

Learning Regular Languages Using Nondeterministic Finite Automata*

Pedro García¹, Manuel Vázquez de Parga¹, Gloria I. Álvarez², and José Ruiz¹

¹ DSIC, Universidad Politécnica de Valencia. Valencia (Spain)

² Pontificia Universidad Javeriana. Cali, Colombia

Abstract. A new general method for inference of regular languages using nondeterministic automata as output has recently been developed and proved to converge. The aim of this paper is to describe and analyze the behavior of two implementations of that method and to compare it with two well known algorithms for the same task. A complete set of experiments has been carried out and the results of the new algorithms improve the existing ones both in recognition rates as in sizes of the output automata.

1 Introduction

The first ideas in the field of regular languages inference focused the description of the target languages using deterministic finite automata (*DFAs*). Many of the algorithms and heuristics proposed in this field use the technique of merging supposedly-equivalent states as a way to generalize the input. The starting point is the *prefix tree acceptor* (*PTA*), which is a tree-shaped automaton that recognizes the sample. The first of these algorithms (1973) is due to Trakhtembrot and Barzdin [12]. It is described as a contraction procedure in a finite tree that represents the words up to a certain length of a regular language. If the given data contains a certain characteristic set of the target language, it finds out the smallest *DFA* that recognizes the language.

The *RPNI* algorithm [10] starts from the *PTA* of the sample also. The merges are done in canonical order (two levels of ordering: length and alphabetical) controlled by the negative samples. The main ideas to improve the performance of *RPNI* have dealt with the order in which the states are merged. The algorithm *EDSM* [9] led to a control strategy called blue-fringe in which one of the states to merge is in the root of a tree. This algorithm, known as RedBlue is considered the state of art of *DFAs* inference by means of state merging. In [1] a new measure to order the merges called shared evidence is proposed. Using the concept of inclusion between the residuals of states, an extension of *RPNI* that enlarges the training set while learning has been proposed in [6].

These methods, which output *DFAs*, do not behave well sometimes when the target language has been obtained using randomly generated automata or

* Work partially supported by Spanish Ministerio de Educación y Ciencia under project TIN2007-60769.

regular expressions. This reason led to researchers to develop algorithms that output *NFAs*. Note that *NFAs* are generally smaller descriptions for a regular language than its equivalent *DFAs*. One of those is the *DeLeTe2* algorithm [5], which output an special type of *NFA* called Residual Finite State Automaton (*RFSA*). A *RFSA* is an automaton with its states being residuals of the language it accepts.

A subclass of the class of *NFAs* called *unambiguous finite automata (UFA)* has been defined and inferred in [4]. One of the properties of *UFA* is that the same target language will be achieved independently of the order in which states are merged. Some algorithms that use the same strategy that *RPNI* but output *NFAs* have been proposed [2]. The first attempt to use the *maximal automata* of the positive sample instead of the *PTA* is done in [13], where every positive sample is considered independently.

Finally, in [7] a general inference method based in states merging has been developed and proved to converge. It has been named *OIL* (order independent learning). It starts building the maximal automaton that recognizes the positive samples and one of its main features is that the convergence is achieved independently from the order in which states are merged. The convergence is proved using the concept of *Universal Automaton* of a language.

This latter fact about order-independent merging convergence opens up new possibilities of learning algorithms. In this paper two new algorithms based on the previous method are proposed. They will be referred to as *MOIL* (minimal order independent learning) and *VOIL* (voting order...). Both have in common that they canonically order the positive samples, and in an incremental way, build the maximal automata of the sample and merge the states in a random order to obtain an irreducible automata, controlled by the negative samples. Running the algorithm several times with different orders, different automata may be obtained. The difference between *MOIL* and *VOIL* is that the former outputs the smallest of the obtained automata, whereas the later keeps all of them and classifies the test samples by a majority vote. The proposed algorithms also use some evidence measure: as sometimes a state in the current automaton could be merged to several previous states, it chooses the merge that makes the resulting automaton to accept more positive samples.

2 Definitions and Notation

2.1 Languages and Automata

A language L is any subset of A^* , the free monoid generated by a finite alphabet A . The elements $x \in A^*$ are called *words* and the neutral element is denoted λ . The complement of L is denoted \bar{L} . The residual of L with respect to the word x is $x^{-1}L = \{y \in A^* : xy \in L\}$.

A (non deterministic) *finite automaton (NFA)* is a 5-tuple $\mathcal{A} = (Q, A, \delta, I, F)$, where Q is a finite set of states, A is an alphabet, $I, F \subseteq Q$ are respectively the set of initial and final states and $\delta : Q \times A \rightarrow 2^Q$ is the transition function, also be denoted as $\delta \subseteq Q \times A \times Q$ and is extended to $Q \times A^*$ as usual. The language

accepted by \mathcal{A} is $L(\mathcal{A}) = \{x \in A^* : \delta(I, x) \cap F \neq \emptyset\}$. The *left language* of a state q with respect to \mathcal{A} is $L_q = \{x \in A^* : q \in \delta(I, x)\}$.

A *sub-automaton* of a NFA $\mathcal{A} = (Q, A, \delta, I, F)$ is any finite automaton $\mathcal{A}' = (Q', A, \delta', I', F')$ where $Q' \subseteq Q$, $I' \subseteq I \cap Q'$, $F' \subseteq F \cap Q'$ and $\delta' \subseteq \delta \cap Q' \times A \times Q'$. If \mathcal{A}' is a sub-automaton of \mathcal{A} then $L(\mathcal{A}') \subseteq L(\mathcal{A})$.

Let $D \subset A^*$ finite. The *maximal automaton* for D is the NFA $MA(D) = (Q, A, \delta, I, F)$ where $Q = \cup_{x \in D} \{(u, v) \in A^* \times A^* : uv = x\}$, $I = \{(\lambda, x) : x \in D\}$, $F = \{(x, \lambda) : x \in D\}$ and for $(u, av) \in Q$, $\delta((u, av), a) = (ua, v)$. So defined $L(MA(D)) = D$.

Let $\mathcal{A} = (Q, A, \delta, I, F)$ be an automaton and let π be a partition of Q . Let $B(q, \pi)$ be the class of π that contains q . The *quotient automaton* of π in \mathcal{A} is $\mathcal{A}/\pi = (Q', A, \delta', I', F')$, where $Q' = Q/\pi = \{B(q, \pi) : q \in Q\}$, $I' = \{B \in Q' : B \cap I \neq \emptyset\}$, $F' = \{B \in Q' : B \cap F \neq \emptyset\}$ and the transition function is $B' \in \delta'(B, a)$ if and only if $\exists q \in B, \exists q' \in B'$ with $q' \in \delta(q, a)$.

The *merge* of states p and q in a finite automaton \mathcal{A} , denoted $merge(\mathcal{A}, p, q)$ is a particular quotient in which one of the blocks of the partition is the set $\{p, q\}$ and the rest are singletons.

An automaton \mathcal{A} is *irreducible* in L if and only if $L(\mathcal{A}) \subseteq L$ and for any non trivial partition π of the states of \mathcal{A} , $L(\mathcal{A}/\pi) - L \neq \emptyset$. \mathcal{A} is irreducible if it is irreducible in $L(\mathcal{A})$.

Given a regular language L , let \mathbf{U} be the finite set of all the possible intersections of residuals of L with respect to the words of A^* , that is, $\mathbf{U} = \{u_1^{-1}L \cap \dots \cap u_k^{-1}L : k \geq 0, u_1, \dots, u_k \in A^*\}$. The *universal automaton* (UA) [3,11] for L is $\mathcal{U} = (\mathbf{U}, A, \delta, I, F)$ where $I = \{q \in \mathbf{U} : q \subseteq L\}$, $F = \{q \in \mathbf{U} : \lambda \in q\}$ and the transition function is such that $q \in \delta(p, a)$ iff $q \subseteq a^{-1}p$.

The UA for a language L does not have any mergible states. A theorem [3] states that every automata that recognizes a subset of a language L can be projected into the UA for L by a homomorphism.

2.2 Grammatical Inference

Regular language learning is the process of learning an unknown regular language from a finite set of labeled examples. A positive (resp. negative) sample of L is any finite set $D_+ \subseteq L$ (resp. $D_- \subseteq \overline{L}$). If it contains positive and negative words it will be denoted as (D_+, D_-) and called a *complete sample*. A *complete presentation* of $L \subseteq \Sigma^*$ is a sequence of all the words of A^* labelled according to their membership to L .

An *inference algorithm* is an algorithm that on input of any sample outputs a representation of a language called hypothesis. The algorithm is *consistent* if the output contains D_+ and is disjoint with D_- . For the family of regular languages, the set of hypotheses, \mathcal{H} , can be the set of NFAs.

The type of convergence that we will use in our algorithms was defined by Gold [8] and is called *identification in the limit*. It is a framework proposed in order to analyze the behavior of different learning tasks in a computational way.

An algorithm \mathcal{LA} identifies a class of languages \mathcal{L} by means of hypotheses in \mathcal{H} *in the limit* if and only if for any $L \in \mathcal{L}$, and any presentation of L , the infinite sequence of hypotheses output by \mathcal{LA} converges to $h \in \mathcal{H}$ such that $L(h) = L$, that is, there exists t_0 such that $(t \geq t_0 \Rightarrow h_t = h_{t_0} \wedge L(h_{t_0}) = L)$, where h_t denotes the hypothesis output by \mathcal{LA} after processing t examples.

Most of the regular language inference algorithms output *DFAs* but recently, an algorithm called DeLeTe2 was proposed in [5]. It converges to an *RFSA* of size in between the sizes of the canonical *RFSA* and of the minimal *DFA* of the target language. It has the inconvenience that it generally outputs non consistent hypotheses. To overcome this difficulty, a program, also called DeLeTe2, has been proposed which obtains the best recognition rates -so far- when the languages to infer are obtained from random *NFAs* or regular expressions.

The generalizing process of the learning algorithms we propose in this paper is based in merging the states of the the maximal automaton of the positive samples in a random order, under the control of the negative samples. They are instances of the general method called OIL, which has been proposed in [7] by the same authors of the current paper.

3 Two Algorithms of the OIL Scheme

A general method was described in [7] which, on input of a set of blocks of positive and negative samples for a target regular language L , obtains an automaton that recognizes L in the limit. The method is called **OIL** (Order Independent Learning) and is described in Algorithm 1.

The method starts building the maximal automata for the first set of positive words $D_+^{(1)}$ and obtains a partition of the set of states such that the quotient automaton is irreducible in $\overline{D_-^{(1)}}$.

For every new block which is not consistent with the previous automaton (otherwise this new block is just deleted), it has to consider the following possibilities:

1. If the new set of negative samples is consistent with the current automaton,
 - It deletes the positive words accepted by the current hypothesis.
 - It builds the maximal automata $MA(D_+^{(i)})$ for the new set of positive words and adds to the set of negative ones the new block, obtaining D_- .
 - It finds a partition of the states of the disjoint union of the previous automaton with $MA(D_+^{(i)})$ such that the quotient automaton is irreducible in $\overline{D_-}$.
2. Otherwise it steps back and runs the algorithm starting with the first set of words, but considering the whole set of negative samples presented so far.

The convergence of the method was proved in [7] using the concepts of *irreducible automaton in a language* and of *universal sample*. To keep this paper self-contained, we recall these definitions and give a brief description of the proof of the convergence of the method.

Algorithm 1. OIL

Require: A sequence of blocks $\langle (D_+^{(1)}, D_-^{(1)}), (D_+^{(2)}, D_-^{(2)}), \dots, (D_+^{(n)}, D_-^{(n)}) \rangle$.

Ensure: An irreducible automaton consistent with the sample (recognizes the target language in the limit).

1: **STEP 1:**

2: Build $MA(D_+^{(1)})$;

3: $D_- = D_-^{(1)}$;

4: Find a partition π of the states of $MA(D_+^{(1)})$ such that $MA(D_+^{(1)})/\pi$ is irreducible in $\overline{D_-}$.

5: **STEP** $i + 1$:

6: Let $\mathcal{A} = (Q, A, \delta, I, F)$ be the output of the algorithm after processing the first i blocks, for $i \geq 1$.

7: $D_- = D_- \cup D_-^{(i+1)}$.

8: **if** \mathcal{A} is consistent with $(D_+^{(i+1)}, D_-^{(i+1)})$ **then**

9: Go to **Step** $i + 2$.

10: **end if**

11: **if** \mathcal{A} is consistent with $D_-^{(i+1)}$ **then**

12: $D_+^{(i+1)'} = D_+^{(i+1)} - L(\mathcal{A})$;

13: Build $MA(D_+^{(i+1)'})$; $//MA(D_+^{(i+1)'}) = (Q', A, \delta', I', F')//$

14: $\mathcal{A}' = (Q \cup Q', A, \delta \cup \delta', I \cup I', F \cup F')$;

15: Find a partition π of $Q \cup Q'$ such that \mathcal{A}'/π is irreducible in $\overline{D_-}$.

16: $\mathcal{A} = \mathcal{A}'/\pi$; Go to **Step** $i + 2$.

17: **end if**

18: **if** \mathcal{A} is not consistent with $D_-^{(i+1)}$ **then**

19: **Run OIL** with input $\langle (D_+^{(1)}, D_-), (D_+^{(2)}, D_-), \dots, (D_+^{(i+1)}, D_-) \rangle$

20: Go to **Step** $i + 2$.

21: **end if**

22: Return \mathcal{A}

A universal sample for L is a finite set $D_+ \subseteq L$ such that if π is any partition of the states of $MA(D_+)$, such that $MA(D_+)/\pi$ is irreducible in L , then $L(MA(D_+)/\pi) = L$. The proof of its existence (and its finiteness) can be seen in [7].

The following facts are also proved in [7]:

1. If $D_+ \subseteq L$ is finite and π is a partition of the states of $MA(D_+)$ such that $MA(D_+)/\pi$ is irreducible in L , then $MA(D_+)/\pi$ is isomorphic to a sub-automaton of \mathcal{U} (the universal automaton of L). If D_+ is a universal sample, then $MA(D_+)/\pi$ accepts L .
2. If D_+ is a universal sample, there exists a finite set $D_- \subseteq \overline{L}$ such that if π is a partition that makes $MA(D_+)/\pi$ to be irreducible in $\overline{D_-}$, then $MA(D_+)/\pi$ is irreducible in L and accepts L .

Based in those facts, the convergence of algorithm **OIL** is proved straight forward. In fact, if $(D_+^{(1)}, D_-^{(1)}), (D_+^{(2)}, D_-^{(2)}), \dots$ is a complete presentation of a

regular language L , there exists a value of n such that $D_+ = \bigcup_{i=1}^n D_+^{(i)}$ is universal for L . In this case,

- If $\bigcup_{i=1}^n D_-^{(i)}$ is enough to guarantee a correct partition of the set of states, OIL will return a correct hypothesis that will not change (line 9).
- Otherwise, there will exist $m > n$ such that $D_- = \bigcup_{i=1}^m D_-^{(i)}$ will avoid any erroneous merging when the first n blocks of positive samples are processed considering always D_- as the control set.

In both cases **OIL** will converge to a correct hypothesis.

3.1 The Algorithms MOIL and VOIL

Based on the previous method we propose two algorithms. They are particular instances of the OIL scheme in which the way to obtain an irreducible automaton (lines 4 and 15 of the method) is specified. So those lines have to be changed by the function described in Algorithm 2. The function *Run* (**Common Part OIL**) of line 3 of both algorithms means to run the OIL scheme with the specific way of obtaining the partition established by Algorithm 2.

Both algorithms have in common that:

- they canonically order the set of positive samples and consider that every block contains just a single word, so $D_+ = \{x_1, x_1, \dots, x_n\}$ with $x_i \ll x_j$ if $i < j$ and consider every block of positive samples as having one word, that is, $D_+^{(i)} = \{x_i\}$.
- they consider the whole set of negative samples from the beginning, so $D_-^{(1)} = D_-$ and $D_-^{(i)} = \emptyset$ if $i > 1$.
- if the current state has several candidates to be merged with, it chooses the state that makes the resulting automaton to accept more positive samples.
- they run the method k times (this can be done since the merges are done randomly and thus, several automata can be obtained).

They differ in how the output is considered. The first algorithm, called **MOIL** (Minimal OIL) outputs the smallest of the k hypotheses, that is, the automaton having smallest number of states. It is described in Algorithm 3.

The second, called **VOIL** (Voting OIL), keeps the k hypotheses and classifies the test sample voting among those hypotheses. It is described in Algorithm 4.

This way of obtaining the partition does not affect to the convergence of the process, so both algorithms converge and they run in time $O(n^2m)$, where n is the sum of the lengths of the positive samples and m the sum of the lengths of the negative ones.

4 Experimental Results

We present the results -both in recognition rates and in size of the inferred automata- of the algorithms MOIL and VOIL and compare them with the results

Algorithm 2. FindPartition(\mathcal{A}, x, D_+, D_-)

Require: A sequence of words (D_+, D_-) , $x \in D_+$ and an automaton \mathcal{A} with states randomly ordered.

Ensure: An irreducible automaton \mathcal{A} consistent with the current sample $(\{x_i : x_i \ll x\})$.

```

1: Build  $MA(x)$ ; //states of  $MA(x)$  randomly ordered after those of  $\mathcal{A}$ //
2:  $\mathcal{A} = \mathcal{A} \cup MA(x)$ ; //disjoint union//
3: for every state  $q_i$  of  $MA(x)$  in order do
4:    $\mathcal{A} = \text{merge}(\mathcal{A}, q_i, q_k)$  where  $q_k$  is any previous state such that:
5:   (1) merge  $(\mathcal{A}, q_k, q_i)$  is consistent with  $D_-$ 
6:   (2) merge  $(\mathcal{A}, q_k, q_i)$  recognizes more words of  $D_+$  than any other merge  $(\mathcal{A}, q, q_i)$ 
7: end for
8: Return  $(\mathcal{A})$ 

```

Algorithm 3. MOIL

```

1: size =  $\infty$ ; output =  $\emptyset$ ;
2: for  $i = 1$  to  $k$  do
3:    $\mathcal{A} = \text{Run}(\text{Common Part OIL})$ ;
4:   if  $\text{size}(\mathcal{A}) < \text{size}$  then
5:     size =  $\text{size}(\mathcal{A})$ ; output =  $\mathcal{A}$ ;
6:   end if
7: end for
8: Return (output)

```

Algorithm 4. VOIL

```

1: output =  $\emptyset$ ;
2: for  $i = 1$  to  $k$  do
3:    $\mathcal{A} = \text{Run}(\text{Common Part OIL})$ ;
4:   output =  $\text{Append}(\text{output}, \mathcal{A})$ ;
5: end for
6: Return (output)

```

obtained by the algorithms RedBlue [9] and DeLeTe2 [5] for the same task. These algorithms constitute the state of art in regular languages inference. The former behaves better when the source comes from random DFAs whereas the latter works better when the source comes from NFAs or from regular expressions.

4.1 Corpora

The regular languages we use in the experiments come from the corpora used to run the algorithms DeLeTe2 [5] and UFA [4]. The target languages in them are randomly generated from three different sources: regular expressions (RE), deterministic (DFA) and nondeterministic (NFA) automata. Once we eliminate repetitions we keep 102 RE, 120 NFAs and 119 DFAs.

We generate 500 different training samples and divide them in five incremental sets of size 100, 200, 300, 400 and 500. We also generate 1000 test samples,

different from the training ones. The length of the samples randomly varies from zero to 18. The samples are labeled by every automaton, obtaining 15 different sets (5 for each of RE, DFA and NFA). The percentage of positive and negative samples in each of the training sets is not controlled.

4.2 Experiments

We have done two basic experiments to compare the behavior of the new algorithms with the previous DeLeTe2 and RedBlue:

1. We run five times the basic method and thus we obtain five (may be) different automata for which we measure:
 - (a) The average size and recognition rate of the five automata.
 - (b) The size and recognition rate of the smallest automata (MOIL).
 - (c) We label the test set according to the majority vote between the five automata measuring this way the recognition rate (VOIL).
2. We fix the training corpus (we use the sets `re_100` and `nfa_100`) and run k times the method, being k an odd number varying from 3 to 15 and proceed as in (1)(c).

The results obtained by the algorithms are summarize in Table 1. We can see that both strategies -choosing the smallest hypothesis and voting among the five output hypotheses- present better recognition rates than those obtained by the algorithm DeLeTe2 and Red Blue when the source comes from random regular expressions or from NFAs. Note that those better recognition rates are obtained with hypotheses which are much smaller than those obtained by the algorithm DeLeTe2.

The size of the output is not reported when the labels come from voting, as in this case several hypotheses participate in the decision and the size can not be compared to the other outputs.

On the other hand the recognition rates of MOIL and VOIL algorithms when the source comes from random DFAs are as poor as they are when one uses the algorithm DeLeTe2 and they are far away of the rates obtained by Red Blue. For the case of DeLeTe2, this fact has been explained saying that the inference methods based on the detection of inclusion relations do not behave well when the target automata do not have those inclusion relations between states. Denis et al. [5] have experimentally shown that this was the case for randomly generated DFAs.

For the case of the algorithms we present, we conjecture that the reason of this behavior is that the size of minimal NFAs (hypotheses) which are equivalent to relatively small DFAs (targets) may be very similar to the size of the DFAs, so the algorithms that narrow the search space to DFAs tend to behave better. On the other hand, relatively small NFAs may have substantially greater equivalent DFAs.

The second experiment is done over the sets of samples `er_100` and `nfa_100` and the value of k indicating the number of automata output by the algorithm varies from 3 to 15, following the odd numbers. Of course it was expected that the size

Table 1. Recognition rates and average size of the smallest hypothesis and recognition rates voting for $k = 5$ compared to those of DeLeTe2 and RedBlue algorithms

Id.	Smallest A.		Vote	DeLeTe2		RedBlue	
	Rate	Size	Rate	Rate	Size	Rate	Size
er_100	93.79	8.27	93.32	91.65	30.23	87.94	10
er_200	97.83	7.80	97.27	96.96	24.48	94.81	9.97
er_300	98.77	7.68	98.68	97.80	31.41	96.46	11.05
er_400	99.20	7.55	99.10	98.49	27.40	97.74	10.43
er_500	99.66	6.82	99.53	98.75	29.85	98.54	10.47
nfa_100	75.00	21.46	76.42	73.95	98.80	68.15	18.83
nfa_200	78.05	35.23	79.94	77.79	220.93	72.08	28.80
nfa_300	81.27	45.81	82.94	80.86	322.13	74.55	36.45
nfa_400	83.87	52.40	85.58	82.66	421.30	77.53	42.58
nfa_500	85.64	58.81	87.06	84.29	512.55	80.88	47.54
dfa_100	60.17	28.01	60.34	62.94	156.89	69.12	18.59
dfa_200	63.05	49.63	63.54	64.88	432.88	77.18	25.83
dfa_300	66.01	65.17	67.41	66.37	706.64	88.53	25.10
dfa_400	69.12	78.66	70.53	69.07	903.32	94.42	21.36
dfa_500	72.29	88.30	73.66	72.41	1027.42	97.88	18.75

Table 2. Recognition rates for different values of k over the set of samples er_100 and nfa_100

k	er_100			nfa_100		
	Smallest H.	Vote	Rec.	Smallest H.	Vote	Rec.
3	92.15	9.79	92.17	74.23	22.77	75.41
5	93.79	8.27	93.32	75.00	21.46	76.42
7	94.90	7.51	93.59	75.32	20.30	77.46
9	94.63	7.42	93.83	75.67	19.90	77.65
11	94.82	7.49	93.82	75.85	19.67	77.83
13	95.01	7.18	93.95	76.32	19.51	77.78
15	95.14	7.10	94.32	76.23	19.10	78.00

of the smallest automaton becomes smaller as k gets larger, so this experiment wanted to measure how the variation of k affects to the recognition rates both average and of the smallest output.

The results are summarized in Table 2. Note that the recognition rates increase as k gets bigger, which would indicate that for this algorithm, the smallest hypothesis output tends to be the best.

5 Conclusions

The general scheme proposed in [7] opens up new possibilities of inference algorithms. Two implementations of this method have been proposed and measured

its performance, both in recognition rates as in size of the output. The results are very promising as they beat -both in recognition rates as in size of output- the algorithm DeLeTe2, which is considered the state of art when samples are taken from random NFAs or regular expressions. In the case of samples taken from random DFAs, the proposed algorithms and DeLeTe2 are far away of the results obtained by the RedBlue.

References

1. Abela, J., Coste, F., Spina, S.: Mutually Compatible and Incompatible Merges for the Search of the Smallest Consistent DFA. In: Paliouras, G., Sakakibara, Y. (eds.) ICGI 2004. LNCS (LNAI), vol. 3264, pp. 28–39. Springer, Heidelberg (2004)
2. Alvarez, G.I., Ruiz, J., Cano, A., García, P.: Nondeterministic Regular Positive Negative Inference NRPNI. In: Díaz, J.F., Rueda, C., Buss, A. (eds.) XXXI CLEI 2005, pp. 239–249 (2005)
3. Conway, J.H.: Regular algebra and finite machines. Chapman and Hall, Boca Raton (1971)
4. Coste, F., Fredouille, D.: Unambiguous automata inference by means of state merging methods. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 60–71. Springer, Heidelberg (2003)
5. Denis, F., Lemay, A., Terlutte, A.: Learning regular languages using RFSAs. *Theoretical Computer Science* 313(2), 267–294 (2004)
6. García, P., Ruiz, J., Cano, A., Alvarez, G.: Inference Improvement by Enlarging the Training Set while Learning DFAs. In: Sanfeliu, A., Cortés, M.L. (eds.) CIARP 2005. LNCS, vol. 3773, pp. 59–70. Springer, Heidelberg (2005)
7. García, P., Vazquez de Parga, M., Alvarez, G., Ruiz, J.: Universal Automata and NFA learning. *Theoretical Computer Science* (to appear, 2008)
8. Gold, E.M.: Language identification in the limit. *Information and Control* 10, 447–474 (1967)
9. Lang, K., Perarlmutter, B., Price, R.: Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In: Honavar, V.G., Slutzki, G. (eds.) ICGI 1998. LNCS (LNAI), vol. 1433, pp. 1–12. Springer, Heidelberg (1998)
10. Oncina, J., García, P.: Inferring Regular Languages in Polynomial Updated Time. In: de la Blanca, P., Sanfeliú, Vidal (eds.) *Pattern Recognition and Image Analysis*. World Scientific, Singapore (1992)
11. Polák, L.: Minimalizations of NFA using the universal automaton. In: Domaratzki, M., Okhotin, A., Salomaa, K., Yu, S. (eds.) CIAA 2004. LNCS, vol. 3317, pp. 325–326. Springer, Heidelberg (2005)
12. Trakhtenbrot, B., Barzdin, Y.: *Finite Automata: Behavior and Synthesis*. North Holland Publishing Company, Amsterdam (1973)
13. Vázquez de Parga, M., García, P., Ruiz, J.: A family of algorithms for non deterministic regular languages inference. In: Parsons, S., Maudet, N., Moraitis, P., Rahwan, I. (eds.) ArgMAS 2005. LNCS (LNAI), vol. 4049, pp. 265–275. Springer, Heidelberg (2006)