O}(\sigma \cdot k^3)$ clauses [13].

Hybrid Models: Each word $w \in S$ is split into a prefix and a suffix $w = uv$ to obtain two samples $S_u = S_u^+ \cup S_u^-$ with $S_u^+ = \{u \mid \exists v \in S^+, w = uv\}$ and $S_u^- = \{u \mid \exists v \in S^-, w = uv\}$, and $S_v = S_v^+ \cup S_v^-$ with $S_v^+ = \{v \mid \exists u \in S^+, w = uv\}$ and $S_v^- = \{v \mid \exists u \in S^-, w = uv\}$. Then, Constraints (2), (3) are used for prefixes of S_u , and (6), (7) for suffixes of S_v . Finally, for each $w = uv$, clauses generated for u are linked to clauses generated for v :

$$\bigvee_{(i, j) \in K^2} p_{u, \overrightarrow{q_1 q_j}} \wedge p_{v, \overrightarrow{q_j q_i}} \wedge f_i \quad (8)$$

$$\bigwedge_{(i, j) \in K^2} (\neg p_{u, \overrightarrow{q_1 q_j}} \vee \neg p_{v, \overrightarrow{q_j q_i}} \vee \neg f_i) \quad (9)$$

Constraints (8) are used for words from S^+ and Constraints (9) for words from S^- .

Efficient decomposition of each word into a prefix and a suffix is crucial. In [12] and [14], we proposed various decomposition strategies. Here, we consider three of them:

- $ILS_k^f(\text{init})$ is based on an Iterated Local Search (ILS) [15] with the fitness f defined as $f(S_u, S_v) = |Pref(S_u)| + k \cdot |Suf(S_v)|$ for optimizing the hybrid model. The search starts with a split init for each word. At each iteration, the best split $w = uv$ is found for the word w selected randomly with a roulette wheel selection based on the weights of words defined by

$\text{weight}_w = 75\%/|S| + 25\% \cdot |w| / (\sum_{w_i \in S} |w_i|)$. The number of iterations is given and diversification is ensured with word selection.

- the **Best suffix model** (S_k^*) optimizes constructions of the suffixes by ordering the set $Suf(S)$. Let $\Omega(u) = \{w \in S \mid u \in Suf(w)\}$ be the set of words accepting u as a suffix. Then, $u_1 \succcurlyeq u_2$ iff $|u_1| \cdot |\Omega(u_1)| \geq |u_2| \cdot |\Omega(u_2)|$. The set of best suffixes is composed of the best suffixes (w.r.t. to \succcurlyeq) that cover S (see [12] for more details).
- the **Best prefix model** (P_k^*) is built in a similar way as the Best suffix model, starting with a selection of the best prefixes [12].

III. REFINED MODELS FOR $(k+1)$ _NFA

We now consider $\lambda \notin S^+$.

A. Building a $(k+1)$ _NFA from a k _NFA

From a k _NFA, we can build a $(k+1)$ _NFA with a single final state without outgoing transitions, and such that each incoming transition to the final state is a duplication of a transition of the k _NFA.

Let $A = (Q^A, \Sigma, \Delta^A, q_1, F)$ be a k _NFA. Then, there always exists a $(k+1)^*$ _NFA $A' = (Q^{A'}, \Sigma, \Delta^{A'}, q_1, F^{A'})$ such that $Q^{A'} = Q^A \cup \{q_{k+1}\}$, $F^{A'} = \{q_{k+1}\}$ and $\Delta^{A'}$:

$$\begin{aligned} \forall i, j \in (Q^A)^2 \quad \delta_{a, q_i q_j}^A &\leftrightarrow \delta_{a, q_i q_j}^{A'} \\ \forall i \in Q^A, j \in F \quad \delta_{a, q_i q_j}^A &\leftrightarrow \delta_{a, q_i q_{k+1}}^{A'} \end{aligned}$$

Sketch of the proof:

- 1) For each word $w \in S^+ \setminus \{\lambda\}$:
Let $w = va$ with $v \in Pref(S)$ and $a \in \Sigma$. Then v is recognized by A and can finish in several states $T \subseteq Q$ (not necessary final states). As $w \in S^+$, at least one transition $\delta_{a, q_i q_j}^A$ with $i \in T$ and $j \in F$ exists. By the rules of transitions creation, $\delta_{a, q_i q_{k+1}}^{A'}$ exists and so word w is recognized by A' .
- 2) For each word $w \in S^- \setminus \{\lambda\}$:
Let $w = va$ with $v \in Pref(S)$ and $a \in \Sigma$. There may be a c_path for v in A that terminates in states $T \subseteq Q$. As $w \in S^-$, if $\delta_{v, q_1 q_j}^A$ exists then $\delta_{a, q_j q_i}^A$ with $j \in T$, $i \in F$ does not exist. Thus, $\delta_{a, q_1 q_{k+1}}^{A'}$ is not created and w is then rejected by A' .

Note that if $\lambda \in S^+$, we can do a very similar construction by considering q_{k+1} and q_1 as both final. Then, the construction is identical with only some few more disjunctions between q_1 and q_{k+1} .

Example 1: Consider a sample $S = (S^+, S^-)$, such that $S^+ = \{a, ab, abba, baa\}$ and $S^- = \{aab, b, ba, bab\}$. An example minimal k _NFA and the corresponding $(k+1)^*$ _NFA are shown in Fig. 1, with the additional transitions marked in blue. The transition $\delta_{a, q_1 q_4}^{A'}$ results from the transition $\delta_{a, q_1 q_2}^A$, while the transition $\delta_{b, q_2 q_4}^{A'}$ results from the transition $\delta_{b, q_2 q_3}^A$.

Example 2: Consider now a different sample S , where $S^+ = \{\lambda, ab, abba, baa\}$, and $S^- = \{aa, aab, b, bab\}$. The k _NFA and its corresponding $(k+1)^*$ _NFA are shown in Fig. 2.

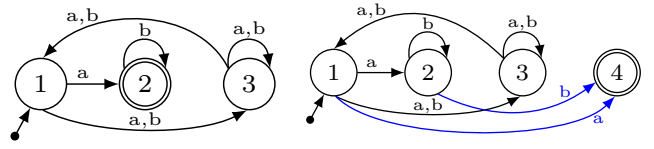


Fig. 1. Example k _NFA (left) and the corresponding $(k+1)^*$ _NFA for sample $S = (\{a, ab, abba, baa\}, \{aab, b, ba, bab\})$

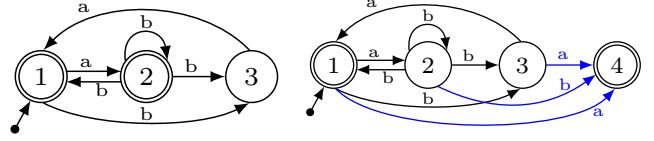


Fig. 2. Example k _NFA (left) and the corresponding $(k+1)^*$ _NFA for sample $S = (\{\lambda, ab, abba, baa\}, \{aa, aab, b, bab\})$

main differences with respect to the previous example are that state q_1 remains final in the $(k+1)^*$ _NFA due to $\lambda \in S^+$, and that state q_4 is not the final state for λ , since we assume the NFA has no λ -transitions. Note also that words ab and $abba$ are accepted in both state q_1 and q_4 . So, the only word that requires state q_4 for being accepted is baa . This is in contrast to Example 1, in which all the words in S^+ were only accepted in state q_4 .

B. Nested NFA

We now constrain more the $(k+1)$ _NFA in order to obtain fewer candidates. Consider:

- F_1 : the set of $(k+1)$ _NFA for S
- F_2 : the set of $(k+1)$ _NFA for S with only one final state
- F_3 : the set of $(k+1)$ _NFA for S with only one final state q_{k+1} and no outgoing transition from the final state
- F_4 : the set of $(k+1)$ _NFA for S with only one final state q_{k+1} and no outgoing transition from the final state, and with the property:

$$\forall u \in Suf(S^+) \cap \Sigma, \forall i \in K, \delta_{u, q_i q_{k+1}} \longrightarrow \bigvee_{j \in K} \delta_{u, q_i q_j}$$

which means that each incoming transition from q_i to q_{k+1} has a "clone" from q_i to at least one q_j which we expect to be final in the k _NFA.

- A' is a $(k+1)^*$ _NFA built by the algorithm presented before.

Then, we have that $A' \in F_4 \subseteq F_3 \subseteq F_2 \subseteq F_1$. It is thus interesting to be as close as possible to A' : this removes symmetries, and the reduction algorithm, from $(k+1)^*$ _NFA to k _NFA shown in Sect. IV succeeds more frequently.

Example 3: To illustrate the importance of the $(k+1)^*$ _NFA, let us consider the following scenario. Assume that we know that there is no NFA of size k_1 and that there exists an NFA of size k_2 , $k_2 > (k_1 + 2)$. Now, if we find a $(k+1)^*$ _NFA of size k_3 , such that $k_2 \geq k_3 > (k_1 + 1)$, then we know that there exists an NFA of size $k_3 - 1$, which means that we lowered the upper bound on the size of the minimal NFA. If

we also find there exists no $(k+1)^*_\text{NFA}$ of size k_4 such that $k_2 > k_4 > (k_1 + 1)$, then we know there is no NFA of size $k_4 - 1$. Hence, we raised the lower bound on the size of the minimal NFA. So, we know now that the size of minimal NFA k_{\min} satisfies $k_1 + 1 < k_4 \leq k_{\min} \leq k_3 - 1 < k_2$. Clearly, if $k_2 = k_1 + 2$, we can find either k_3 or k_4 but not both. However, it still proves the usefulness of the $(k+1)^*_\text{NFA}$.

C. Refining and simplifying $(k+1)_\text{NFA}$ models

We cannot model exactly A' as described before, but we can generate $(k+1)^*_\text{NFA}$. The idea here is thus to refine and simplify models to obtain $(k+1)^*_\text{NFA}$. In what follows, we consider $\lambda \notin S^+$.

Since we consider only one final state, q_{k+1} , we can omit the f_i Boolean variables, and Constraints (3) and (4) simplify to:

$$p_{w, \overrightarrow{q_1 q_{k+1}}} \quad (10)$$

for each word $w \in S^+$. This removes $\mathcal{O}(2k \cdot |S^+|)$ constraints and $\mathcal{O}(k \cdot |S^+|)$ auxiliary variables for Tseitin transformation of (3), and $\mathcal{O}((k-1) \cdot |S^+|)$ clauses in Constraints (4).

Constraint (5) can also be replaced with:

$$\neg p_{w, \overrightarrow{q_1 q_{k+1}}} \quad (11)$$

for each word of S^- . This replaces $k \cdot |S^-|$ binary clauses with $|S^-|$ unit clauses.

These simplifications are worth with the prefix and suffix models. Moreover, for the suffix model, Constraint (6) can be simplified to:

$$\bigvee_{i \in K} \delta_{a, \overrightarrow{q_i q_{k+1}}} \leftrightarrow p_{a, \overrightarrow{q_i q_{k+1}}} \quad (12)$$

and (7) to:

$$\bigwedge_{i \in K} (p_{w, \overrightarrow{q_i q_{k+1}}} \leftrightarrow (\bigvee_{j \in K} \delta_{a, \overrightarrow{q_i q_j}} \wedge p_{v, \overrightarrow{q_j q_{k+1}}})) \quad (13)$$

Finally, the complexity of the suffix model is reduced to $\mathcal{O}(\sigma \cdot k^2)$ instead of $\mathcal{O}(\sigma \cdot k^3)$.

For the hybrid model, construction is similar as before, but Constraints (8) and (9) are simplified consequently: the index i is removed from \bigvee and \bigwedge , and changed to $k+1$ in paths and transitions:

$$\bigvee_{j \in K} p_{u, \overrightarrow{q_1 q_j}} \wedge p_{v, \overrightarrow{q_j q_{k+1}}} \quad (14)$$

$$\bigwedge_{j \in K} (\neg p_{u, \overrightarrow{q_1 q_j}} \vee \neg p_{v, \overrightarrow{q_j q_{k+1}}}) \quad (15)$$

The hybrid model has now the complexity $\mathcal{O}(\sigma \cdot k^2)$.

D. Model specific refinements

The models presented thus far allow us to generate $(k+1)_\text{NFA}$, but they do not guarantee that the obtained automaton is reducible to a k_NFA . Hence, they do not guarantee obtaining the $(k+1)^*_\text{NFA}$. To improve the number of times we succeed, we now refine again the inference models. However, these refinements are not applicable to each previously

presented model, but only to the prefix model of Sect. III-C, which we will call P_{k+1} .

We now add some constraints to obtain the P_{k+1}^R model. Let f_i^k be Boolean variables meaning that i is a possible final state of the k_NFA :

- No negative word finishes in a final state of the k_NFA :

$$\bigwedge_{i \in K} \bigwedge_{w \in S^-} (p_{w, \overrightarrow{q_1 q_i}} \rightarrow \neg f_i^k). \quad (16)$$

- Each final state of the k_NFA is reached by at least one positive word w ($w = va$, $v \in \text{Pref}(S)$, and $a \in \Sigma$). The paths from state q_1 to q_{k+1} (the unique final state of the $(k+1)_\text{NFA}$) and from q_1 to q_i (final state of the k_NFA) are the same except for the last transition:

$$\bigwedge_{i \in K} (f_i^k \rightarrow \bigvee_{w \in S^+} \bigvee_{j \in K} (p_{v, \overrightarrow{q_1 q_j}} \wedge \delta_{a, \overrightarrow{q_j q_i}} \wedge \delta_{a, \overrightarrow{q_j q_{k+1}}})) \quad (17)$$

- Each positive word finishes at least in one final state of the k_NFA :

$$\bigwedge_{w \in S^+} \bigvee_{i \in K} (p_{w, \overrightarrow{q_1 q_i}} \wedge f_i^k). \quad (18)$$

Note that this refinement, which generates $(k+1)_{R_\text{NFA}}$, is only worth for the prefix models. For the other models, we still have to compute the suffixes for each pair of states, and we come back to the complexity of the k_NFA models.

IV. A REDUCTION ALGORITHM: FROM $(k+1)_\text{NFA}$ TO k_NFA

Let us consider we have a $(k+1)_\text{NFA}$ of F_4 . Then, each transition $\delta_{a, \overrightarrow{q_i q_{k+1}}}$ can be removed, and each state j such that there is $\delta_{a, \overrightarrow{q_i q_j}}$ must be considered as ‘‘possibly’’ final. In the worst case, it means 2^k possibilities of k_NFA (each of the k states is either final or not). Each of these NFAs must be tested on S to be validated. This is tractable, but we now present a simpler reduction algorithm.

Given a $(k+1)_\text{NFA}$, we can try to reduce it to a k_NFA with the following procedure:

- 1) Based on the variables $p_{w, \overrightarrow{q_1 q_i}}$ for $i \in K$ and $w \in S^-$, determine the set of candidate final states as $Q^A \setminus \{q_j\}$, where $j \in K$ and $p_{w, \overrightarrow{q_1 q_j}} = \text{true}$. This means that states in which any negative example can be reached cannot be final.
- 2) If the set of candidate final states is empty, the algorithm terminates, and the k_NFA cannot be obtained from the given $(k+1)_\text{NFA}$.
- 3) Otherwise, given a non-empty set of candidate final states, for each word $w \in S^+$ test whether the word can be reached in any candidate state by investigating variables $p_{w, \overrightarrow{q_1 q_i}}$, for these states.
- 4) If the test in the previous step returns a negative result for some word $w \in S^+$ (i.e., the word cannot be reached in any of the candidate final states), the algorithm terminates, and the k_NFA cannot be obtained from the given $(k+1)_\text{NFA}$.

- 5) Otherwise, the k _NFA can be obtained by removing the transitions leading to state $(k + 1)$ and setting all candidate states to be final.

The worst-case complexity of the algorithm is in $O(k \cdot |S|)$, since Step 1 requires $k \cdot |S^-|$ tests and Step 3 requires at most $k \cdot |S^+|$ tests. Note also, that for a $(k + 1)$ *_NFA obtained by P_{k+1}^R model the algorithm will always succeed.

Example 4: Consider the $(k + 1)$ *_NFA shown in Fig. 1. Step 1 determines the singleton set of candidate final states $\{q_2\}$ (all words $w \in S^-$ can be reached in either q_1 or q_3 , so they are excluded). Since the set is not empty, we move to step 3, and test whether each word $w \in S^+$ is reachable in state q_2 . Hence, we terminate with a k _NFA as shown in Fig. 1.

Example 5: Consider now a $(k + 1)$ _NFA shown in Fig. 3 obtained for the sample S from Example 1. With only one additional transition, $\delta_{b, q_1 q_2}$, the k _NFA cannot be built, since after step 1, the set of candidate states is empty.

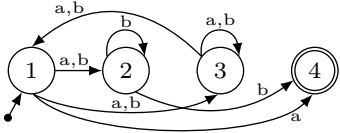


Fig. 3. Example $(k + 1)$ _NFA for sample $S = (\{a, ab, abba, baa\}, \{aab, b, ba, bab\})$

Example 6: Consider now a $(k + 1)$ _NFA shown in Fig. 4 obtained for the sample $S = (S^+, S^-)$ such that $S^+ = \{ab, abba, ba, baa\}$ and $S^- = \{aa, aab, b, bab\}$. After step 1 the set of candidate states is $\{q_1\}$ (words aa and aab are not reachable at all; words b and bab exclude states q_2 and q_3). The tests in step 3 for words $ab, abba,$ and ba succeed, but it turns out that word baa can only be reached in state q_2 which is not a candidate final state. Hence, we conclude that the k _NFA cannot be built from the given $(k + 1)$ _NFA.

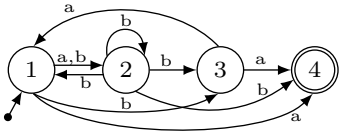


Fig. 4. Example $(k + 1)$ _NFA for sample $S = (\{ab, abba, ba, baa\}, \{aa, aab, b, bab\})$

Let us view Examples 4–6 from the perspective of constraints (16)–(18). In Example 3 it is easy to verify that all three constraints are satisfied. In Example 4, after evaluating Constraint (16), we set all f_i^k variables to *false*, thus satisfying also Constraint (17). However, we are then unable to satisfy Constraint (18). A similar observation is made for Example 5, even though we do not initially set all f_i^k variables to *false*. In this case, constraint (18) cannot be satisfied for word baa .

V. EXPERIMENTATION

A. Models for k _NFA vs. $(k + 1)$ _NFA inference

The algorithms were implemented in Python using libraries such as PySAT [16]. The experiments were carried out on a computing cluster with Intel-E5-2695 CPUs, and a fixed limit of 10 GB of memory. Running times were limited to 15 minutes, including model generation and solving time. We used the Glucose [17] SAT solver with default options.

In Tables I and II we compare the different models previously presented for k _NFA and $(k + 1)$ _NFA. Note that $ILS_k^f(init)$ is used with 3 different initial splits: random split ($ILS_k^f(r)$), split based on best prefix ($ILS_k^f(P_k^*)$), and split based on best suffix ($ILS_k^f(S_k^*)$). These experiments were carried out on state-of-the-art instances, described in [12]. These instances can be divided into three categories corresponding to the sizes of the alphabet (2, 5, and 10). The number of positive and negative words are the same in each instance and vary from 10 to 100 in increments of 10 for each category. There are thus 30 instances in total. The value of k we used for the experiments is the best known lower bound for each instance found in various papers.

For each type of model, the k _NFA and $(k + 1)$ _NFA approaches were tested. The columns of Table I and Table II show the average number of variables and clauses as well as the number of resolved instances, and among the solved instances, the number of SAT instances. The column Ctime shows the cumulative execution time (generation of the model and solving time). Since the maximum execution time is 900 seconds, each category can consume at most 9000 seconds and thus, 27000 seconds for all the instances.

Based on Table I, we see that the size of the alphabet Σ does not influence much the size of the instances: $|\Sigma|$ does not appear in the complexity which is upper bounded by σ ; but in practice, the number of prefixes (or suffixes) is related to $|\Sigma|$ (the probability to have common prefixes is higher when $|\Sigma|$ is small). The size of the instances significantly depends on the models, and on the samples (number of prefixes or suffixes). S_k and P_k^* generate the largest models, and the difference with other models is significant.

In Table II, we can notice that either $ILS_k(r)$, $ILS_k(S_k^*)$, or S_k^* are able to solve more than half of the instances of the tested samples. S_k^* is the fastest model for the total generation time and solving time. It enables us to save 3725 seconds for $(k + 1)$ _NFA which can then be used for the reduction to a k _NFA.

On average, $(k + 1)$ models are smaller than k _NFA models in terms of the number of variables. Whatever the model, more $(k + 1)$ _NFA can be inferred than k _NFA. This is due to both smaller instances and more constrained instances. Although $(k + 1)$ models can solve more instances, the cumulative time is always smaller than for k _NFA: this means that solved instances are also solved faster. $ILS_k^f(init)$ is not very sensitive to the initial splitting of words. However, it seems that the random initialization enables us to obtain smaller SAT instances. It seems that better initialization such as with best

TABLE I
COMPARISON OF THE DIFFERENT MODELS FOR k _NFA AND $(k + 1)$ _NFA. THE TESTED INSTANCES ARE DIVIDED INTO THREE CATEGORIES ACCORDING TO THEIR ALPHABET SIZE. EACH CATEGORY CONTAINS 10 INSTANCES RANGING FROM 20 TO 200 WORDS.

		Size of the alphabet = 2					Size of the alphabet = 5					Size of the alphabet = 10				
		<i>var.</i>	<i>cl.</i>	<i>solved</i>	<i>SAT</i>	<i>Ctime</i>	<i>var.</i>	<i>cl.</i>	<i>solved</i>	<i>SAT</i>	<i>Ctime</i>	<i>var.</i>	<i>cl.</i>	<i>solved</i>	<i>SAT</i>	<i>Ctime</i>
P_k	k	18 049	66 282	2	1	7 204	19 797	70 949	4	3	5 930	10 823	37 208	6	5	3 638
	$k + 1$	19 721	72 214	3	2	6 425	22 418	80 100	5	5	4 828	12 836	43 852	7	7	2 836
S_k	k	162 589	600 680	2	1	7 203	130 589	471 852	2	2	7 209	51 103	177 707	6	5	3 618
	$k + 1$	18 447	67 607	3	2	6 309	19 354	69 095	4	4	5 463	10 760	36 641	6	6	3 613
$ILS_k^f(r)$	k	24 586	94 912	2	1	7 220	22 073	82 352	4	3	5 625	10 659	38 354	5	4	4 569
	$k + 1$	7 718	28 059	4	3	5 765	12 987	46 119	5	5	4 551	8 971	30 486	7	7	2 767
$ILS_k^f(P_k^*)$	k	24 166	93 393	2	1	7 219	22 045	82 305	4	3	5 594	10 583	38 080	5	4	4 593
	$k + 1$	7 834	28 485	3	2	6 328	12 943	45 961	5	5	4 564	9 051	30 760	7	7	2 782
$ILS_k^f(S_k^*)$	k	15 781	62 423	2	1	7 220	17 383	65 431	4	3	5 596	9 280	33 536	5	4	4 595
	$k + 1$	12 891	47 189	4	3	5 468	17 621	62 884	5	5	4 892	9 978	34 003	7	7	3 015
P_k^*	k	107 425	401 231	2	1	7 204	100 673	366 869	3	3	6 392	39 232	138 115	6	5	4 762
	$k + 1$	12 402	45 381	3	2	6 305	15 202	54 167	5	5	4 972	8 625	29 271	6	6	3 612
S_k^*	k	15 811	62 503	3	2	6 307	18 062	67 697	4	3	5 669	9 890	35 495	5	4	4 559
	$k + 1$	12 962	47 384	3	2	6 315	17 646	62 969	6	6	3 749	9 961	33 949	7	7	2 747

TABLE II
SUMMARY OF THE COMPARISON OF THE DIFFERENT MODELS FOR k _NFA AND $(k + 1)$ _NFA.

		<i>var.</i>	<i>cl.</i>	<i>solved</i>	<i>SAT</i>	<i>Ctime</i>
P_k	k	16 223	58 146	12	9	16 772
	$k + 1$	18 325	65 389	15	14	14 089
S_k	k	114 760	416 747	10	8	18 031
	$k + 1$	16 187	57 781	13	12	15 385
$ILS_k^f(r)$	k	19 106	71 873	11	8	17 413
	$k + 1$	9 892	34 888	16	15	13 083
$ILS_k^f(P_k^*)$	k	18 931	71 259	11	8	17 406
	$k + 1$	9 943	35 069	15	14	13 674
$ILS_k^f(S_k^*)$	k	14 148	53 797	11	8	17 411
	$k + 1$	13 496	48 025	16	15	13 374
P_k^*	k	82 443	302 072	11	9	18 358
	$k + 1$	12 077	42 940	14	13	14 889
S_k^*	k	14 587	55 232	12	9	16 535
	$k + 1$	13 523	48 101	16	15	12 810

prefixes or best suffixes stay stuck in local minima, close to the initialization.

The statistical analysis of the k _NFA vs. $(k + 1)$ _NFA models shows no statistically significant differences in the execution times if we assume the execution time of 900 seconds for the unsolved instances. However, when we focus only on the solved instances, we get statistically significant differences based on the Kruskal-Wallis test (p -value = $1.4 \cdot 10^{-6}$). The post-hoc Dunn's analysis shows that the significant differences are between all ILS-based models and the rest of the models (in favor of the other models, which have lower mean ranks). There are no significant differences between k _NFA and $(k + 1)$ _NFA for the same model type.

B. $(k + 1)$ _NFA to k _NFA reduction algorithm

The results shown in Tables I and II were obtained for the base and refined models, excluding the refinements of Sect. III-D. Consequently, the vast majority of $(k + 1)$ _NFA could not be reduced to k _NFA. In the second experiment,

all instances have been tested with the P_k model for k _NFA, $(k + 1)$ _NFA and reducible $(k + 1)_{R_NFA}$ with a maximum running time of 15 minutes (900 seconds).

Table III details the results for instances solved and proved SAT. For each instance, the k column provides three values corresponding to the sizes of the k _NFA, $(k + 1)$ _NFA, and reducible $(k + 1)_{R_NFA}$, respectively. Columns *var.* and *cl.* represent the number of variables and clauses. Column *Stime* is the generation and solving time, while *Rtime* is the running time of the reduction algorithm. Finally, *Ttime* is the total running time.

For each instance the best time to obtain a k -state NFA (either directly or through reduction) is bolded. A star (\star) is added when the reduction of the reducible $(k + 1)_{R_NFA}$ permits to find a k _NFA whereas it is not possible directly using the model for k _NFA. When it takes longer to reduce $(k + 1)_{R_NFA}$ to a k _NFA than to find the k _NFA, a bullet (\bullet) is added. When the running time exceeded 15 minutes, nothing is written. Clearly, reduction time is only provided for $(k + 1)$ _NFA and reducible $(k + 1)_{R_NFA}$.

We can observe in Table III that reducible $(k + 1)_{R_NFA}$ obtains the best total time for 12 of the 14 instances. Moreover, for instances *st-2-20* and *ww-10-70* NFA is found by reduction of the $(k + 1)_{R_NFA}$ whereas no NFA is found by k _NFA. It proves that it is worth using reducible $(k + 1)_{R_NFA}$ rather than k _NFA. We can note that only instances *st-5-30* and *ww-10-30* can be reduced from $(k + 1)$ _NFA to k _NFA, and the total running time is better only once than for reducible $(k + 1)_{R_NFA}$.

The number of variables and clauses of reducible $(k + 1)_{R_NFA}$ is bigger than for $(k + 1)$ _NFA but it does not impact the efficiency. Moreover, the reduction time is irrelevant with respect to the solving time.

VI. CONCLUSION

Grammatical inference consists in learning a formal grammar, in our case as an NFA. We have presented how we can

TABLE III
DETAILED RESULTS FOR P_k MODEL FOR k_NFA , $(k + 1)_NFA$ AND REDUCIBLE $(k + 1)_R_NFA$. THE RESULTS INCLUDE ONLY SOLVED AND PROVED SAT INSTANCES.

Instance	k	var.	cl.	S _{time}	R _{time}	T _{time}
st-2-10	4	1 276	4 250	1.67		1.67
	5	1 550	5 128	1.88	0.01	1.89
	5 R	1754	5942	0.96	0.01	0.97
st-2-20	7	6 461	23 113			
	8	7 232	25 806	121.66	0.01	121.68
	8 R	8 359	30 313	185.44	0.02	185.46*
st-5-10	3	1 098	3 445	1.82		1.82
	4	1 440	4 455	1.77	0.00	1.78
	4 R	1 563	4 948	1.07	0.01	1.07
st-5-20	4	4 024	13 464	1.99		1.99
	5	4 950	16 465	1.85	0.00	1.85
	5 R	5 354	18 089	1.06	0.01	1.07
st-5-30	5	8 020	27 720	4.84		4.84
	6	9 473	32 593	1.66	0.01	1.67
	6 R	10 373	36 228	2.23	0.05	2.28
st-5-40	5	9 390	32 550	19.77		19.77*
	6	11052	38157	11.58	0.01	11.59
	6 R	12257	43002	20.62	0.02	20.63
st-5-50	6	15 564	55 100			
	7	17 836	62 955	354.36	0.02	354.38
	7 R	20 706	74 327			
ww-10-10	3	831	2 431	1.71		1.71
	4	1 104	3 118	1.54	0.00	1.54
	4 R	1 227	3 611	1.27	0.00	1.27
ww-10-20	3	2 613	8 255	1.65		1.65
	4	3 440	10 730	1.63	0.00	1.63
	4 R	3 683	11 713	1.01	0.06	1.07
ww-10-30	4	1 443	4 485	1.60		1.60
	5	1 840	5 550	1.58	0.00	1.58
	5 R	3 479	11 459	1.06	0.01	1.06
ww-10-40	4	3 624	11 900	1.39		1.39
	5	4 375	14 145	1.53	0.01	1.53
	5 R	5 179	17 389	1.00	0.03	1.03
ww-10-50	4	5 364	17 850	1.67		1.67
	5	6 500	21 390	2.44	0.01	2.45
	5 R	7 504	25 444	1.59	0.01	1.60
ww-10-60	5	10 995	37 800	27.18		27.18
	6	12 888	44 018	3.23	0.01	3.24
	6 R	14 693	51 283	12.27	0.03	12.30
ww-10-70	5	14 525	50 190			
	6	17 064	58 654	121.27	0.01	121.28
	6 R	19 169	67 129	349.95	0.02	349.98*

benefit from a very simple property to refine and improve some already refined models from [12]. The property says that if there is a k_NFA , there is also a $(k + 1)_NFA$. This property has been strengthened to bring us closer to the $(k + 1)^*_NFA$ that one could build manually.

Thanks to the proposed refinements, we managed to obtain smaller models, with a lower complexity decreasing from $\mathcal{O}(\sigma \cdot k^3)$ to $\mathcal{O}(\sigma \cdot k^2)$. With some experiments, we have shown that the $(k + 1)_NFA$ are easier and faster to infer than the k_NFA . We have also presented an algorithm to reduce $(k + 1)_NFA$ to k_NFA . Although the $(k + 1)$ models are closer to the $(k + 1)^*_NFA$ obtainable with the simple algorithm given in this paper, the reduction algorithm does not always succeed. We have shown that given some model specific refinements, we can keep the low complexity of the model and infer $(k + 1)_R_NFA$ that can be very quickly reduced to k_NFA .

In the future, we plan to refine again the $(k + 1)_NFA$

models for the reduction algorithm to succeed more often. For example, we want to add constraints such as the ones for the P_k model. We also plan to integrate symmetry breaking techniques to reduce the initial search space. But to this end, we will have to be very cautious and verify that symmetry breaking constraints are compatible with the $(k + 1)_NFA$ models.

REFERENCES

- [1] C. de la Higuera, *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.
- [2] T. Jastrzab, Z. J. Czech, and W. Wiecek, "Parallel algorithms for minimal nondeterministic finite automata inference," *Fundam. Informaticae*, vol. 178, no. 3, pp. 203–227, 2021. [Online]. Available: <https://doi.org/10.3233/FI-2021-2004>
- [3] W. Wiecek, *Grammatical Inference - Algorithms, Routines and Applications*, ser. Studies in Computational Intelligence. Springer, 2017, vol. 673.
- [4] F. Denis, A. Lemay, and A. Terlutte, "Learning regular languages using rfsas," *Theor. Comput. Sci.*, vol. 313, no. 2, pp. 267–294, 2004.
- [5] M. Vázquez de Parga, P. García, and J. Ruiz, "A family of algorithms for non deterministic regular languages inference," in *Proc. of CIAA 2006*, ser. LNCS, vol. 4094. Springer, 2006, pp. 265–274.
- [6] M. Tomita, "Dynamic construction of finite-state automata from examples using hill-climbing," *Proc. of the Fourth Annual Conference of the Cognitive Science Society*, pp. 105–108, 1982.
- [7] P. Dupont, "Regular grammatical inference from positive and negative samples by genetic search: the GIG method," in *Proc. of ICGI 94*, ser. LNCS, vol. 862. Springer, 1994, pp. 236–245.
- [8] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*, 1st ed. Elsevier Science, 2006.
- [9] T. Jastrzab, "On parallel induction of nondeterministic finite automata," in *Proc. of ICCS 2016*, ser. Procedia Computer Science, vol. 80. Elsevier, 2016, pp. 257–268.
- [10] —, "Two parallelization schemes for the induction of nondeterministic finite automata on PCs," in *Proc. of PPAM 2017*, ser. LNCS, vol. 10777. Springer, 2017, pp. 279–289.
- [11] —, "A comparison of selected variable ordering methods for NFA induction," in *Proc. of ICCS 2019*, ser. LNCS, vol. 11540. Springer, 2019, pp. 741–748.
- [12] F. Lardeux and E. Monfroy, "Optimized models and symmetry breaking for the NFA inference problem," in *33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021, Washington, DC, USA, November 1-3, 2021*. IEEE, 2021, pp. 396–403. [Online]. Available: <https://doi.org/10.1109/ICTAI52525.2021.00065>
- [13] —, "GA and ILS for optimizing the size of NFA models," in *The 8th International Conference on Metaheuristics and Nature Inspired Computing (META)*, Marrakech, Morocco, Oct. 2021. [Online]. Available: <https://hal.univ-angers.fr/hal-03284541>
- [14] —, "Improved SAT models for NFA learning," in *International Conference in Optimization and Learning (OLA)*, Catania, Italy, Jun. 2021. [Online]. Available: <https://hal.univ-angers.fr/hal-03284571>
- [15] T. Stützle and R. Ruiz, *Iterated Local Search*. Cham: Springer International Publishing, 2018, pp. 579–605. [Online]. Available: https://doi.org/10.1007/978-3-319-07124-4_8
- [16] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python toolkit for prototyping with SAT oracles," in *SAT*, 2018, pp. 428–437. [Online]. Available: https://doi.org/10.1007/978-3-319-94144-8_26
- [17] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *Proc. of IJCAI 2009*, 2009, pp. 399–404.