Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT
Theoretical Computer Science 313 (2004) 295–312

Theoretical Computer Science

www.elsevier.com/locate/tcs

Inference of ω-languages from prefixes

C. de la Higuera*, J.C. Janodet

EURISE, Faculté des Sciences et Techniques, Université Jean Monnet, 23 rue Paul Michelon, Cedex 02, F-42023 Saint-Etienne, France

Abstract

Büchi automata are used to recognize languages of infinite strings. Such languages have been introduced to describe the behavior of real-time systems or infinite games. The question of inferring them from infinite examples has already been studied, but it may seem more reasonable to believe that the data from which we want to learn is a set of finite strings, namely the prefixes of accepted or rejected infinite strings. We describe the problems of identification in the limit and polynomial identification in the limit from given data associated to different interpretations of these prefixes: a positive prefix is universal (respectively existential) when all the infinite strings of which it is a prefix are in the language (respectively when at least one is); the same applies to the negative prefixes. We prove that the classes of regular ω-languages (those recognized by Büchi automata) and of deterministic ω-languages (those recognized by deterministic Büchi automata) are not identifiable in the limit, whatever interpretation for the prefixes is taken. We give a polynomial algorithm that identifies the class of safe languages from positive existential prefixes and negative universal prefixes. We show that this class is maximal for polynomial identification in the limit from given data, in the sense that no superclass can even be identified in the limit.

MSC: 68T05

Keywords: Infinite string languages; Grammatical inference; Identification in the limit; Polynomial identification from given data

1. Introduction

Grammatical inference [5,8,14] deals with the general problem of automatic learning machines (grammars or automata) from structured data, and more usually strings.

* Corresponding author.
E-mail addresses: cdlh@univ-st-etienne.fr (C. de la Higuera), janodet@univ-st-etienne.fr (J.C. Janodet).

0304-3975/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2003.11.009

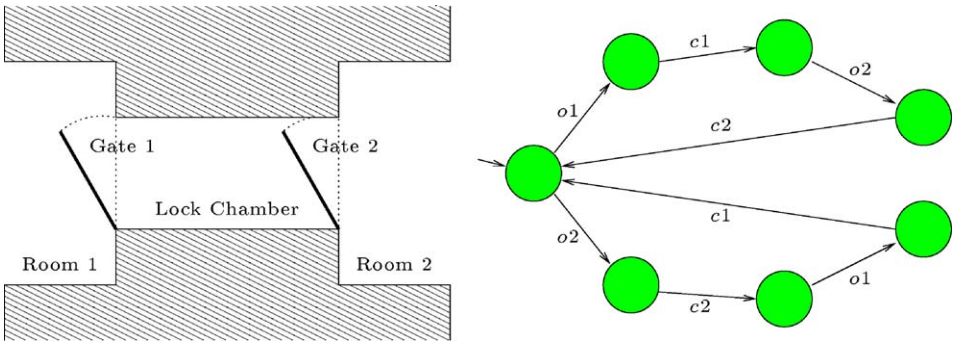


Fig. 1. A two-gate lock chamber ($o =$ open, $c =$ closed).

Between the different syntactic objects from formal language theory, most attention has been paid to the case of deterministic finite automata (dfa), although if some results on different types of grammars are known. The question of learning automata on infinite strings has hardly been studied.

The study of these infinite string automata was motivated by decision problems in mathematical logic. They provide a normal form for certain monadic second-order theories [4]. Later work concerned the relationship between these automata, the semantics of modal and temporal logics, and their application to critical reactive systems [18]. By reactive is implied a software whose purpose is to interact with its environment, and by critical one where mistakes or anomalies can have serious consequences, that can cost much more than the actual benefit made by the use of the software. This is the case for instance of automatic pilots, operating systems or nuclear station automatic supervisors.

The development of such software requires program proving capacities. In particular a property known as *safety*, which expresses that something bad will never occur during the execution of the system, must be verified. Current examples of safety properties are mutual exclusion or deadlock avoidance [10]. These properties are described formally in temporal logics like *Propositional Temporal Logic* (*PTL*), whose models, Kripke structures, can be described by Büchi automata. Consequently, Büchi automata make it possible to model with the same formalism the critical systems and the logical properties that they must satisfy. Effective proof algorithms (model checking) can then be developed [18].

Nevertheless, the formal specifications of the critical software, and even more, their properties, are difficult to write for a non-specialist of automata and temporal logics. Let us take the example of a lock chamber with two gates giving access to a safe deposit, represented by the automaton depicted in Fig. 1. One enters the lock chamber by gate 1 and one leaves it by gate 2 (or vice versa), but gate 2 should be allowed to open only if gate 1 is closed (and vice versa).

The safety property "gates 1 and 2 are never open at the same time" is written, in *PTL*: $\Box(\neg p_1 \vee \neg p_2)$, where property p_i is that gate i is open (the box $\Box$ should be

interpreted as "always"). If a non-specialist is not able to describe a system and its properties, he may be able on the other hand to give examples of good and bad behaviors of the system. These examples are sequences of events such as $o_1c_1o_2c_2o_2c_2o_1c_1\ldots$ and $o_2c_2o_1c_1\ldots$, which are good behaviors, or $o_1o_2\ldots$ and $o_1c_1o_1\ldots$, which are bad behaviors. The same applies to the logical properties the system must satisfy. Our objective is thus to learn automatically the Büchi automaton by collecting only positive and negative examples.

The problem of learning automata on infinite strings poses a first delicate problem: whatever the way of recovering the data (batch of examples, on line learning, use of an oracle or a teacher), is it reasonable to consider data which would be infinite strings? Let us recall that even with an alphabet of size 2 the set of infinite strings is uncountable. In previous research on learning this type of automaton, the choice was to use data coming from the countable subset of the ultimately periodic strings (of type uv^ω, u and v being finite strings). Saoudi and Yokomori [15] define a (restricted) class of local languages, and prove the learnability of these languages from positive examples; Maler and Pnueli [12] adapt Angluin's L* algorithm [2] and make it possible to learn a particular class of automata with the assistance of a polynomial number of equivalence and membership queries.

Nevertheless, we wish the learning of an automaton to be done from experimental data received from the potential users of a system. The data will therefore necessarily be finite strings. And the interpretation of these strings can vary. A finite string u can be a positive prefix, in the sense that one will be able to say that all its (infinite) continuations are good, or that at least one of its continuations is. The same kind of interpretations exists for the negative prefixes.

In this article we are thus interested in the inference of various types of machines on infinite strings, but from prefixes. In Section 2 we shall give the definitions concerning the ω-languages, and in Section 3 those necessary to the comprehension of the learning problems; this yields a family of identification problems. In Section 4 we establish several learnability results, by showing that for the majority of the defined problems, identification in the limit of the classes of ω-regular languages and ω-deterministic languages is not possible. But in the special case of *safe* languages, a positive result concerning polynomial identification is given. The algorithm presented in Section 4 is improved in Section 5.

2. Definitions

2.1. Finite strings, languages and automata

An alphabet Σ is a finite non-empty set of symbols called letters. Σ^* denotes the set of all finite strings over Σ. A language L over Σ is a subset of Σ^*. In the following, letters are indicated by $a,b,c,\ldots$, strings by $u,v,\ldots,z$, and the empty string by λ. Let $\mathbb{N}$ be the set of all non-negative integers.

A deterministic finite automaton (dfa) is a quintuple $A=\langle Q,\Sigma,\delta,F,q_0\rangle$, where Σ is an alphabet, Q is a finite set of states, $q_0\in Q$ is an initial state, $\delta: Q\times\Sigma\to Q$ is a

transition function, and $F \subseteq Q$ is a set of marked states, called (in this case) the final states.

As usual, δ can be recursively extended to Σ^* by defining $\delta(q,\lambda)=q$ and $\delta(q,a.w)=\delta(\delta(q,a),w)$ for all $q\in Q$, $a\in\Sigma$, $w\in\Sigma^*$. Let $\mathscr{L}(A)$ denote the language recognized by automaton A:

$$\mathscr{L}(A)=\{w\in\Sigma^* \mid \delta(q_0,w)\in F\}.$$

It is well known that the languages recognized by dfa form the family of regular languages. This class is considered as a borderline case for grammatical inference [5].

2.2. Infinite strings and ω-languages

We mainly use the notations from [16].

An infinite string u (or ω-string) over Σ is a mapping $\mathbb{N}\to\Sigma$. Such a string is written $a_0a_1\ldots a_n\ldots$, with $a_i\in\Sigma$. Let Σ^ω denote the set of all ω-strings over Σ. An ω-language over Σ is a set of infinite strings, thus a subset of Σ^ω.

Let L and K be two languages over Σ. We define

$$L^\omega=\{u\in\Sigma^\omega \mid u=u_0.u_1.\ldots.u_n.\ldots \text{ and } u_i\in L \text{ for all } i\in\mathbb{N}\},$$
$$K.L^\omega=\{u\in\Sigma^\omega \mid u=u_1.u_2 \text{ and } u_1\in K \text{ and } u_2\in L^\omega\}.$$

An ω-language L is ω-regular *iff* there exist $n\in\mathbb{N}$ and two finite sequences of regular languages $(A_i)_{i\in 1..n}$ and $(B_i)_{i\in 1..n}$ such that

$$L=\bigcup_{i=1}^{n} A_i.B_i^\omega$$

Let $\mathscr{P}\mathrm{ref}(u)$ be the set of all finite prefixes of an infinite string u. Finally, given an ω-language L, let

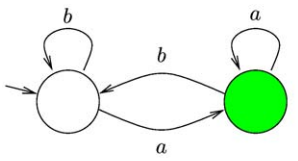
$$\mathscr{P}\mathrm{ref}(L)=\bigcup_{u\in L}\mathscr{P}\mathrm{ref}(u).$$

2.3. Automata on infinite strings

Büchi automata [4] recognize languages of infinite strings. These languages are actually used to model reactive systems [18] and infinite games [16].

A Büchi automaton is a quintuple $A=\langle Q,\Sigma,\delta,F,q_0\rangle$, where Σ is an alphabet, Q is a finite set of states, $q_0\in Q$ is an initial state, $\delta: Q\times\Sigma\to 2^Q$ is a transition function and $F\subseteq Q$ is a set of marked states. These states are distinguished from the others but not "final", as the strings are infinite.

A run of A on an ω-string u is a mapping $C_u:\mathbb{N}\to Q$ such that:
(1) $C_u(0)=q_0$ and
(2) $C_u(i+1)\in\delta(C_u(i),u_i)$, for all $i\in\mathbb{N}$.
Note that C_u is undefined if at some point, $C_u(i)$ is undefined.



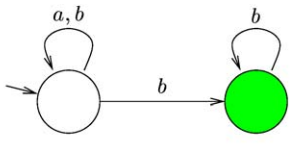


Fig. 2. Büchi automata 2a and 2b recognize $(b^*a)^\omega$ and $(a+b)^*b^\omega$, respectively. Their marked states are in gray.

An ω-string u is accepted by A *iff* there exists a state of F which appears infinitely often in a run of A on u. Let $\mathscr{L}(A)$ be the set of all accepted ω-strings by A. We can show [16] that an ω-language L is ω-regular *iff* $L = \mathscr{L}(A)$ for some Büchi automaton A.

An automaton is deterministic *iff* $|\delta(q,a)| \leqslant 1$ for all states q and letters a.

Let $\mathbb{R}\mathrm{eg}_\omega(\Sigma)$ be the class of all ω-regular languages and $\mathbb{D}\mathrm{et}_\omega(\Sigma)$ the class of all ω-languages which are recognized by a deterministic Büchi automaton. Unlike what happens in the case of finite automata, $\mathbb{D}\mathrm{et}_\omega(\Sigma) \subset \mathbb{R}\mathrm{eg}_\omega(\Sigma)$ but $\mathbb{D}\mathrm{et}_\omega(\Sigma) \neq \mathbb{R}\mathrm{eg}_\omega(\Sigma)$. Indeed, consider the language $(b^*a)^\omega$ of strings with an infinite number of a. This language is accepted by the deterministic automaton 2a but its complementary $(a+b)^*b^\omega$ is not deterministic, although it is recognized by the non-deterministic automaton 2b (Fig. 2).

2.4. ω-Safe languages and DB-machines

Since Lamport's seminal paper [10], it is usual to distinguish two classes of ω-languages, namely those that have a *safety* property and those that have a *liveness* property. In the domain of critical systems, a liveness property expresses the fact that "something good is necessarily going to happen in the system", e.g. its termination, the absence of starvation or the guaranty of some service. On the other hand, a safety property expresses that "something bad will never happen in the system"; examples of such properties are mutual exclusion or the absence of deadlock. It was proved, using topological arguments [3], that for every ω-language, there exists one safety property and one liveness property such that the language is the set of strings that satisfy either the safety property or the liveness one.

In substance, a language is safe when the limits of all its prefixes are themselves in the language.

Definition 1. An ω-language L is safe [1] iff

$$\forall w \in \Sigma^\omega, (\forall u \in \mathscr{P}\mathrm{ref}(\omega), \exists v \in \Sigma^\omega \text{ such that } u.v \in L) \Rightarrow w \in L.$$

It can be easily shown that this definition is equivalent to

$$\forall w \in \Sigma^\omega, \mathscr{P}\mathrm{ref}(\omega) \subseteq \mathscr{P}\mathrm{ref}(L) \Rightarrow w \in L$$

or even by taking the negation of the last line to:

$$\forall w \in \Sigma^\omega, w \notin L \;\Rightarrow\; (\exists u \in \mathscr{P}\mathrm{ref}(\omega) \text{ such that } \forall v \in \Sigma^\omega, u.v \notin L).$$

Let $\mathbb{S}\mathrm{afe}_\omega(\Sigma)$ denote the class of all safe ω-regular languages.

b^*a^ω is not a safe language. Indeed, every prefix b^k of b^ω (which is not in the language) is a prefix of $b^k a^\omega$ (which is in the language). On the other hand, $b^*a^\omega + b^\omega$ is safe. It follows that $\mathbb{S}\mathrm{afe}_\omega(\Sigma) \neq \mathbb{D}\mathrm{et}_\omega(\Sigma)$ and we are going to show (Theorem 2) that $\mathbb{S}\mathrm{afe}_\omega(\Sigma) \subset \mathbb{D}\mathrm{et}_\omega(\Sigma)$.

A *DB*-machine is a deterministic Büchi automaton where $F = Q$. As the automaton need not be complete, the strings that are not accepted are those that simply cannot be parsed through the automaton.

Theorem 2. *L is a safe ω-regular language iff L is recognized by a DB-machine.*

We introduce the following definitions in order to prove the theorem:

Definition 3. A language $P \subseteq \Sigma^*$ is a regular prefix language if and only if:
(1) P is regular,
(2) every prefix of a string of P is a string of P:

$$\forall u \in \Sigma^*, \forall a \in \Sigma, u.a \in P \;\Rightarrow\; u \in P$$

and
(3) every string of P is a proper prefix of another string of P:

$$\forall u \in P, \exists a \in \Sigma \text{ such that } u.a \in P$$

A dfa $A = \langle Q, \Sigma, \delta, F, q_0\rangle$ is a prefix automaton (or prefix dfa) if and only if:
(1) every state is final and
(2) every state is alive: $\forall q \in Q, \exists a \in \Sigma$ such that $\delta(q,a) \in Q$.

Proposition 4. (1) *If L is an ω-regular language, then $\mathscr{P}ref(L)$ is a regular prefix language.*

(2) *If P is a regular prefix language, then there exists a prefix automaton which recognizes P.*

(3) *If $A = \langle Q, \Sigma, \delta, F, q_0\rangle$ is a prefix automaton, then the language $\mathscr{L}(M)$ recognized by the DB-machine $M = \langle Q, \Sigma, \delta, F, q_0\rangle$ is ω-regular and satisfies $\mathscr{L}(A) = \mathscr{P}ref(\mathscr{L}(M))$.*

Proof. Notice that several different ω-languages can have the same prefix language.

L is ω-regular, so there exist two sequences of regular languages $(A_i)_{i\in 1..n}$ and $(B_i)_{i\in 1..n}$ such that $L = \bigcup_{i=1}^n A_i.B_i^\omega$. Notice also that the following holds:

$$\mathscr{P}\mathrm{ref}\left(\bigcup_{i=1}^n A_i.B_i^\omega\right) = \bigcup_{i=1}^n \mathscr{P}\mathrm{ref}(A_i) \cup A_i.B_i^*.\mathscr{P}\mathrm{ref}(B_i).$$

So $\mathscr{P}\mathrm{ref}(L)$ is a regular language which is closed by prefixes. Moreover, let $u \in \mathscr{P}\mathrm{ref}(L)$. There exists $v \in \Sigma^\omega$ such that $u.v \in L$. Let a be the first letter of v. Then $u.a$ is a prefix of $u.v$, so $u.a \in \mathscr{P}\mathrm{ref}(L)$, hence $\mathscr{P}\mathrm{ref}(L)$ is a regular prefix language.

Let P be a regular prefix language. P is recognized by a dfa A which is minimal and has no dead state. As P is prefix, every state of this automaton is final. Moreover let q be a state of A and u a string such that $\delta(q_0, u) = q$. By the definition of a prefix language, there exists $a \in \Sigma$ such that $u.a \in P$. So $\delta(q, a)$ is necessarily defined, i.e. q is alive.

Let $A = \langle Q, \Sigma, \delta, F, q_0 \rangle$ be a prefix automaton. Consider the corresponding DB-machine $M = \langle Q, \Sigma, \delta, F, q_0 \rangle$. Let us prove that $\mathscr{P}\mathrm{ref}(\mathscr{L}(M)) = \mathscr{L}(A)$. Let $u \in \mathscr{P}\mathrm{ref}(\mathscr{L}(M))$. There exists $w \in \Sigma^\omega$ such that $u.w \in \mathscr{L}(M)$. It is clear that $\delta(q_0, u) \in Q$, so $u \in \mathscr{L}(A)$. Conversely, let $u \in \mathscr{L}(A)$ and $q = \delta(q_0, u)$. As q is alive, we can build two strings v and w such that $\delta(q, v) = q'$ and $\delta(q', w) = q'$. Clearly, the run $C_{u.v.w^\omega}$ goes infinitely often through state q'. So $u.v.w^\omega \in \mathscr{L}(M)$, thus $u \in \mathscr{P}\mathrm{ref}(\mathscr{L}(M))$. □

Proof of Theorem 2. Let L be a language recognized by a DB-machine $M = \langle Q, \Sigma, \delta, Q, q_0 \rangle$ and $w \in \Sigma^\omega$. Assume that every prefix $w(1..n)$ of w can be continued into a string u_n of L recognized by M. The mapping $C : \mathbb{N} \to Q$ such that $C(0) = q_0$ and $C(i+1) = \delta(C(i), u_i(i)) = \delta(C(i), w_i)$ is a run of M on w. Since all the states of M are marked, this run is successful, so $w \in L$. Hence, L is a safe ω-regular language.

Conversely, let L be a safe ω-regular language. By Proposition 4, $\mathscr{P}\mathrm{ref}(L)$ is a regular prefix language which is recognized by some prefix automaton $A = \langle Q, \Sigma, \delta, Q, q_0 \rangle$. We claim that L is recognized by the DB-machine $M = \langle Q, \Sigma, \delta, Q, q_0 \rangle$. Indeed, by Proposition 4, $\mathscr{L}(M)$ satisfies $\mathscr{P}\mathrm{ref}(\mathscr{L}(M)) = \mathscr{L}(A)$. Moreover, by the first part of this proof, $\mathscr{L}(M)$ is a safe language (since M is a DB-machine). So L and $\mathscr{L}(M)$ are both safe languages such that $\mathscr{P}\mathrm{ref}(L) = \mathscr{P}\mathrm{ref}(\mathscr{L}(M)) = \mathscr{L}(A)$. Assume that there exists a string $w \in L$ and not in $\mathscr{L}(M)$ (or vice versa). As $\mathscr{P}\mathrm{ref}(L) = \mathscr{P}\mathrm{ref}(\mathscr{L}(M))$, every prefix of w is in $\mathscr{P}\mathrm{ref}(\mathscr{L}(M))$. Since $\mathscr{L}(M)$ is a safe language, w itself is in $\mathscr{L}(M)$ which is inconsistent with the hypothesis. So $L = \mathscr{L}(M)$. □

Corollary 5. *Let L and L' be two safe ω-regular languages.*

$$\mathscr{P}ref(L) = \mathscr{P}ref(L') \quad iff \quad L = L'.$$

Proof. $\Leftarrow$ is straightforward.

$\Rightarrow$ is an immediate consequence of the previous proof. □

3. Identifying from prefixes

We have to deal with the two following issues:

- one consists in explaining the meaning of the statements "u is a positive prefix of the ω-language L" and "u is a negative prefix of the ω-language L". The meaning of prefixes and the interesting cases to be studied depend on the context of our problem;
- the other is to decide in what way do we want convergence to be defined.

3.1. On positive and negative prefixes

A finite string can be a positive prefix of some ω-language either because all its (infinite) continuations are in the ω-language, or because at least one is. On the other hand, a finite string can be a negative prefix of some ω-language either because none of its (infinite) continuations are in the ω-language, or because at least one is not.

We formalize these notions as follows:

Definition 6. Let $u \in \Sigma^*$.
(1) u is an $\exists$-positive prefix of L iff $\exists v \in \Sigma^\omega$ such that $u.v \in L$.
(2) u is an $\forall$-positive prefix of L iff $\forall v \in \Sigma^\omega$: $u.v \in L$.
(3) u is an $\exists$-negative prefix of L iff $\exists v \in \Sigma^\omega$ such that $u.v \notin L$.
(4) u is an $\forall$-negative prefix of L iff $\forall v \in \Sigma^\omega$: $u.v \notin L$.

Given an ω-language L, let $\mathscr{P}_\forall(L)$ denote the set of all $\forall$-positive prefixes of L, $\mathscr{P}_\exists(L)$ the set of all $\exists$-positive prefixes of L, $\mathscr{N}_\forall(L)$ the set of all $\forall$-negative prefixes of L and $\mathscr{N}_\exists(L)$ the set of all $\exists$-negative prefixes of L.

Two finite sets S_+ and S_- of finite strings form together a *sample* $S = \langle S_+, S_- \rangle$ of (p,n)-examples for an ω-language L if and only if $S_+ \subseteq \mathscr{P}_p(L)$ and $S_- \subseteq \mathscr{N}_n(L)$. We shall manipulate samples as sets, and write $\langle S_+, S_- \rangle \subseteq \langle S'_+, S'_- \rangle$ when $S_+ \subseteq S'_+$ and $S_- \subseteq S'_-$.

For instance, on automaton 2a, $L = (b^*a)^\omega$, $\mathscr{P}_\forall(L) = \mathscr{N}_\forall(L) = \emptyset$ and $\mathscr{P}_\exists(L) = \mathscr{N}_\exists(L) = \Sigma^*$.

Relationships between $\mathscr{P}_\exists(L), \mathscr{P}_\forall(L), \mathscr{N}_\forall(L), \mathscr{N}_\exists(L)$ and $\mathscr{P}\mathrm{ref}(L)$ depend essentially on the properties of the quantifiers:

Proposition 7. *For all ω-languages L,*

$$\mathscr{P}_\exists(L) = \mathscr{P}ref(L),$$

$$\mathscr{P}_\exists(L) \cap \mathscr{N}_\forall(L) = \mathscr{P}_\forall(L) \cap \mathscr{N}_\exists(L) = \emptyset,$$

$$\mathscr{P}_\exists(L) \cup \mathscr{N}_\forall(L) = \mathscr{P}_\forall(L) \cup \mathscr{N}_\exists(L) = \Sigma^*,$$

$$\mathscr{P}_\forall(L) = \mathscr{N}_\forall(\Sigma^\omega \backslash L) \quad \textit{and} \quad \mathscr{P}_\exists(L) = \mathscr{N}_\exists(\Sigma^\omega \backslash L).$$

Proof. Straightforward. □

3.2. Identification in the limit from prefixes

We discuss here the fact that identification in the limit [7] can lead, in the case of infinite prefixes, to two alternative but equivalent definitions. The second one is based on the existence of *characteristic samples*, and allows better proofs.

Definition 8. Given a language L and $p, n \in \{\forall, \exists\}$, a complete (p,n)-prefix presentation of L is an infinite sequence of pairs $\langle u_i, label(u_i) \rangle$ satisfying the following

conditions:
(1) if $label(u_i) = +$ then $u_i \in \mathscr{P}_p(L)$;
(2) if $label(u_i) = -$ then $u_i \in \mathscr{N}_n(L)$;
(3) for all $w \in \mathscr{P}_p(L)$, there exists $i \in \mathbb{N}$ such that $w = u_i$ and $label(u_i) = +$;
(4) for all $w \in \mathscr{N}_n(L)$, there exists $i \in \mathbb{N}$ such that $w = u_i$ and $label(u_i) = -$.
Given a complete (p,n)-prefix presentation of some ω-language L, and k a positive integer, denote $S_k = \{\langle u_i, label(u_i)\rangle \mid i \leqslant k\}$.

It will be necessary to systematically consider a class $\mathbb{L}$ of languages and an associated class $\mathbb{R}$ of representations. The latter one will have to be strong enough to represent the whole class of languages, i.e.

$$\forall L \in \mathbb{L}, \exists R \in \mathbb{R} \text{ such that } \mathscr{L}(R) = L.$$

Definition 9. Let $\mathbb{L}$ be a class of languages represented by some class of representations $\mathbb{R}$. An $(\mathbb{L},\mathbb{R})$-learning algorithm is a program that takes as input a sample of labeled strings and outputs a grammar from $\mathbb{R}$.

We can now simply adjust Gold's definition of identification in the limit [7]:

Definition 10 (Gold identification). A class $\mathbb{L}$ of languages is identifiable in the limit from complete (p,n)-prefix presentations for a class $\mathbb{R}$ of representations if and only if there exists an $(\mathbb{L},\mathbb{R})$-learning algorithm $\mathfrak{A}$ such that given any language L in $\mathbb{L}$ and any complete (p,n)-prefix presentation of L, $\exists k \in \mathbb{N}$ such that $\forall j \geqslant k$, $\mathscr{L}(\mathfrak{A}(S_j)) = L$.

An alternative definition is that there exists a finite set of prefixes with which identification is ensured:

Definition 11 (Identification by characteristic sets). A class $\mathbb{L}$ of languages is identifiable in the limit by characteristic sets of (p,n)-prefixes for a class $\mathbb{R}$ of representations if and only if there exists an $(\mathbb{L},\mathbb{R})$-learning algorithm $\mathfrak{A}$ such that:
(1) given a sample $\langle S_+, S_-\rangle$ of prefixes, with $S_+ \subseteq \mathscr{P}_p(L)$ and $S_- \subseteq \mathscr{N}_n(L)$, $\mathfrak{A}$ returns H in $\mathbb{R}$ consistent with $\langle S_+, S_-\rangle$, i.e. such that $S_+ \subseteq \mathscr{P}_p(\mathscr{L}(H))$ and $S_- \subseteq \mathscr{N}_n(\mathscr{L}(H))$;
(2) for each representation $R \in \mathbb{R}$ of a language L in $\mathbb{L}$, there exists a finite characteristic sample $\langle CS_+, CS_-\rangle$, such that, on $\langle S_+, S_-\rangle$ with $CS_+ \subseteq S_+ \subseteq \mathscr{P}_p(L)$ and $CS_- \subseteq S_- \subseteq \mathscr{N}_n(L)$, $\mathfrak{A}$ returns a hypothesis H equivalent to R.

Theorem 12. *Let $\mathbb{L}$ be a class of languages, and $\mathbb{R}$ a class of representations for $\mathbb{L}$. The two following propositions are equivalent*:
- *$\mathbb{L}$ is identifiable in the limit from complete (p,n)-prefix presentations for a class $\mathbb{R}$ of representations*;
- *$\mathbb{L}$ is identifiable in the limit by characteristic sets of (p,n)-prefixes for a class $\mathbb{R}$ of representations*.

Proof. (i) $\Rightarrow$ (ii): Let L be any language in $\mathbb{L}$, represented by some R in $\mathbb{R}$. Take any complete (p,n)-prefix presentation of L, and let k be the position where $\mathfrak{A}$ from Definition 10 converges. Then (step 1), either S_k is a characteristic sample (for Definition 11) and we are done, or there exists a super-sample of S_k, say $S_{k'}$, such that $\mathfrak{A}(S_{k'})$ is equivalent to R. In that case $\mathfrak{A}$, over the new complete presentation $S_{k'}$ converges at a later rank. Such a rank exists as L is identifiable in the limit from complete (p,n)-prefix presentations. Denote this rank k (again) and return to step 1. We argue that this process can only happen finitely many times, thus leading to the construction of a characteristic sample. Indeed, if the process can take place an infinite number of time, this implies an infinite (p,n)-prefix presentation, from which convergence of algorithm $\mathfrak{A}$ does not take place. If such a sequence exists it cannot be complete.

(ii) $\Rightarrow$ (i): If $\mathbb{L}$ is identifiable in the limit by characteristic sets of (p,n)-prefixes, then algorithm $\mathfrak{A}$ from Definition 11 can be used to identify any language in $\mathbb{L}$ from a complete (p,n)-prefix presentation: as soon as the strings in the characteristic sample have appeared, identification is ensured.

3.3. Identification in the limit from polynomial prefixes and time

In this section, we adapt the definitions of Gold [8] and de la Higuera [5]. Other paradigms than identification in the limit are known, but they are often either similar to these or harder to establish. A comparison between different models can be found in [14].

The size of a representation R, denoted $\|R\|$, is polynomially related to the size of its encoding. In the case of a deterministic automaton, the number of states is a relevant measure, since the alphabet has constant size. The size of a sample S of finite strings is the sum of the length of all strings in S.

We now adapt the definition of identification in the limit from polynomial time and data [5,8] to the case of learning from prefixes. This definition takes better care of practical considerations: for instance with this definition, deterministic finite automata are learnable whereas context-free grammars or non-deterministic automata are not.

Definition 13 (Polynomial identification). A class $\mathbb{L}$ of ω-languages is (p,n)-identifiable in the limit from polynomial time and finite prefixes for a class $\mathbb{R}$ of representations if and only if there exists an algorithm $\mathfrak{A}$ and two polynomials $\alpha(\,)$ and $\beta(\,)$ such that:

(1) given a finite sample $\langle S_+, S_-\rangle$ of prefixes of size m, with $S_+ \subseteq \mathscr{P}_p(L)$ and $S_- \subseteq \mathscr{N}_n(L)$, $\mathfrak{A}$ returns a hypothesis $H \in \mathbb{R}$ in $\mathcal{O}(\alpha(m))$ time and H is consistent with $\langle S_+, S_-\rangle$;

(2) for all representations R of size k of a language L in $\mathbb{L}$, there exists a finite characteristic sample $\langle CS_+, CS_-\rangle$ of size at most $\mathcal{O}(\beta(k))$ such that, on $\langle S_+, S_-\rangle$ with $CS_+ \subseteq S_+ \subseteq \mathscr{P}_p(L)$ and $CS_- \subseteq S_- \subseteq \mathscr{N}_n(L)$, $\mathfrak{A}$ returns a hypothesis $H \in \mathbb{R}$ which is equivalent to R.

3.4. The problem of learning ω-languages from their prefixes

We have now defined the different parameters of the problem. The main question is: "can the class $\mathbb{L}$ of ω-regular languages represented by $\mathbb{R}$ be learned following the criterion **c** from a sample of (p,n)-examples?"

The classes $\mathbb{L}$ we are interested in are those defined in Section 2. The representation classes are $\mathbb{B}$A (Büchi automata) for $\mathbb{R}\text{eg}_\omega(\Sigma)$, $\mathbb{D}$BA (deterministic Büchi automata) for $\mathbb{D}\text{et}_\omega(\Sigma)$ and $\mathbb{D}$BM (*DB*-machines) for $\mathbb{S}\text{afe}_\omega(\Sigma)$. The criteria **c** will be identification in the limit, **idlim**, and identification in the limit from polynomial time and prefixes, **polyid**. The examples of positive and negative prefixes will be defined according to the different combinations of the quantifiers $\exists$ and $\forall$.

Hence a learning problem will be completely specified when given:
(1) the class of languages and its representation class;
(2) the convergence criterion;
(3) the interpretation one gives to positive and negative prefixes.
A problem will thus be a triple $\langle \mathbb{L}_\mathbb{R}, \mathbf{c}, \mathbf{i}\rangle$ where the criterion **c** will be **idlim** (identification in limit) or **polyid** (identification in the limit from polynomial time and prefixes) and the interpretation **i** will be a pair (p,n) in $\{\exists,\forall\}\times\{\exists,\forall\}$.

Example 14. The problem $\langle \mathbb{S}\text{afe}_\omega(\Sigma)_{\mathbb{D}\text{BM}}, \textbf{idlim}, (\exists,\forall)\rangle$ is the one of identification in the limit of the class $\mathbb{S}\text{afe}_\omega(\Sigma)$ where the languages are represented by *DB*-machines and a presentation made of existential positive prefixes and universal negative prefixes (see Definition 6) is given.

Such a problem will have a *positive status* if this class is actually learnable with the chosen criterion, a *negative status* if it is not and an *unknown status* if the problem is unsolved.

4. Results

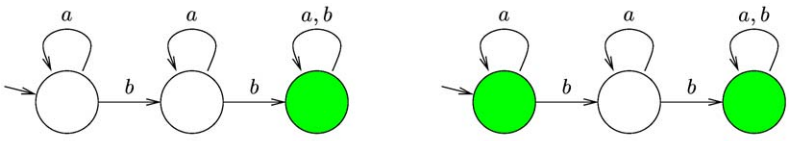
We give two types of results. The first concerns classes $\mathbb{R}\text{eg}_\omega(\Sigma)$ and $\mathbb{D}\text{et}_\omega(\Sigma)$ for which identification in the limit from prefixes is impossible. The second concerns the class of safe languages, for which identification in the limit from polynomial time and prefixes is proved.

4.1. General properties

We first notice (by a straightforward reduction) that polynomial identification only holds when identification in the limit also holds: if $\langle \mathbb{L}_\mathbb{R}, \textbf{idlim}, sign\rangle$ has a negative status, then so does $\langle \mathbb{L}_\mathbb{R}, \textbf{polyid}, sign\rangle$.

A necessary condition for the identification of a class of languages is that any pair of languages from the class can be effectively separated by some prefix:

Lemma 15. *Let $\mathbb{L}$ be a class of ω-languages and $\mathbb{R}$ a class of representations for $\mathbb{L}$. If there exist L_1 and L_2 in $\mathbb{L}$ such that $L_1 \neq L_2$, $\mathscr{P}_p(L_1) = \mathscr{P}_p(L_2)$ and $\mathscr{N}_n(L_1) = \mathscr{N}_n(L_2)$, then the problem $\langle \mathbb{L}_\mathbb{R}, \textbf{idlim}, (p,n)\rangle$ has a negative status.*



$L_1 = a^*ba^*b(a+b)^\omega$ $L_2 = a^\omega + a^*ba^*b(a+b)^\omega$

Fig. 3. Automata 3a and 3b accepting respectively languages $a^*ba^*b(a+b)^\omega$ and $a^\omega + a^*ba^*b(a+b)^\omega$.

Proof. Suppose that an algorithm $\mathfrak{A}$ identifies the class $\mathbb{L}$; then L_1 and L_2 have respective characteristic samples CS_1 and CS_2. But L_1 and L_2 are consistent with $CS_1 \cup CS_2$. Hence either L_1 or L_2 is not identified. □

As an immediate consequence of Lemma 15, we prove:

Theorem 16. *For any class of representations* $\mathbb{R}$ *and for all* $(p,n) \in \{\exists,\forall\} \times \{\exists,\forall\}$, *problems* $\langle \mathbb{R}eg_\omega(\Sigma)_\mathbb{R}, \textbf{idlim}, (p,n)\rangle$ *and* $\langle \mathbb{D}et_\omega(\Sigma)_\mathbb{R}, \textbf{idlim}, (p,n)\rangle$ *have negative status.*

Proof. We use the same counter-example, shown in Fig. 3, to prove that neither the class of all ω-regular languages, nor that of all ω-deterministic ones are identifiable in the limit (and furthermore polynomially identifiable from prefixes). The languages accepted by automata 3a and 3b are, respectively, $L_1 = a^*ba^*b(a+b)^\omega$ and $L_2 = a^\omega + a^*ba^*b(a+b)^\omega$. Whatever the choice of quantifiers p and n, languages $\mathscr{P}_p$ and $\mathscr{N}_n$ are identical in both cases.

Formally:

$$\mathscr{P}_\exists(L_1) = \mathscr{P}_\exists(L_2) = \Sigma^*,$$

$$\mathscr{P}_\forall(L_1) = \mathscr{P}_\forall(L_2) = a^*ba^*b(a+b)^*,$$

$$\mathscr{N}_\exists(L_1) = \mathscr{N}_\exists(L_2) = a^* + a^*ba^*,$$

$$\mathscr{N}_\forall(L_1) = \mathscr{N}_\forall(L_2) = \emptyset. \qquad \square$$

4.2. On the identification of safe languages

The previous result is very negative, but hardly surprising. It implies that learning requires either to consider a subclass of languages, and/or to change the convergence criterion. It is surely not reasonable to choose a less demanding criterion than identification in the limit; we shall thus concentrate on a subclass of ω-deterministic languages in the sequel: the safe ω-languages. We first prove that the class of prefix languages is polynomially identifiable in the limit from given data. This result is closely related to that of identifying safe ω-languages from given prefixes as in fact we can match both problems.

Proposition 17. *The class of regular prefix languages, represented by prefix dfa, is polynomially identifiable in the limit from given data.*

Algorithm 1. RPNIprefixes1

Input: a sample of positive and negative strings $S = \langle S_+, S_- \rangle$
Output: a prefix automaton $A = \langle Q, \Sigma, \delta, F, q_0 \rangle$

$A \leftarrow \mathtt{RPNI}(S_+, S_-)$
if A is not a prefix *dfa* **then**
  $max\_neg \leftarrow \mathtt{max}\{\mathtt{length}(u) \mid u \in S_-\}$
  for all $w \in S_+$ such that
    $w \in \mathscr{P}\mathrm{ref}(S_-)$ and
    $w \notin \mathscr{P}\mathrm{ref}(S_+ \setminus \{w\})$
  do
    Compute v of length max_neg such that
      $\mathscr{P}\mathrm{ref}(v) \cap S_- = \emptyset$ and $w \in \mathscr{P}\mathrm{ref}(v)$
    $S_+ \leftarrow S_+ \cup \{v\}$
  end for
  $A \leftarrow \mathtt{PTA}(S_+)$
  $Q \leftarrow Q \cup \{q_f\}$; $F \leftarrow Q$
  for all $a \in \Sigma$ **do** $\delta(q_f, a) \leftarrow q_f$
  for all $q \in Q$ such that q is a leaf **do** $\forall a \in \Sigma, \delta(q, a) \leftarrow q_f$
end if
return A

Proof. To prove the above proposition we use algorithm RPNIprefixes1 (see algorithm 1) which identifies in the limit prefix dfa from a sample of positive and negative strings, if there exists a prefix dfa consistent with this sample; this condition is granted as we are in an identification setting.

RPNIprefixes1 makes use, as a subroutine, of RPNI [13] which can identify a dfa from positive and negative data. The main features of RPNI are:

- RPNI takes as input two finite sets of finite strings S_+ and S_-;
- RPNI constructs a dfa consistent with S_+ and S_- in time polynomial in $\|S_+ \cup S_-\|$;
- RPNI identifies dfa in polynomial time from given data;
- the first object RPNI builds is the prefix tree acceptor (PTA): this is the largest dfa with no useless [1] state recognizing exactly S_+.

If $\langle S_+, S_- \rangle$ contains a characteristic sample of the target language L, then RPNI returns a prefix automaton A that accepts language L [13]. If $\langle S_+, S_- \rangle$ does not contain any characteristic sample, RPNI returns an automaton which is consistent with $\langle S_+, S_- \rangle$, but may be neither prefix nor even transformable into a prefix automaton. In that case RPNIprefixes1 transforms the PTA into a consistent prefix automaton.

Indeed the function $\mathtt{PTA}(S_+)$ builds the PTA corresponding to S_+ in which extra strings are added whose positive labeling does not introduce inconsistency; testing ($w \in \mathscr{P}\mathrm{ref}(S_-)$ and $w \notin \mathscr{P}\mathrm{ref}(S_+ \setminus \{w\})$) allows to know which states of the PTA have no successors; these states must then lead to a new universal [2] state q_f whenever the new transition is not used by some negative string: such a transition always exists

[1] A state is useless if it does not lead to an accepting state, or is not accessible from the initial state.
[2] A state is universal if by any letter there is a transition to the same state.

since the data is supposed to be consistent. Building a polynomial implementation is straightforward. □

An alternative and more efficient algorithm is proposed in the next section; RPNIprefixes2 returns a compatible non-trivial prefix automaton, even when the characteristic sample is not included in the data. We now turn to the case of the safe languages:

Theorem 18. $\langle \mathbb{S}safe_\omega(\Sigma)_{\mathbb{D}BM}, \textbf{polyid}, (\exists,\forall)\rangle$ *has a positive status.*

Proof. We show that the conditions of Definition 13 are met:

(1) Let L be a safe language. On any sample $\langle S_+, S_-\rangle$ of $(\exists,\forall)$-prefixes for L, by Proposition 17 a prefix dfa accepting S_+ and rejecting S_- can be returned in polynomial time. In constant time this automaton is transformed into a DB-machine M by changing the acceptance criterion. Furthermore $S_+ \subseteq \mathscr{P}\text{ref}(\mathscr{L}(M))$ and $S_- \cap \mathscr{P}\text{ref}(\mathscr{L}(M)) = \emptyset$.

(2) Let L be a safe language and M a DB-machine accepting L. Let A be the prefix automaton associated with M. Let $\langle CS_+, CS_-\rangle$ be a characteristic sample for A and RPNI. Let now $\langle S_+, S_-\rangle$ be a sample of examples such that $CS_+ \subseteq S_+ \subseteq \mathscr{P}_p(L)$ and $CS_- \subseteq S_- \subseteq \mathscr{P}_n(L)$ and $S_+ \subseteq \mathscr{L}(A)$ and $S_- \cap \mathscr{L}(A) = \emptyset$. Notice that the size of $\langle CS_+, CS_-\rangle$ is polynomial in that of A which in turn is the same as the size of M. On input $\langle S_+, S_-\rangle$, RPNI returns an automaton A' equivalent to A. By construction, the DB-machine M' associated to A' is such that $\mathscr{P}\text{ref}(\mathscr{L}(M')) = \mathscr{L}(A') = \mathscr{L}(A) = \mathscr{P}\text{ref}(\mathscr{L}(M))$. So by Corollary 5, $\mathscr{L}(M) = \mathscr{L}(M')$ holds. □

Theorem 19. *If a class* $\mathbb{L}$ *of* ω*-regular languages strictly contains* $\mathbb{S}afe_\omega(\Sigma)$ *and* $\mathbb{R}$ *is a class of machines for* $\mathbb{L}$*, then* $\langle \mathbb{L}_{\mathbb{R}}, \textbf{idlim}, (\exists,\forall)\rangle$ *has a negative status.*

Proof. It is sufficient to find two ω-regular languages that share the same sets of positive and negative prefixes: let $\mathbb{L}$ be a class strictly containing $\mathbb{S}\text{afe}_\omega(\Sigma)$ and L a language in $\mathbb{L}$ but not in $\mathbb{S}\text{afe}_\omega(\Sigma)$. As L is ω-regular, $\mathscr{P}_\exists(L)$ is a prefix language. But in that case there exists a language L' in $\mathbb{S}\text{afe}_\omega(\Sigma)$ such that $\mathscr{P}_\exists(L) = \mathscr{P}_\exists(L')$ and $\mathscr{N}_\forall(L) = \mathscr{N}_\forall(L')$. So by Lemma 15, it follows that L is not identifiable. □

Theorems 18 and 19 allow us to deduce a final result concerning learning from $(\forall,\exists)$-prefixes. An ω-language L is *co-safe iff* its complementary $\Sigma^\omega \backslash L$ is a safe language. We denote $\mathbb{C}\text{o-safe}_\omega(\Sigma)$ the family of co-safe ω-regular languages. Co-safe languages are accepted by co-DB-machines, i.e., complete Büchi automata with a unique marked state which is a universal state.

Theorem 20. $\langle \mathbb{C}o\text{-}safe_\omega(\Sigma)_{\mathbb{C}DBM}, \textbf{polyid}, (\forall,\exists)\rangle$ *has a positive status. Furthermore, for any class* $\mathbb{L}$ *of* ω*-regular languages strictly containing* $\mathbb{C}o\text{-}safe_\omega(\Sigma)$ *and* $\mathbb{R}$ *a class of machines for* $\mathbb{L}$*,* $\langle \mathbb{L}_{\mathbb{R}}, \textbf{idlim}, (\forall,\exists)\rangle$ *has a negative status.*

Proof. Any complete prefix presentation by $(\forall,\exists)$ of a co-safe language L is a complete prefix presentation by $(\exists,\forall)$ of the safe language $\Sigma^\omega \backslash L$ since $\mathscr{N}_\forall(\Sigma^\omega \backslash L) = \mathscr{P}_\forall(L)$

and $\mathcal{N}_{\exists}(\Sigma^{\omega}\backslash L) = \mathcal{P}_{\exists}(L)$. Moreover, the construction of a co-DB-machine from a DB-machine can be done in linear time by completing it with a universal state which becomes the marked state. From Theorem 18, the problem $\langle \mathbb{S}\text{afe}_{\omega}(\Sigma)_{\mathbb{D}BM}, \textbf{polyid}, (\exists, \forall)\rangle$ has a positive status, and so has $\langle \mathbb{C}\text{o-safe}_{\omega}(\Sigma)_{\mathbb{C}\text{DBM}}, \textbf{polyid}, (\forall, \exists)\rangle$. □

5. A constructive prefix *DFA* inference algorithm

The algorithm proposed in Section 4 identifies polynomially and in the limit from given data any prefix automaton. It is nevertheless practically a useless algorithm: one is never sure to have a characteristic sample inside his learning data, and returning the PTA with some added edges is not convincing. We give here a specific prefix automaton learning algorithm. It is based on `RPNI` [13], and uses notations from [6] (see algorithm 2).

Algorithm `RPNIprefixes2` adds to S_+ all prefixes of S_+ and goes through a typical state merging routine. The only problem is to make sure that every merge leads to an automaton that can be completed in some way into a prefix automaton. To do this, each positive state has to stay alive: there must be at least one infinite string leading from this state that avoids every negative state.

The function `Choose_transition` returns a triple $\langle q, a, q'\rangle$ corresponding to the transition $\delta(q,a) = q'$ where $\delta(q,a)$ is undefined and $q' \notin \texttt{Tested}(q,a)$. This function is usually implemented by ordering the states of the Prefix Tree Acceptor using a length-lexicographic order, and returning the triple $\langle q, a, q'\rangle$ such that $q' \leqslant q$ where the merge of q and q' has not been tested before. For this to work, renumbering of states, after merging, is always done by taking the smallest number.

Other functions have been tested, `EDSM` (Evidence Driven State Merging) type functions have been shown preferable [11] in practice; these would allow different merges, associate a score to each merge; this score takes into account the amount of states that will be recursively merged, and an `EDSM Choose_transition` chooses the merge with highest score.

The function `Possible`$(\delta(q,a) = q')$ returns `True` if adding to δ rule $\delta(q,a) = q'$ does not lead to an inconsistency, `False` otherwise. Inconsistency appears when after merging the automaton either accepts some negative example, or rejects a positive one.

Inconsistency is tested on the current automaton on which rule $\delta(q,a) = q'$ is added. It can have two causes:

(1) There exists two strings $u.a.w$ and $v.w$ such that $\delta(q_0,u) = q$ and $\delta(q_0,v) = q'$ and either $u.a.w \in S_+$, $v.w \in S_-$ or $u.a.w \in S_-, v.w \in S_+$.

(2) A state is no more alive; a state q is alive if it can still lead to an accepting state: $\exists w \in \Sigma^{\omega}$ such that $(\{u \mid \delta(q_0,u) = q\}.\mathcal{P}\text{ref}(\omega)) \cap S_- = \emptyset$. This ensures that the current automaton (and thus by induction the last one) can be transformed into a prefix dfa.

The main elements of the proof of `RPNIprefixes2` are:

- The algorithm returns a prefix automaton (by construction).
- The `Possible` test ensures that all states are alive and that at any moment the automaton can be transformed into a consistent prefix automaton.

- In the case where a characteristic sample (for RPNI) is included, no transformation will take place.

Algorithm 2. RPNIprefixes2

Input: a sample of positive and negative strings $S = \langle S_+, S_- \rangle$
Output: a prefix automaton A defined by δ, F_+, F_-

$S_+ \leftarrow S_+ \cup \mathscr{P}\text{ref}(S_+)$
$n \leftarrow 0$
for all $a \in \Sigma$ **do** $\texttt{Tested}(q_0, a) \leftarrow \emptyset$
$F_+ \leftarrow \{q_0\}$; $F_- \leftarrow \emptyset$
while there are some unmarked strings in $S_+ \cup S_-$ **do**
  $\langle q, a, q' \rangle \leftarrow \texttt{Choose\_transition}()$
  if $\texttt{Possible}(\delta(q, a) = q')$ **then**
    $\delta(q, a) \leftarrow q'$
    for all unmarked strings in $S_+ \cup S_-$ **do**
      if $\delta(q_0, w) = q$ **then** $\texttt{mark}(w)$
      if $w \in S_+$ **then** $F_+ \leftarrow F_+ \cup \{q\}$
      if $w \in S_-$ **then** $F_- \leftarrow F_- \cup \{q\}$
    end for
  else
    $\texttt{Tested}(q, a) \leftarrow \texttt{Tested}(q, a) \cup \{q'\}$
  end if
  if $\|\texttt{Tested}(q, a)\| = n + 1$ {impossible merging} **then**
    $n \leftarrow n + 1$ {creation of a new state}
    $Q \leftarrow Q \cup \{q_n\}$
    $\delta(q, a) \leftarrow q_n$
    for all unmarked strings in $S_+ \cup S_-$ **do**
      if $\delta(q_0, w) = q$ **then** $\texttt{mark}(w)$
      if $w \in S_+$ **then** $F_+ \leftarrow F_+ \cup \{q\}$
      if $w \in S_-$ **then** $F_- \leftarrow F_- \cup \{q\}$
    end for
    for all $a \in \Sigma$ **do** $\texttt{Tested}(q_n, a) \leftarrow \emptyset$
  end if
end while
$Q \leftarrow F_+$ {conversion into a consistent prefix *dfa*}
for all $q \in F_+$ such that $\forall a \in \Sigma, \delta(q, a) \notin Q$ **do**
  choose w of minimal length such that
    $(\{u \in \Sigma^* \mid \delta(q_0, u) = q\}.\mathscr{P}\text{ref}(\omega)) \cap S_- = \emptyset$
  $Q \leftarrow Q \cup \{q_i^w \mid 0 < i < \|w\|\}$
  $F_+ \leftarrow F_+ \cup \{q_i^w \mid 0 < i < \|w\|\}$
  for $i = 0$ to $\|w\|$ **do** $\delta(q_{i-1}^w, w_i) \leftarrow q_i^w$
  for all $a \in \Sigma$ **do** $\delta(q_{\|w\|}^w, a) \leftarrow q_{\|w\|}^w$
end for

- Finally, the algorithm works in polynomial time.

We refer the interested reader to [6] for a complete proof, given in the case of *dfa*infer, but basically translatable to RPNIprefixes2.

6. Conclusion

This work is a first approach to the problem of learning or identifying automata on infinite strings from finite prefixes. A certain number of open questions and new research directions can be proposed. Among those we mention:

- We give no results concerning the case where all prefixes are existential, i.e., for the problem $\langle \mathbb{X}_{\mathbb{Y}}, \textbf{criterion}, (\exists, \exists) \rangle$. It is rather easy to show that for all the classes of languages studied in this paper, the status will be negative. It nevertheless seems relevant to find a class of languages (undoubtedly rather restricted) for which the status would be positive.
- Following the tracks of [15] or [12], ω-languages learning from queries still leads to interesting questions (see also [17] for recent work). Prefix queries (membership queries on the prefixes), equivalence queries, or infinite pattern queries (does this string appear an infinite number of times in one/all string(s)?) are candidate queries.
- On real data (produced by a system), typical behaviors may be that the type of automata corresponding to real world tasks has the characteristic to have a large alphabet, but few outgoing transitions per state. In this context simplification by typing of the alphabet [9] is undoubtedly a track to be retained, in order to speed-up the inference process.

Acknowledgements

The authors would like to thank Maurice Nivat who suggested the problem and the different reviewers who contributed positively to the paper.

References

[1] B. Alpern, A. Demers, F. Schneider, Defining liveness, Inform. Process. Lett. 21 (1985) 181–185.
[2] D. Angluin, On the complexity of minimum inference of regular sets, Inform. and Control 39 (1978) 337–350.
[3] A. Bouajjani, J. Fernandez, S. Graf, C. Rodriguez, J. Sifakis, Safety for branching time semantics, in: Proc. 18th Internat. Coll. on Automata, Languages and Programming, Lecture Notes in Computer Science, Vol. 510, Springer, Wien, Austria, 1991, pp. 76–92.
[4] J. Büchi, On a decision method in restricted second order arithmetic, in: E. Nagel, P. Suppes, A. Tarski (Eds.), Proc. 1st Internat. Cong. on Logic, Methodology and Philosophy of Science, Stanford University Press, Stanford, Canada, 1960, pp. 1–11.
[5] C. de la Higuera, Characteristic sets for polynomial grammatical inference, Mach. Learning 27 (1997) 125–138.
[6] C. de la Higuera, J. Oncina, E. Vidal, Identification of dfa's: data-dependant versus data-independant algorithms, in: Proc. 3rd Internat. Coll. on Grammatical Inference (ICGI'96), Lecture Notes in Artificial Intelligence, Vol. 1147, Springer, Montpellier, France, 1996, pp. 313–325.
[7] M. Gold, Language identification in the limit, Inform. and Control 10 (1967) 447–474.

[8] M. Gold, Complexity of automaton identification from given data, Inform. and Control 37 (1978) 302–320.
[9] C. Kermorvant, C. de la Higuera, Learning languages with help, in: P. Adriaans, H. Fernau, M. van Zaannen (Eds.), Grammatical Inference: Algorithms and Applications, Proc. ICGI '00, Lecture Notes in Artificial Intelligence, Vol. 2484, Springer, Berlin, Heidelberg, 2002, pp. 161–173.
[10] L. Lamport, Proving the correctness of multiprocess programs, IEEE Trans. Software Eng. 3 (2) (1977) 125–143.
[11] K. Lang, B. Pearlmutter, R. Price, Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm, in: Proc. 4th Internat. Coll. on Grammatical Inference (ICGI'98), Lecture Notes in Artificial Intelligence, Vol. 1433, Springer, Ames, Iowa, USA, 1998, pp. 1–12.
[12] O. Maler, A. Pnueli, On the learnability of infinitary regular sets, in: Proc. 4th Ann. Workshop on Computational Learning Theory (COLT'91), Morgan Kaufmann, Santa Cruz, California, USA, 1991, pp. 128–136.
[13] J. Oncina, P. Garcia, Identifying regular languages in polynomial time, in: H. Bunke (Ed.), Advances in Structural and Syntactic Pattern Recognition, Vol. 5, World Scientific, Singapore, 1992, pp. 99–108.
[14] R. Parekh, V. Honavar, On the relationship between models for learning in helpful environments, in: Proc. 5th Internat. Coll. on Grammatical Inference (ICGI'2000), Lecture Notes in Computer Science, Vol. 1891, Springer, Lisbon, Portugal, 2000, pp. 207–220.
[15] A. Saoudi, T. Yokomori, Learning local and recognizable ω-languages and monadic logic programs, in: J. Shawe-Taylor, M. Anthony (Eds.), Proc. 1st European Conf. on Computational Learning Theory (EuroCOLT'93), Oxford University Press, London, UK, 1994, pp. 157–169.
[16] W. Thomas, Automata on infinite objects, Handbook of Theoretical Computer Science, Vol. B, Chap. 4, Elsevier, Amsterdam, North-Holland, 1990, pp. 135–191.
[17] D.G. Thomas, M.H. Begam, K.G. Subramanian, S. Gnanasekaran, Learning of regular bi-ω languages, in: P. Adriaans, H. Fernau, M. van Zaannen (Eds.), Grammatical Inference: Algorithms and Applications, Proc. ICGI '00, Lecture Notes in Artificial Intelligence, Vol. 2484, Springer, Berlin, Heidelberg, 2002, pp. 283–292.
[18] M. Vardi, P. Wolper, Automata-theoretic techniques in modal logics of programs, J. Comput. Systems Sci. 32 (1986) 183–221.