Theoretical Computer Science 448 (2012) 41–46

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

# Polynomial characteristic sets for DFA identification[☆]

Pedro García, Damián López [*], Manuel Vázquez de Parga

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

ARTICLE INFO

*Article history:*
Received 4 April 2011
Received in revised form 14 March 2012
Accepted 29 April 2012
Communicated by J. Karhumaki

*Keywords:*
DFA identification
Grammatical inference

ABSTRACT

We study the order in *Grammatical Inference* algorithms, and its influence on the polynomial (with respect to the data) identification of languages. This work is motivated by recent results on the polynomial convergence of data-driven grammatical inference algorithms. In this paper, we prove a sufficient condition that assures the existence of a characteristic sample whose size is polynomial with respect to the minimum DFA of the target language.

© 2012 Elsevier B.V. All rights reserved.

## 1. Introduction

A *Grammatical Inference* (GI) algorithm is a method that tries to obtain a representation of a target language $L$ from some information about $L$ [1–3]. In this work we study the inference of deterministic automata for regular string languages using complete presentation (samples that belong or not to the target language) [4–8]. We note here that there are other results that, using the same data presentation, tackle the inference of non-deterministic automata [9,10] as well as results that study the identification of languages using queries [11–13].

A common approach to the GI of regular string languages takes into account an initial machine that represents the input data. Some states of this machine are then merged in order to obtain some generalization. It is worth noting that the generalization obtained depends both on the data supplied as well as on the order in which the states of the machine are traversed.

An important issue to determine whether a given GI algorithm has good behavior or not, is the amount of information needed to identify the target language. In that way it is important the concept of *characteristic sample* of a target language for an inference algorithm. Given a class of languages $H$, the characteristic sample for an algorithm $A$ and a language $L \in H$ is defined as a set of words in $L$ (usually denoted by $D_+$) and a set of words not in $L$ (denoted by $D_-$) such that: whenever the algorithm $A$ is run with input $(D_+, D_-)$ the algorithm outputs a correct representation of $L$; this representation does not change even though more words are added to the input.

The model of learning called identification in the limit by Gold, establishes that an algorithm identifies a class of languages *$H$ in the limit* if and only if every language in the class has associated with it a characteristic set for that algorithm [14]. Taking into account Gold's work, as well as results by Pitt [15], Angluin and Smith [16] and de la Higuera [17] defines *polynomial time and data identifiability*, as an extension of Gold's definitions which consider characteristic sets of polynomial size.

As mentioned above, the most referred results on the inference of regular languages from complete presentation propose algorithms based on the merging of states: the *RPNI* algorithm proposed by Oncina and García [7] and the *Blue-Fringe* algorithm by Lang et al. [8]. In both approaches, from an initial representation of the input information (usually a Moore machine of the training set), the algorithms consider a fixed order to traverse of the states of the machine, usually the

[☆] Work partially supported by the Spanish Ministerio de Economía y Competitividad under research project TIN2011-28260-C03-01 and Universidad Politécnica de Valencia grant PAID-2019-06-2011.

[*] Corresponding author. Tel.: +34 96 3877007; fax: +34 963877359.
*E-mail addresses:* pgarcia@dsic.upv.es (P. García), dlopez@dsic.upv.es (D. López), mvazquez@dsic.upv.es (M. Vázquez de Parga).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.04.042

canonical order. In [18], de la Higuera et al. propose an algorithmic scheme able to consider a broader set of orderings (the chosen order is an input parameter of the method), including any *data-driven* one. In that work, the order in which the states of the initial representation are promoted (considered as states of the automaton to be output) is a consequence of the order in which the states are merged. In fact, the authors do not distinguish among both orders. In that article, it is proved that, whenever the (merging) order does not consider the input data, then, for that algorithm, there exists a polynomial characteristic sample for any language. The authors also show that, given any size of automata, there is at least one data-driven order for which it is possible to find automata with non-polynomial characteristic set. This seems to indicate that any *interesting* data-driven order would imply an exponential characteristic sample.

All these results led to the GI community to think that *Blue-Fringe* algorithm (which was somewhat inspired by the results in [18]) to have no polynomial characteristic set, because *Blue-Fringe* merging order is in fact data-driven. Despite this assumption, *Blue-Fringe* became the *state of art* algorithm because of its experimental behavior in practical tasks, which outperformed the results of previous approaches. In fact, the merging order used by *Blue-Fringe* led this algorithm to be more data-efficient with respect to other GI algorithms, even when the training set does not include a characteristic set for the target language.

Nevertheless, it has been recently proved that *Blue-Fringe* algorithm has a polynomial characteristic set [19]. The proof of this takes into account that the *Blue-Fringe* promotion order does not depend on the merging order. This last result motivates the study of the order influence in the polynomial convergence of GI algorithms.

## 2. Definitions and notation

Let $\Sigma$ be a finite alphabet and let $\Sigma^*$ be the set of possible strings over $\Sigma$. Let also $\lambda$ denote the empty string. A *language* $L$ over $\Sigma$ is a subset of $\Sigma^*$. For any given set $q$, we will denote the cardinality of $q$ with $|q|$. Given $x \in \Sigma^*$, if $x = uv$ with $u, v \in \Sigma^*$, then $u$ (resp. $v$) is called *prefix* (resp. *suffix*) of $x$. Let us denote with $Pr(L)$ the set of prefixes of $L$.

A *Deterministic Finite Automaton* (*DFA*) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where $Q$ is a finite set of states, $\Sigma$ is an alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $\delta : Q \times \Sigma \to Q$ is the transition function. The language accepted by an automaton $A$ is denoted $L(A)$.

Given any *DFA* $A$, any two states $p$ and $q$ of $A$ are usually said to be equivalent if and only if, for any $x \in \Sigma^*$, $\delta(p, x) \in F$ if and only if $\delta(q, x) \in F$. This relation allows to obtain the *minimal DFA* for the language $L(A)$ (the automaton with the smallest set of states that accept the language). We note that this minimal automaton is unique up to isomorphism.

A Moore machine is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, \Phi)$, where $\Sigma$ (resp. $\Gamma$) is the input (resp. output) alphabet, $\delta$ is a partial function that maps $Q \times \Sigma$ in $Q$ and $\Phi$ is a function that maps $Q$ in $\Gamma$ called the *output function*. The function $\delta$ can be extended in a natural way to consider strings over $\Sigma$.

Throughout this paper, the behavior of $M$ will be given by the partial function $t_M : \Sigma^* \to \Gamma$ defined as $t_M(x) = \Phi(\delta(q_0, x))$ for every $x \in \Sigma^*$ such that $\delta(q_0, x)$ is defined.

A *DFA* $A = (Q, \Sigma, \delta, q_0, F)$ can be simulated by a Moore machine $M = (Q, \Sigma, \{+, -\}, \delta, q_0, \Phi)$, where $\Phi(q) = +$ if $q \in F$ and $\Phi(q) = -$ otherwise. Then, the language defined by $M$ is $L(M) = \{x \in \Sigma^* : \Phi(\delta(q_0, x)) = +\}$.

Given two disjoint finite sets of strings $D_+$ and $D_-$, we define the $(D_+, D_-)$*-prefix tree Moore machine* (*PTMM*$(D_+, D_-)$) as the Moore machine having $\Gamma = \{+, -, ?\}$, $Q = \mathrm{Pr}(D_+ \cup D_-)$, $q_0 = \lambda$ and $\delta(u, a) = ua$ if $u, ua \in Q$ and $a \in \Sigma$. For every state $u$, the value of the output function associated to $u$ is $+$, $-$ or ? (undefined) depending on whether $u$ belongs to $D_+$, to $D_-$ or to $Q - (D_+ \cup D_-)$ respectively.

A Moore machine $M = (Q, \Sigma, \{+, -, ?\}, \delta, q_0, \Phi)$ is *consistent* with $(D_+, D_-)$ if $\forall x \in D_+$ we have $t_M(x) = +$ and $\forall x \in D_-$ we have $t_M(x) = -$.

## 3. Convergence of GI algorithms using polynomial data

In this section we will prove a condition that, when fulfilled by a GI algorithm, assures the existence of a polynomial characteristic set.

In order to do so, we first propose a general framework that unifies all the previous results. This framework will be also useful to prove the main result.

### 3.1. A general framework

Algorithm 3.1 summarizes the most relevant GI algorithms. Please note that this algorithm puts aside efficiency. We now show that this algorithm can be considered as a general framework able to implement any previous result.

Note that, in order to implement *RPNI* algorithm with the proposed scheme, it is enough to take into account which list (*NonEqStatesList* or *MergibleStatesList*) has been first updated in order to choose among promotion or merge (line 16). Note also that, in order to implement *Blue-Fringe* algorithm, it is enough to consider the merging of states whenever the list of promotable states is empty.

Let us recall here the algorithm scheme proposed in [18]. The fact that this scheme does not use a *PTMM* to represent the training data can be considered secondary. In order to avoid extra notation, we will describe it in terms of a *PTMM* representation of the training set.

**Algorithm 3.1** Inference from complete presentation. A general scheme.

```
Input: $M = PTMM(D_+, D_-) = (Q, \Sigma, \{0, 1, ?\}, \delta, q_0, \Phi)$;
Input: An order among the states of the input PTMM
Output: A Moore Machine consistent with respect to the input data
 1: Method
 2: $red = \{\lambda\}$
 3: $blue = \{q \in Q : q = \delta(p, a),\ p \in red \wedge a \in \Sigma\} - red$
 4: while $blue \neq \emptyset$ do
 5:   $NonEqStatesList = MergibleStatesList = \emptyset$
 6:   for all $q \in blue$ /* traversed following the given order */ do
 7:     $merged = False$
 8:     for all $p \in red$ /* traversed following the given order */ do
 9:       if $(p, q)$ are mergable then
10:         $AppendTo(MergibleStatesList,\ (p, q))$
11:         $merged = True$
12:       end if
13:     end for
14:     if not $merged$ then $AppendTo(NonEqStatesList,\ q)$ end if
15:   end for
16:   Set option value among $\{merge,\ promote\}$
17:   if $option = promote$ then
18:     $red = red \cup \{q\} : q = First(NonEqStatesList)$
19:   else
20:     Let $(p, q) \in MergibleStatesList$ be chosen following whichever criterion, and deterministically merge them
21:   end if
22:   $blue = \{q \in Q : q = \delta(p, a),\ p \in red \wedge a \in \Sigma\} - red$
23: end while
24: Return($M$);
25: End Method.
```

In any given iteration of that scheme, a score for every $p \in red$ and $q \in blue$ is obtained, no matter whether the merge is possible or not. This set of scores drives the traverse of the pairs of *red* and *blue* states. In the traverse of the pairs of states, it is checked whether or not the merging of the states is possible. If so, the pair is merged and the *blue* set is updated. If the merge is not possible, the algorithm checks if the state $q$ has been considered to be merged with all states in the *red* set. In that case, the state $q$ is promoted to *red*, otherwise, the *blue* set is updated and the set of scores is reset. The algorithm ends when the *blue* set is empty.

Note that, in order to use Algorithm 3.1 to implement the algorithm by de la Higuera et al., it is necessary to compute the score among the *red* and *blue* states before the loop that traverses the *blue* states (line 6). The scores obtained guide the traverse of both *blue* and *red* sets, and therefore, the loops of lines 6 and 8 are also reduced to just one loop. The update of the *NonEqStatesList* (line 14) has also to be modified (to check whether or not all the possible merges have been considered), as well as included into the loop. Finally, the *option* flag is set to *merge* or *promote* (line 6) taking into account which list is updated first (in the same way the *RPNI* algorithm was previously adapted to our scheme).

## 3.2. A stronger result

We now will consider the proposed framework to prove a sufficient condition that assures the existence of a polynomial characteristic set for any given regular language. We first show that, for any regular language $L$, and whichever the order Algorithm 3.1 considers, it is possible to obtain a polynomial characteristic sample that identifies $L$.

The usual way to compute the characteristic set for any given language $L$ is based on the definition of the *minimal set of test states*. Thus, given $A = (Q, \Sigma, \delta, q_0, F)$ the minimum DFA for $L$, the set $S \subset \Sigma^*$ is a minimal set of test states if for every $q \in Q$ there exists only one string $x \in S$ such that $\delta(q_0, x) = q$.

Usually, for each state $q$, the set $S$ contains the first string in canonical order that reaches $q$ (note that, so defined, $S$ is minimal ($Card(S) = Card(Q)$) and prefix closed). We note here that, for any order used by Algorithm 3.1, no state is considered before any of its prefixes. This follows from the fact that Algorithm 3.1 chooses a state among those in the *blue* set. In the following, we will consider only this kind of *effective* orders.

Whichever the effective order chosen, it can be used to obtain a prefix closed minimal set of test states.

**Example 1.** Let the automaton in Fig. 1. Let us also consider the alphabetical order.

In order to obtain the minimal prefix closed test states set it is necessary to find, for each state $q$, the first string that reaches $q$ such that it does not visit a state twice. The alphabetic order does not give priority to shorter strings, thus, the

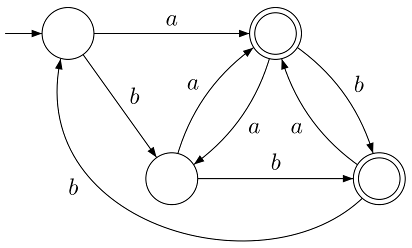

**Fig. 1.** Automaton example.

**Algorithm 3.2** Algorithm to obtain the characteristic set for a language $L$.

```
Input: The minimal DFA $A$ for $L$
Output: The polynomial characteristic set for $L$
 1: Method
 2: Let $S$ be the minimal set of test states for $A$
 3: $E = \{\lambda\}$
 4: Let $S' = S\Sigma - S$
 5: Let $T$ be a matrix indexed by the strings $u \in S \cup S'$ and $e \in E$ that stores the membership of the string $ue$ to the
    language
 6: while there exist two undistinguished $u, v \in S$ and a symbol $a \in \Sigma$ such that $T[ua, e] \neq T[va, e]$ for some $e \in E$ do
 7:    $E = E \cup \{ae\}$
 8: end while
 9: $(D_+, D_-) =$ Data in $T$
10: Return($D_+,\ D_-$);
11: End Method.
```

minimal prefix closed set of test states for this example is $S = \{\lambda, a, aa, aab\}$. Note that the set of test states obtained according the canonic order would have been $S = \{\lambda, a, b, ab\}$.

Let us also note that, for instance, a set of test states using the alphabetical order is $S = \{\lambda, a, aa, aaab\}$, but it is not prefix closed. Note also that, to make this set closed under prefixes implies that it would be non-minimal.

Taking into account any prefix closed minimal set of test states, Algorithm 3.2 shows the way to obtain two sets $D_+(S)$ and $D_-(S)$. A rough bound for the size of $D_+(S) \cup D_-(S)$ is easily seen to be quadratic in the size of $Q$.

In this algorithm, the condition of the loop in line 6 refers to two *undistinguished* states, that is, two elements $u$ and $v$ in $S$ such that there is no $w \in E$ such that just one of the strings $uw$ or $vw$ belongs to the language $L$. In order to find two such states, it is possible to use the matrix $T$ and look for two identical rows indexed by elements in $S$. Example 2 illustrates this procedure.

**Example 2.** Let us also consider the automaton in Fig. 1 and the prefix closed minimal set of test states $S = \{\lambda, a, aa, aab\}$.

The following table summarizes the process of obtaining the characteristic set. For the sake of clarity, we represent separately the elements in $S$ and those in $S\Sigma - S$. Initially the only column available is the one with label $\lambda$. The 1 and 0 entries in the table represent if the strings obtained by concatenation of the strings that label the row and column belong or not to the language $L$.

| | $\lambda$ | $b$ |
|---|---|---|
| $\lambda$ | 0 | 0 |
| $a$ | 1 | 1 |
| $aa$ | 0 | 1 |
| $aab$ | 1 | 0 |
| $b$ | 0 | 1 |
| $ab$ | 1 | 0 |
| $aaa$ | 1 | 1 |
| $aaba$ | 1 | 1 |
| $aabb$ | 0 | 0 |

Note that, taking into account just the column labeled $\lambda$, the undistinguished elements in $S$ are $\{\lambda, aa\}$ and $\{a, aab\}$. It is possible to distinguish the first one using the suffix $b$. Once the table is filled in, all the elements in $S$ are distinguished, therefore, the sets $D_+(S)$ and $D_-(S)$ for the language are the following:

$$D_+(S) = \{a, ab, bb, aaa, aab, aaab, aaba, aabab\}$$
$$D_-(S) = \{\lambda, b, aa, abb, aabb, aabbb\}.$$

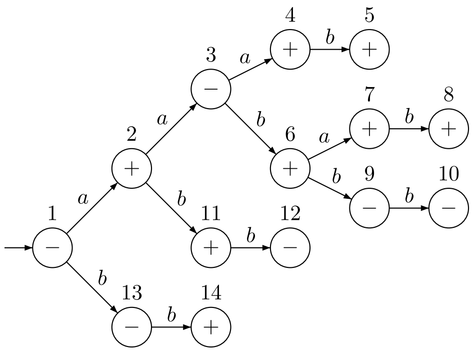

**Fig. 2.** *PTMM* for the example (characteristic) training set.

We now prove that, if the minimal set of test states was built using the same order Algorithm 3.1 uses, then, the sets $D_+(S)$ and $D_-(S)$ are a characteristic sample that identifies $L$.

**Theorem 3.** *Any GI algorithm, such that the promotion of states is independent from the input set, has a polynomial characteristic set, no matter the order followed to carry out the merging of states.*

**Proof.** Let us consider any order over a finite subset of $\Sigma^*$ as defined previously. Let $S \subset \Sigma^*$ be a prefix-closed minimal set of test states obtained according the defined order. Let also $D_+(S)$ and $D_-(S)$ be the positive and negative sets of strings obtained from $S$.

Let us assume first that promotion has priority over merging. We first will prove that $red \subseteq S$ always hold.

Initially, $red = \{\lambda\}$ and $blue = \Sigma$. Let us consider any given iteration such that $red \subsetneq S$, then $blue = red\Sigma - red \subset (S\Sigma \cup S)$. Note that there is at least one state in $blue \cap S$ which, by construction of the characteristic sample, can be distinguished from any state in *red*. Let $q$ denote the first of those states, $q$ appears also the first in *NonEqStateList* and is promoted to *red* (line 18 in Algorithm 3.1). Eventually, all the elements in $S$ will be promoted and thus $red = S$ and $blue = S\Sigma - S$. At that moment, again by the construction of the characteristic sample, each $q \in blue$ can be distinguished from any element in $S$ but just one. Therefore, the criterion followed to merge the remaining states in *blue* is irrelevant.

Let us assume now that promotion has no priority over merge. We prove now that the order in which the merges are carried out affects only when the training set is not characteristic. Under these conditions, whenever $red \subset S$ it is fulfilled that $blue \subseteq S\Sigma \cup S$. By construction of the characteristic sample, for any state $q \in blue$ there is only one *red* state $p$ such that the pair $(p, q)$ is in *MergibleStatesList*. Once the merging of the pair of states $(p, q)$ has been carried out, the *red* set does not change and the *blue* set continues being included into $S\Sigma \cup S$. □

**Example 4.** Let the automaton $A$ in Fig. 1 and the characteristic sample for the language $L(A)$ obtained in Example 2. The $PTMM(D_+, D_-)$ is the one shown in Fig. 2.

The symbol inside each state represents the output value for that state. Note also the figure shows also the numbering of the states according the alphabetical order. The *red* and *blue* sets are initialized respectively to {1} and {2, 13}. The first state to analyze is state 2 and added to the *NonEqStatesList* as well as state 13 which is analyzed afterwards. Thus, the state 2 is promoted to the red set, and therefore, $red = \{1, 2\}$ and $blue = \{3, 11, 13\}$.

The second iteration analyzes first the state 3 which is added to the list of *NonEqStatesList*, as well as state 11 and state 13. Thus, the algorithm's only option is to promote the first state in *NonEqStatesList*, and thus, $red = \{1, 2, 3\}$ and $blue = \{4, 6, 11, 13\}$.

The next iteration starts analysing state 4 which is found to be mergable with state 2, and therefore it is added to the *MergibleStatesList*. State 6 is the next state took into account and it is added to *NonEqStatesList* because it is not equivalent to any state in *red* (as well as state 11). The last state analyzed is state 13 which is found to be equivalent to state 3 and therefore added to *MergibleStatesList*. First, let us note that the two possible merges in this stage consider the pairs of states (2, 4) and (3, 13) (which will be of interest at the end of this example). In this run, we will choose first to promote rather than to merge. Therefore, the first state in *MergibleStatesList* is added to the red set, and thus $red = \{1, 2, 3, 6\}$ and $blue = \{4, 7, 9, 11, 13\}$.

The analysis of the *blue* states carried out in the next iteration detects that the pair of states (2, 4) can be merged, as well as the pairs (2, 7), (1, 9), (6, 11) and (3, 13). It is worth noting here that: first, each *blue* state can only be merged with one *red* state; second that the possible merges detected in previous iterations are also considered in this last iteration, and therefore, when a characteristic sample is used, it does not matter which choice is done in previous iterations because the output of the algorithm is always the same.

## 4. Conclusions

The experimental behavior of *Blue-Fringe* algorithm proves that the merging order in a GI algorithm is important to obtain good results in applied tasks. This is mainly due to the fact that a guided order can take profit from *evidences* in the training set. In general, the consideration of guided orders in a GI algorithm leads to a more data-efficient method.

The proof that *Blue-Fringe* algorithm has a polynomial characteristic set, which was assumed not to exist by the GI community, motivates this work. In this paper we prove a sufficient condition for GI algorithms to have a polynomial characteristic sample. The result allows the consideration of any interesting data-driven criterion to establish the merging order of a GI algorithm. Thus, the use of *ad-hoc* orders in the application of GI algorithms to real tasks, under some conditions, could lead on the one hand to very efficient algorithms with respect to the data, and on the other hand, does not threaten the polynomial convergence which can be achieved.

## References

[1] Y. Sakakibara, Recent advances of grammatical inference, Theoretical Computer Science 185 (1997) 15–45.
[2] C. de la Higuera, A bibliographical study of grammatical inference, Pattern Recognition 38 (2005) 1332–1348.
[3] C. de la Higuera, Grammatical Inference. Learning Automata and Grammars, Cambridge University Press, 2010.
[4] E.M. Gold, Complexity of automaton identification from given data, Information and Control 37 (1978) 302–320.
[5] B.A. Trakhtenbrot, Ya.M. Barzdin, Finite automata. Behavior and Synthesis, North-Holland Pub. Co, 1973.
[6] K.J. Lang, Random DFA's can be approximately learned from sparse uniform examples, in: Proceedings of the fifth annual workshop on Computational learning theory, 1992, pp. 45–52.
[7] J. Oncina, P. García, Inferring regular languages in polynomial updated time, in: Pattern Recognition and Image Analysis, volume 1, World Scientific, 1992, pp. 49–61.
[8] K.J. Lang, B.A. Pearlmutter, R.A. Price, Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm, in: 4th International Colloquium, ICGI-98, LNAI 1433 (1998) 1–12.
[9] F. Denis, A. Lemay, A. Terlutte, Learning regular languages using rfsa, Theoretical Computer Science 313 (2) (2004) 267–294.
[10] P. García, M. Vázquez de Parga, G.I. Álvarez, J. Ruiz, Universal automata and NFA learning, Theoretical Computer Science 407 (1–3) (2008) 192–202.
[11] D. Angluin, Learning regular sets from queries and counterexamples, Information and Computation 75 (1987) 87–106.
[12] D. Angluin, Queries and concept learning, Machine Learning 2 (4) (1988) 319–342.
[13] M.J. Kearns, U.V. Vazirani, An Introduction to Computational Learning Theory, The MIT Press, 1994.
[14] E.M. Gold, Language identification in the limit, Information and Control 10 (1967) 447–474.
[15] L. Pitt, Inductive inference, dfas, and computational complexity, in: Proc. 2nd Workshop on Analogical and Inductive Inference, LNAI 397 (1989) 18–44.
[16] D. Angluin, C.H. Smith, Inductive inference: theory and methods, Computing Surveys 15 (3) (1983) 237–269.
[17] C. de la Higuera, Characteristic sets for polynomial grammatical inference, Machine Learning 27 (1997) 125–138.
[18] C. de la Higuera, J. Oncina, E. Vidal, Identification of dfa: data-dependent vs data-independent algorithms, in: 3rd International Colloquium, ICGI-96, LNAI 1147 (1996) 313–325.
[19] P. García, M. Vázquez de Parga, D. López, J. Ruiz, Learning automata teams, in: 10th International Colloquium, ICGI-10, LNAI 6339 (2010) 52–65.