[9] Knoerr, A. *Globals Models of Natural Boundaries: Theory and Applications.* Report in Pattern Theory 148. Brown University. (1988).

[10] Mardia,K.,V., Hainsworth,T.J. . A spatial thresholding method for image segmentation. *IEEE Trans. on PAMI.* **vol 10**,(1988), 919-927.

[11] Ripley,B.D. Recognizing organisms from their shape - a case study in image analysis. *Proc. XVth International Biometrics Conference*, Budapest, (1990), 259-263.

[12] Ripley,B.D. Classification and clustering in spatial and image data. 15 Jahrestagung von Gesellschaft fur Klassifikation, Salzburg. (1991).

[13] Serra, J. *Image Analysis and Mathematical Morphology.* (Academic Press, London, 1982).

[14] Su Bu-Qing, Liu Ding-Yuan *Computational Geometry. Curve and Surface Modeling.* (Academic Press, 1989).

INFERRING REGULAR LANGUAGES IN POLYNOMIAL UPDATED TIME

J. Oncina and P. García

Departamento Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Valencia, Spain

ABSTRACT*

An algorithm is described such that, given a set of positive data and a set of negative data of an unknown regular language, obtains a Deterministic Finite Automaton consistent with the data. The update time of this algorithm is $O(p^3n)$ where p is the sum of the lengths of all the strings constituting the positive data and n is the sum of the lengths of those corresponding to the negative data. Moreover, this algorithm identifies any regular language in the limit.

1. Introduction

The inductive inference paradigm[1] is the basis of an approach to the automatic learning problem. In this framework there are a lot of theoretical results that tells us what is learnable by means of algorithmic strategies, and give us the general properties of the inference algorithms[2]. On the other hand, in connection with the Syntactical Pattern Recognition framework, many grammatical inference algorithms exist that can be used in the learning phase of pattern recognition tasks[3,4,5,6].

If we revise the (regular grammar) inference algorithms proposed in the last twenty years, we can immediately observe that only positive data is taken into account. This is

* Work partially supported by the Spanish CICYT under grant TIC-0448/89

rather a paradoxical situation. It is well known that the class of regular languages can not be correctly identified from only positive presentation; but any recursively enumerable class of languages is identifiable using a complete presentation (with positive and negative data)[7]. What is the reason for not making use of the negative information? Very often, when a new method is proposed, there is no explanation of the reason for this limitation and when this explanation exists, it is usually argued that the use of this information constitues an intactable problem. To support this last argument it is usual to resort to the Gold result establishingt hat the problem of finding a DFA with a minimum number of states and compatible with a complete presentation is NP-Hard[7]. The origin of this somewhat misleading relation between the minimum compatible DFA problem and the regular language inference problem is, however, rather clear. In fact, any algorithm that would construct a DFA with a minimum number of states compatible with all the data already processed, has the property of identifying any regular language in the limit . From the Gold result it follows that an inference method like this can not be efficient. However, using this result to conclude the intractability of the regular language inference problem is erroneous. This conclusion is particularly surprising if we observe that in the very same paper in wich the probable intractability of the minimum compatible DFA problem is established[4], Gold proposes a polynomial algorithm (that doesn't find a minimum DFA for each finite set of data) that allows us to identify any regular language in the limit.

In this paper we propose a new algorithm that can identify the class of regular languages from a complete presentation. Given an arbitrary sample, this method obtains a (non necessarily minimum) DFA compatible with the sample. Moreover, it this sample includes a "characteristic set", the algorithm is shown to supply the minimum DFA compatible with the sample. The proposed method is based on state clustering. It is well known that if S_+ is a "structurally complete" sample of a regular language L (all the transitions of the unknown source automaton $A(L)$ are used for the acceptation of the strings in S_+) there exists a partition π on the state set of the prefix tree acceptor of S_+, $PT(S_+)$, such that $PT(S_+)/\pi$ is isomorphic to $A(L)$. If we can take advantage of the words in Σ^*-L to reject some partitions, then the inference problem reduces to a guided search in the lattice of all the possible partitions. Unfortunately the search space grows exponentially with the size of the state set in $PT(S_+)$ and then with the size of S_+.

Instad of attempting exhaustive search, the proposed algorithm tries to merge pairs of states in $PT(S_+)$ according to an specific order and only does it if the automaton that the result rejects all the negative data. The algorithm obtains in polynomial time with the size of the sample, a DFA compatible with this sample, and when the sample is large enough the obtained automaton is the minimum DFA for the unknown language.

This algorithm, like the one proposed by Gold[4] has the disadvantage of not being incremental. However, while the Gold's algorithm has the additionaldisadvantage that it doesn' t generalize unless the sample has a characteristic set. On the other hand our algorithm is free of this inconvenience and then it is better for use in learning tasks.

2. Basic Concepts And Notation

Let Σ be a fixed finite alphabet of symbols. The set of all finite strings of symbols from Σ is denoted by Σ^*. The length of a string u is denoted by $|u|$. The unique string of length zero is denoted by λ. The concatenation of two strings u, w is denoted by uv. If a string $u = vw$ we say that v (w) is a prefix (suffix) of u. A language is any subset of Σ^*.

If L is a language over Σ, we define the set of prefixes over L as:

$$Pr(L) = \{ u \in \Sigma^* \mid \exists v \in \Sigma^*, uv \in L \}$$

and the set of tails of u in L as:

$$T_L(u) = \{ v \in \Sigma^* \mid uv \in L \}$$

A finite automaton (FA) A is defined by a five-tuple (Q,Σ,δ,q_0,F) where Q is a finite set of states, q_0 is the initial state, $F \subseteq Q$ is the set of final states and $\delta: Q\times\Sigma \to 2^Q$ is the transition function. A is deterministic if for all $q \in Q$ and for all $a \in \Sigma$, $\delta(q,a)$ has at most one element. The language accepted by A is denoted by $L(A)$. A language is regular iff it is accepted by a FA. We say that q is an a-successor of p If $p \in \delta(q,a)$.

If $A = (Q,\Sigma,\delta,q_0,F)$ is a FA and π is a partition of Q, we denote by $B(q,\pi)$ the only block that contains q and we denote the quotient set $\{ B(q,\pi) \mid q \in Q \}$ as Q/π. Given a FA A and a partition π over Q, we define the quotient automaton A/π as:

$$A/\pi = (Q/\pi,\Sigma,\delta',B(q_0,\pi),\{B \in Q/\pi \mid B \cap F \neq \emptyset\})$$

where δ' is defined as:

$$\forall B,B' \in Q/\pi,\ \forall a \in \Sigma,\ B' \in \delta'(B,a) \text{ if } \exists q,q' \in Q,\ q \in B,\ q' \in B' : q' \in \delta(q,a).$$

Given A and π over Q, it is easyly seen that $L(A) \subseteq L(A/\pi)$.

Given $A = (Q,\Sigma,\delta,q_0,F)$, $L(A)=L$, the partition π_L defined as $B(q,\pi_L)=B(q',\pi_L)$ iff $\forall x \in \Sigma^*\ \delta(q,x) \cap F \neq \emptyset$ iff $\delta(q',x) \cap F \neq \emptyset$ produces a A/π_L that is the AFD that has the minimum number of states and accepts $L(A)$; this automaton is called the canonical automaton of the language L and we denote it by $A(L)$.

Given L, we can also define the canonical automaton $A(L) = (Q,\Sigma,\delta,q_0,F)$ as:

$$Q = \{ T_L(u) \mid u \in Pr(L) \};\ q_0 = T_L(\lambda);\ F = \{ T_L(u) \mid u \in L \};$$

$$\delta(T_L(u),a) = T_L(ua) \text{ where } u,ua \in Pr(L)$$

A Sample S of a language L is a finite set of words that we can represent as $S=(S_+,S_-)$ where S_+ is a subset of L (positive sample) and S_- is included in the complementary language of L (Negative Sample).

Let S_+ be the positive sample from a regular language L, we say that S_+ is structurally complete if all the transitions of $A(L)$ are used in the acceptance of the strings in S_+.

Let S_+ be a positive sample from a regular language L, we can define the prefix tree acceptor of S_+ as $PT(S_+) = (Pr(S_+),\Sigma,,\delta,\lambda,S_+)$ where δ is defined as: $\delta(u,a) = ua$ where $u, ua \in Pr(S_+)$. This automaton only accepts the strings belonging to S_+.

It is well known that if S_+ is a structurally complete sample of a regular language L, then there exists a partition π on the states of $PT(S_+)$ such that $PT(S_+)/\pi$ is the $A(L)$.

3. Inference Algorithm

We denote as the lexicographic order in Σ^* as "<". Given a positive sample S_+ and a partition π over the set of all the prefixes of all the strings of the sample, the order "<" allows to define an order between the blocks of the partition.

Definition 3.1

Let S_+ be a positive sample, let π be a partition over $Pr(S_+)$ and let B_i,B_j be two blocks of π. We say that $B_i < B_j$ iff some $u \in B_i : \forall v \in B_j,\ u < v$ exists.

3.1 Operation Joint

Given a partition π over $Pr(S_+)$ and given $B_i,B_j \in \pi$ we define the set $J(\pi,B_i,B_j)$ as:

$$J(\pi,B_i,B_j) = \{ B \in \pi \mid B \neq B_i, B \neq B_j \} \cup \{ B_i \cup B_j \}$$

In the rest of the paper we assume that the subindex of a block is the same as the subindex of the smallest string belonging to the block. If a block has only one string we will represent it indistinctly as a block or as a string.

3.2 The Inference Algorithm

Given a sample $S = (S_+,S_-)$ of an unknown regular language L, this algorithm produces an automaton $PT(S_+)/\pi_r$ where π_r can be recursively obtained as follows:

$\pi_0 = Pr(S_+) = \{ u_0, .. , u_r \}$	(we suppose that the the prefixes are indexed in lexicographical order, therefore $u_0=\lambda$)
$\pi_n = J(\pi_{n-1},B,u_n)$	if $\exists B, B' \in \pi_{n-1}$ such that both B and u_n are a-successors of B' and $B<u_n$ and is the first a-successor of B' that fulfill $S_- \cap L(A_0/J(\pi_{n-1},B,u_n)) = \varnothing$.
$\pi_n = J(\pi_{n-1},B,u_n)$	if $B < u_n$ and is the first block in π_{n-1} such that $S_- \cap L(A_0/J(\pi_{n-1},B,u_n))=\varnothing$.
$\pi_n = \pi_{n-1}$	otherwise

3.3 Example

Let $S_+ = \{ 111, 000, 11101, 01 \}$ be a positive sample and let $S_- = \{ 0, 1, 00, 11 \}$ be a negative sample. In this example we will show the automaton A_0/π_n for each step of the above algorithm.

The set of the prefixes of S_+ in lexicographical order is:

$$< \lambda, 0, 1, 00, 01, 11, 000, 111, 1110, 11101 >$$

The first step is to build $A_0=PT(S_+)$. Therefore the first automaton is the automaton of the Figure 3.1.

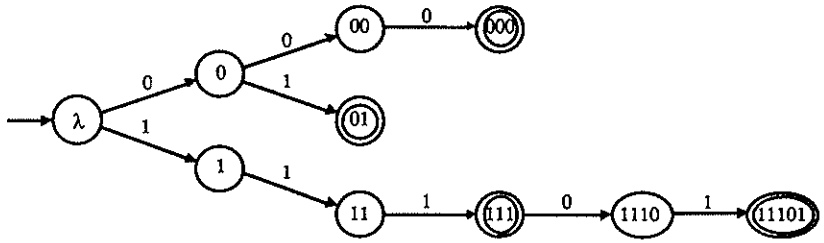


Fig.3.1. Initial Automaton: A_0/π_0

Obtaining π_1, $(u_1 = 0)$:

As λ has not any 0-successor smaller than 0,then we try to merge the state 0 with the state λ; in other words, we perform the operation $J(\pi_0,\lambda,0)$, and we show the automaton

$A_0/J(\pi_0,\lambda,0)$ shown in Figure 3.2. and we can see that this automaton doesn't fulfill $L(A_0/J(\pi_0,\lambda,0)) \cap S_- = \varnothing$ ($1 \in S_-$ is accepted by the automaton) .

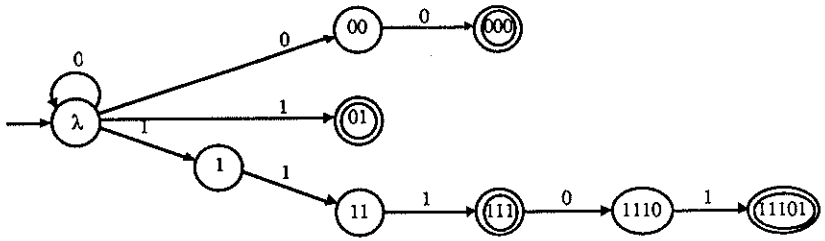


Fig.3.2. $A_0/J(\pi_0,\lambda,0)$ (not compatible with the sample)

Now there are no more states to try to merge with u_1 (= 0), and we have that $\pi_1 = \pi_0$.

Obtaining π_2, ($u_2 = 1$)

The algorithm obtains $A_0/J(\pi_1,\lambda,1)$ and accepts $11 \in S_-$. Next we try with $A_0/J(\pi_1,0,1)$ that also accepts 11. Then we have $\pi_2 = \pi_1$.

Obtaining A_3, ($u_3 = 00$)

$A_0/J(\pi_2,\lambda,00)$ accepts $0 \in S_-$

$A_0/J(\pi_2,0,00)$ accepts $00 \in S_-$

$A_0/J(\pi_2,1,00)$, shown in the figure Figure 3.3, rejects all the negative data. Thus $\pi_3 = J(\pi_2,1,00)$

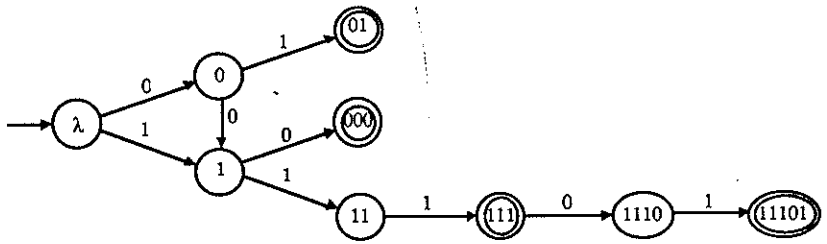


Fig.3.3. $A_0/J(\pi_2,1,00)$

The corresponding partition is:

$$\pi_3 = \{ \lambda, 0, \{ 1, 00 \}, 01, 11, 111, 1110, 11101 \}$$

The next automata for A_0/π_n, $n=4,...,9$ are shown in figures 3.4 to 3.9. The final automaton, A_0/π_9, recognizes the language over the alphabet $\{0,1\}$ of all the words where the difference between the number of 0's and the number of 1's is a multiple of 3.

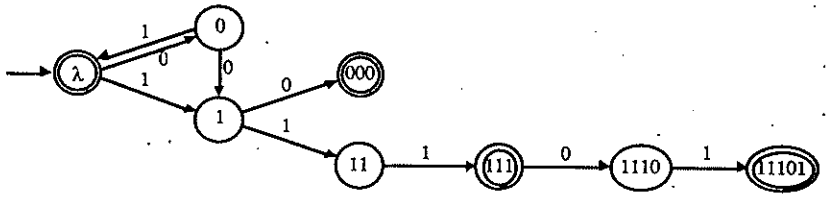


Fig.3.4. $A_0/J(\pi_3,\lambda,01)$

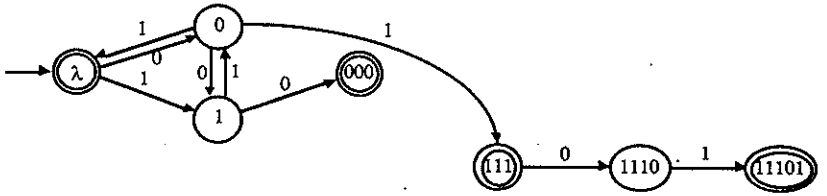


Fig.3.5. $A_0/J(\pi_4,0,11)$

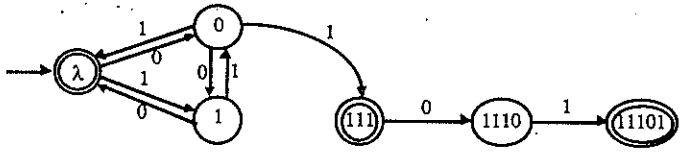


Fig.3.6. $A_0/J(\pi_5,e,000)$

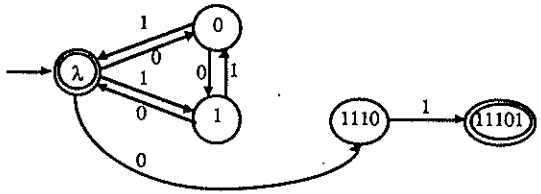


Fig.3.7. $A_0/J(\pi_6,e,111)$

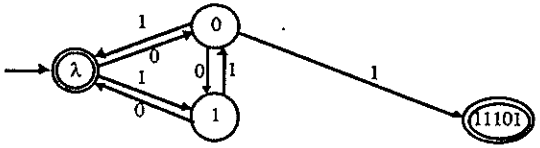


Fig.3.8. $A_0/J(\pi_7,0,1110)$

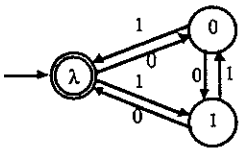


Fig.3.9. $A_0/J(\pi_8,e,11101)$

4. Identification In The Limit

The concept of identification in the limit was introduced by Gold[1]. In this model, an algorithm $\mathcal{A}$ reads strings of a presentation and for each readIn string it conjectures a DFA as the solution. We say that the algorithm $\mathcal{A}$ identifies the DFA A in the limit iff it for any presentation of $L(A)$ the infinite sequence of automata conjetured by $\mathcal{A}$ converges to a DFA that accepts the same language as the original A. It should be noted that the convergence point will depend on the presentation.

To show that the here proposed inference algorithm effectively identifies any regular language in the limit, we will see that for all possible DFA a finite sample (that grows polynomically with the size of the automaton) exists which, if it is included in the set of strings reads by the automaton , then the algorithm produces this DFA. Correspondingly, as for as a sample can be arbitrarily large we can guarantee that this sample is included in the sample that the algorithm reads.

4.1 Preliminary definitions

Short prefix (*Sp*):

$$Sp(L) = \{ w \mid w \in Pr(L) \; y \; \neg\exists v: T_L(w) = T_L(v) \text{ and } v < w \}$$

We can observe that for a given language L there must be as many *short prefixes* as states in $A(L)$.

Nucleus:

$$N(L) = \{ ua \mid ua \in Pr(L) \text{ and } u \in Sp(L) \} \cup \{ \lambda \}$$

It is clear that if $ua \in N(L)$ then $u \in Sp(L)$. If we call n the number of states in $A(L)$, it is clear that, in the worst case, the number of strings in $N(L)$ is $n|\Sigma| + 1$.

Complete sample:

We say that a sample $S = (S_+,S_-)$ is complete if it fulfils the following two properties:

$\forall u \in N(L)$, $\exists v: uv \in S_+$; and if $u \in L$, then $v=\lambda$.

$\forall u \in Sp(L)$, $\forall v \in N(L)$: $\delta(q_0,u) \neq \delta(q_0,v)$ either *$\exists uu' \in S_+ : \exists vu' \in S_-$* or *$\exists vv' \in S_+ : \exists uv' \in S_-$*

The first rule is to guarantee that S_+ is a structurally complete sample. With the second rule we can assure that, if we try to merge two disctint states (respect to $A(L)$)belonging to $N(L)$, then a negative string exist that does not allow us do it. We can observe that in the worst case the number of samples that we need is:

$$|S_+| \text{ and } |S_-| < n^2 . |\Sigma|$$

Lemma 4.1

Let L be a regular language, let $S=(S_+,S_-)$ be a complete sample of L, let π a partition in $Sp(S_+)$, let $u \in N(L)$ and $v \in Sp(L)$ and suppose that $u,v \in B \in \pi$. Then

$$L(A_0/\pi) \cap S_- = \emptyset \implies T_L(u) = T_L(v).$$

Proof:

suppose that this is not true, then *$\exists u,v$: $u \in N(L)$, $v \in Sp(L)$: $L(A_0/\pi) \cap S_- = \emptyset$* and $T_L(u) \neq T_L(v)$. Then from the construction of S_+ and S_- it is clear that either *$\exists vv' \in S_+$: $uv' \in S_-$* or *$\exists uu' \in S_+$: $vu' \in S_-$*. If we suppose that the first is true (we can show the other in a similar way), and becouse both u and v belong to the same block of the partition, then in A_0/π all the tails for v must be tails for u, then as $vv' \in L(A_0/\pi)$, uv' must also belong to $L(A_0/\pi)$ but also $uv' \in S_-$, and this is in contradiction with $L(A_0/\pi) \cap S_- = \emptyset$

❑

Lemma 4.2

Let L be a regular language, let $S = (S_+, S_-)$ be a complete sample of L, let π a partition over $Pr(S_+)$ and let u,v be two strings such that $T_L(u) = T_L(v)$. Then

$$S_- \cap L(A_0/\pi) = \emptyset \implies S_- \cap L(A_0/J(\pi,u,v)) = \emptyset.$$

Proof:

Suppose that this is not true. Then *$\exists u,v$:* $T_L(u)=T_L(v)$, $S_- \cap L(A_0/\pi) = \emptyset$ but $S_- \cap L(A_0/J(\pi,u,v)) \neq \emptyset$. Let $x \in S_- \cap L(A_0/J(\pi,u,v))$, it is obvious that $x \in L(A_0/J(\pi,u,v)) - L(A_0/\pi)$. Then x can be written as $x=x_1x_2$ where either $u \in \delta(e,x_1)$ and $\delta(v,x_2) \cap F \neq \emptyset$ or $v \in \delta(e,x_1)$ and $\delta(u,x_2) \cap F \neq \emptyset$, because otherwise $x \in L(A_0/\pi)$. But since $T_L(u)=T_L(v)$ then $x \in L$ and this is contradictory with $x \in S_-$.

❑

Let π_i be the partition over $Pr(S_+)$ that is produced by the algorithm in the iteration i. In this partition the meger of states of A_0 only have affected the states $\{u_0,..,u_i\}$ (for the states larger than u_i, the automaon maintain with the tree shape of A_0). If $A_0/\pi_i=(Q_i,S,d_i,I,F_i)$ let $A'_i=(Q'_i,S,d'_i,I,F'_i)$ be the subacceptor of A_0/π_i induced for the

elimination of the states $\{u_{i+1},...,u_r\}$. We will see that if S is a complete sample, in each step of the algorithm we obtain a quotient automaton A_0/π_i) such as A'_i is isomorphic to a subacceptor of $A(L)$.

Theorem 4.1

Let S be a complete sample of a regular language L, let $A(L)=(\Sigma,Q,q_0,F,\delta)$ and let $A_0/\pi_i=(\Sigma,Q_i,q_0,F_i,\delta_i)$ then $Q'_i\subseteq Q$, $F'_i\subseteq F$, $\delta'_i\subseteq\delta$.

Proof: (By induction)

If $i=0$ this is obvious, $A_0/\pi_0=PT(S_+)$ then:

$Q'_0=\{\lambda\}$ because $u_0=\lambda$

$$F_0=\begin{cases}\{\lambda\} \text{ if } \lambda\in S_+\\ \varnothing \text{ otherwise}\end{cases}$$

$\delta'_i=\varnothing$

We suppose the theorem holds for $i-1$ and we will prove it will also hold for i. We will going to distinguish three cases, according to the three possibilities of the algorithm.

1) suppose that $\exists B,B'\in\pi_{i-1}$: $B<u_i$ and both B and u_i are a-successors of B' and also B is the first a-successor of B' that fullfil $S_-\cap L(A_0/J(\pi_{i-1},B,u_i))=\varnothing$

(the situation is represented in the figure 4.1)

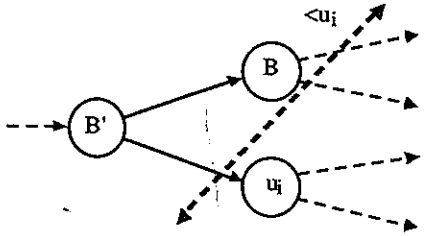


Fig. 4.1: Case 1)

Since B and u_i are a-successors of B' it is clear that, in $A(L)$, $T_L(u_i)=T_L(B)$ and since $S_-\cap L(A_0/\pi_{i-1})=\varnothing$ then from the Lemma 4.2 it is always true that $S_-\cap L(A_0/J(\pi_{i-1},B,u_i))=\varnothing$, then when we construct $\pi_i=J(\pi_{i-1},B,u_i)$ it is true that $u_i\in B$ and then $Q'_i=Q'_{i-1}\subseteq Q$. It is also clear that if $u_i\in F_i$ by construction of S_+ a string v in B must exist such that $v<u_i$ and $v\in F_i$, then $F'_i=F'_{i-1}\subseteq F$. And since both the transition (B',a,u_i) and (B',a,B) talso exist then $\delta'_i=\delta'_{i-1}\subseteq\delta$.

Note that, if $u_i\notin N(L)$ and if $B'<u_i$ is such that (B',a,u_i), then since $B'<u_i$, B' must belong to $Sp(L)$ and as since the positive sample we have for all the strings $xa\in N(L)$ a string xav, in particular we have a string such that $x=B'$ and since $xa\in L(N)$ then the transition (B',a,xa) must exist and it also is obvious that $xa<u_i$ and then xa must belong to a block $B<u_i$ and now we have the same conditions as at the beginning of this point. Then for the next points we can suppose that $u_i\in N(L)$.

2) Suppose now that the first condition is not true and the second condition holds, then: $\exists B<u_i$ and B is the first block of π_{i-1} that fullfil $S_-\cap L(A_0/J(\pi_{i-1},B,u_n))=\varnothing$. (the situation is represented in the figure 4.2)

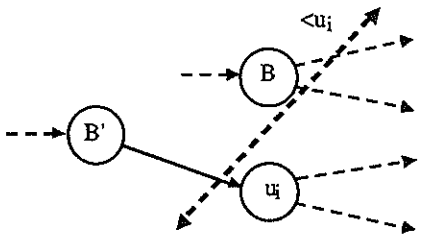


Fig 4.2: Case 2)

Since the first condition is not true u_i must belong to $N(L)$ and also if we call B' as the predecessor of u_i then u_i is the first a-successor of B'. Since $B<u_i$ then $B\in Pc(L)$ and since $u_i\in N(L)$ then from the Lemma 1 we have that $T_L(B)=T_L(u_i)$ and when we construct π_i it is clear that $u_i\in B$ and then $Q'_i=Q'_{i-1}\subseteq Q$ it is also clear that if $u_i\in F_i$, by the construction of S_+ a string v in B must exist such that $v<u_i$ and $v\in F_i$, then $F'_i=F'_{i-1}\subseteq F$. And since $\delta'_i=\delta'_{i-1}\cup\{(B',a,B)\}$ and as this transition comes from (B',a,u_i) and $T_L(u_i)=T_L(B)$ it is clear that $(B',a,B)\in\delta$ and then $\delta'_i\subseteq\delta$.

3) Suppose now that $\neg\exists B<u_i$ that satisfy $S_-\cap L(A_0/J(\pi_{n-1},B,u_n))=\varnothing$ using Lemma 4.2 this is equivalent to $\neg\exists B<u_i$: $T(B)=T(u_i)$ then we have that $u_i\in Sp(L)$ and then and then $u_i\in Q$ and then $Q'_i=Q'_{i-1}\cup\{u_i\}\subseteq Q$. Moreover for the same reason it is also true that $F'_i\subseteq F$ and for ending, if B' is the predecessor of u_i and as $B'\in Sp(L)$ then it is clear that $(B',a,u_i)\in\delta$ and then $\delta'_i=\delta_{i-1}\cup\{(B',a,u_i)\}\subseteq\delta$.

❑

Theorem 4.2

This proposed algorithm identifies in the limit the class of regular languages.

Proof:

Since we are studying the inference in the limit we can suppose that the sample is complete. Let L be the regular language we are searching for. We call $A(L)=(\Sigma,Q,e,F,\delta)$, and let $A_r=(\Sigma,Q_r,e,F_r,\delta_r)$ be the automaton obtained from the algorithm. We must show that $Q_r=Q$, $F_r=F$ and $\delta_r=\delta$.

It is clear from Theorem 4.1 that $Q_r \subseteq Q$, $F_r \subseteq F$ and $\delta_r \subseteq \delta$. We go on to show that $Q \subseteq Q_r$. Suppose it is false; then $x \in Sp(L)$ and $x \in Q-Q_r$ must exist from the construction of S_+ then a block B in Q_r must exist such that $x \in B$ it is clear that $B \in Pc(L)$ and since $L(A_0/\pi_r) \cap S_- = \varnothing$ then for the Lemma 1 we have that $T_L(x)=T_L(B)$ and this is in contradiction with $x \in Q-Q_r$.

Since F contains also states and all the states are represented in S_+ then $F \subseteq F_r$. And in the same way, since S_+ is structurally complete then $\delta \subseteq \delta_r$. then we have what we wanted to show.

❑

We are going to determine the complexity of this algorithm in the worst case. This case is when we can not merge any state. In what follws we define p and n as:

$$p=\sum_{x \in S_-} |x| \quad \text{and} \quad n=\sum_{x \in S_+} |x|$$

To obtain the searched automaton we must compute all the series $A(S,r)$ of automata; that is a total of p automata. To compute $A(S,i)$ from $A(S,i-1)$, the worst case is when we try with all the previous states and we have only found it inadequate with the last string of the negative sample. Then we have tried i times, and since the time needed to know if the negative sample is accepted for a nondeterministic automaton is in the worst case proportional to $e \,.\, n$ where e is the number of states. Then the time needed to build $A(S,n)$ must be *i. e.* n, consequently, since the worst case is when we can never merge any state, then the number of states is as large as p. Therefore, the overall complexity is

$$\sum_{i=1}^{p} i.p.n \in O(p^3.n)$$

5. References

[1] E. M. Gold. "Language identification in the limit". *Information and Control*, 10: pp 447-474, 1967.

[2] D. Angluin and C. H. Smith. "Inductive inference: theory and methods". *Computing Surveys*, 15(3), pp 237-269, 1983.

[3] K. S. Fu and T. L. Booth. "Grammatical Inference: Introduction and Survey". parts 1 and 2. *IEEE Trans. Sys. Man and Cyber.*, SMC-5: 95-111, pp 409-423, 1975.

[4] R. C. González and M. G. Thomason. "Syntactic Pattern Recognition, an introduction". *Addison-Wesley*, Reading Mass., 1978.

[5] K. S. Fu. "Syntactic Pattern Recognition and Applications". *Prentice-Hall*, New York, 1982.

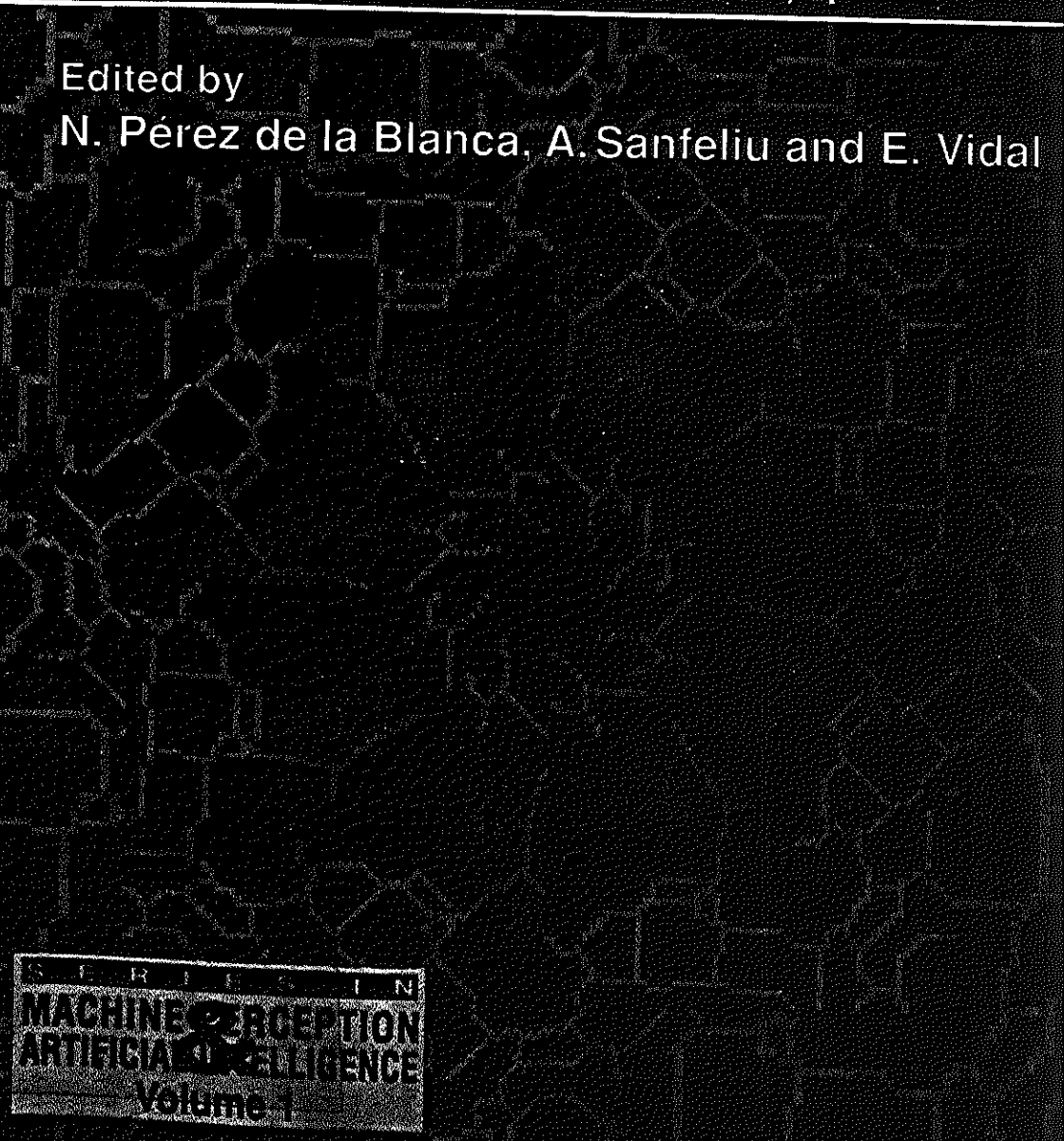
[6] L. Miclet. "Grammatical Inference". In *Syntactic and Structural Pattern Recognition*. H. Bunke and A. San Feliu (eds.) *World Scientific*, pp 237-290, 1990.

[7] E. M. Gold. "Complexity of automaton identification from given data". *Information and Control*, 37 pp 302-320, 1978.

PATTERN RECOGNITION AND IMAGE ANALYSIS

Selected Papers from the IVth Spanish Symposium

Edited by
N. Pérez de la Blanca, A. Sanfeliu and E. Vidal



World Scientific

Pérez de la Blanca
Sanfeliu
Vidal

PATTERN RECOGNITION AND IMAGE ANALYSIS

ISBN 981-02-0881-2