Inference of ω-Languages from Prefixes

Colin DE LA HIGUERA and Jean-Christophe JANODET

EURISE, Université de Saint-Etienne
France
cdlh@univ-st-etienne.fr, janodet@univ-st-etienne.fr

Abstract. Büchi automata are used to recognize languages of infinite words. 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 words, namely the prefixes of accepted or rejected infinite words. 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 words 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, whichever 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.

1 Introduction

Grammatical inference [5, 7, 11] deals with the general problem of automatic learning machines (grammars or automata) from structured data, and more usually words. Between the different syntactic objects from formal language theory, most attention has been paid to the case of deterministic finite automata (*dfa*), even if some results on different types of grammars are known. On the other hand the question of learning automata on infinite words has hardly been studied.

The study of these 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 and the semantics of modal and temporal logics [14]. Today, these automata are used to model critical reactive systems. 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 software. This is the case for instance of automatic pilots, operating systems or nuclear station automatic supervisors.

The development of such software requires automatic program proving capacities. It is wished in particular that properties known as safety, which expresses that something bad will never occur during the execution of the system, are verified. Current examples of safety properties are mutual exclusion or deadlock avoidance [1]. These properties are described formally in temporal logics like *PTL* (*Propositional Temporal Logic*), whose models, Kripke structures, can be modeled by Büchi automata [14]. Consequently, Büchi automata make it possible to model with the same formalism the critical systems and the logical properties that they must satisfy and to develop effective proof algorithms (model checking).

Nevertheless, the formal specifications of the critical software, and more still, 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. 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). This system is represented by the automaton below:

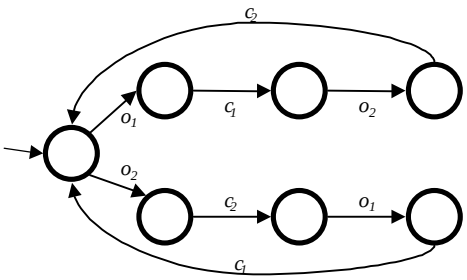

Fig. 1. A two-gate lock chamber (*o*=open, *c*=closed)

The safety property "gates 1 and 2 are never open at the same time" is written, in *PTL*: $\Box(\text{not } p_1 \vee \text{not } p_2)$, where property p_i is that "gate i is open". 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, $o_1\ c_1\ o_2\ c_2\ o_2\ c_2\ o_1\ c_1\ldots$ and $o_2\ c_2\ o_1\ c_1\ldots$, which are "good" behaviors, or $o_1\ o_2\ldots$ and $o_1\ c_1\ o_1\ldots$, 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 words 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 words? Let us recall that with an alphabet of size 2 the set of infinite words is already uncountable. In previous research, the choice was to use data coming from the countable subset of the ultimately periodic words (of type uv^{ω}, u and v being finite words). Saoudi and Yokomori [12] define a (restricted) class of local languages, and prove the learnability of these languages from positive examples; Maler and Pnueli [9] 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 words. And the interpretation of these words can vary. A finite word 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 one of its continuations at least 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 words, from prefixes. In section 2 we will give the definitions concerning the ω-languages, and in section 3 those necessary to the comprehension of the learning problems. In section 4 we establish several learnability results, by showing that for the majority of the alternatives, identification in the limit of the classes of ω-regular languages and ω-deterministic languages is not possible. A positive result concerning the polynomial identification of safe languages is given.

2 Definitions

2.1 Finite Words, Languages and Automata

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

A deterministic finite automaton (*dfa*) is a quintuple $A=<Q, \Sigma, \delta, F, q_0>$ where Σ is an alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta: Q\times\Sigma \rightarrow Q$ is a transition function, and $F \subseteq Q$ is a set of marked states, called the final states.

We define recursively:

- $\delta(q_i, \lambda) = q_i$
- $\delta(q_i, a.w) = \delta(\delta(q_i, a), w)$

$L(A)$, the language recognized by automaton A is $\{w \in \Sigma^*: \delta(q_0, w) \in F\}$.

It is well known that the languages recognized by *dfa*s form the family of regular languages. This class is considered as a borderline case for grammatical inference [7].

2.2 Infinite Words and ω-Languages

We mainly use the notations from [13].

An infinite word u (or ω-word) over Σ is a mapping $\mathbf{N} \rightarrow \Sigma$. Such a word is written $u(0)u(1)...u(n)...$, with $u(i) \in \Sigma$. Σ^ω denotes the set of all ω-words over Σ. An ω-language over Σ is a set of infinite words, thus a subset of Σ^ω.

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

$L^\omega = \{u \in \Sigma^\omega / u = u_0u_1...: \forall i \in \mathbf{N}\ u_i \in L\}$ and

$KL^\omega = \{u \in \Sigma^\omega / u = u_1u_2: u_1 \in K \text{ and } u_2 \in L^\omega\}$

An ω-language L is ω-regular *iff* there exists two finite sequences of regular languages $<A_i>_{i\in[n]}$ and $<B_i>_{i\in[n]}$ such that $L=\bigcup_{i=1}^{i=n} A_i B_i^{\omega}$.

Let $Pref(u)$ denote the set of all finite prefixes of an infinite word u.

Given an ω-language L, $Pref(L)=\bigcup_{u\in L} Pref(u)$.

2.3 Automata on Infinite Words

Büchi automata [4] are used to recognize languages of infinite words. These languages are actually used to model reactive systems [14] and infinite games [13].

A Büchi automaton is a quintuple $A=<Q, \Sigma, \delta, F, q_0>$ where Σ is an alphabet, Q is a finite set of states, $q_0\in Q$ is an initial state, $\delta: Q\times\Sigma\rightarrow 2^Q$ is a transition function, and $F\subseteq Q$ is a set of marked states.

A run of A on an ω-word u is a mapping C_u: $\mathbf{N}\rightarrow Q$ such that:

(i) $C_u(0)= q_0$

(ii) $\forall i\in \mathbf{N}$, $C_u(i+1)\in\delta(C_u(i), u(i))$

Note that C_u is undefined if at some point $C_u(i)$ is undefined.

An ω-word u is accepted by A *iff* there exists a state of F which appears infinitely often in a run of A on u. Let $L(A)$ be the set of all accepted ω-words by A. We can show [13] that an ω-language L is ω-regular *iff* $L=L(A)$ for some Büchi automaton A.

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

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

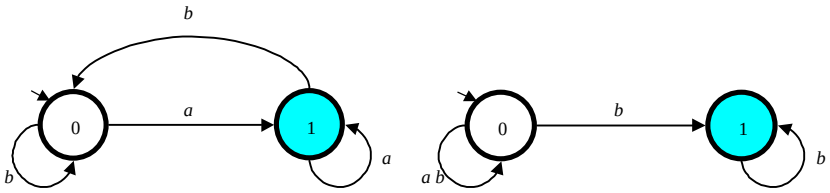

Fig. 2. Büchi automata 2a and 2b recognize $(b^*a)^{\omega}$ and $(a+b)^*b^{\omega}$. Their marked states are in gray.

2.4 ω_{ω}-Safe Languages and *DB*-Machines

An ω-language L is safe [1] *iff*

$\forall w\in\Sigma^{\omega}$, $(\forall u\in Pref(w), \exists v\in\Sigma^{\omega}: uv\in L) \Rightarrow w\in L$

ie

$\forall w\in\Sigma^{\omega}$, $Pref(w)\subseteq Pref(L) \Rightarrow w\in L$

that is to say,

$\forall w\in\Sigma^{\omega}$, $w\notin L \Rightarrow (\exists u\in Pref(w): \forall v\in\Sigma^{\omega}\ uv\notin L)$

Let $\mathbf{Safe}_{\omega}(\Sigma)$ denote the class of all safe ω-regular languages.

b^*a^{ω} is not a safe language. Indeed, every prefix b^n of b^{ω} (which is not in the language) is a prefix of $b^n a^{\omega}$ (which is in the language). On the other hand, $b^*a^{\omega} + b^{\omega}$ is safe. It follows that $\mathbf{Safe}_{\omega}(\Sigma) \neq \mathbf{Det}_{\omega}(\Sigma)$ and we are going to show (Theorem 1) that $\mathbf{Safe}_{\omega}(\Sigma) \subset \mathbf{Det}_{\omega}(\Sigma)$.

A *DB*-machine is a deterministic Büchi automaton where $F=Q$.

Theorem 1. *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 previous theorem:

Definition 1. *$P\subseteq\Sigma^*$ is a regular prefix language if and only if:*

1. *P is regular;*
2. *every prefix of a word of P is a word of P: $\forall u\in\Sigma^* \forall a\in\Sigma: ua\in P\Rightarrow u\in P$;*
3. *every word of P is a proper prefix of another word of P: $\forall u\in P\ \exists a\in\Sigma: ua\in P$.*

Definition 2. *A dfa A is a prefix automaton (or prefix dfa) if and only if*

1. *every state is final;*
2. *every state is alive: $\forall q\in Q, \exists a\in\Sigma: \delta(q, a)\in Q$.*

Proposition 1.

1. *If L is an ω-regular language, then Pref(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 $A=<Q, \Sigma, \delta, Q, q_0>$ is a prefix automaton, then the language L(M) recognized by the DB-machine $M=<Q, \Sigma, \delta, Q, q_0>$ is ω-regular and satisfies $L(A)=Pref(L(M))$.*

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

1) Let $P=Pref(L)$. L is ω-regular, so there are sequences of regular languages $<A_i>_{i\in[n]}$ and $<B_i>_{i\in[n]}$ such that $L=\bigcup_{i=1}^{i=n} A_i B_i^{\omega}$. $Pref(\bigcup_{i=1}^{i=n} A_i B_i^{\omega})=\bigcup_{i=1}^{i=n} Pref(A_i)\cup A_i B_i^* Pref(B_i)$ is a regular language which is closed by prefixes. Let $u\in P$. As $P=Pref(L)$, there exists $v\in\Sigma^{\omega}$ such that $uv\in L$. Let a be the first letter of v. Then ua is a prefix of uv, so $ua\in P$.

2) Let P be a regular prefix language. P is recognized by a *dfa A* which is minimal but not necessarily complete (*ie*, we remove its dead-state if necessary). As P is prefix, every state of this automaton is final. Finally, let q be a state of A and u a word such that $\delta(q_0, u)=q$. By the definition of a prefix language, there exists $a\in\Sigma$ such that $ua\in P$. So $\delta(q, a)\in Q$, thus q is alive.

3) Let $A=<Q, \Sigma, \delta, Q, q_0>$ be a prefix automaton. Consider the corresponding *DB*-machine $M=<Q, \Sigma, \delta, Q, q_0>$. Let us prove that $Pref(L(M))=L(A)$. Let $u \in Pref(L(M))$. Then there exists $w \in \Sigma^\omega$ such that $uw \in L(M)$. It is clear that $\delta(q_0, u) \in Q$, so $u \in L(A)$. Conversely, let $u \in L(A)$ and $q=\delta(q_0, u)$. As q is alive, we can build two words v and w such that $\delta(q, v)=q'$ and $\delta(q', w)=q'$. Clearly, the run C_{uvw^ω} goes infinitely often through state q'. So $uvw^\omega \in L(M)$ and $u \in Pref(L(M))$.

Proof of Theorem 1. Let L be a language recognized by a *DB*-machine $M=<Q, \Sigma, \delta, Q, q_0>$ and $w \in \Sigma^\omega$. Assume that every prefix u_n of w can be continued into a word of L recognized by M. The mapping C_w: $\mathbf{N} \rightarrow Q$ such that $C_w(0)=q_0$ and $\forall i \in \mathbf{N}$, $C_w(i+1)= \delta(C_w(i), u_i(i))= \delta(C_w(i), w(i))$ 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 1, $Pref(L)$ is a regular prefix language which is recognized by some prefix automaton $A=<Q, \Sigma, \delta, Q, q_0>$. We claim that L is recognized by the *DB*-machine $M=<Q, \Sigma, \delta, Q, q_0>$. Indeed, by Proposition 1, the language $L(M)$ satisfies $Pref(L(M))= L(A)$. Moreover, by the first part of this proof, $L(M)$ is a safe language (since M is a *DB*-machine). So L and $L(M)$ are both safe languages such that $Pref(L)=Pref(L(M))=L(A)$. Assume that there exists a word w in L and not in $L(M)$ (or *vice-versa*). As $Pref(L)=Pref(L(M))$, every prefix of w is in $Pref(L(M))$. Since $L(M)$ is a safe language, w itself is in $L(M)$, which is impossible. So $L=L(M)$.

Corollary 1. *Let L and L' be two safe ω-regular languages. Pref(L)=Pref(L')⇔L=L'* .

Proof. ⇐ is straightforward. ⇒ is an immediate consequence of the previous proof.

3 Learning ω-Regular Languages from their Prefixes

One of the main difficulties consists in explaining the meaning of "*p* is a positive prefix of the ω-language *L*" and "*n* 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.

Definition 3.

1. *p is an ∃-positive prefix of L iff* $\exists u \in \Sigma^\omega$, $pu \in L$
2. *p is a ∀-positive prefix of L iff* $\forall u \in \Sigma^\omega$, $pu \in L$
3. *n is an ∃-negative prefix of L iff* $\exists u \in \Sigma^\omega$, $nu \notin L$
4. *n is a ∀-negative prefix of L iff* $\forall u \in \Sigma^\omega$, $nu \notin L$

Given an ω-language L, let $P_\forall(L)$ denote the set of all ∀-positive prefixes of L, $P_\exists(L)$ the set of all ∃-positive prefixes of L, $N_\forall(L)$ the set of all ∀-negative prefixes of L, and $N_\exists(L)$ the set of all ∃-negative prefixes of L.

Two finite sets $S+$ and $S-$ of finite words form together a set of (*p, n)-examples* for an ω-language L if and only if $S+ \subseteq P_p(L)$ and $S- \subseteq N_n(L)$.

For instance, on the automaton 2a, $L=((a+b)^*a)^\omega$, $P_\forall(L)=N_\forall(L)=\varnothing$ and $P_\exists(L)=N_\exists(L)=\Sigma^*$.

We can also remark that for all ω-languages L, $P_\exists(L)=Pref(L)$ and

$P_\exists(L)\cap N_\forall(L)=P_\forall(L)\cap N_\exists(L)=\emptyset$ $\quad$ $P_\exists(L)\cup N_\forall(L)=P_\forall(L)\cup N_\exists(L)=\Sigma^*$

$P_\forall(L)=N_\forall(\Sigma^\omega\backslash L)$ $\quad$ $P_\exists(L)=N_\exists(\Sigma^\omega\backslash L)$

3.1 On Convergence Criteria

In this section, we adapt the definitions of Gold [5] and de la Higuera [7]. 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 [11].

It will be useful to systematically consider a class $\mathcal{L}$ of languages and an associated class $\mathcal{R}$ of representations. The latter one will have to be strong enough to represent the whole class of languages, *i.e.* $\forall L\in\mathcal{L}, \exists r\in\mathcal{R}: L(r)=L$.

The size of a representation (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 a constant size.

All the classes we consider are recursively enumerable. Moreover, for Büchi automata and finite words, given $x\in\{\exists, \forall\}$, the problems "$w\in P_x(L(A))$?" and "$w\in N_x(L(A))$?" are decidable, so the definition of identification in the limit from prefixes can be presented as follows:

Definition 4. *A class $\mathcal{L}$ of ω-languages is (p, n)-identifiable in the limit for a class $\mathcal{R}$ of representations if and only if there exists an algorithm A such that:*

1. *given a finite set $<S+, S->$ of prefixes, with $S+\subseteq P_p(L)$ and $S-\subseteq N_n(L)$, A returns h in $\mathcal{R}$ consistent with $<S+, S->$;*
2. *for all representations r of a language L in $\mathcal{L}$, there exists a finite characteristic set $<CS+, CS->$, such that, on $<S+, S->$ with $CS+\subseteq S+\subseteq P_p(L)$ and $CS-\subseteq S-\subseteq N_n(L)$, A returns a hypothesis h equivalent to r.*

We now adapt the definition of polynomial identification in the limit from fixed data [5, 7] 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 5. *A class $\mathcal{L}$ of ω-languages is (p, n)-polynomially identifiable in the limit from fixed finite prefixes for a class $\mathcal{R}$ of representations if and only if there exists an algorithm A and two polynomials $\alpha()$ and $\beta()$ such that:*

1. *given a set $<S+, S->$ of prefixes of size m[1], with $S+\subseteq P_p(L)$ and $S-\subseteq N_n(L)$, A returns h in $\mathcal{R}$ in $\mathcal{O}(\alpha(m))$ time and h is consistent with $<S+, S->$;*
2. *for all representations r of size n of a language L in $\mathcal{L}$, there exists a characteristic set $<CS+, CS->$ of size at most $\beta(n)$, such that, on $<S+, S->$ with $CS+\subseteq S+\subseteq P_p(L)$ and $CS-\subseteq S-\subseteq N_n(L)$, A returns a hypothesis h equivalent to r.*

[11] The size of a set S of finite words is the sum of the length of all the words in S.

3.2 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 $\mathcal{L}$ of ω-regular languages represented by $\mathcal{R}$ be learned following the criterion $\mathcal{C}$ from a set of (p, n)-examples?

The classes $\mathcal{L}$ we are interested in are those defined in section 2. The representation classes are *B*-Aut (Büchi automata) for $\mathbf{Reg}_\omega(\Sigma)$, *DB*-Aut (deterministic Büchi automata) for $\mathbf{Det}_\omega(\Sigma)$ and *DB*-Mach (*DB*-machines) for $\mathbf{Safe}_\omega(\Sigma)$. The criteria will be identification in the limit and polynomial identification in the limit from fixed prefixes. The examples of positive and negative prefixes will be defined according to the different combinations of the quantifiers ∃ and ∀.

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 $<\mathcal{L}_\mathcal{R}$, *criterion*, *interpretation*> where *criterion* will be *idlim* (identification in limit) or *polyid* (polynomial identification in the limit from fixed prefixes) and *interpretation* will be a pair (p, n) such that p and $n \in \{\exists, \forall\}$.

Example. The problem $<\mathbf{Safe}_\omega(\Sigma)_{DB\text{-Mach}}$, *idlim*, $(\exists, \forall)>$ is the one of identification in the limit of the class $\mathbf{Safe}_\omega(\Sigma)$ where the languages are represented by *DB*-machines and a presentation made of existential positive prefixes and universal negative prefixes (see definition 3) 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 $\mathbf{Reg}_\omega(\Sigma)$ and $\mathbf{Det}_\omega(\Sigma)$, for which identification in the limit from prefixes is impossible. The second concerns the class of safe languages, for which polynomial identification in the limit by fixed prefixes is proved.

4.1 General Properties

We first give a straightforward reduction property; we establish that polynomial identification only holds when identification in the limit also holds: if $<\mathcal{L}_\mathcal{R}$, *idlim*, *sign*> has a negative status, so does $<\mathcal{L}_\mathcal{R}$, *polyid*, *sign*>.

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 1. *Let $\mathcal{L}$ be a class of ω-languages and $\mathcal{R}$ a class of representations for $\mathcal{L}$. If there exist L_1 and L_2 in $\mathcal{L}$ such that $L_1 \neq L_2$, $P_p(L_1)=P_p(L_2)$ and $N_n(L_1)=N_n(L_2)$, then the problem $<\mathcal{L}_\mathcal{R}$, idlim, (p, n)> has a negative status.*

Proof. Suppose that an algorithm A identifies class $\mathcal{L}$; then L_1 and L_2 have respective characteristic sets 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.

Theorem 2. *For any class of representations $\mathcal{R}$, $\forall p$, $n \in \{\exists, \forall\}$, $<\mathbf{Reg}_{\omega}(\Sigma)_{\mathcal{R}}$, idlim, $(p, n)>$ and $<\mathbf{Det}_{\omega}(\Sigma)_{\mathcal{R}}$, idlim, $(p, n)>$ have negative status.*

Proof. We will use the same counter-example, shown in Figure 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 given prefixes). The languages accepted by automata 3a and 3b are respectively $L_1=a^{\omega}+a^*ba^*b(a+b)^{\omega}$ and $L_2=a^*ba^*b(a+b)^{\omega}$. Whatever the choice of quantifiers p and n, languages P_p and N_n are identical in both cases.

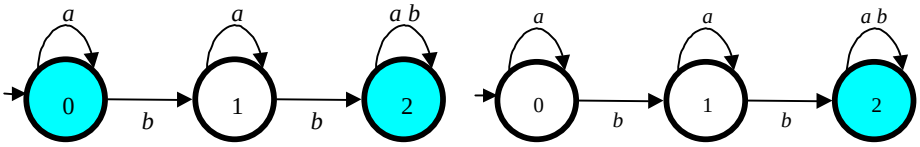

Fig. 3. Automata 3a and 3b accept respectively languages $a^{\omega}+a^*ba^*b(a+b)^{\omega}$ and $a^*ba^*b(a+b)^{\omega}$.

Formally:

$P_{\exists}(L_1)=P_{\exists}(L_2)=\Sigma^*$ $\quad$ $N_{\exists}(L_1)=N_{\exists}(L_2)=a^*+a^*ba^*$

$P_{\forall}(L_1)=P_{\forall}(L_2)=a^*ba^*b(a+b)^*$ $\quad$ $N_{\forall}(L_1)=N_{\forall}(L_2)=\varnothing$

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 will thus concentrate on a subclass of ω-deterministic languages in the sequel: the safe ω-languages. We first prove that the associated class of prefix languages is polynomially identifiable in the limit from given data:

Proposition 2. *The class of regular prefix languages, represented by prefix dfas, is polynomially identifiable in the limit from given prefixes.*

Proof. To prove the above proposition we use algorithm *RPNI*-prefixes1. An alternative and more efficient algorithm, that can return a compatible non trivial prefix automaton, even when the characteristic set is not included in the data is proposed in the appendix. As for *RPNI*-prefixes1, it makes use, as a sub-routine, of *RPNI* [10] which can identify a *dfa* from positive and negative data (typically two finite sets of finite words $S+$ and $S-$).

The first object *RPNI* builds is the prefix tree acceptor (*pta*): this is the largest *dfa* with no useless[2] states recognizing exactly $S+$.

---

[2] A state is useless if it does not lead to an accepting state, or is not accessible from the initial state.

Algorithm *RPNI*-prefixes1

```
Input: S=<S+,S-> (a set of positive words S+, and
of negative words S-)
Output: a prefix automaton (<Q, Σ, δ, F, q0>)
Begin
 A←RPNI(S+, S-);
 If A is a prefix dfa
    then return A
    else max_neg←max{length(u): u∈S-};
         For all w in S+ s.t. w∈Pref(S-) and
                   w∉(Pref(S+)\{w}) do
              Compute v of length max_neg s.t.
                  Pref(v)∩S-=∅ and w∈Pref(v);
              S+←S+∪{v};
         A←PTA(S+);
         Q←Q∪{qf}; F←Q;
         For all a in Σ do δ(qf, a)← qf;
         For all q in Q such that q is a leaf do
              For all a in Σ do δ(q, a)← qf;
         Return A
end.
```

If <*S*+, *S*-> contains a characteristic set of the target language *L*, *RPNI* returns a prefix automaton *A* that accepts language *L* [10]. If <*S*+, *S*-> does not contain a characteristic set, *RPNI* returns an automaton which is consistent with <*S*+, *S*->, but may be neither prefix nor even transformable into a prefix automaton. In that case *RPNI*-prefixes1 transforms the *pta* into a consistent prefix automaton.

Indeed function PTA(S+) constructs the *pta* corresponding to *S*+ in which are added extra words whose positive labeling does not introduce inconsistency; testing (w∈Pref(S-) and w∉Pref(S+)\{w}) allows to know which states of the *pta* have no successors; these states must then lead to a new universal[3] state q_f whenever the new transition is not used by some negative word: such a transition always exists since the data is supposed to be consistent. Building a polynomial implementation is straightforward.

Theorem 3. *<**Safe**$_{\wp(\Sigma)_{DB\text{-}Mach}}$, polyid, (∃, ∀)> a has positive status.*

Proof. We show that the conditions of definition 5 are met:

i. Let *L* be a safe language. On any pair of sets <*S*+, *S*-> of (∃, ∀)-prefixes for *L*, by proposition 2 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 $S^+ \subseteq Pref(L(M))$ and $S^- \cap Pref(L(M)) = \emptyset$.

ii. Let *L* be a safe language, and *M* a *DB*-machine accepting *L*. Let *A* be the prefix automaton associated with *M*. Let <*CS*+, *CS*-> be a characteristic set for *A* and *RPNI*.

[3] A state is universal if by any letter there is a transition to the same state.

Let now $<S+, S->$ be such that $CS+ \subseteq S+$, $CS- \subseteq S-$, $S+ \subseteq L(A)$ and $S- \cap L(A) = \emptyset$. Notice that the size of $<CS+, CS->$ is polynomial in that of A which in turn is the same as the size of M. On input $<S+, S->$ *RPNI* returns an automaton A' equivalent to A. By construction, the *DB*-machine M' associated to A' is such that $Pref(L(M'))=L(A')=L(A)=Pref(L(M))$. By corollary 1 $L(M)=L(M')$ holds.

Theorem 4. *If $\mathcal{L}$ strictly contains $\mathbf{Safe}_{\omega}(\Sigma)$ and $\mathcal{K}$ is a class of machines for $\mathcal{L}$, $<\mathcal{L}_{\mathcal{K}}$, idlim, $(\exists,\forall)>$ has a negative status.*

Proof. Let $\mathcal{L}$ be a class containing strictly $\mathbf{Safe}_{\omega}(\Sigma)$ and L a language in $\mathcal{L}$ but not in $\mathbf{Safe}_{\omega}(\Sigma)$. $P_{\exists}(L)$ is a prefix language. But in that case there exists a language L' in $\mathbf{Safe}_{\omega}(\Sigma)$ such that $P_{\exists}(L)=P_{\exists}(L')$ and $N_{\forall}(L)=N_{\forall}(L')$. By lemma 1, it follows that $\mathcal{L}$ is not identifiable.

Theorems 3 and 4 allow us to deduct a final result concerning learning from $(\forall, \exists)$-prefixes. An ω-language L is co-safe *iff* its complementary $\Sigma^{\omega} \setminus L$ is a safe language. We denote $\mathbf{Co\text{-}Safe}_{\omega}(\Sigma)$ 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 5. *$<\mathbf{Co\text{-}Safe}_{\omega}(\Sigma)_{co\text{-}DB\text{-}Mach}$, polyid, $(\forall, \exists)>$ has a positive status. Furthermore for any class $\mathcal{L}$ strictly containing $\mathbf{Co\text{-}Safe}_{\omega}(\Sigma)$, and $\mathcal{K}$ a class of machines for $\mathcal{L}$, $<\mathcal{L}_{\mathcal{K}}$, idlim, $(\forall, \exists)>$ has 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 $\Sigma^{\omega} \setminus L$, since $P_{\exists}(\Sigma^{\omega} \setminus L)=N_{\exists}(L)$ and $N_{\forall}(\Sigma^{\omega} \setminus L)=P_{\forall}(L)$. 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 3, the problem $<\mathbf{Safe}_{\omega}(\Sigma)_{DB\text{-}Mach}$, *polyid*, $(\exists, \forall)>$ has a positive status, and so has $<\mathbf{Co\text{-}Safe}_{\omega}(\Sigma)_{\omega\text{-}DB\text{-}Mach}$, *polyid*, $(\forall, \exists)>$.

5 Conclusion

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

The problem $<?_{\mathcal{L}}$, *criterion*, $(\exists, \exists)>$. It is rather easy to show that for all the classes of languages studied in this paper, the status will be negative. It seems relevant to find a class of languages (undoubtedly rather restricted) for which the status would be positive.

Learning from prefix queries (membership queries on the prefixes) and equivalence queries.

Improvement of the inference algorithm (*RPNI*-prefixes) for the learning of the prefix languages. The algorithm proposed is polynomial. It is however neither easy to implement, nor (probably) does it perform well in practice.

Lastly, the validation of this algorithm on real data (produced by a system), remains to be done. The type of automata corresponding to real world tasks has the characteristic to have an important alphabet, but few outgoing transitions per state. In this context simplification by typing of the alphabet [3] is undoubtedly a track to be retained.

6 Acknowledgement

The authors would like to thank Maurice Nivat who suggested the problem.

References

1. B. Alpern, A.J. Demers and F.B. Schneider. Defining Liveness. *Information Processing Letters* 21, 181-185, 1985.
2. D. Angluin. On the Complexity of Minimum Inference of Regular Sets. *Information and Control* 39, 337–350, 1978.
3. M. Bernard and C. de la Higuera. Apprentissage de Programmes Logiques par Inférence Grammaticale, *Revue d'Intelligence Artificielle*, 14/3-4, 375-396, 2001.
4. J.R. Büchi. On a decision method in restricted second order arithmetic. *Proc. Cong. Logic Method and Philos. Of Sci.*, Stanford Univ. Press, California, 1960.
5. M.E. Gold. Complexity of Automaton Identification from Given Data, *Information and Control*, 37, 302-320, 1978.
6. C. de la Higuera, J. Oncina and E. Vidal. Identification of dfa's:data dependant vs data-independant algorithms,.*in Proceedings of ICGI* '96, LNAI 1147, Springer-Verlag, 1996.
7. C. de la Higuera. Characteristic Sets for Polynomial Grammatical Inference, *Machine Learning* 27, 125-138, 1997.
8. K. Lang, B.A. Pearlmutter and R.A. Price. Results of the Abbadingo One DFA Learning Competition and a New Evidence-Driven State Merging Algorithm, in Grammatical Inference, *Proceedings of ICGI '98*, LNAI 1433, Springer Verlag, 1-12, 1998.
9. O. Maler and A. Pnueli. On the Learnability of Infinitary Regular Sets, *Proc. 4th COLT*, 128-136, Morgan Kauffman, San Mateo, 1991.
10. J. Oncina and P. Garcia. Identifying Regular Languages in Polynomial Time, in *Advances in Structural and Syntactic Pattern Recognition*, H. Bunke ed., Series in Machine Perception and Artificial Intelligence 5, 99-108, 1992.
11. R.J. Parekh and V. Honavar. On the relationship between Models for Learning in Helpful Environments, in *Proceedings of ICGI 2000*, LNAI 1891, Springer Verlag, 207-220, 2000.
12. A. Saoudi and T. Yokomori. Learning Local and Recognizable ω-languages and Monadic Logic Programs, in *Proceedings of EUROCOLT*, LNCS, Springer Verlag, 1993.
13. W. Thomas. Automata on infinite objects, *Handbook of Theoretical Computer Science* (Van Leewen ed.), 133-191, North-Holland, Amsterdam, 1990.
14. M.Y. Vardi and P. Wolper. Automata-Theoretic Techniques in Modal Logics of Programs, Journal of Computer and Systems Science 32, 183-221, 1986.

Appendix: 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 set 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* [10], and uses notations from [6].

Algorithm *RPNI*-prefixes2 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 will be completable into a prefix automaton. To do this each positive state has to stay alive: there must be at least one infinite word leading from this state that avoids every negative state.

```
Algorithm RPNI-prefixes2
Input: S=<S+, S->
Output: a prefix automaton (defined by δ, F+, F-)
Begin
  (*Initializations*)
  S+←S+∪Pref(S+); n←0;
  ∀a∈Σ, Tested(q0, a)←∅; F+←{q0}; F-←∅;
  While there are some unmarked words in S+∪S- do
    <q, a, q'>←chose_transition();
    If Possible(δ(q, a)=q')
         then δ(q, a)←q';
              For all unmarked w in S+ do
                   If δ(q0, w)=q" then mark (w);
                                        F+←F+∪{q"};
              For all unmarked w in S- do
                   If δ(q0, w)=q" then mark (w);
                                        F-←F-∪{q"};
         else Tested(q, a)←Tested(q, a) ∪{q'};
    If |Tested(q, a)|=n+1 (*impossible to merge *)
         then (*creation of a new state*)
              n←n+1; Q←Q∪{qn}; δ(q, a)← qn;
              For all unmarked w in S+ do
                   If δ(q0, w)=q" then mark(w);
                                        F+←F+∪{q"};
              For all unmarked w in S- do
                   If δ(q0, w)=q" then mark(w);
                                        F-←F-∪{q"};
    ∀a∈Σ, Tested(qn, a)←∅;
End_while;
(* conversion into a consistent prefix dfa*)
Q←F+;
```

```
For all q∈F+ such that ∀a∈Σ δ(q, a)∉Q
        chose w minimal such that
              ({u: δ(q0, u)=q}.Pref(w))∩S- = ∅;
        Q←Q∪{q_i^w: 0<i<|w|};
        F+←F+∪{q_i^w: 0<i<|w|};
        δ(q, w(0))←q_1^w;
        For all i from 0 to |w| do
                                  δ(q_{i-1}^w, w(i))←q_i^w;
        For all a in Σ do δ(q_{|w|}^w, a)←q_{|w|}^w;
End.
```

`Function Chose_transition`: returns a triplet <*q*, *a*, *q'*> corresponding to the transition $\delta(q, a)=q'$ where $\delta(q, a)$ is undefined and $q' \notin$ `Tested`(q, a). Different functions can work. Typically EDSM type functions have been shown preferable [8].

`Function Possible(δ(q, a)=q'):` returns `True` if adding to δ rule (q, a, q') does not lead to an inconsistency, `False` otherwise.

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

- there exists two words *uaw* and *vw* such that $\delta(q_0, u)=q$ and $\delta(q_0, v)=q'$ and $uaw \in S+$, $vw \notin S-$, and $uaw \notin S-$, $vw \in S+$.
- a state is no more alive; a state *q* is alive if it can still lead to an accepting state: $\exists w \in \Sigma^{\omega} / (\{u: \delta(q_0, u)=q\}.Pref(w)) \cap S- = \emptyset$. This insures 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 *RPNI*-prefixes2 are:

- The algorithm returns a prefix automaton (by construction).
- The `possible` test insures 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 set (for *RPNI*) is included, no transformation will take place.
- Finally, the algorithm works in polynomial time.

We refer the reader to [6] for a complete proof (in the case of *dfa*s, but the proof can easily be adapted to the case of prefix automata).