Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT
Theoretical Computer Science 313 (2004) 267–294

Theoretical Computer Science

www.elsevier.com/locate/tcs

Learning regular languages using RFSAs☆

François Denis[a], Aurélien Lemay[b,*], Alain Terlutte[b]

[a] *LIF-CMI, UMR 6166, 39, rue F. Joliot Curie, 13453 Marseille Cedex 13, France*
[b] *GRAPPA—LIFL, Bât M3, Université de Lille I, 59655 Villeneuve d'Ascq Cedex, France*

Abstract

Residual languages are important and natural components of regular languages and several grammatical inference algorithms naturally rely on this notion. In order to identify a given target language L, classical inference algorithms try to identify words which define identical residual languages of L. Here, we study whether it could be interesting to perform a tighter analysis by identifying inclusion relations between the residual languages of L. We consider the class of Residual Finite State Automata (RFSAs). An RFSA A is a NonDeterministic Automaton whose states corresponds to residual languages of the language L_A it recognizes. The inclusion relations between residual languages of L_A can be naturally materialized on A. We prove that the class of RFSAs is not polynomially characterizable. We lead some experiments which show that when a regular language is randomly drawn by using a nondeterministic representation, the number of inclusion relations between its residual languages is very important. Moreover, its minimal RFSA representation is much smaller than its minimal DFA representation. Finally, we design a new learning algorithm, DeLeTe2, based on the search for the inclusion relations between the residual languages of the target language. We give sufficient conditions for the identifiability of the target language. We experimentally compare the performance of DeLeTe2 to those of classical inference algorithms.

Keywords: Grammatical inference; Machine learning; Regular languages; Finite state automata

1. Introduction

The subject of this paper is grammatical inference of regular languages. In this field, regular languages are mostly represented by Deterministic Finite Automata (DFAs):

☆ This work was partially supported by the "projet TIC du CPER TACT—région Nord—Pas de Calais".
* Corresponding author.
E-mail addresses: fdenis@cmi.univ-mrs.fr (F. Denis), lemay@univ-lille3.fr (A. Lemay), terlutte@lifl.fr (A. Terlutte).

0304-3975/$ - see front matter © 2003 Published by Elsevier B.V.
doi:10.1016/j.tcs.2003.11.008

operations on DFAs are fast, and every regular language is recognized by a unique minimal DFA. However, it is well known that using DFAs is not the optimal way to represent regular languages. The size of the minimal DFA of languages as simple as $\Sigma^*0\Sigma^n$ is exponential with respect to n. Thus, these languages cannot be efficiently learnt by classical algorithms. It seems natural to learn regular languages by using nondeterministic representations [3,4,16]. We presented in [5,6] a new class of nondeterministic finite automata, the class of Residual Finite State Automata (RFSAs), based on the notion of residual languages. The residual language of a language L with respect to a word u is the set of words v such that uv is in L. As it is stated by the Myhill–Nerode Theorem, a language is regular if and only if it has a finite number of residual languages. An RFSA is a nondeterministic automaton each state of which is naturally associated with a residual language of the language it recognizes. RFSAs have interesting properties which have been studied in [6]: for instance, every regular language L can be represented by a unique minimal canonical RFSA whose states correspond with the *prime* residual languages of L, i.e. the residual languages which are not unions of other residual languages; moreover, canonical RFSAs can be exponentially smaller than their equivalent minimal DFAs.

T. Yokomori presented in [16] an algorithm that infers NFAs from *Minimally Adequate Teacher* [1] (i.e. using membership and equivalence queries). The NFAs output by his algorithm are RFSAs (but T. Yokomori did not bring out this kind of automata). A first learning algorithm using this representation has been described in [4].

We will prove in Section 3 that it is not possible to infer a regular language by using a sample whose size is polynomial in the size of its canonical RFSA. But this worst case negative result does not necessarily mean that RFSAs are unavailing for grammatical inference. In order to know whether RFSAs can be useful, we will first compare the sizes of minimal DFAs and canonical RFSAs of randomly generated regular languages. This comes down to studying the ratio between the number of prime residual languages of a randomly generated regular language and the total number of its residual languages. We also study the number of inclusion relations between the residual languages of the randomly generated languages, as such inclusion relations are used to build canonical RFSAs and can be exploited in learning algorithms. We will consider several ways to generate regular languages, based on classical representations: DFAs, NFAs and regular expressions. We will show in Section 4 that experimental results highly depend on the way regular languages are generated. When a language is generated by a random DFA, most of its residual languages are prime, entailing that its canonical RFSA is quite as big as its equivalent minimal DFA, and there are few inclusion relations between its residual languages. But when a regular language is generated by a random NFA or a random regular expression, many residual languages are composite (not prime), entailing that its canonical RFSA is much smaller than its minimal DFA, and that there are a lot of inclusion relations between its residual languages.

From the grammatical inference point of view, we are interested in one particular advantage of RFSAs. The classical algorithms, such as RPNI [12,15], look for words defining identical residual languages and merge states associated with these words to obtain a DFA. Using RFSAs allows us to perform a tighter analysis and to look for

inclusion relations between residual languages. A saturation operator is then used to materialize them as transitions in the automaton. We will present in Section 5 the learning algorithm DeLeTe2 that follows this method. DeLeTe2 looks for inclusion relations between the residual languages of the target language, uses them to add transitions to the current automaton, and finally outputs an RFSA. We show that DeLeTe2 runs in polynomial time.

Section 6 will present experimental results which show that DeLeTe2 performs better than other classical learning algorithms when regular languages are generated using nondeterministic representations.

2. Preliminaries

The reader may refer to [11,17] for classical definitions and proofs on formal language theory. The notions of prime and composite residual languages and RFSAs have been introduced in [5] and studied in [6].

2.1. Regular languages, regular expressions and automata

Let Σ be a finite alphabet and let Σ^* be the set of words on Σ. We denote by ε the empty word and by $|u|$ the length of a word u. We simply denote by uv the concatenation of two words u and v. A *language* is a subset of Σ^*. For any word u and any language L, we define $Pref(u) = \{v \in \Sigma^* \mid \exists w \in \Sigma^*,\ vw = u\}$ and $Pref(L) = \bigcup_{u \in L} Pref(u)$. We denote by $<$ the length-lexicographic order. We denote by $\bar{L}$ the complementary of a language L, i.e. the set of words of Σ^* not in L. For any languages L_1 and L_2 over Σ, we define $L_1 \cdot L_2 = \{u_1u_2 \mid u_1 \in L_1 \text{ and } u_2 \in L_2\}$. For any language L, we define $L^0 = \{\varepsilon\}$ and for any integer n, $L^{n+1} = L^n \cdot L$; we also define $L^* = \bigcup_{n \geqslant 0} L^n$ and $L^+ = \bigcup_{n \geqslant 1} L^n$.

A *nondeterministic finite automaton* (NFA) is a 5-tuple $A = \langle \Sigma, Q, Q_0, F, \delta \rangle$ where Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, δ is the transition function defined from $Q \times \Sigma$ to 2^Q. As usual, we also denote by δ the extended transition function defined from $2^Q \times \Sigma^*$ to 2^Q. We take the number of states of an NFA as a measure of its size. Two NFAs $A = \langle \Sigma, Q, Q_0, F, \delta \rangle$ and $A' = \langle \Sigma, Q', Q'_0, F', \delta' \rangle$ are *isomorphic* if there exists a bijection $f : Q \to Q'$ such that $f(Q_0) = Q'_0$, $f(F_0) = F'_0$ and for any states $q_1, q_2 \in Q$ and any letter $x \in \Sigma$, $q_2 \in \delta(q_1, x)$ iff $f(q_2) \in \delta'(f(q_1), x)$. Two NFAs are *equivalent* if they recognize the same language.

An NFA is *deterministic* (DFA) if Q_0 contains exactly one element q_0 and if $\forall q \in Q$, $\forall x \in \Sigma$, $Card(\delta(q,x)) \leqslant 1$. A DFA is *complete* if $\forall q \in Q$, $\forall x \in \Sigma$, $Card(\delta(q,x)) = 1$. An NFA is *trimmed* if $\forall q \in Q$, $\exists w_1, w_2 \in \Sigma^*$, $q \in \delta(Q_0, w_1)$ and $\delta(q, w_2) \cap F \neq \emptyset$. A word $u \in \Sigma^*$ is recognized by an NFA if $\delta(Q_0, u) \cap F \neq \emptyset$ and the language L_A recognized by A is the set of words recognized by A. We denote by $Rec(\Sigma^*)$ the class of recognizable languages over Σ^*. There exists a unique minimal DFA that recognizes a given recognizable language (minimal with respect to the number of states and unique up to an isomorphism).

A *regular expression* e denotes a *regular language* $L(e)$ if
- $e=\emptyset$ and $L(e)=\emptyset$;
- $e=\varepsilon$ and $L(e)=\{\varepsilon\}$;
- $e=x$ and $L(e)=\{x\}$ where $x\in\Sigma$;
- $e=e_1+e_2$ and $L(e)=L(e_1)\cup L(e_2)$;
- $e=e_1\cdot e_2$[1] and $L(e)=L(e_1)\cdot L(e_2)$;
- $e=e_1^*$ and $L(e)=(L(e_1))^*$.

A language is *regular* if it is denoted by a regular expression. The Kleene's Theorem states that the class of regular languages $Reg(\Sigma^*)$ is identical to $Rec(\Sigma^*)$.

2.2. Residual languages and RFSAs

For any language L and any word u over Σ, the *residual* language of L associated with u (also called Brzozowski derivative [2] or left quotient) is defined by $u^{-1}L=\{v\in\Sigma^*\mid uv\in L\}$ and we say that u is a *characterizing word* for $u^{-1}L$. The number of residual languages of a language L is finite if and only if L is regular (Myhill–Nerode theorem). A residual language is *composite* if it is equal to the union of the residual languages it strictly contains i.e. $u^{-1}L$ is composite if and only if $u^{-1}L=\bigcup\{v^{-1}L\mid v^{-1}L\subsetneq u^{-1}L\}$. A residual language is *prime* if it is not composite.

Let $A=\langle\Sigma,Q,Q_0,F,\delta\rangle$ be an NFA and let $q\in Q$. We denote by $L_{A,q}$ the language $\{v\mid\delta(q,v)\cap F\neq\emptyset\}$. If A is a trimmed DFA, $L_{A,q}$ is always a residual language of L_A; moreover, if A is minimal, for every nonempty residual language $u^{-1}L_A$, there exists a unique $q\in Q$ such that $L_{A,q}=u^{-1}L_A$.

Definition 1 (Denis et al. [7]). A Residual Finite State Automaton (RFSA) is an NFA $A=\langle\Sigma,Q,Q_0,F,\delta\rangle$ such that, for each state $q\in Q$, $L_{A,q}$ is a residual language of L_A.

Trimmed DFAs are RFSAs. It can be proved that for each prime residual $u^{-1}L_A$ of an RFSA A, there exists a state q such that $L_{A,q}=u^{-1}L_A$ [7]. A state q of an RFSA is said to be *prime* (resp. composite) if the residual language it defines is prime (resp. composite).

Two operators are defined in [5] about the notion of residual language: the saturation and the reduction operators.

Definition 2. The saturation operator. Let $A=\langle\Sigma,Q,Q_0,F,\delta\rangle$ be an NFA. We define $A^s=\langle\Sigma,Q,Q_0^s,F,\delta^s\rangle$ where $Q_0^s=\{q\in Q\mid L_{A,q}\subseteq L_A\}$ and $\forall q\in Q$, $\forall x\in\Sigma$, $\delta^s(q,x)=\{q'\in Q\mid L_{A,q'}\subseteq x^{-1}L_{A,q}\}$. We say that A is *saturated* if $A=A^s$.

Proposition 1 (Denis et al. [7]). *For any NFA A, $A^s=(A^s)^s$ and for every state q of A, $L_{A,q}=L_{A^s,q}$. As a consequence, A and A^s are equivalent and if A is an RFSA, then A^s is also an RFSA.*

Example 1 shows a minimal DFA A and the equivalent saturated RFSA A^s.

[1] For sake of clarity, we often omit the $\cdot$ and simply denote $e=e_1e_2$.

Definition 3. The reduction operator. Let $A = \langle \Sigma, Q, Q_0, F, \delta \rangle$ be a saturated NFA, and let q be a state of Q.

If q is such that $L_{A,q} = \bigcup \{L_{A,q'} \mid q' \in Q \setminus \{q\}$ and $L_{A,q'} \subseteq L_{A,q}\}$, we define $\phi(A, q) = \langle \Sigma, Q', Q'_0, F', \delta' \rangle$ where

- $Q' = Q \setminus \{q\}$,
- $Q'_0 = Q_0 \cap Q'$,
- $F' = F \cap Q'$, and
- $\delta'(q', x) = \delta(q', x) \cap Q'$ for every $q' \in Q'$ and every $x \in \Sigma$.

Otherwise, we define $\phi(A, q) = A$.

So, the reduction operator simply removes a state q from a saturated NFA A whenever the language associated with q can be expressed with the help of the other states of A.

Proposition 2 (Denis et al. [7]). *Let A be a saturated NFA and let q and q' be two distinct states of A. Then,*

- *$\phi(A, q)$ is saturated,*
- *$L_{\phi(A,q),q'} = L_{A,q'}$ and*
- *A and $\phi(A, q)$ are equivalent.*

Therefore, if A is an RFSA, then $\phi(A, q)$ is an RFSA.

Definition 4. A saturated NFA A is *reduced* if for any state q of A, we have $\phi(A, q) = A$.

There may exist several equivalent reduced saturated NFAs. However, it can be shown that every regular language is recognized by a unique reduced saturated RFSA.

Proposition 3 (Denis et al. [7]). *Every regular language L is recognized by a unique (up to an isomorphism) reduced saturated RFSA A which is called the* canonical RFSA *of L. The number of states of A is exactly the number of prime residual languages of L.*

Therefore, a canonical RFSA may have far fewer states than the equivalent minimal DFA: for example, the canonical RFSA of $\Sigma^* 0 \Sigma^n$ has $n+2$ states while the equivalent minimal DFA has 2^{n+1} states. However, it can be shown that the ratio between the number of states of the minimal DFA and the canonical RFSA for a one-letter alphabet regular language is at most quadratic.

Proposition 4 (Denis et al. [7]). *Let L be a regular language over a one-letter alphabet. Let n_R (resp. n_P) be the number of nonempty residual languages (resp. prime residual languages) of L. Then,*

$$n_R \leqslant (n_P - 1)^2 + 2.$$

The number of transitions of a canonical RFSA is not minimal and it can be shown that there may exist several equivalent nonisomorphic RFSAs having as many states

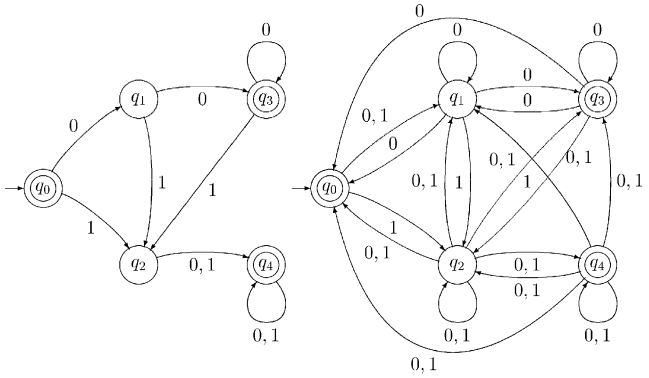

Fig. 1. The minimal DFA A recognizing $\varepsilon + 00^+ + 0^*1\Sigma^+$ and its corresponding saturated RFSA A^s.

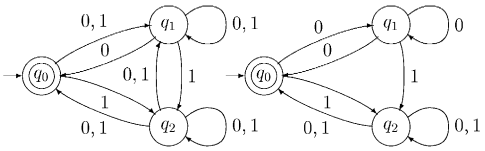

Fig. 2. The canonical RFSA and the simplified canonical RFSA of the language $\varepsilon + 00^+ + 0^*1\Sigma^+$.

as the equivalent canonical RFSA but fewer transitions. However, the definition below describes a way to rule out some redundant transitions, while preserving the recognized language.

Definition 5. Let $A = \langle \Sigma, Q, Q_0, F, \delta \rangle$ be a canonical RFSA. The *simplified* canonical RFSA of L_A is the NFA $A' = \langle \Sigma, Q, Q_0', F, \delta' \rangle$ such that $Q_0' = \{q \in Q_0 \mid \nexists q' \in Q_0, L_{A,q} \subsetneq L_{A,q'}\}$ and $\forall q' \in Q,\ \forall x \in \Sigma,\ \delta'(q',x) = \{q \in \delta(q',x) \mid \nexists q'' \in \delta(q',x),\ L_{A,q} \subsetneq L_{A,q''}\}$.

It is shown in [7] that the simplified canonical RFSA of a regular language L is an RFSA which recognizes L.

Example 1. Let A be the minimal DFA recognizing $L_A = \varepsilon + 00^+ + 0^*1\Sigma^+$; A is described in Fig. 1. The residual languages of L_A are $L_{q_0} = \varepsilon + 00^+ + 0^*1\Sigma^+ = \varepsilon^{-1}L_A$, $L_{q_1} = 0^+ + 0^*1\Sigma^+ = 0^{-1}L_A$, $L_{q_2} = \Sigma^+ = 1^{-1}L_A$, $L_{q_3} = 0^* + 0^*1\Sigma^+ = (00)^{-1}L_A$, $L_{q_4} = \Sigma^* = (10)^-1L_A$.

We have $L_{q_1} \subset L_{q_2}$; L_{q_3} and L_{q_4} are composite since $L_{q_3} = L_{q_0} \cup L_{q_1}$ and $L_{q_4} = L_{q_0} \cup L_{q_1} \cup L_{q_2} \cup L_{q_3}$. The prime states are q_0, q_1, q_2. The saturated NFA A^s is described in Fig. 1; the equivalent canonical RFSA and the simplified canonical RFSA are described in Fig. 2.

RFSAs have a minimal canonical form, as DFAs have. However, most computations on RFSAs are as complex as on NFAs.

Proposition 5 (Denis et al. [7]). *The following problems are PSPACE-complete*:
(1) *deciding whether an NFA is saturated*;
(2) *deciding whether an NFA is an RFSA*;
(3) *deciding whether a DFA is a canonical RFSA*.

2.3. Identification in the limit from polynomial time and data

The learning model of *Identification in the limit from polynomial time and data* has been introduced in [9] where Gold proved that regular languages represented by DFAs are learnable in this framework (see [10] for a comment on this result and related works). An *example* of a language L over Σ^* is a pair (u,e) where $e=1$ if $u\in L$ (positive example) and $e=0$ otherwise (negative example). A *sample* of L is a finite set of examples of L. The size of a sample is the sum of the lengths of the words it contains. For any sample S of L, we denote $S_+=\{u\,|\,(u,1)\in S\}$ and $S_-=\{u\,|\,(u,0)\in S\}$.

Here, we only consider the class of regular languages $Reg(\Sigma^*)$. We consider three *representation schemes*: *DFAs*, *RFSAs* and *NFAs*.

Definition 6 (de la Higuera [10]). We say that $Reg(\Sigma^*)$ is *identifiable in the limit from polynomial time and data* using the representation scheme R if there exist two algorithms $\mathscr{T}$ and $\mathscr{L}$ such that for any target language $L\in Reg(\Sigma^*)$ and any representation r of L in the representation scheme R:
- for any sample S, $\mathscr{L}$ computes a representation in R compatible with S in time polynomial in the size of S,
- $\mathscr{T}$ with input r computes a *characteristic* sample S_L whose size is polynomial in the size of r,
- if a sample S contains S_L, $\mathscr{L}$ with input S computes a representation r' equivalent to r.

$Reg(\Sigma^*)$ is identifiable in the limit from polynomial time and data using the representation scheme of DFAs [9]. Since this seminal paper, alternative algorithms have been proposed to infer DFAs in this framework [12,15]. For example, the *Regular Positive and Negative Inference* (RPNI) algorithm [15] is a polynomial time algorithm which outputs a DFA consistent with the input sample and which can be used to identify regular languages from polynomial time and data. Given a learning sample S, RPNI.
- constructs the prefix tree acceptor[2] A that recognizes the finite language S_+ (each word of $Pref(S_+)$ can be identified with exactly one state of A);
- orders the states of A according to the corresponding words and searches the pairs of states according to this order;
- merges pairs of states and propagates the merges in order to keep a deterministic automaton, under the control of S_-.

[2] A prefix tree acceptor is a DFA with a tree-like structure, i.e. such that each state has at most one predecessor.

In other words, the RPNI algorithm tries to identify prefixes of the target language which define equal residual languages, under the control of negative examples.

Later on, variants of the RPNI algorithm have been introduced. Rather than performing merges in a predetermined order, they try to perform those merges that are supported by the most evidence (evidence-driven state merging algorithms) [13].

However, all these algorithms fail to efficiently infer languages as simple as $\Sigma^*0\Sigma^n$ because these languages have exponentially long DFA representations. So the question whether regular languages are identifiable in the limit from polynomial time and data by using more concise representations arises naturally. C. de la Higuera gave in [10] a necessary condition for $Reg(\Sigma^*)$ to be identifiable in the limit from polynomial time and data based on the following notion:

Definition 7 (de la Higuera [10]). $Reg(\Sigma^*)$ is *polynomially characterizable* using the representation scheme R if there exists a function $\mathscr{T}$ such that

- for any language $L \in Reg(\Sigma^*)$ and any representation $r \in R(L)$, $\mathscr{T}(r)$ is a sample of L whose size is polynomial in the size of r,
- for any pair of distinct languages (L, L') represented by (r, r'), L is not consistent with $\mathscr{T}(r')$ or L' is not consistent with $\mathscr{T}(r)$.

Proposition 6 (de la Higuera [10]).

- *If $Reg(\Sigma^*)$ is identifiable in the limit from polynomial time and data using a representation scheme R then it is polynomially characterizable using R.*
- *$Reg(\Sigma^*)$ is not polynomially characterizable using representation by NFAs (even if $Card(\Sigma) = 1$) and therefore $Reg(\Sigma^*)$ is not identifiable in the limit from polynomial time and data using NFAs.*

3. Regular languages are not polynomially characterizable using RFSAs

As the canonical RFSA of a regular language may be much more concise than the minimal equivalent DFA, it could then be an interesting target for grammatical inference. Is $Reg(\Sigma^*)$ identifiable in the limit from polynomial time and data using RFSAs? We show below that the answer is no, except when the alphabet contains only one letter (but in this case, the sizes of DFAs and RFSAs are not very different).

Proposition 7. *The class of regular languages over Σ is polynomially characterizable using RFSAs if and only if $Card(\Sigma) = 1$.*

Proof. Let Σ be a one-letter alphabet. Proposition 4 shows that the size of the canonical RFSA of a regular language over Σ is not smaller than the square root of the size of its equivalent minimal DFA. Therefore, any inference algorithm that identifies $Reg(\Sigma^*)$ using DFAs can be used to identify $Reg(\Sigma^*)$ in the limit from polynomial time and data using RFSAs. Then, Proposition 6 shows that $Reg(\Sigma^*)$ is polynomially characterizable using RFSAs.

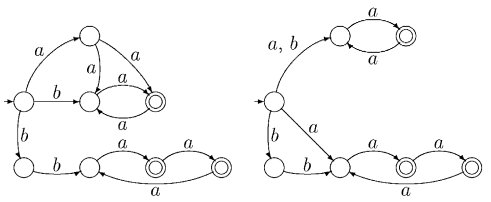

Fig. 3. Automata A_1 and A_2 for $p_1 = 2$ and $p_2 = 3$ (some transitions have been omitted for sake of clarity).

Suppose now that $Card(\Sigma) \geqslant 2$ and let a, b be two different letters from Σ. The construction which follows is similar to the one used in [10]. Let $p_1, \ldots, p_k$ be distinct prime numbers and let $L_{p_i} = a^*\backslash(a^{p_i})^* = \{a^{np_i+m} \mid n \in \mathbb{N},\ 0 < m < p_i\}$, for each i, $1 \leqslant i \leqslant k$. Let us consider the two canonical RFSAs A_1 and A_2 recognizing the languages

$$L_{A_1} = aa^+ + \bigcup_{i=1}^{k} b^i L_{p_i} \quad \text{and} \quad L_{A_2} = \bigcup_{i=1}^{k} (a + b^i) L_{p_i}.$$

Automaton A_1 has $\Sigma_{i=1}^k p_i + k + 1$ states and automaton A_2 has $\Sigma_{i=1}^k p_i + k$ states (see Fig. 3). For any polynomial P, we can choose prime numbers such that $\Pi_{i=1}^k p_i$ is larger than $P(\Sigma_{i=1}^k p_i + k + 1)$. Check that:

$$\begin{aligned}
L_{A_1} \cap a^* &= aa^+, \\
L_{A_2} \cap a^* &= a\left(\bigcup_{i=1}^{k} L_{p_i}\right) = a\left(\bigcup_{i=1}^{k} (a^*\backslash(a^{p_i})^*)\right) \\
&= a\left(a^* \Big\backslash \bigcap_{i=1}^{k} (a^{p_i})^*\right) = a(a^*\backslash(a^{\Pi_{i=1}^k p_i})^*), \\
L_{A_1} \cap b(a+b)^* &= L_{A_2} \cap b(a+b)^*, \\
L_{A_1} \cap (a+b)^{<\Pi_{i=1}^k p_i} &= L_{A_2} \cap (a+b)^{<\Pi_{i=1}^k p_i}.
\end{aligned}$$

Therefore, any set S whose size is smaller than $\Pi_{i=1}^k p_i$ satisfies $S \cap L_{A_1} = S \cap L_{A_2}$ and $S \cap \overline{L_{A_1}} = S \cap \overline{L_{A_2}}$.

Suppose now that $Reg(\Sigma^*)$ is polynomially characterizable using RFSAs. Let $\mathscr{T}$ be the function that computes characteristic samples and let P be a polynomial such that $size(\mathscr{T}(A)) \leqslant P(size(A))$. Let k be such that $\Pi_{i=1}^k p_i > P(\Sigma_{i=1}^k p_i + k + 1)$ and let $S_1 = \mathscr{T}(A_1)$ and $S_2 = \mathscr{T}(A_2)$. L_{A_1} is consistent with S_2 and L_{A_2} is consistent with S_1 which is contradictory. □

Using proof of Proposition 7 (for one-letter alphabet) and Proposition 6, we immediately obtain the following corollary:

Corollary 1. *The class of regular languages over Σ is identifiable in the limit from polynomial time and data using RFSAs if and only if $Card(\Sigma) = 1$.*

We use the number of states of an NFA as a measure of its size, but results still hold if we use the number of transitions instead (since the number of transitions is at most equal to $Card(Q)^2 \times Card(\Sigma)$).

4. Experimental study of residual languages

In the previous section, we proved that regular languages are not identifiable in the limit from polynomial time and data using RFSAs. This negative result entails no efficient inference algorithm can output the canonical RFSA of the target language. Nevertheless, this does not mean that an inference algorithm must not look for an RFSA representation of the target language, not necessarily the smallest one. There are two main differences between DFAs and RFSAs. First, all residual languages of a language are represented as individual states of any DFA recognizing it while only some of them, namely the prime residual languages, need to be represented as states of an RFSA. Second, inclusion relations can be represented within RFSAs by using the saturation operator. DFAs and RFSAs of a given regular language are not very different when only few inclusion relations hold between its residual languages. On the other hand, RFSAs could be a much more appropriate representation scheme than DFAs for target languages whose residual languages satisfy a lot of inclusion relations.

In this section, we experimentally study the inclusion properties of the residual languages of randomly drawn regular languages. We generate regular languages according to three different random processes:

- from randomly drawn DFAs,
- from randomly drawn NFAs,
- from randomly drawn regular expressions.

All the methods, we use are inherently biased as there are no natural probability distribution on the set of regular languages. All of them depend on some parameters and, when it is possible, we study the impact of the variations of these parameters on the generated languages.

4.1. Languages generated using random DFAs

We tested two methods of DFAs generation. The first one, developed in [14], allows to uniformly draw a minimal DFA among all minimal DFAs of a given size, for languages defined on a two-letter alphabet. The second method is the "naive" one: first, the size of the alphabet and the number of states are chosen; next, one state is randomly chosen so as to be initial, each state has a probability t% of being terminal and the successor of each state is randomly drawn for each letter; only minimal DFAs are kept. [3]

[3] We have led other experiments by uniformly drawing DFAs of a given size (instead of only keeping *minimal* DFAs): they have provided very similar results.

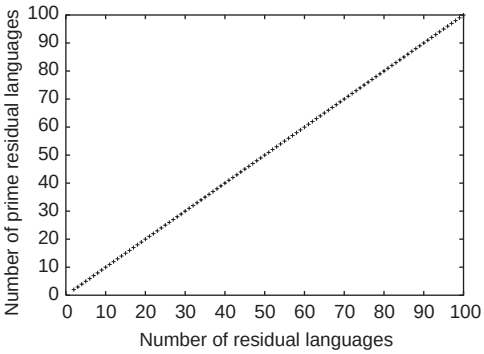

Fig. 4. Minimal DFAs and canonical RFSAs have almost surely the same size.

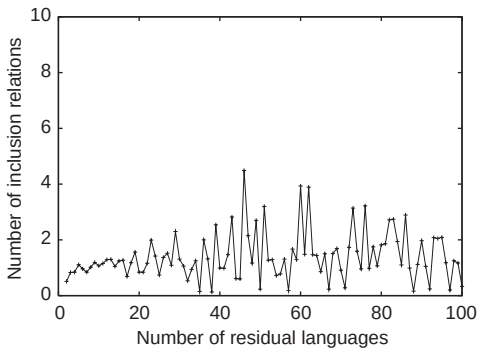

Fig. 5. Nearly no inclusion relations between residual languages.

We present here an experimental analysis relying on the first generation method. Similar results have been obtained with the second method for different sizes of the alphabet.

Fig. 4 illustrates the average size of canonical RFSAs of languages having a minimal DFA of a given size: 100 languages have been generated for each size of minimal DFAs. Note that almost all residual languages are prime and that consequently, the size of the canonical RFSA of observed languages is almost the same as the size of their minimal DFA.

Fig. 5 illustrates the average number of inclusion relations between their residual languages. There are nearly no inclusion relations between residual languages and this property explains why almost all residual languages are prime.

This suggests that looking for a RFSA representation of the target language will provide no improvement for languages generated this way.

4.2. Languages generated using random NFAs

In order to generate random NFAs, we choose a number of states n, the size $|\Sigma|$ of the alphabet, the number of transitions per state n_δ, and the probabilities p_I and p_T for each state of being initial or terminal. Each state has exactly n_δ successors in the NFA, the letter and the destination state of each transition are drawn randomly. Note that after trimming the automaton, states can have fewer than n_δ transitions. We have also made experiments with other methods. For example, we can choose only one state so as to be initial. Experiments presented in the appendix show that results are quite robust to variations in the generating process.

We present results obtained with $n = 10$, $|\Sigma| = 2$, $n_\delta = 2$ and $p_I = p_T = 0.5$. 100 000 NFAs have been generated this way.

As for the previous generation method, we first compare the sizes of minimal DFAs and canonical RFSAs of the randomly drawn regular languages. Fig. 6 shows the average number of prime residual languages relatively to the total number of residual languages, i.e. the average size of the canonical RFSA relatively to the size of the equivalent minimal DFA. Vertical bars indicate the standard deviation. We also show in this figure both theoretical bounds on the size of the canonical RFSA. Languages with large minimal DFAs must have a significant number of composite residual languages.

Fig. 7 illustrates the repartition of the sizes of the generated languages represented either using minimal DFAs or canonical RFSAs. For example the point $(10, 18)$ on the RFSAs curve indicates that 18% of the generated languages have a canonical RFSA of 10 states. We first observe that the average size of canonical RFSAs (10.0) is lower than the average size of minimal DFAs (26.2). Furthermore, the standard deviation is much greater for DFAs (17.1) than for RFSAs (2.9). Consequently, a significant number of regular languages have minimal DFAs larger than their canonical RFSAs.

The number of transitions can also be used as a measure of the size of an automaton. Fig. 8 shows the average number of transitions in canonical RFSAs (with respect to the number of states of the minimal DFA) and Fig. 9 shows the average number of transitions in *simplified* canonical RFSAs. The curve $2 \times x$ indicates the maximum

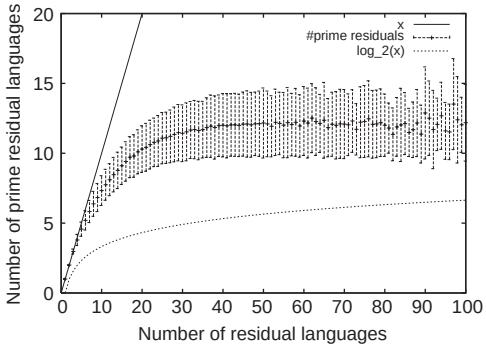

Fig. 6. Minimal DFAs can be significantly larger than canonical RFSAs.

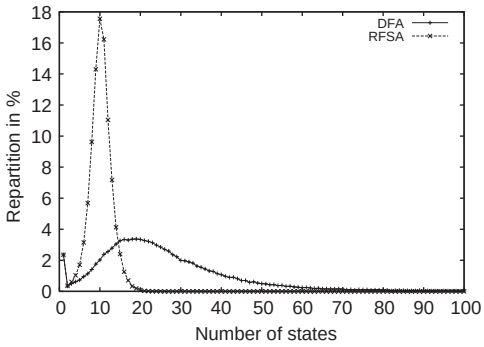

Fig. 7. Size of canonical RFSAs and minimal DFAs.

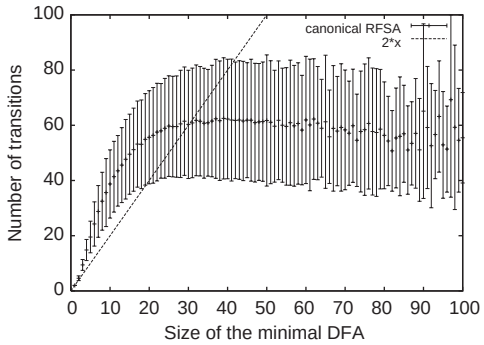

Fig. 8. Number of transitions of the canonical RFSA.

number of transitions in the minimal DFA: the total number of transitions is also significantly smaller for canonical RFSAs than for minimal DFAs.

Fig. 10 shows the average number of inclusion relations between residual languages relatively to the number of residual languages of generated languages. The number of inclusions is quite high and this is a hint that the saturation operator might be used as a generalization operator in the learning process.

We have repeated these experiments with several values of parameters and we have always obtained similar results. See the appendix for additional results about these experiments.

To sum up, regular languages generated using randomly drawn NFAs have a lot of composite residual languages and many inclusion relations hold among their residual languages. So, the use of canonical RFSAs seems appropriate to represent this kind of languages: the canonical RFSA is significantly smaller than the minimal DFA, and both reduction operator and saturation operator can be applied efficiently on minimal DFAs of these languages.

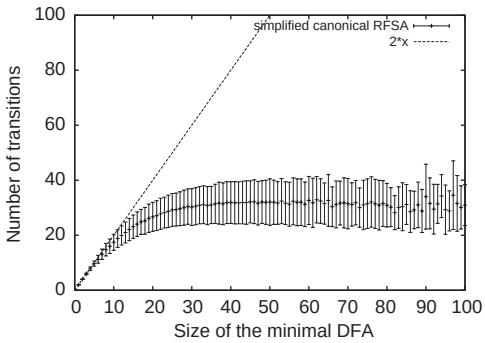

Fig. 9. Number of transitions of the simplified canonical RFSA.

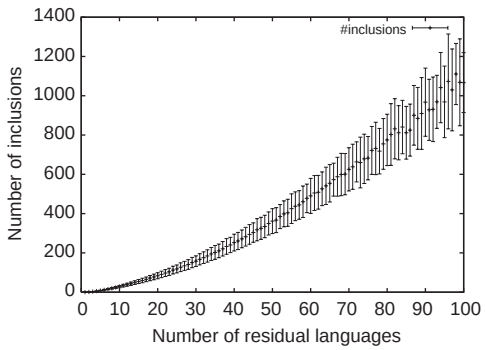

Fig. 10. Number of inclusion relations between residual languages.

4.3. Languages generated using random regular expressions

To generate random regular expressions, we consider a set $Op = \{\emptyset, 0, 1, ^*, \cdot, +\}$ of operators, we choose an upper bound n_{op} for the number of operators that will be used and we define a probability distribution p on Op. The root operator is chosen among Op using p: if the root operator is 0-ary, the construction stops; if it is 1-ary, the procedure is called recursively with parameter $n_{op} - 1$ to build its argument and when it is binary, it is called with $\lceil n_{op}/2 \rceil$ on one branch and $\lceil (n_{op} - 1)/2 \rceil$ on the other.

We first present results obtained with $n_{op} = 100$, $p_\varepsilon = 2\%$, $p_0 = p_1 = 5\%$, $p_\star = 13\%$, $p_\cdot = 50\%$, $p_+ = 25\%$: 100 000 languages have been generated.

Fig. 11 shows the repartition of the sizes of minimal DFAs and canonical RFSAs and Fig. 12 shows the average size of canonical RFSAs of the generated languages with respect to the size of their minimal DFAs. The last curve is similar to the one of Fig. 6: the canonical RFSA is significantly smaller than the minimal DFA in general. As previously, we indicated the theoretical bounds on the size of the canonical RFSA.

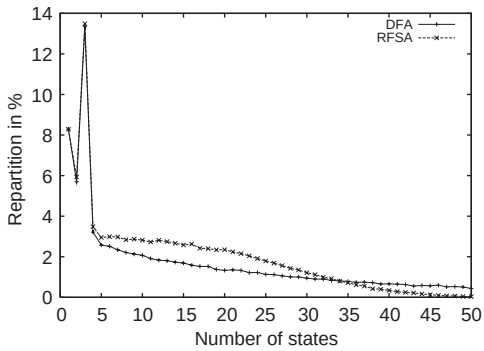

Fig. 11. Size of canonical RFSAs and minimal DFAs.

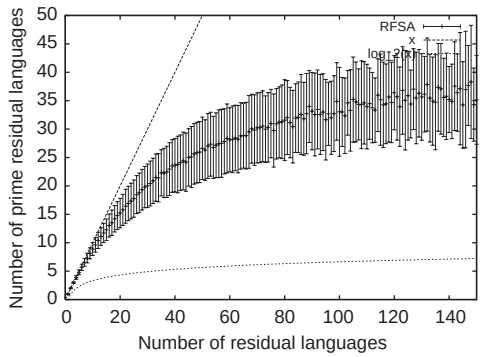

Fig. 12. Size of canonical RFSAs w.r.t. size of minimal DFAs.

Again, the number of composite residual languages can be important with respect to the total number of residual languages.

Figs. 13 and 14 indicate the number of transitions of the canonical RFSA and that of the simplified canonical RFSA. Again, the number of transitions of the simplified canonical RFSA is clearly smaller than the number of transitions of the equivalent minimal DFA.

The number of inclusion relations between residual languages of languages generated using regular expressions is shown in Fig. 15, which looks like Fig. 10. So, similar conclusions can be drawn.

4.4. Conclusion

These results tend to highlight two kinds of regular languages. The first one is composed of languages that have few composite residual languages and few inclusion relations between the residual languages. Previous experiments show that generating languages by randomly drawing DFAs provides this kind of languages. RFSAs do not

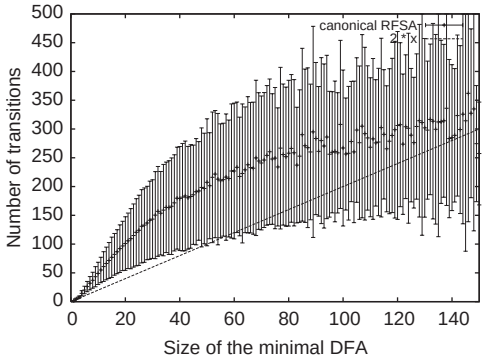

Fig. 13. Number of transitions of the canonical RFSA.

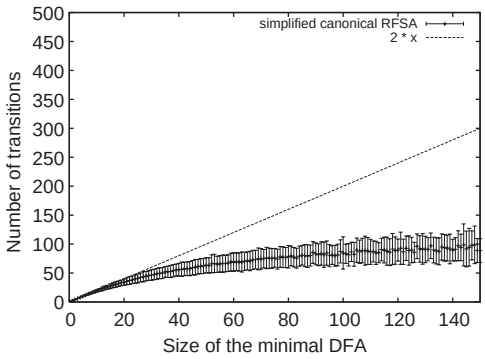

Fig. 14. Number of transitions of the simplified canonical RFSA.

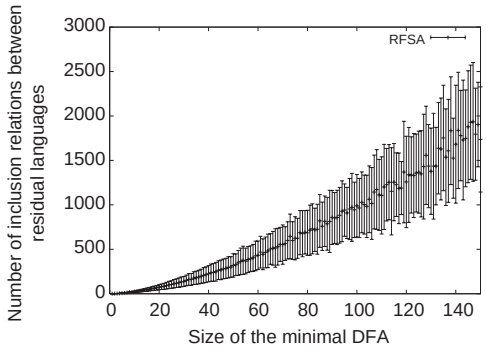

Fig. 15. Number of inclusion relations between residual languages.

seem to be an appropriate representation for this kind of languages. The second one is composed of languages for which a lot of inclusion relations hold between their residual languages. Previous experiments show that generating languages by drawing nondeterministic representations can lead to this second kind of languages. In this case, the canonical RFSA is significantly smaller than the minimal DFA, and both reduction and saturation operators can be used efficiently to either reduce the number of states of automata, or add transitions.

This suggests an approach for grammatical inference: look for inclusion relations between residual languages and use a saturation-based algorithm. This is the approach that we develop in Section 5.

These results also raise the problem of a natural representation of regular languages: none of the previously described procedures is more natural or more artificial than the others. As a consequence, artificial learning benchmarks should not only be based on procedures that choose DFAs.

We also raise the naive question: are regular languages occurring in practical cases of the first kind (size(canonical RFSA) $\simeq$ size(minimal DFA)), or of the second kind (size(canonical RFSA) $\ll$ size(minimal DFA)) or something in-between?

5. Learning using residual languages

Experiments lead in previous section tend to point out that regular languages can be such that a lot of inclusion relations between residual languages hold. We present in this section a new learning algorithm which tries to detect such inclusion relations and we study its theoretical properties. These results extend the study made in [4].

Classical learning algorithms such as RPNI build a prefix tree acceptor from the positive examples, and evaluate whether languages associated with their states are equivalent; if so, these states are merged. Previous experiments showed that it could be interesting to go one step further and look for inclusions of languages instead of equivalences. We present here a learning algorithm (DeLeTe2) based on this approach. We first introduce its target automaton: an RFSA whose size is intermediary between the size of the canonical RFSA and the one of the minimal DFA.

5.1. Saturated subautomata of the minimal DFA

Let $A = \langle \Sigma, Q, q_0, F, \delta \rangle$ be a minimal DFA. For every state q of A, let u_q be the smallest word of Σ^* such that $\delta(q_0, u_q) = q$ (so, $u_{q_0} = \varepsilon$). We assume that $Q = \{q_0, \ldots, q_n\}$ is ordered according to u_q, i.e. $q_i < q_j$ iff $u_{q_i} < u_{q_j}$. Let $A^s = \langle \Sigma, Q, Q_0^s, F, \delta^s \rangle$.

For every word u, the automaton $A_u = \langle \Sigma, Q^u, Q_0^u, F^u, \delta^u \rangle$ is obtained from A^s by deleting the states q such that $u_q > u$: $Q^u = \{q \in Q \mid u_q \leqslant u\}$, $Q_0^u = Q_0^s \cap Q^u$, $F^u = F \cap Q^u$, $\delta^u(q,x) = \delta^s(q,x) \cap Q^u$.

There is a finite number of subautomata A_u. When u is bigger than u_{q_n}, the subautomaton A_u is A^s itself. What is the smallest u such that $L_A = L_{A_u}$?

Proposition 8. *Let p be the greatest prime state of A. The word u_p is the smallest word such that A_{u_p} and A are equivalent.*

Proof. All states greater than p in A are composite. The automaton A_{u_p} is obtained by saturation of A and reduction of the states greater than p. Both operations preserve the language recognized by the automaton A [5].

On the other hand, if u is smaller than u_p, the subautomaton A_u does not contain p. Since p is a prime state, there exists a word w which does not belong to any residual language included in $u_p^{-1}L_A$. The word u_pw is not recognized by the automaton A_u. □

Note that the automaton A_{u_p} only depends on the language L_A.

Example 2. The greatest prime state of the automaton A described in Fig. 1 is q_2 and A_{u_2} is the canonical RFSA described in Fig. 2.

It is possible to build examples where the automaton A_{u_p} is exponentially smaller than A.

5.2. A characteristic sample

Let L be a regular language and let A be its minimal DFA. A sample S is *complete for inclusion relations* (*complete* for short) if it provides complete information about inclusion relations between the residual languages whose smallest characterizing word is not greater than u_p, where p is the greatest prime state of A. More formally, let $SP(L)=\{u_q \mid q \leqslant p\}$ and $K(L)=\{u_qx \mid q \leqslant p,\ x\in\Sigma,\ \delta(q,x)\neq\emptyset\}$ (SP stands for Short Prefix and K for Kernel).

Definition 8. Let L be a regular language. A sample S is *complete for inclusion relations* if
- $\forall u \in SP(L)\cup K(L),\ u \in Pref(S_+)$
- $SP(L)\cap L \subseteq S_+$,
- $\forall u \in SP(L),\ \forall v \in SP(L)\cup K(L),\ u^{-1}L \not\subseteq v^{-1}L \Rightarrow \exists w$ such that $uw \in S_+$ and $vw \in S_-$.

Note that for every regular language L, there exists a complete sample whose size is $\mathcal{O}(n^5)$ where n is the size of the minimal DFA of L: this sample contains at most $\mathcal{O}(n^2)$ words whose length is at most $\mathcal{O}(n^3)$ (the third point implies that S can contain words uw where $|u|\leqslant n$ and w is a word of $u^{-1}L\cap\overline{v^{-1}L}$ for instance, so $|w|\leqslant n^2$).

Let S be a sample, let $u,\ v \in Pref(S_+)$. We define:
- $u \prec v$ if no word w exists such that $uw \in S_+$ and $vw \in S_-$,
- $u \simeq v$ if $u \prec v$ and $v \prec u$.

It is clear that for any sample S and any words $u,v \in Pref(S_+)$, we have $u^{-1}L = v^{-1}L \Rightarrow u \simeq v$ and $u^{-1}L \subseteq v^{-1}L \Rightarrow u \prec v$.

Lemma 1. *Suppose that the sample S is complete for inclusion relations and let $u \in SP(L),\ v \in SP(L)\cup K(L)$. We have $u \prec v \Rightarrow u^{-1}L \subseteq v^{-1}L$.*

Proof. Straightforward since $\forall u \in SP(L)$, $\forall v \in SP(L) \cup K(L)$, $u^{-1}L \not\subseteq v^{-1}L \Rightarrow \exists w$ such that $uw \in S_+$ and $vw \in S_-$. This implies that $u \not\prec v$. □

Example 3. Let L be the language defined in Example 1. We have $SP(L) = \{\varepsilon, 0, 1\}$, $K(L) = \{0, 1, 00, 01, 10, 11\}$. The smallest set which is complete for inclusion relations, is $S = S_+ \cup S_-$ where $S_+ = \{\varepsilon, 00, 11, 010, 10\}$ and $S_- = \{0, 1, 01, 001\}$.

We have the following relations between elements of $SP(L)$ and elements of $SP(L) \cup K(L)$ (if u is the label of a row and v the label of a column, a word w in the array means that $uw \in S_+$ and $vw \in S_-$).

	ε	0	1	00	01	10	11
ε	$\prec$	ε	ε	$\prec$	ε	$\prec$	$\prec$
0	0	$\prec$	$\prec$	$\prec$	$\prec$	$\prec$	$\prec$
1	1	1	$\prec$	1	$\prec$	$\prec$	$\prec$

The conclusion of Lemma 1 can be checked from this array.

5.3. The DeLeTe2 algorithm

We now present an algorithm that builds an NFA from a sample of a target language L. If the sample is complete for inclusion relations of L, the algorithm builds the automaton A_{u_p} of the language L. Starting with an empty automaton, the algorithm considers prefixes of positive examples as characterization of states. A new state is added to the current set of states when it is supposed to be nonequivalent with previous ones. New transitions are deduced from inclusion relations that can exist between this new state and previous ones, which roughly corresponds to a local saturation of the current automaton. The algorithm stops when either the current automaton is consistent with the input sample or when it has explored all the prefixes of positive examples.

Input: a sample S of a language L.
Let $Pref = \{u_0, \ldots, u_n\}$ be the set of prefixes of S_+
ordered using the length-lexicographic order.
Let $Q = Q_0 = F = \delta = \emptyset$.
Let $u = \varepsilon$.
Loop
  If $\exists u' \in Q$ such that $u \simeq u'$
  Then
    % u is equivalent to a state in Q
    delete $u\Sigma^*$ from $Pref$
  Else
    % add u to the sets of states and add the corresponding transitions
    $Q = Q \cup \{u\}$
    $Q_0 = Q_0 \cup \{u\}$ if $u \prec \varepsilon$

$F = F \cup \{u\}$ if $u \in S_+$
$\delta = \delta \cup \{(u', x, u) \mid u' \in Q,\ u'x \in Pref,\ u \prec u'x\}$
$\cup \{(u, x, u') \mid u' \in Q,\ ux \in Pref,\ u' \prec ux\}$
End If
If u is the last word of *Pref* **Or If** $A = \langle \Sigma, Q, Q_0, F, \delta \rangle$ is consistent with S
Then
Exit Loop
Else
Let $u =$ next word in *Pref*
End If
End Loop

Output: the automaton $A = \langle \Sigma, Q, Q_0, F, \delta \rangle$.

Example 4. On the previous example, the algorithm needs three steps to recover the target automaton.

(1) In the first step, the state ε is added. As $\varepsilon \prec \varepsilon$, the state ε is initial. The word ε belongs to S_+, the state ε is final. There is no relation $\varepsilon \prec x$ or $x \prec \varepsilon$ for $x \in \{0, 1\}$ and $x \in Pref$; thus no transition has to be added.
(2) In the second step, the state 0 is added because $0 \not\simeq \varepsilon$. As $0 \not\prec \varepsilon$, the state 0 is not initial. As 0 does not belong to S_+, the state 0 is not final. We have the relations $0 \prec \varepsilon 0$, $0 \prec \varepsilon 1$, $0 \prec 00$, $0 \prec 01$, $\varepsilon \prec 00$. The corresponding transitions are added.
(3) In the third step, state 1 is added because $1 \not\simeq \varepsilon$ and $1 \not\simeq 0$. As $1 \not\prec \varepsilon$, the state 1 is not initial. The word 1 does not belong to S_+, then the state 1 is not final. We have the relations $1 \prec \varepsilon 1$, $1 \prec 01$, $1 \prec 10$, $1 \prec 11$, $\varepsilon \prec 10$, $0 \prec 10$, $\varepsilon \prec 11$, $0 \prec 11$. The corresponding transitions are added.

This automaton is consistent with S, the algorithm halts (Fig. 16).

Theorem 1. *If the input sample of the DeLeTe2 algorithm is complete for inclusion relations for a language L, it outputs the saturated subautomaton A_{u_p} of L.*

Proof. We can prove that, at each beginning of the loop, we have $u \leqslant u_p$. At the beginning of the *else* part, u belongs to $SP(L)$. Then the set Q is included in $SP(L)$. Due to the definition of a complete sample and to Lemma 1, u is an initial state if $u \prec \varepsilon$, i.e. $u^{-1}L \subseteq L$; u is a final state if $u \in S_+$, i.e. $\varepsilon \in u^{-1}L$. The added transitions are

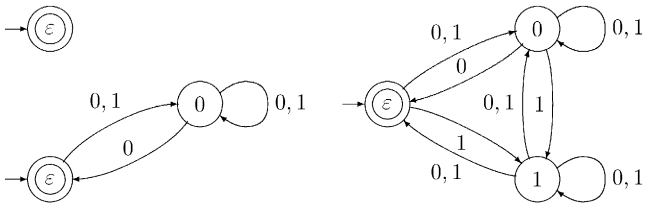

Fig. 16. The three steps.

all transitions such that $u \prec u'x$, i.e. $(u'x)^{-1}L \supseteq u^{-1}L$ or $u' \prec ux$, i.e. $(ux)^{-1}L \supseteq u'^{-1}L$. Thus, at the end of the loop, the automaton A is the subautomaton A_u of the minimal DFA recognizing L. When the automaton A is consistent with the sample, we have the subautomaton A_{u_p}. □

Note that, if the sample is not complete for the inclusion relation, this algorithm does not necessarily give an automaton which is consistent with that sample. For a theoretical purpose, it can easily be transformed in order to have a consistency algorithm, i.e. an algorithm which always outputs an automaton consistent with the input sample. For example, we can check whether the output automaton is consistent or not. If it is not consistent, then output the prefix tree acceptor of the sample. A different solution is chosen in the implemented algorithm (see Section 6).

Given a minimal DFA, deciding whether a given state is prime is a PSPACE-complete problem [5]. Therefore, computing the smallest sample which is complete for the language recognized by a given DFA is not feasible. However, it is always possible to compute within polynomial time from a given DFA *some* complete sample: use $SP'(L) = \{u_q \mid q \in Q\}$ and the corresponding $K'(L)$. The resulting sample would be:

$$\begin{aligned} S_+ = {} & \{uw \mid u \in SP'(L) \cup K'(L) \text{ and } w \text{ is the smallest word of } u^{-1}L\} \\ & \cup \{uw \mid u \in SP'(L), v \in SP'(L) \cup K'(L) \text{ and } w \text{ is the smallest} \\ & \text{word of } u^{-1}L \cap \overline{v^{-1}L}\}, \\ S_- = {} & \{vw \mid u \in SP'(L), v \in SP'(L) \cup K'(L) \text{ and } w \text{ is the smallest word} \\ & \text{of } u^{-1}L \cap \overline{v^{-1}L}\}. \end{aligned}$$

To sum up, although regular languages cannot be identified in the limit from polynomial time and data when RFSA is chosen as a representation scheme (see Section 3), it is still possible to design algorithms that take advantage of this representation. DeLeTe2 builds an NFA consistent with its input sample in polynomial time. For each regular language L, it is possible to build a *characteristic* sample S_L (for example the above complete sample) whose size is polynomial with respect to the size of any DFA recognizing L. The building process of the characteristic sample can be done in polynomial time with respect to the size of any DFA recognizing L. If a sample containing S_L is given, DeLeTe2 outputs an RFSA recognizing L.

As a consequence, except for the fact that it outputs an NFA, DeLeTe2 identifies regular languages in the limit from polynomial time and data (relatively to the DFA representation scheme). As the goal of grammatical inference is mainly to produce a function able to label new examples, NFA is an acceptable representation. Indeed, deciding whether a word is recognized by an NFA can be done in polynomial time.

6. Some experimental results

In this section, we compare DeLeTe2 to other grammatical inference algorithms: RPNI [15], and Red–Blue (RB) (by H. Juillé and J.B. Pollack, implemented by

K. Lang) which is a variant of RPNI that uses evidence driven state merging (EDSM, see [13]).

6.1. Implementation of DeLeTe2

Here, we do not suppose that the input sample is complete for the inclusion relations, and so, to perform our experiments, we use a variant of the algorithm presented above which is more efficient and which always computes a consistent automaton. The modifications do not alter theoretical results: the algorithm remains a learning algorithm in the conditions mentioned above.

The inclusion relation between residual languages is transitive and right-invariant for concatenation.[4] As the $\prec$ relation is an estimation of this relation, it should also be transitive and right-invariant for concatenation. So, whenever the algorithm estimates that $q \prec q'$, it also adds all the $\prec$ relations needed to have a relation both transitive and right-invariant for concatenation. Then it adds all the corresponding transitions and checks whether the modified automaton is consistent with the input sample. If not, the initial $\prec$ relation is declared invalid and is undone. In the other case, the initial $\prec$ relation and all the deduced $\prec$ relations are considered valid and the modification of the current automaton is accepted.

Note that the modified algorithm is now a consistency algorithm, i.e. it always outputs an automaton which is consistent with the input sample.

6.2. Experimental protocol

We built several benchmarks using the generating methods described in Section 4: the procedure to generate *NFAs* has been used with $n = 20$, $n_\delta = 2$, $|\Sigma| = 2$, $p_I = p_T = 10\%$. The procedure to generate regular expressions has been used with $nb_{op} = 50$, $|\Sigma| = 2$, $p_\varepsilon = 0.025$, $p_0 = p_1 = 0.05$, $p_* = 0.125$, $p_\cdot = 0.5$ and $p_+ = 0.25$. Studies have also been made using procedures generating DFAs: in this last case DeLeTe2 has worse results than RPNI and Red–Blue, which can be understood easily considering that, for this generation method, there are nearly no inclusion relations between distinct residual languages, so the approach we propose here is less effective than algorithms using equivalence relations. A target language being drawn, examples are drawn the following way: we choose l randomly in $[0, 15]$, and we create a word w of length l, each letter of w being chosen by flipping a coin.

One experiment consists of: choosing a generation method (NFAs or regular expressions), generating a language, choosing a number of training examples (50, 100, 150 or 200), generating a training set, generating a test set containing 1000 words and training each algorithm on the training set. In order to have results significantly higher than majoritary vote, only experiments the generated language of which have

[4] I.e. for any residual languages $u^{-1}L$ and $v^{-1}L$, if $u^{-1}L \subseteq v^{-1}L$, then for each word w, $(uw)^{-1}L \subseteq (vw)^{-1}L$.

more than 20% of negative examples and more than 20% of positive examples in the learning sample have been kept.

We compare the following algorithms: Majoritary Vote (*MAJ*), DeLeTe2 (*DLT2*), *RPNI*, and Red–Blue (*RB*). We compare them using two methods: we first observe the average recognition rate of the output automaton of each algorithm on the test set, we then do matches (denoted *algo1–algo2* in the table) where we count the number of experiments where one algorithm is better than another (in terms of recognition rate), and we count a tie when the difference is not significant (using the Mc Nemar test, see [8]). Results of those matches are denoted: won_by_algo1+won_by_algo2+nb_ties.

For each generation method, each size of training set and each algorithm, 30 experiments are performed. We also perform basic statistical tests: n_r is the average number of distinct residual languages of the generated languages, n_p is their average number of prime residual languages, n_i is the average number of inclusion relations between distinct residual languages of L and $|A_{u_p}|$ is the average number of states of the target automaton of DeLeTe2.

6.3. Results

Against RPNI, DeLeTe2 won 130 matches, it lost 24 matches and there were 86 ties; against Red–Blue, it won 127 matches, it lost 33 matches and there were 80 ties. So, we can say that DeLeTe2, while very basic, is better than the other two algorithms on benchmarks generated using NFAs and regular expressions. Details on the experiments described in this paper can be found at http://www.grappa.univ-lille3.fr/~lemay/tcs04/ (Table 1).

Table 1

Benchmark	nfa_50	nfa_100	nfa_150	nfa_200	expreg_50	expreg_100	expreg_150	expreg_200
MAJ	68.6%	68.7%	65.0%	67.9%	65.0%	66.7%	62.4%	62.4%
RB	66.5%	68.5%	70.7%	70.8%	77.5%	82.0%	88.1%	90.9%
RPNI	66.5%	68.7%	72.2%	71.0%	81.2%	82.5%	85.2%	90.6%
DLT2	69.3%	74.4%	76.7%	78.9%	81.3%	91.4%	92.0%	95.7%
DLT-RPNI	14+6+10	19+3+8	18+3+9	23+1+6	17+2+11	19+1+10	11+4+15	9+4+17
DLT-RB	16+8+6	17+4+9	19+4+7	21+2+7	9+8+13	19+3+8	15+3+12	11+1+18
n_r	126,4	123,3	131,6	120,3	6,8	7,0	9,7	9,2
n_p	22,1	22,6	21,4	24,7	5,8	5,8	6,7	6,1
n_i	2172,7	2124,4	2093,3	1834,5	16,4	16,2	38,3	39,0
$\|A_{u_p}\|$	99.5	94.8	91.7	110.0	6.6	6.7	8.6	8.3

Note: Columns represent the generation method and the size of the training sample; rows represent the different algorithms, the matches and the statistical tests; cells represent the recognition rate (for the different algorithms), the results of the matches and the values of the statistical tests

7. Conclusion

The most classical strategy used in grammatical inference of regular languages consists in identifying the words which define identical residual languages and then merging the corresponding states in the current automaton. This strategy naturally leads to build a DFA in order to identify the target language.

We have proposed here an alternative strategy: look for the inclusion relations between residual languages and then saturate the current automaton. This new strategy naturally leads to the RFSA representation of regular languages. Both theoretical and experimental results given in this paper show that this new approach is interesting and promising.

This paper also raises the problem of representation of languages: properties of randomly generated regular languages highly depend on the representation used to generate them. Two kinds of languages are highlighted here: for the first kind of languages, most residual languages are prime and there are few inclusion relations between them, for the second kind of languages, most residual languages are composite and there are many inclusion relations between them. Both those kinds of languages should be studied in benchmarks. It is likely that in some practical cases, target languages are constituted of languages close to the first kind whereas in some other cases, they are composed of languages close to the second kind. This could determine the kind of learning algorithm to use.

Appendix. Variation of parameters for NFAs generation

Figs. 17 and 18 illustrate the repartition of the size of minimal DFAs and canonical RFSAs for $n_\delta = 2$, 3 and 4. Results are similar to results presented in Section 4.2, until the number of transitions is too large and the generation method becomes degenerated and nearly always outputs Σ^*.

Figs. 19 and 20 illustrate the repartition of sizes of DFAs and RFSAs depending on the variation on $|\Sigma|$ (for $|\Sigma| = 2$, 4, and 6). Previous results still hold until the alphabet

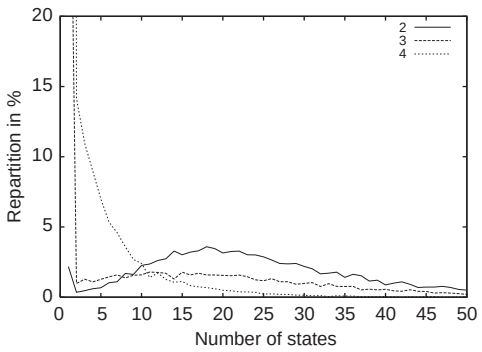

Fig. 17. Variation of the size of the minimal DFA depending on n_δ.

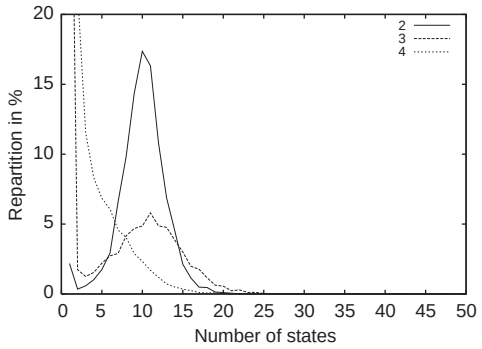

Fig. 18. Variation of the size of the canonical RFSA depending on n_δ.

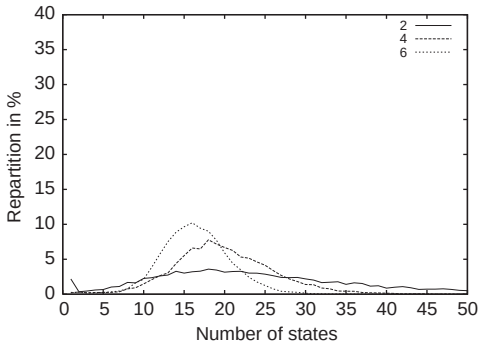

Fig. 19. Variation of the size of the minimal DFA depending on $|\Sigma|$.

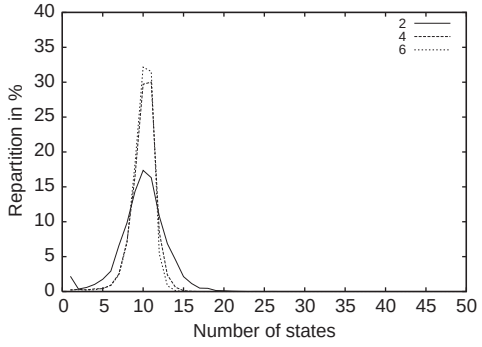

Fig. 20. Variation of the size of the canonical RFSA depending on $|\Sigma|$.

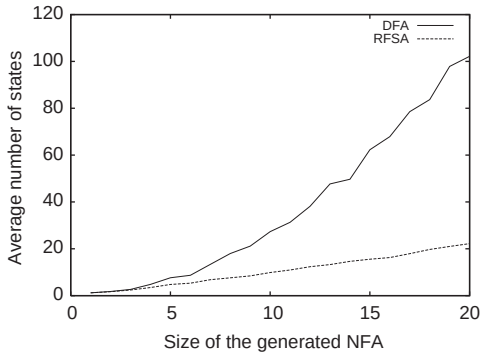

Fig. 21. Average size of minimal DFAs and canonical RFSAs.

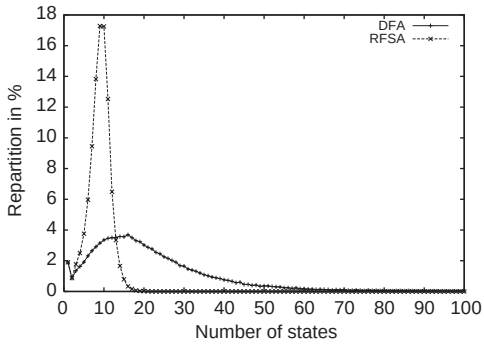

Fig. 22. Size of canonical RFSAs and minimal DFAs.

is too large with respect to n_δ: in extreme cases, the transition function becomes deterministic, and results are to be compared with results on languages generated using DFAs. This tends to indicate that there is some kind of continuity between results obtained using random DFAs and random NFAs.

Fig. 21 indicates the average size of minimal DFAs and canonical RFSAs for languages generated with NFAs whose size varies from 1 to 20. 100 languages have been generated for each size. Again, the difference of size between both representations significantly increases when the size of the generated languages grows.

Figs. 22 and 23 illustrate results obtained with a slightly different generation method for NFAs. Instead of having a certain probability for each state of being initial, only one of them is chosen so as to be initial. The other components of the automaton are generated with the same method as in Section 4.2, and same parameters are chosen ($n = 10$, $|\Sigma| = 2$, $n_\delta = 2$ and $p_T = 0,5$). Fig. 22 indicates the repartition of the generated languages represented either by minimal DFAs or canonical RFSAs (to compare with

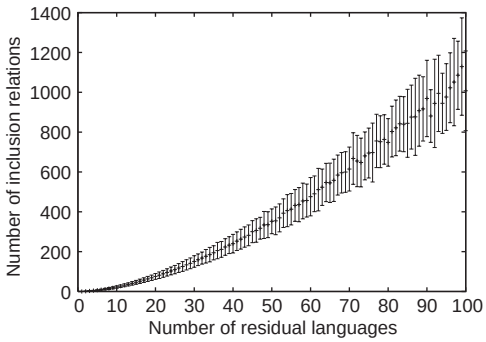

Fig. 23. Number of inclusion relations between residual languages.

Fig. 7). Fig. 23 shows the average number of residual languages with respect to the number of residual languages of generated languages (to compare with Fig. 10). Results are similar to those obtained with the method of Section 4.2.

References

[1] D. Angluin, Learning regular sets from queries and counterexamples, Inform. Comput. 75 (2) (1987) 87–106.
[2] J.A. Brzozowski, Derivatives of regular expressions, J. ACM 11 (1964) 481–494.
[3] F. Coste, D. Fredouille, Efficient ambiguity detection in c-nfa, in: Grammatical Inference: Algorithms and Applications, Lecture Notes in Artificial Intelligence, Vol. 1891, Springer, Berlin, 2000.
[4] F. Denis, A. Lemay, A. Terlutte, Learning regular languages using nondeterministic finite automata, in: ICGI'2000, Proc. 5th Internat. Colloquium on Grammatical Inference, Lecture Notes in Artificial Intelligence, Vol. 1891, Springer, Berlin, 2000, pp. 39–50.
[5] F. Denis, A. Lemay, A. Terlutte, Residual finite state automata, in: STACS 2001, Lecture Notes in Computer Science, Vol. 2010, Springer, Berlin, 2001, pp. 144–157.
[6] F. Denis, A. Lemay, A. Terlutte, Residual finite state automata, Fund. Inform. 51 (4) (2002) 339–368.
[7] F. Denis, A. Lemay, A. Terlutte, Some language classes identifiable in the limit from positive data, in: ICGI 2002, Lecture Notes in Artificial Intelligence, Vol. 2484, Springer, Berlin, 2002, pp. 63–76.
[8] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput. 10 (7) (1998) 1895–1923.
[9] E.M. Gold, Complexity of automaton identification from given data, Inform. and Control 37 (1978) 302–320.
[10] C. de la Higuera, Characteristic sets for polynomial grammatical inference, Mach. Learning 27 (1997) 125–137.
[11] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, MA, 1979.
[12] K.J. Lang, Random DFA's can be approximately learned from sparse uniform examples, in: Proc. 5th Annu. Workshop on Computer Learning Theory, ACM Press, New York, NY, 1992, pp. 45–52.
[13] K.J. Lang, B.A. Pearlmutter, R.A. Price, Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm, in: Proc. 4th Internat. Colloquium on Grammatical Inference—ICGI 98, Lecture Notes in Artificial Intelligence, Vol. 1433, Springer, Berlin, 1998, pp. 1–12.

[14] C. Nicaud, Etude du comportement des automates finis et des langages rationnels. Ph.D. Thesis, Université de Marne la Vallée, 2001.
[15] J. Oncina, P. Garcia, Inferring regular languages in polynomial update time, in: Pattern Recognition Image Anal., 1992, pp. 49–61.
[16] T. Yokomori, Learning nondeterministic finite automata from queries and counterexamples, in: D. Michie, K. Furukawa, Clarendon S. Muggleton (Eds.), Machine Intelligence 13: Machine Intelligence and Inductive Learning, 1994.
[17] S. Yu, Handbook of Formal Languages, Regular Languages, Vol. 1, Springer, Berlin, 1997, pp. 41–110 (Chapter 2).