

# **Identification of *DFA*: data-dependent versus data-independent algorithms**

C. de la Higuera  
LIRMM,  
161 rue Ada,  
34 392 Montpellier  
Cedex 5, France.  
<http://www.lirmm.fr/~cdlh>

J. Oncina  
Dpto. Lenguajes y  
Sistemas Informáticos  
Universidad de Alicante,  
E-03071 Alicante, Spain.  
[joncina@dlsi.ua.es](mailto:joncina@dlsi.ua.es)

E. Vidal  
Dpto. Sistemas Informáticos y  
Computación, Universidad  
Politécnica de Valencia  
46071 Valencia, Spain.  
[evidal@iti.upv.es](mailto:evidal@iti.upv.es)

## **Abstract.**

Algorithms that infer deterministic finite automata from given data and that comply with the identification in the limit condition have been thoroughly tested and are in practice often preferred to elaborate heuristics. Even if there is no guarantee of identification from the available data, the existence of associated characteristic sets means that these algorithms converge towards the correct solution. In this paper we construct a framework for algorithms with this property, and consider algorithms that use the quantity of information to direct their strategy. These data dependent algorithms still identify in the limit but may require an exponential characteristic set to do so. Nevertheless preliminary practical evidence suggests that they could perform better.

**Keywords:** *DFA*, grammatical inference, identification in the limit.

## **Introduction**

Identification of deterministic finite automata (*DFA*) is possible in polynomial time through characteristic sets. The emphasis on deterministic finite automata is justified by the fact that algorithms treating the inference problem for *DFA* can be nicely adapted for larger classes of grammars, for instance even linear grammars [SG94, T88, T94], subsequential transducers [OGV93] or tree grammars. They can even be transposed to solve the inference problem for context-free grammars, when the data is presented as unlabelled trees [S92]. These results are algebraic and do not depend on the algorithms.

The question of identifying a target deterministic finite automaton is the earliest in grammatical inference. In [G67] the problem is studied under the on-line paradigm and in 1978 Gold gives an answer in the given data paradigm [G78]. Other early works on the problem include Trakhenbrot and Barzdin who give a general identification algorithm for this task [TB73]. The algorithmic difficulty of the task is studied in Pitt's 1989 paper [P89]. In this he describes the on-line learning paradigm in which, by using Angluin's results [A87] on oracle learning, it can be proven that

for the reasonable definition of polynomial time learning, *DFA* cannot be identified in polynomial time. This result nevertheless must be compared with experimental results (such as [L92] or [D94]) where the performance of Trakhenbrot and Barzdin's [TB73] algorithm or others are analysed. The encouraging results lead to believe that in practical cases an approximate identification of *DFA* is possible.

Other techniques have since been devised to infer *DFA*. Oncina and García [OG92] proposed, along the lines of Trakhenbrot & Barzdin's algorithm, *RPNI*<sup>1</sup>, a polynomial algorithm for inference of *DFA*, with the property of identification in the limit. This algorithm has been linked with work on the search space where a correct (inferred) automaton is defined as one obtained through merging states from the Prefix Tree Acceptor, which is the tree-like *DFA* that accepts only the positive data. This leads to exploring the lattice of partitions [DMV94]. Dupont proposed alternative methods based on genetic algorithms to explore this lattice [D94].

*RPNI* is an optimistic algorithm: at any one step two states are compared and the question is: can they be merged? No positive evidence can be produced; merging will take place each time that such a merge does not produce inconsistency. Obviously an early mistake can have disastrous effects and we note from Lang [L92] that a breadth-first exploration of the lattice (which corresponds to a breadth wise construction of the automaton) is likely to be better as in such a way more examples will be present to test the proposed merges.

In this paper we aim to compare the different ways this construction can take place. We prove that any data-independent ordering will allow for identification in the limit. We also prove that if we take the heuristic: try to merge those two states for which most evidence is available, the algorithm identifies in the limit, but the characteristic set associated to this heuristic can be exponential. Nevertheless preliminary experimental evidence suggests that the heuristic is interesting.

## 1) Definitions

An alphabet is a finite non-empty set of distinct symbols. For a given alphabet  $\Sigma$ , the set of all finite strings of symbols from  $\Sigma$  is denoted  $\Sigma^*$ . The empty string is denoted  $\lambda$ . For a string  $w$ ,  $|w|$  denotes the length of  $w$ . A language  $L$  over  $\Sigma$  is a subset of  $\Sigma^*$ .

A deterministic finite automaton (*DFA*) over  $\Sigma$  is a 5-tuple  $A=(Q, \Sigma, \delta, q_0, F)$  where  $Q$  is a finite set of states,  $F$  a subset of  $Q$ , denoting the set of final states of  $A$ ;  $\delta$  is the transition function:  $Q \times \Sigma \rightarrow Q$ .  $\delta$  is recursively extended to a function:  $Q \times \Sigma^* \rightarrow Q$  as follows:

$$\begin{aligned} \delta(q, \lambda) &= q \text{ and} \\ \text{if } w &= xw', \text{ with } x \in \Sigma \text{ and } w' \in \Sigma, \delta(q, w) = \delta(\delta(q, x), w'). \end{aligned}$$

A *DFA*  $A$  accepts a string  $w$  iff  $\delta(q_0, w) \in F$ . The language recognized by an automaton is the set of all strings accepted by the automaton. Two automata are

---

<sup>1</sup>Regular Positive and Negative Inference

equivalent *iff* they recognize the same language. An automaton is complete if its associated transition function  $\delta$  is total. The size of a *DFA* is the number of its states, which is polynomially related to the number of bits needed to encode it.

Gold presented in 1978 [G78] a model for identification, where a sample of labelled strings  $\langle S+, S- \rangle$ , with  $S+$  a set of positive instances, and  $S-$  a set of negative instances, is presented to the inference algorithm that must return a representation compatible with  $\langle S+, S- \rangle$ . The further conditions are that for each language there exists a characteristic sample with which the algorithm returns a correct representation, and, to avoid collusion (or cheating), this must be monotonous in the sense that if correctly labelled examples are added to the characteristic set, then the algorithm infers the same language. Gold proved that deterministic finite automata were identifiable (in this model) in polynomial time from given data. Further work in this model has contributed the following results: alternative algorithms have been proposed to infer *DFA* [OG92], Even linear grammars have been proven identifiable in polynomial time [T88]&[SG94]; these techniques have been extended to universal linear grammars in [T94]. Following the same idea deterministic even linear grammars are identifiable in polynomial time from positive examples only [KMT95] and the same holds for total subsequential functions [OGV93]. The algorithms provided in these papers have been implemented to deal with practical problems in the fields of speech, pattern recognition or automatic translation.

To take into account the fact that the length of the examples must depend polynomially in the size of the concept to be learnt we use the following definition [H95], which generalizes Gold's results [G78]. The size of a sample  $\langle S+, S- \rangle$  (denoted  $size(S+ \cup S-)$ ) is the sum of the lengths of all the strings in the sample.

**Definition 1** A representation class  $R$  is polynomially identifiable from given data *iff* there exist two polynomials  $p()$  and  $q()$  and an algorithm  $A$  such that:

- 1) Given any sample  $\langle S+, S- \rangle$ , of size  $n$ ,  $A$  returns a representation  $R$  compatible with  $\langle S+, S- \rangle$  in  $O(p(n))$  time.
- 2) For each representation  $R$  of size  $n$ , there exists a characteristic sample  $\langle CS+, CS- \rangle$  of size less than  $q(n)$  for which, if  $S+ \supseteq CS+$ ,  $S- \supseteq CS-$ ,  $A$  returns a representation  $R'$  equivalent with  $R$ .

By this definition algorithm  $A$  is a polynomial learner. With this definition, and considering (as it is usual) that the size of a *DFA* is the number of its states, Gold's 1978 result can be restated as follows:

**Gold's theorem (1978)** *DFA* are polynomially identifiable from given data.

In fact his result is even stronger as for any *DFA* a characteristic set can be also computed in polynomial time. The definition above and Gold's theorem are not trivial: in [H95] is proven that the following classes of representations do not admit polynomial identification from given data:

- Context-free grammars
- Linear grammars
- Simple deterministic grammars

- Non deterministic finite automata.

## 2) The general converging algorithm.

We aim to give a very general class of algorithms that converge in polynomial time. The class is based on merging algorithms as defined by Trakhenbrot & Barzdin [TB73] or Oncina & García [OG92]. For this purpose a general yet abstract data structure is needed. Obviously, for specific instantiations of the algorithm, the data structures need not be so heavy. Moreover research in the direction of finding interesting data structures for grammatical inference is necessary: better structures mean faster algorithms and hence should help produce in the foreseeable future nice results.

The instances will be given by two sets  $S+$ ,  $S-$  of strings. For each string in  $S+ \cup S-$  a mark will enable to know if the string can be read in the automaton, *i.e.* as the automaton may be incomplete, if through reading the string, one ends in a state of the automaton or not.

The initial state will be denoted  $q_0$ , and all states will be indexed by positive integers. The transition function  $\delta$  will consist in an array with two entries, one corresponding to the states, and the other to the alphabet. The value of a cell  $\delta(q, x)$  is (the index of) the state reached from  $q$  by  $x$ . If the edge has not yet been computed, the value will be undefined.

Furthermore a special structure is needed to remember which merges have already been tested: the corresponding array has 2 entries and should be read:  $q' \in \text{Tested}(q, x)$  iff adding  $\delta(q, x)=q'$  has been tested.

Finally given an automaton  $A$ , and sets  $\langle S+, S- \rangle$  a state in  $Q$  can be positive-final if some string in  $S+$  terminates in the state, negative-final if some string in  $S-$  terminates in the state, or indeterminate if no string in  $S+ \cup S-$  terminates in the state. Obviously if a state is indeterminate, it can be labelled final or non-final (in the classical sense) without breaking the inconsistency of the automaton towards  $\langle S+, S- \rangle$ . For this purpose we define two sets of (indices of) states  $F+$  and  $F-$ . In our notations a positive-final state will belong to  $F+$ , and a negative-final state to  $F-$ .

For example at some point the data structure could be:

$\delta$	$a$	$b$	$F+ = \{1\}$ $F- = \{0\}$
0	1	0	The intended meaning is that:
1		2	$\delta(q_0, aba)=q_1$ so $aba \in L$ .
2	1		$\delta(q_0, bbaa)=\text{undefined}$ .

We propose an algorithm that uses a function ( $\phi$ ) as a parameter:  $\phi$  can be of any type provided the following specifications are met: given an automaton  $A$ ,  $\phi$  returns a triple  $\langle q, x, q' \rangle$  such that

- 1  $\delta(q, x)$  is undefined and
- 2  $q' \notin \text{Tested}(q, x)$ .

In such a case  $\phi$  is called *admissible*. Furthermore a function  $\phi$  is called *data-independent* if it does not need information about the data  $\langle S^+, S^- \rangle$  to return its result. Otherwise  $\phi$  is *data-dependent*.

The following algorithm converges:

**Algorithm *DFAinfer***( $\phi$ );  
Input:  $S = \langle S^+, S^- \rangle$   
Output: an automaton (defined by  $\delta, F^+, F^-$ )  
{Initialisations}  
 $n \leftarrow 0; \forall x \in \Sigma, \text{Tested}(q_0, x) \leftarrow \emptyset; F^+ \leftarrow \emptyset; F^- \leftarrow \emptyset;$

While there are some unmarked strings in  $S^+ \cup S^-$  do  
 $\langle q, x, q' \rangle \leftarrow \phi(A);$   
If **Possible**( $\delta(q, x) = q'$ )  
then  $\delta(q, x) \leftarrow q'$ ;  
For each unmarked  $w$  in  $S^+$  do  
if  $\delta(q_0, w) = q$   
then mark( $w$ );  $F^+ \leftarrow F^+ \cup \{q\};$   
For each unmarked  $w$  in  $S^-$  do  
if  $\delta(q_0, w) = q$   
then mark( $w$ );  $F^- \leftarrow F^- \cup \{q\};$   
else  $\text{Tested}(q, x) \leftarrow \text{Tested}(q, x) \cup \{q'\};$   
if  $\forall i \leq n, q_i \in \text{Tested}(q, x)$   
then  $n \leftarrow n + 1;$   
 $Q \leftarrow Q \cup \{q_n\};$   
 $\delta(q, x) \leftarrow q_n$   
For each unmarked  $w$  in  $S^+$  do  
if  $\delta(q_0, w) = q$  then mark( $w$ );  
 $F^+ \leftarrow F^+ \cup \{q\};$   
For each unmarked  $w$  in  $S^-$  do  
if  $\delta(q_0, w) = q$  then mark( $w$ );  
 $F^- \leftarrow F^- \cup \{q\};$   
 $\forall x \in \Sigma, \text{Tested}(q_n, x) \leftarrow \emptyset;$

End-while;

**Function *Possible*** ( $\delta(q, x) = q'$ ): Boolean;  
returns True if adding to  $\delta$  the rule  $(q, x, q')$  does not lead to inconsistency, if not False.

This can be checked easily by testing if in the automaton obtained through adding the transition  $\delta(q, x) = q'$  to the current automaton there are two strings  $uxw$  and  $vw$  such that:

- $\delta(q_0, u) = q$  and  $\delta(q_0, v) = q'$  and
- $uxw \in S^+, vw \in S^-,$  or  $uxw \in S^-, vw \in S^+.$

Before proving that the algorithm converges let us show how it works on an example:

### Example 1

Let  $\Sigma = \{a, b\}$   $S^+ = \{\lambda, ab, aaa, aabaa, aaaba\}$   $S^- = \{aa, baa, aaab\}$

- $F^+ \leftarrow \{q_0\}$   $F^- \leftarrow \emptyset$   $n \leftarrow 0$
- $\phi(A) = \langle q_0, a, q_0 \rangle$  **Possible** returns False, due to the presence of  $\lambda$  in  $S^+$ ,  $aa$  in  $S^-$ . So  $\text{Tested}(q_0, a) = \{q_0\}$ . As  $\forall i \leq n$   $q_i \in \text{Tested}(q_0, a)$ ,  $q_1$  is created and  $\delta(q_0, a) \leftarrow q_1$ ,  $n \leftarrow 1$ .
- $\phi(A) = \langle q_0, b, q_0 \rangle$  **Possible** returns True,  $\delta(q_0, b) \leftarrow q_0$ .
- $\phi(A) = \langle q_1, a, q_0 \rangle$  **Possible** returns False, due to the presence of  $\lambda$  in  $S^+$ ,  $aa$  in  $S^-$ . So  $\text{Tested}(q_1, a) = \{q_0\}$ .
- $\phi(A) = \langle q_1, a, q_1 \rangle$  **Possible** returns False, due to the presence of  $aaa$  in  $S^+$ ,  $aa$  in  $S^-$ . So  $\text{Tested}(q_1, a) = \{q_0, q_1\}$ . As  $\forall i \leq n$   $q_i \in \text{Tested}(q_1, a)$ ,  $q_2$  is created and  $\delta(q_1, a) \leftarrow q_2$ ,  $n \leftarrow 2$ .
- $\phi(A) = \langle q_1, b, q_0 \rangle$  **Possible** returns True,  $\delta(q_1, b) \leftarrow q_0$ .
- $\phi(A) = \langle q_2, a, q_0 \rangle$  **Possible** returns False, due to the presence of  $\lambda$  in  $S^+$ ,  $aaab$  in  $S^-$ . So  $\text{Tested}(q_2, a) = \{q_0\}$ .
- $\phi(A) = \langle q_2, a, q_1 \rangle$  **Possible** returns False, due to the presence of  $\lambda$  in  $S^+$ ,  $aaab$  in  $S^-$ . So  $\text{Tested}(q_2, a) = \{q_0, q_1\}$ .
- $\phi(A) = \langle q_2, a, q_2 \rangle$  **Possible** returns False, due to the presence of  $aaa$  in  $S^+$ ,  $aa$  in  $S^-$ . So  $\text{Tested}(q_2, a) = \{q_0, q_1, q_2\}$ . As  $\forall i \leq n$   $q_i \in \text{Tested}(q_2, a)$ ,  $q_3$  is created and  $\delta(q_2, a) \leftarrow q_3$ ,  $n \leftarrow 3$ .
- $\phi(A) = \langle q_2, b, q_0 \rangle$  **Possible** returns False, due to the presence of  $aabaa$  in  $S^+$ ,  $aa$  in  $S^-$ . So  $\text{Tested}(q_2, b) = \{q_0\}$ .
- $\phi(A) = \langle q_2, b, q_1 \rangle$  **Possible** returns True,  $\delta(q_2, b) \leftarrow q_1$ .
- $\phi(A) = \langle q_3, a, q_0 \rangle$  **Possible** returns True,  $\delta(q_3, a) \leftarrow q_0$ .
- $\phi(A) = \langle q_3, b, q_0 \rangle$  **Possible** returns False, due to the presence of  $\lambda$  in  $S^+$ ,  $aaab$  in  $S^-$ . So  $\text{Tested}(q_3, b) = \{q_0\}$ .
- $\phi(A) = \langle q_3, b, q_1 \rangle$  **Possible** returns False, due to the presence of  $aaaba$  in  $S^+$ ,  $aa$  in  $S^-$ . So  $\text{Tested}(q_3, b) = \{q_0, q_1\}$ .
- $\phi(A) = \langle q_3, b, q_2 \rangle$  **Possible** returns True,  $\delta(q_3, b) \leftarrow q_2$ .

All strings can be read on this automaton, so the algorithm halts, with solution as in Figure 1. Notice that  $F^+ = \{q_0, q_3\}$ ,  $F^- = \{q_2\}$ . So state  $q_1$  can be indifferently labelled positive or negative: in both cases the automaton will be consistent with the data.

The function  $\phi$  used in this example will be formally defined later: it corresponds to a classical breadth-wise research based on a total ordering on  $\Sigma$  (here  $a < b$ ).

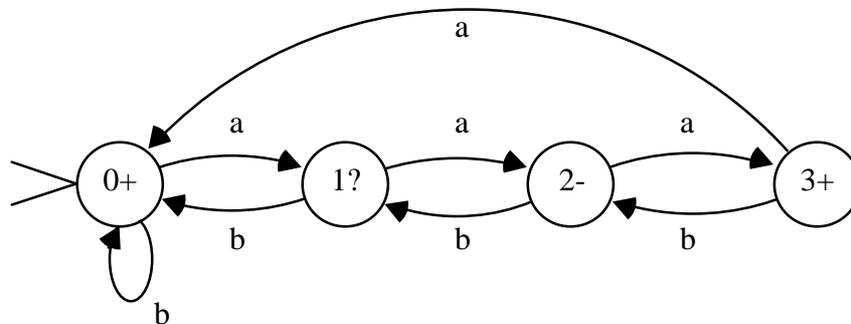


Figure 1

**Theorem 1**

For any polynomial data-independent function  $\phi$ ,  $DFAinfer(\phi)$  is a polynomial inference algorithm for  $DFA$ .

**Proof**

Let us first prove that the algorithm is polynomial in  $m=size(S+\cup S-)$ . Function *Possible* insures us that the solution returned is always consistent with  $S$ . The number of states of the inferred automaton is thus at most  $m$ . At each pass one of the following quantities increases:

- the number of states
- the number of *Tested* triplets.

As both are polynomially bounded, so is the overall complexity of the algorithm (provided the specified function  $\phi$  is polynomial).

It remains to be proven that for every regular language  $L$  defined by a  $DFA$ , there exists a polynomial characteristic set, depending of  $\phi$ , for which, when this set is included in a set of strings compatible with  $L$ , an equivalent  $DFA$  is inferred.

Let  $M$  be the minimum canonical automaton, complete, equivalent to the target automaton  $A$ . The characteristic set is computed in the following way: in algorithm *DFAinfer*, after the call to function  $\phi(\langle q, x, q' \rangle \leftarrow \phi(A))$  increment the characteristic set  $\langle CS+, CS- \rangle$  as follows:

If  $\delta_M(q, x) \neq q'$  then add  $w$  to  $CS+$ ,  $w'$  to  $CS-$  such that:

$$\begin{aligned} &\text{either } w=uxv, w'=u'v, && \text{and } \delta_A(q_0, u)=q \quad \delta_A(q_0, u')=q' \\ &\text{or } w=uv, w'=u'xv, && \text{and } \delta_A(q_0, u)=q' \quad \delta_A(q_0, u')=q. \end{aligned}$$

The existence of such a pair  $\langle w, w' \rangle$  is a consequence of Nerode's Theorem: in the minimal  $DFA$ , any two different states can be separated by some suffix. Moreover there exists a separating suffix of length at most the number of states of the automaton [H78]. Hence at most as many pairs of strings  $\langle w, w' \rangle$  are needed as there are tests that need to fail. This quantity is bounded by  $n^2|\Sigma|$ . As  $\phi$  does not depend on  $\langle S+, S- \rangle$ , adding strings to the learning set cannot modify the behaviour of the algorithm: the strings we have added are compatible with  $A$ , hence with  $M$ , and any merge that has been accepted beforehand cannot be refused by the added strings. This remains true if a new computation is started with a consistent set of

strings including the constructed  $\langle CS^+, CS \rangle$

•

In [OG92] the authors propose the following function  $\phi_\alpha$ <sup>2</sup>

Let  $\langle \Sigma \rangle$  be a total order on  $\Sigma$ . Consider the following order on  $Q \times \Sigma \times Q$ :

$$\langle q_i, x, q_j \rangle \ll_\alpha \langle q_k, x, q_m \rangle \text{ iff } \begin{array}{l} i < k \\ \text{or } i=k \text{ and } x <_\Sigma y \\ \text{or } i=k \text{ and } x=y \text{ and } j < m. \end{array}$$

$\phi_\alpha(A)$  returns the smallest triple  $\langle q_i, x, q_j \rangle$  for  $\ll_\alpha$  such that

- 1  $\delta(q_i, x)$  is undefined
- and 2  $q_j \notin \text{Tested}(q_i, x)$ .

Other alternative algorithms are possible and infer in the limit, provided the function  $\phi$  does not depend on the test set. For example, if *a priori* knowledge makes us believe that the letters in the alphabet do not have same probability of appearance a depth-first strategy would be better:

$$\langle q_i, x, q_j \rangle \ll_\beta \langle q_k, x, q_m \rangle \text{ iff } \begin{array}{l} x <_\Sigma y \\ \text{or } x=y \text{ and } i < k \\ \text{or } x=y \text{ and } i=k \text{ and } j < m. \end{array}$$

$\phi_\beta(A)$  returns the smallest triple  $\langle q_i, x, q_j \rangle$  for  $\ll_\beta$  such that

- 1  $\delta(q_i, x)$  is undefined
- and 2  $q_j \notin \text{Tested}(q_i, x)$ .

The data of example 1 contains more *a*'s than *b*'s. If this was known then for a function  $\phi_\beta$  constructed upon an order  $\ll_\beta$  (with  $a <_\Sigma b$ ), **DFAInfer**( $\phi_\beta$ ) now returns the automaton depicted Figure 2.

---

<sup>2</sup> In this paper we have slightly modified the setting: the natural object under construction in [OG93] (but also in [TP73], [L92], [DMV94]...) is the prefix tree acceptor (*PTA*), which is through successive merges modified into a graph. In the algorithm we propose the *PTA* is not constructed, it is implicit.

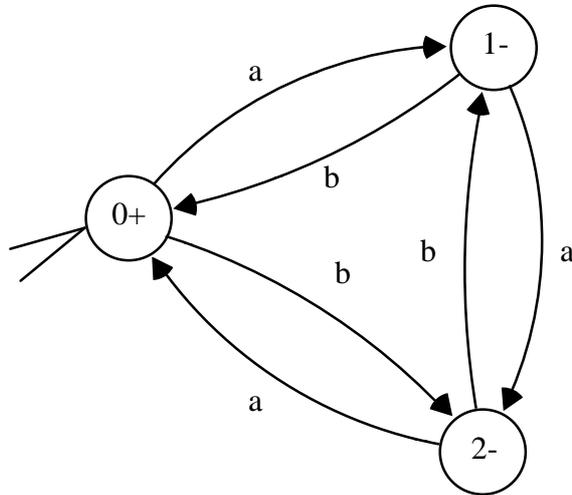


Figure 2

Comparing both automata and considering that alternative orderings could be proposed, a natural question is the following: does there exist for each regular language a universal polynomial characteristic set, one for which, when included in a set  $\langle S+, S- \rangle$ , any (correct) algorithm would give the same (correct) solution? A general (negative) answer is given in [H95]: for the wide class of all polynomial algorithms, some *DFA* only admit infinite characteristic sets. But if we consider the algorithms of the family  $DF\text{Ainfer}(\phi)$  we get the following result:

**Theorem 2**

For any regular language represented by a *DFA*  $A$  with  $n$  states, there exists a finite set  $\langle CS+, CS- \rangle$  for which, given any superset  $\langle S+, S- \rangle$ , any admissible function  $\phi$ ,  $DF\text{Ainfer}(\phi)$  returns a *DFA* equivalent to  $A$ .

Proof

Let  $M$  be the target automaton recognizing language  $L$ .

We construct a characteristic set as follows:

For each state  $q_i$  let  $K(i) = \{w \in \Sigma^* : \delta(q_0, w) = q_i \text{ and no substring of } w \text{ is in } K(i)\}$

Now construct  $\langle CS+, CS- \rangle$ :

For each  $\langle q_i, x, q_j \rangle$  such that  $\delta(q_i, x) \neq q_j$  by Nerode's theorem [H78] we have:

$\exists uxw, vw \in \Sigma^*$  with  $u \in K(i)$  and  $v \in K(j)$  and

either  $uxw \in L$  and  $vw \notin L$ ,  $uxw \in CS+$ ,  $vw \in CS-$

or  $uxw \notin L$  and  $vw \in L$ ,  $uxw \in CS-$ ,  $vw \in CS+$ .

The construction is obviously finite.

Now let  $S+ \supseteq CS+$ ,  $S- \supseteq CS-$ ,  $\phi$  be any function.

$DF\text{Ainfer}(\phi)$  identifies  $L$  in the limit:

We will only sketch the proof. In the automaton under construction, each state is reachable from  $q_0$ . If not this state would not have been created. Thus if at some

moment function  $\phi$  proposes as triplet  $\langle q_i, x, q_j \rangle$  then  $q_i$  and  $q_j$  are reachable through a path without cycles. This path is correct in the target automaton, where it also is cycle-free and thus corresponds to some string in  $K(i)$  (and respectively  $K(j)$ ). If  $\delta(q_i, x) \neq q_j$  then two strings exist in the characteristic set that distinguish  $\delta(q_i, x)$  and  $q_j$ . Thus,  $DF\text{Ainfer}(\phi)$  achieves identification in the limit. •

### Example 2

For the automaton depicted in Figure 2

$$K(0)=\{\lambda\} \quad K(1)=\{a, bb\} \quad K(2)=\{aa, b\}$$

Thus the following set is Characteristic for all  $\phi$ s:

$$CS^+ = \{\lambda, aaa, ab, aabb, ba, bbaa, bbb\} \quad CS^- = \{a, aa, aab, b, baa, bb, bba\}$$

This characteristic set is not overestimated in size: given a class of functions  $\Phi$  and an automaton  $A$ , a pair  $\langle S^+, S^- \rangle$  is characteristic for  $(\Phi, A)$  if given any function  $\phi$  in  $\Phi$ ,  $DF\text{Ainfer}(\phi)$  identifies  $A$ .

### Proposition 1

Given any polynomial  $p()$ , there exists an integer  $n$  and a DFA  $A$  of size  $n$ , such that no set  $\langle S^+, S^- \rangle$ , with  $size(S^+ \cup S^-) \leq p(n)$ , is characteristic for  $(\Phi, A)$ , where  $\Phi$  is the set of all admissible functions.

Proof

Let us consider  $\forall w \in \Sigma$  the function  $\phi_w(A)$ :

If  $w$  can be read in  $A$  then apply  $\phi_w$ , else return  $\langle q, x, q' \rangle$  such that for  $w=uxv$ ,  $u$  is the longest prefix one can read in  $A$ , and  $q'$  is minimal such that  $q' \notin \text{Tested}(q, x)$ .

Suppose now that the target language is  $\Sigma^{n-1}$ , the set of all strings of length  $n-1$ . Then consider the run of  $DF\text{Ainfer}(\phi_w)$  on data  $\langle S^+, S^- \rangle$ : if no mistake has been done in the first  $n$  tests of *possible*, we have at step  $n$  the following automaton:  $\delta(q_0, x_1) = q_1, \dots, \delta(q_{n-2}, x_{n-2}) = q_{n-1}$  with  $w = x_1 \dots x_n$ . At the next step,  $\langle q_{n-1}, x_n, q \rangle$  must be considered (for some  $q$ ). For the merge to be avoided, as is necessary,  $S^+$  or  $S^-$  must contain at least one string prefixed by  $w$ . As this holds for any  $w$ , and the number of such strings is exponential, we have established the result. •

## 3) A data driven algorithm

A reasonable idea is that the less information one has before trying to merge two states the more likely an error will appear. This fact has already been noticed by Lang [L92], and been used to justify a breadth-wise exploration. Yet the idea can be enhanced by using a data-dependant function  $\phi$  that would return the triple  $\langle q, x, q' \rangle$  for which most information is available: different measures of this "quantity of information" are possible. We chose here to count the number of strings exiting a given state, or exiting through a given edge.

We propose to adapt algorithm  $DF\text{Ainfer}(\phi)$ , in order to maintain the following data structure: given a state  $q$  of the automaton  $A$  we are constructing,

$$\begin{aligned} \# + \text{strings\_per\_state}(q) &= \wedge \{ w \in \Sigma^* : \exists u \in \Sigma^* \delta_A(q_0, u) = q \text{ and } uw \in S^+ \} \wedge \\ \# - \text{strings\_per\_state}(q) &= \wedge \{ w \in \Sigma^* : \exists u \in \Sigma^* \delta_A(q_0, u) = q \text{ and } uw \in S^- \} \wedge. \end{aligned}$$

and given a state  $q$ , and a symbol  $x$ ,

$$\begin{aligned} \# + \text{strings\_per\_edge}(\langle q, x \rangle) &= \wedge \{ w \in \Sigma^* : \exists u \in \Sigma^* \delta_A(q_0, u) = q \text{ and } uxw \in S^+ \} \wedge. \\ \# - \text{strings\_per\_edge}(\langle q, x \rangle) &= \wedge \{ w \in \Sigma^* : \exists u \in \Sigma^* \delta_A(q_0, u) = q \text{ and } uxw \in S^- \} \wedge. \end{aligned}$$

And  $\phi_\gamma$  returns the triple  $\langle q, x, q' \rangle$  such that:

- $\langle q, x, q' \rangle$  is valid:
  - 1  $\delta(q, x)$  is undefined
  - 2  $q' \notin \text{Tested}(q, x)$ .
- $\min\{\# + \text{strings\_per\_edge}(\langle q, x \rangle), \# - \text{strings\_per\_state}(q')\}$   
 $+ \min\{\# - \text{strings\_per\_edge}(\langle q, x \rangle), \# + \text{strings\_per\_state}(q')\}$   
 is maximal.

Algorithm  $\mathbf{DFainfer}(\phi_\gamma)$  is polynomial and infers an automaton consistent with the data. But does it infer in the limit ?

### Proposition 2

$\mathbf{DFainfer}(\phi_\gamma)$  identifies any regular language in the limit.

Proof

A consequence of theorem 2. For the characteristic set as defined in theorem 2  $\mathbf{DFainfer}(\phi_\gamma)$  identifies the automaton in the limit. •

It can be checked that on the data from example 1,  $\mathbf{DFainfer}(\phi_\gamma)$  infers the same automaton as with  $\phi_\beta$ . The algorithm "rediscovers" the fact that the  $a$ 's are more important than the  $b$ 's, and thus follows the good order.

Nevertheless the required characteristic set is no longer guaranteed to be polynomial:

### Proposition 3

There exists for each  $n$  some automata for which  $\mathbf{DFainfer}(\phi_\gamma)$  needs a non polynomial characteristic set to identify in the limit.

Proof

A consequence of proposition 1: to a proposed characteristic set one can always add data so that the run of the algorithm will simulate any  $\phi_w$ , for any given string  $w$ . •

## 4) Experimental work

The protocol we have followed is proposed in [D94] and based upon previous benchmarks ([T82] and [MG94]). The automata are all relatively small, and thus, the presented results can only give an indication of what to expect in equivalent conditions. Experimentation on larger automata is yet to be done. The 15 following test languages have been defined<sup>3</sup>:

- $L_1$ :  $a^*$
- $L_2$ :  $(ab)^*$
- $L_3$ : any string not ending with an odd number of  $b$ 's followed by an odd number of  $a$ 's.
- $L_4$ : any string without more than 2 consecutive  $a$ 's.
- $L_5$ : any string with an even number of  $a$ 's and an even number of  $b$ 's.
- $L_6$ : any string such that the number of  $a$ 's differs of the number of  $b$ 's by 0 modulo 3.
- $L_7$ :  $a^* b^* a^* b^*$
- $L_8$ :  $a^* b$
- $L_9$ :  $(a^* + c^*) b$
- $L_{10}$ :  $(aa)^* (bbb)^*$
- $L_{11}$ : any string with an even number of  $a$ 's and an odd number of  $b$ 's.
- $L_{12}$ :  $a(aa)^* b$
- $L_{13}$ : any string with an even number of  $a$ 's
- $L_{14}$ :  $(aa)^* ba^*$
- $L_{15}$ :  $bc^* b + ac^* a$

For each language 10 independent learning samples have been randomly created from these *DFA*, with as a restricting condition that the sample must be structurally complete<sup>4</sup>. The generation of the sample is pursued until the sample size is three times as much as the size of the structurally complete sample. In our case we have used exactly the same data as in [D94].

Our data-dependent method<sup>5</sup> is compared with *RPNI*, which appears in the literature as one of the best methods. Our data-dependent method corresponds to the algorithm from section 2, with function  $\phi_\gamma$ . The comparison is summarized in table 1. The results are computed as in [D94]. All strings up to length 9 (length 7 for languages  $L_9$  and  $L_{15}$ ) are presented to each of the inferred automata. Strings present in the training set are not counted. Then the result (appearing in column 2 for *RPNI*, in column 3 for *DDDI*) is the sum of the proportion of correctly classified positive strings, and of the proportion of correctly classified negative strings, divided by 2. Again this formula is given and justified in [D94]. Column 4 gives the size of the canonical complete automaton that is to be identified.

---

<sup>3</sup> Input alphabet is  $\{a\}$  for  $L_1$ ,  $\{a, b, c\}$  for  $L_9$  and  $L_{15}$ ,  $\{a, b\}$  for all the others.

<sup>4</sup> A sample is structurally complete if every edge in the automaton is used by some string in the data. For more detail see [MG94], [D94].

<sup>5</sup> We have called it *DDDI*: Data Driven Dfa Inference.

	<i>RPNI</i>	<i>DDDI</i>	Size
<i>L1</i>	100,00	100,00	2
<i>L2</i>	96,60	96,60	3
<i>L3</i>	100,00	100,00	5
<i>L4</i>	100,00	100,00	4
<i>L5</i>	62,90	61,98	5
<i>L6</i>	89,82	89,78	4
<i>L7</i>	92,04	88,67	5
<i>L8</i>	100,00	100,00	3
<i>L9</i>	99,09	99,11	3
<i>L10</i>	97,18	97,18	6
<i>L11</i>	88,31	88,08	5
<i>L12</i>	100,00	100,00	4
<i>L13</i>	89,92	94,99	3
<i>L14</i>	98,78	94,56	4
<i>L15</i>	95,34	95,34	5
Mean	93,87	93,75	

**Table 1**

The results remain inconclusive. The fact that the data does not have any specificity (same frequency for each letter...) offers no advantage to a data-driven method. Exhaustive tests with training data obtained through other distributions than the uniform one are necessary.

## 5) Conclusions

"Polynomial identification from given data" is a non trivial condition leading to interesting algorithms in grammatical inference. We have proven that this property could be extended to a wide class of algorithms. The property is lost for data-dependent algorithms but through our preliminary experiments we are convinced that this class of algorithms represents a promising trend of research.

## Bibliography

- [A87] Angluin D. (1987). Queries and concept learning. *Machine Learning* **2**, 319-342.
- [DMV94] Dupont P., Miclet L. & Vidal E. (1994). What is the search space of the regular inference? *Proceedings of the International Colloquium on Grammatical Inference ICGI-94* (pp. 25-37). Lecture Notes in Artificial Intelligence **862**, Springer-Verlag. Edited by R. Carrasco and J. Oncina.
- [D94] Dupont P. (1994). Regular Grammatical Inference from positive and negative samples by genetic search: the GIG method. *Proceedings of the International Colloquium on*

- Grammatical Inference ICGI-94* (pp. 236-245). Springer-Verlag Series in Artificial Intelligence **862**. Edited by R. Carasco and J. Oncina.
- [G67] Gold E.M. (1967). Language identification in the limit. *Inform. & Control*. **10**, 447-474.
- [G78] Gold E.M. (1978). Complexity of Automaton Identification from given Data. *Information and Control* **37**, 302-320.
- [H78] Harrison M.A. (1978). Introduction to Formal Language Theory. Reading: Addison-Wesley.
- [H95] de la Higuera C. (1995). Characteristic sets for Grammatical Inference. In *Proceedings of the International Colloquium on Grammatical Inference ICGI-96*.
- [KMT95] Koshiba, T., Mäkinen, E. & Takada, Y. (1995). Learning Deterministic Even Linear Languages from Positive Examples. *Proceedings of ALT '95*, Lecture Notes in Artificial Intelligence **997**, Springer-Verlag.
- [L92] Lang K.J. (1992). Random DFA's can be approximately Learned from Sparse Uniform Examples, *Proceedings of COLT 1992*, pp 45-52.
- [MG94] Miclet L. & de Gentile C. Inférence Grammaticale à partir d'Exemples et de Contre-exemples : deux algorithmes optimaux (BIG et RIG) et une version Heuristique (BRIG), *Actes des JFA-94*, Strasbourg, France, pp. F1-F13, 1994.
- [OG92] Oncina J. & García P. (1992) Inferring Regular Languages in Polynomial Updated Time. In *Pattern Recognition and Image Analysis*, World Scientific (49-61).
- [OGV93] Oncina J., García P. & Vidal E. (1993). Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**, 448-458.
- [P89] Pitt, L. (1989). Inductive inference, *dfas* and computational complexity. *Proceedings of the International Workshop on Analogical and Inductive Inference* (pp. 18-44). Lecture Notes in Artificial Intelligence **397**, Springer-Verlag.
- [SG94] Sempere J.M. & García P. (1994). A characterisation of Even Linear Languages and its application to the Learning Problem. *Proceedings of the International Colloquium on Grammatical Inference ICGI-94* (pp. 38-44). Lecture Notes in Artificial Intelligence **862**, Springer-Verlag.
- [S92] Sakakibara Y. (1992). Efficient Learning of Context-free Grammars from Positive Structural Examples. *Inf. and Comp.* **97**, 23-60.
- [T88] Takada Y. (1988). Grammatical inference for even Linear languages based on control sets. *Information Processing Letters* **28**, 193-199.
- [T94] Takada Y. (1994). A Hierarchy of Language Families Learnable by Regular Language Learners. *Proceedings of the International Colloquium on Grammatical Inference ICGI-94* (pp. 16-24). Lecture Notes in Artificial Intelligence **862**, Springer-Verlag.
- [T82] Tomita M. (1982). Dynamic construction of Finite -Automata from Examples Using Hill Climbing, *Proc. of the 4th annual Cognitive Science Conference*, USA, pp. 105-108, .
- [TB73] Trakhenbrot B. & Barzdin Y.(1973). Finite automata: Behavior and Synthesis. North Holland Pub., Amsterdam.
- [Y93] Yokomori T. (1993). Learning non deterministic Finite Automata from queries and Counterexamples. *Machine Intelligence* **13**. Furukawa, Michie & Muggleton eds., Oxford Univ. Press.

