

# A Family of Algorithms for Non Deterministic Regular Languages Inference\*

Manuel Vázquez de Parga, Pedro García, and José Ruiz

Universidad Politécnica de Valencia

Departamento de Sistemas informáticos y Computación, Camino de Vera s/n  
46022 Valencia, Spain

{mvazquez, pgarcia, jruiz}@dsic.upv.es

<http://www.dsic.upv.es/users/tlcc/tlcc.html>

**Abstract.** We present in this paper a new family of algorithms for regular languages inference from complete presentation.

Every algorithm of this family, on input of the sets of words  $(D_+, D_-)$ , obtains for every  $x$  in  $D_+$  at least a non deterministic finite automaton (*NFA*) which accepts  $x$  and is consistent with  $D_-$ . This automaton is, besides, irreducible in the sense that any further merging of states accepts words of  $D_-$ . The output of the algorithm is a *NFA* which consists of the collection of *NFAs* associated to each word of  $D_+$ . Every algorithm of the family converges to a automaton for the target language.

We also present the experiments done to compare one of the algorithms of the family with two other well known algorithms for the same task. The results obtained by our algorithm are better, both in error rate as in the size of the output.

## 1 Introduction

The classical algorithms for regular languages identification typically output a deterministic finite automaton (*DFA*). One of the best known algorithms of this kind is the *RPNI* (Regular Positive and Negative Inference) algorithm [8], which converges to the minimal *DFA* of the target language. Its method is to merge the states in the prefix tree Moore machine of the sample in lexicographical order and to propagate the merges to keep the automaton deterministic, under the condition of not merging states that represent positive samples with those which represent negative ones.

Starting up from the idea that a non deterministic automaton (*NFA*) is generally a smaller description for a regular language than its equivalent *DFA*s, it has recently been proposed an algorithm called *DeLeTe2* [3] whose output is a special type of *NFA* called *RFSA* (Residual Finite State Automata) characterized by the fact that its states are residuals of the language it recognizes.

The authors of *DeLeTe2* show that when the target automaton is a randomly generated *DFA*, this algorithm behaves worse than *RPNI*, but the opposite way

---

\* Work partially supported by Spanish CICYT under TIC2003-09319-C03-02.

happens if the target automaton is generated using random regular expressions or *NFAs*.

We propose in this work a family of algorithms such that each of them, on input of a sample, outputs a non deterministic automaton. They all infer the class of regular languages in the limit. The method is to obtain, for each word of the positive sample, at least an irreducible consistent automaton (the merging of any two states in it, makes the resulting automaton to accept negative words). The way to do it is merging states in the automaton that just recognizes the word. The method is flexible as things like the number of subautomata inferred for each word or the order of state merging can be changed without affecting to the convergence of the process. The experiments done clearly show that this method obtains better results than both the classical *RPNI* and *DeLeTe2* algorithms.

The article is structured as follows: After this introduction, section 2 contains some preliminary definitions and notation, section 3 contains the theoretical basis of the method while section 4 contains the method itself and section 5 contains two examples for better understanding of the method. Finally, the experiments done (section 6) and the conclusions (section 7) end the job.

## 2 Preliminary Definitions and Notation

### 2.1 Finite Automata

An *alphabet* is any non empty finite set of symbols. A word over an alphabet  $\Sigma$  is any finite sequence of symbols in  $\Sigma$ , the *empty word* is denoted as  $\lambda$ ,  $\Sigma^*$  is the set of all the words over  $\Sigma$ , which is a *free monoid* under the concatenation of words. Given a word  $x = uv$ , with  $u, v \in \Sigma^*$ ,  $u$  is called prefix of  $x$ . The set of the prefixes of  $x$  is called  $Pr(x)$ .

A *language* over  $\Sigma$  is any subset of  $\Sigma^*$ . The *concatenation* of two languages  $L_1$  and  $L_2$  will be denoted as  $L_1L_2$ . The residual language of  $L$  associated to  $x$  is  $x^{-1}L = \{y \in \Sigma^* | xy \in L\}$ .

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

Given  $P \subseteq Q$  and  $a \in \Sigma$ ,  $\delta(P, a) = \bigcup_{q \in P} \delta(q, a)$ . The function  $\delta$  is extended to words writing  $\delta(P, \lambda) = P$  and  $\delta(P, xa) = \delta(\delta(P, x), a)$ , for every  $a \in \Sigma$ ,  $x \in \Sigma^*$ . The language accepted by  $\mathcal{A}$  will be denoted as  $L(\mathcal{A})$ , that is,  $L(\mathcal{A}) = \{x \in \Sigma^* : \delta(I, x) \cap F \neq \emptyset\}$ . The *left language* of a state  $q$  with respect to  $\mathcal{A}$  is  $L_q = \{x \in \Sigma^* : q \in \delta(I, x)\}$ . Two automata are equivalent if they accept the same language.

A finite automaton  $\mathcal{A}$  is *deterministic* if  $Card(I) = 1$  and for every state  $q$  and every symbol  $a$ ,  $Card(\delta(q, a)) \leq 1$ .

A *subautomaton* of a non deterministic finite automaton  $\mathcal{A} = (Q, \Sigma, \delta, I, F)$  is any finite automaton  $\mathcal{A}' = (Q', \Sigma, \delta', I', F')$  where  $Q' \subseteq Q$ ,  $I' \subseteq I \cap Q'$ ,  $F' \subseteq F \cap Q'$  and  $\delta' \subseteq \delta \cap Q' \times \Sigma \times Q'$ .

It is easily seen that if  $\mathcal{A}'$  is a subautomaton of  $\mathcal{A}$  then  $L(\mathcal{A}') \subseteq L(\mathcal{A})$ .

Given  $\mathcal{A} = (Q, \Sigma, \delta, I, F)$  and  $\mathcal{B} = (Q', \Sigma, \delta', I', F')$  the function  $\varphi : Q \rightarrow Q'$  is a *homomorphism* from  $\mathcal{A}$  to  $\mathcal{B}$  if  $\varphi(I) \subseteq I'$ ,  $\varphi(F) \subseteq F'$  and  $\varphi(\delta(q, a)) \subseteq \delta'(\varphi(q), a)$  for any  $q$  in  $Q$  and  $a$  in  $\Sigma$ .

The subautomaton of  $\mathcal{B}$  induced by  $\varphi(Q)$  is denoted as  $\varphi(\mathcal{A})$ . It follows that  $L(\mathcal{A}) \subseteq L(\varphi(\mathcal{A})) \subseteq L(\mathcal{B})$ .

The *merge* of states  $p$  and  $q$  in a finite automaton  $\mathcal{A} = (Q, \Sigma, \delta, I, F)$  is defined as follows:  $merge(\mathcal{A}, p, q) = (\varphi(Q), \Sigma, \delta', I', F')$  where  $\varphi(q) = p$  and  $\forall r \neq q, \varphi(r) = r$ , also  $I' = \varphi(I)$ ,  $F' = \varphi(F)$  and  $(r, a, s) \in \delta$  if and only if  $(\varphi(r), a, \varphi(s)) \in \delta'$ .

It follows that  $L(\mathcal{A}) \subseteq L(merge(\mathcal{A}, p, q))$ .

Given a language  $L$ , let  $U = \{u_1^{-1}L \cap \dots \cap u_k^{-1}L : k \geq 0, u_1, \dots, u_k \in \Sigma^*\}$ . The *universal automaton* [1,2,7,9] for  $L$  is defined as  $\mathcal{U} = (U, \Sigma, \delta, I, F)$  with:

- $I = \{q \in U : q \subseteq L\}$ .
- $F = \{q \in U : \lambda \in q\}$ .
- The transition function is such that  $q \in \delta(p, a)$  iff  $q \subseteq a^{-1}p$ .

Related to the universal automaton we have the following:

**Theorem 1.** [2] *Let  $\mathcal{U} = (U, \Sigma, \delta, I, F)$  the universal automaton for  $L \subseteq \Sigma^*$ . Then:*

1.  $L(\mathcal{U}) = L$ .
2. *For any automaton  $\mathcal{A} = (Q, \Sigma, \delta_A, I_A, F_A)$  such that  $L(\mathcal{A}) \subseteq L$ , the function  $\varphi : Q \rightarrow U$  defined as  $\varphi(q) = \bigcap_{u \in L_q} u^{-1}L$  is an automata homomorphism.*

## 2.2 Grammatical Inference

Grammatical inference is the discipline that deals with learning formal languages from either a positive or a complete sample.

A positive (resp. negative) sample of  $L$  is any finite set  $D_+ \subseteq L$  (resp.  $D_- \subseteq \bar{L}$ ). In the case it contains positive and negative words it will be denoted as  $(D_+, D_-)$ .

An *inference algorithm* is an algorithm that on input of any sample outputs a representation of a language. The algorithm is *consistent* if the output contains  $D_+$  and is disjoint with  $D_-$ .

The type of convergence that we will use in our algorithms was defined by Gold [5,6] and is called *identification in the limit*.

An algorithm  $A$  identifies a class of languages  $\mathcal{L}$  by means of hypothesis 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 hypothesis output by  $A$  converges to  $h$  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  $A$  after processing  $t$  examples.

One of the best known algorithms that identifies  $\mathcal{L}_3$  (the family of regular languages) using deterministic finite automata is the *RPNI* (Regular positive and negative inference) [8]. Given a sample, the algorithm builds the prefix

tree Moore machine, whose states are the prefixes of the sample and whose transitions are of the form  $\delta(x, a) = xa$ . An output 1, 0 or ? is assigned to every state depending whether it is associated to a positive sample, to a negative one or to a prefix of a word of the sample which is not itself a word of it. The algorithm merges every state with the previous ones in lexicographical order and propagates the merges to keep the determinism under the condition that no state with output 0 can be merged with a state with output 1. The merging of a state with output  $s$  with a state with output ? gives out a state with output  $s$ .

Among the recently proposed algorithms that output non deterministic finite automata is the algorithm *DeLeTe2* [3]. It outputs a special type of automata called *RFSA* (Residual finite state automaton). A finite automaton  $\mathcal{A} = (Q, \Sigma, \delta, I, F)$  is a *RFSA* if for every  $q \in Q$  the language  $\{x | \delta(q, x) \cap F \neq \emptyset\}$  is a residual language of  $L(\mathcal{A})$ .

### 3 Subautomata Associated to a Word in a Language

**Definition 1.** *An automaton  $\mathcal{A} = (Q, \Sigma, \delta, I, F)$  is irreducible in a regular language  $L$  if and only if  $L(\mathcal{A}) \subseteq L$  and for any pair of states  $p$  and  $q$  in  $Q$  we have that  $L(\text{merge}(\mathcal{A}, p, q)) - L \neq \emptyset$ .*

**Proposition 1.** *Let  $\mathcal{A} = (Q, \Sigma, \delta, I, F)$  irreducible in a regular language  $L$ . Then  $\mathcal{A}$  is isomorphic to a subautomaton of  $\mathcal{U}$ , the universal automaton for  $L$ .*

*Proof.* Let  $\varphi : Q \rightarrow \mathcal{U}$  the homomorphism of Theorem 1. As  $\mathcal{A}$  is irreducible in  $L$  so is irreducible in  $L(\mathcal{A})$ . As  $\mathcal{A}$  does not have mergible states  $\varphi$  is injective [4], then given states  $p$  and  $q$  in  $Q$  with  $p \neq q$  it holds that  $\varphi(p) \neq \varphi(q)$ . The automaton  $\varphi(\mathcal{A})$  induced in  $\mathcal{U}$  by  $\varphi(Q)$  is a subautomaton of  $\mathcal{U}$ . Let  $\delta'$  the transition function of  $\varphi(\mathcal{A})$  and let  $\delta''$  be the restriction of  $\delta'$  such that for every  $q$  in  $Q$  and  $a$  in  $\Sigma$ ,  $\delta''(\varphi(q), a) = \varphi(\delta(q, a))$ . The automaton  $\mathcal{B} = (\varphi(Q), \Sigma, \delta'', \varphi(I), \varphi(F))$  is isomorphic to  $\mathcal{A}$  and is a subautomaton of  $\mathcal{U}$ .

**Definition 2.** *A decomposition of a finite automaton  $\mathcal{A}$  is any collection  $(\mathcal{A}_i)_{i \in I}$  of subautomata of  $\mathcal{A}$  such that  $L(\mathcal{A}) = \bigcup_{i \in I} L(\mathcal{A}_i)$ .*

**Definition 3.** *Given a word  $x$ , we will denote  $\mathcal{A}_x$  the minimal deterministic finite automaton without useless states for the language  $\{x\}$ , that is,  $\mathcal{A}_x = (Q, \Sigma, \delta, I, F)$  where  $Q = \text{Pr}(x)$ ,  $I = \{\lambda\}$ ,  $F = \{x\}$  and for any  $u, ua$  in  $\text{Pr}(x)$   $\delta(u, a) = ua$ .*

**Definition 4.** *Let  $L$  be a language over the alphabet  $\Sigma$  and let  $x \in L$ . A subautomaton associated to  $x$  in  $L$  is any finite automaton  $\mathcal{A}$  obtained by means of any sequence of state merging in  $\mathcal{A}_x$  and such that  $\mathcal{A}$  is irreducible in  $L$ .*

It is clear that for every automaton  $\mathcal{A}$  associated to  $x$  in  $L$ ,  $x \in L(\mathcal{A})$ .

**Proposition 2.** *Let  $L$  be a regular language and  $x \in L$ . A subautomaton associated to  $x$  in  $L$  can be obtained knowing a finite number of words in  $\Sigma^* - L$ .*

*Proof.* Let  $x$  be a word in  $L$  and let  $u, v, z$  be a factorization of  $x$ , that is,  $x = uvz$ . If the states related to  $u$  and to  $uv$  can not be merged, any word in  $uv^*z - L$  will prevent to do so. If  $n$  is the number of states of  $\mathcal{A}_x$ , the number of words enough to guarantee that a subautomaton associated to  $x$  in  $L$  is obtained is bounded above by  $n^2$ .

**Proposition 3.** *Let  $L$  be a regular language. There exists a finite set  $M$  of words in  $L$  such that  $L$  is accepted by the NFA defined by the collection of subautomata associated to the words in  $M$ .*

*Proof.* As any subautomaton associated to any word in  $M$  is a subautomaton of  $\mathcal{U}$ , the collection of all the subautomata associated to the different words of  $L$  recognizes  $L$ . The finiteness is deduced from the fact that if  $\mathcal{A}$  is the subautomaton of  $\mathcal{U}$  associated to  $x$  in  $L$  and  $y \in L - L(\mathcal{A})$ , the subautomaton  $\mathcal{A}'$  associated to  $y$  is different from  $\mathcal{A}$ . Then, as the number of subautomata of  $\mathcal{U}$  is finite,  $M$  is finite.

## 4 A Family of Algorithms for Inference of $\mathcal{L}_3$ Using Non Deterministic Finite Automata

### 4.1 The Family WASRI

Based on the concepts that have been previously exposed, we describe now a family of algorithms that will be called *WASRI* (word associated subautomata regular inference). Every member of this family infers the class of regular languages in the limit and is described in Algorithm 1.

---

#### Algorithm 1. WASRI Scheme

---

**Input:**  $(D_+, D_-)$ .

**Output:** NFA consistent with  $(D_+, D_-)$ .

**Method:**

$\mathcal{A} = (Q, \Sigma, \delta, I, F)$  with  $Q = \delta = I = F = \emptyset$

**For every**  $x \in D_+$

**If**  $x \notin L(\mathcal{A})$

Obtain at least a finite automaton irreducible for  $x$  in  $\Sigma^* - D_-$ .

**For every**  $\mathcal{A}' = (Q', \Sigma, \delta', I', F')$  so obtained, where  $Q \cap Q' = \emptyset$

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

**End For**

**End For**

**Returns**  $(\mathcal{A})$

**End**

---

**Theorem 2.** *Any algorithm of the WASRI family infers the class of regular languages in the limit.*

*Proof.* Let  $L \subseteq \Sigma^*$  and suppose we have a complete presentation of  $L$ . From Proposition 3, there exists a finite set of words in  $L$  that give a collection of subautomata that recognizes  $L$ , and for every word, the number of words in  $\bar{L}$  needed to obtain those subautomata is also finite (Proposition 2), these two facts assure the convergence of the process, as after a finite amount of time those words will appear and  $L$  will be identified.

As it can be deduced from the above scheme, what will make the algorithms in *WASRI* differ from each other will be facts like:

- The number of subautomata which are inferred for each word.
- The order in the process of merging states in the way to obtain each subautomata.

Although it has been omitted in the above scheme an algorithm of the family may introduce a subsequent selection in the collection of automata obtained so far. For example, if we have obtained several automata for each word (using different merging order criteria), it is possible to select one or some of them (for example the smallest in size). Some of the automata in the output collection may also be eliminated, when the resulting automaton doing so still accepts  $D_+$ .

Let us see next the description of one of the most basic members of the family:

## 4.2 WASRI1

Let  $D = (D_+, D_-)$  be a complete sample of the language, with  $D_+ = \{x_1, \dots, x_n\}$ . The Algorithm 2 describes one of the elements of the family, that will be called *WASRI1*.

*WASRI1* obtains a finite automaton  $\mathcal{A}$  compatible with  $(D_+, D_-)$  in the following way: For every word  $x \in D_+$  which is not recognized by the current automaton, it starts building  $\mathcal{A}_x$  and obtains a subautomaton associated to  $x$  merging states in  $\mathcal{A}_x$  following the lexicographical order, under the condition that the state merging does not make the resulting automaton to accept any of the negative samples. The subautomaton for that word is added to the current one.

Finally, it checks if the automaton resulting from the deletion of the subautomata obtained for any of the words of  $D_+$  still recognizes  $D_+$ . If that is the case the subautomaton is effectively deleted.

The complexity of the algorithm *WASRI1* is  $kn^2|D_-|$ , where  $k$  is an integer,  $n$  is the length of the longest word of  $D_+$  and  $|D_-|$  is the sum of the lengths of the negative words of the sample. It is worth to mention that the temporal complexity depends on the length of every word and not on the sum of the lengths of the input (i. e. the size of the prefix tree acceptor of the sample). This fact makes *WASRI1* to run much faster, under the same conditions, than the rest of the algorithms that will be compared to it in this work.

If  $\mathcal{A} = (Q, \Sigma, \delta, I, F)$  and  $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, I_i, F_i)$  by  $\mathcal{A} = \mathcal{A} \cup \mathcal{A}_i$  we mean that the automaton  $\mathcal{A}$  becomes  $(Q \cup Q_i, \Sigma, \delta \cup \delta_i, I \cup I_i, F \cup F_i)$ . Under the same conditions as before, by  $\mathcal{A} = Delete(\mathcal{A}, \mathcal{A}_i)$ , we mean that the automaton  $\mathcal{A}$  becomes  $(Q - Q_i, \Sigma, \delta - \delta_i, I - I_i, F - F_i)$ .

**Algorithm 2****WASRI1**( $D_+, D_-$ ) $\mathcal{A} = (Q, \Sigma, \delta, I, F)$  with  $Q = \delta = I = F = \emptyset$  $list = \emptyset$ **For**  $i = 1$  **to**  $i = n$ **If**  $x_i \notin L(\mathcal{A})$  $\mathcal{A}_i = \mathcal{A}_{x_i}$  with  $Q_i = \{q_{i_0}, \dots, q_{i_m}\} = Pr(x_i)$ **For**  $j = 1$  **to**  $j = m$ **For**  $k = 0$  **to**  $k = j - 1$ **If**  $q_{i_k} \in Q_i$ **If**  $L(merge(\mathcal{A}_i, q_{i_k}, q_{i_j})) \cap D_- = \emptyset$ **Then**  $\mathcal{A}_i = merge(\mathcal{A}_i, q_{i_k}, q_{i_j})$ **End If****End If****End For****End For** $\mathcal{A} = \mathcal{A} \cup \mathcal{A}_i$  $list = Add(list, \mathcal{A}_i)$ **End For****For**  $i = 1$  **to**  $Length(list)$ **If**  $D_+ \subseteq L(Delete(\mathcal{A}, \mathcal{A}_i))$  **Then**  $\mathcal{A} = Delete(\mathcal{A}, \mathcal{A}_i)$ **End For****Return**( $\mathcal{A}$ )**End**

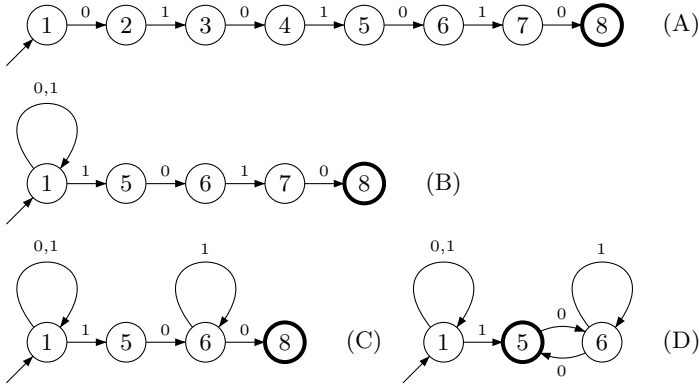
## 5 Two Examples

We present in this section two examples of run for better understanding of *WASRI1*, the first one describes most of the situations that may appear in it, that is, besides the merging of states it reflects the fact that some of the input words may be recognized by the previous automaton and also the fact that some of the subautomata may be deleted and the automaton still recognizes  $D_+$ . The second is an example to show that this algorithm may treat some situations in a much more efficient way than the rest of algorithms that have been compared with it.

### 5.1 Example of Run

Let us suppose that the input to *WASRI1* is  $D_+ = \{0, 000011, 001, 0101010\}$  and  $D_- = \{01000010\}$ .

We will describe the process of the word 0101010 as if this word were the first input to *WASRI1*. The automaton  $\mathcal{A}_x$  for  $x = 0101010$  is depicted in Fig. 1 (A). States 2,3 and 4 can be merged with state 1 as the resulting automaton does not accept the negative sample. The output to these first merges is in Fig. 1 (B). The next states that can be merged in lexicographical order are states 6 and 7, as the previous possible merges would give an automaton that accepts the

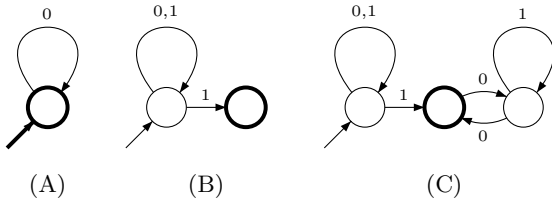


**Fig. 1.** Successive merges done by algorithm *WASRI1* while processing the word 0101010 on input  $D_+ = \{0101010, 0, 001, 000011\}$  and  $D_- = \{01000010\}$

negative sample. The resulting automaton is Fig. 1 (C). Finally, merging states 5 and 8 gives automaton in Fig. 1 (D), which is minimal for this word.

Considering the complete input  $D_+ = \{0, 000011, 001, 0101010\}$ , the automaton for the first word 0 is in Fig. 2 (A), while Fig. 2 (B) and (C) are respectively the outputs for the words 000011 and 0101010. You should observe at this point that the word 001 produces no output as it is recognized by the current automaton so far (Fig. 2 (A) and (B)).

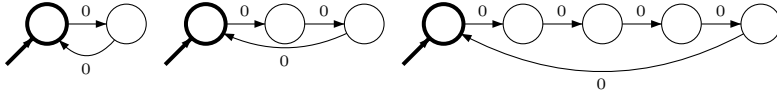
Finally, as the deletion of the automaton corresponding to the word 000011 still recognizes  $D_+$ , *WASRI1* algorithm outputs automata (A) and (C) in Fig.2.



**Fig. 2.** Non deterministic automata on input  $D_+ = \{0, 000011, 001, 0101010\}$  and  $D_- = \{01000010\}$  before deleting superfluous automata.

### 5.2 A Nice Example

Let us suppose that the target language is  $L = \{x \in 0^* : |x| \text{ is a multiple of } 2, 3 \text{ or } 5\}$ . The minimal *DFA* for  $L$  has the same number of states than its canonical *RFSA* (30 states). If the input for *WASRI1* algorithm is  $D_+ = \{0^2, 0^3, 0^5\}$  and  $D_- = \{0, 0^{11}\}$ , it outputs the automaton depicted in Fig. 3 which recognizes  $L$ , while algorithms like *RPNI* or *DeLeTe2* are far away from convergence with this input.



**Fig. 3.** Output automata given by WASRI on input  $D_+ = \{0^2, 0^3, 0^5\}, D_- = \{0, 0^{11}\}$

## 6 Experiments

The aim of the experiments is to analyze the behavior of one of the implementations of the algorithm *WASRI* and to compare it with the *DeLeTe2* algorithm, which has been reported as to have a better behavior than the classical inference algorithms if the target automata are randomly generated as NFAs. We also include the recognition rates obtained by *RPNI* algorithm using the same set of experiments. Both the training/test samples and the *DeLeTe2* program used in these experiments are provided by their authors and are available at Aurélien Lemay’s web page <http://www.grappa.univlille3.fr/~lemay/>.

Two kinds of experiments are reported in Table 1, depending on the source of the training and test samples: *er\_\** if they come from regular expressions and *nfa\_\** from NFAs. The number in the identifier of the experiment represents the number of training samples. Each experiment consists of 30 different languages to be learned and has 1000 test samples. Table 1 reports the recognition rate and the average size of the inferred hypothesis. These results are calculated as follows: each test sample is presented to the inference program, the program tags the sample as belonging to the target language or not, if this classification agrees with the real sample tag, the sample is considered correct and increases a counter; at the end, the number of correct samples is divided by 1000 (the total of test samples) and this value is reported as recognition rate. The average size is computed adding up the number of states of the 30 hypothesis generated in each experiment and dividing by 30.

The algorithm implemented for experiments obtains, for every word in  $D_+$ , two automata: The first one is obtained applying *WASRI* (the order is lexicographical and every state tries to get merged with the previous ones). The

**Table 1.** Inference results with *RPNI*, *DeLeTe2* and *Wasri* algorithms

Iden.	<i>RPNI</i>		<i>DeLeTe2</i>		<i>WASRI</i>	
	Recogn. rate	Avg. size	Recogn. rate	Avg. size	Recogn. rate	Avg. size
er_50	76.36%	9.63	81.3%	32.43	89.15%	15,93
er_100	80.61%	14.16	91.4%	30.73	93.0%	25,36
er_150	84.46%	15.43	92.0%	60.96	95.88%	25,73
er_200	91.06%	13.3	95.7%	47.73	95.79%	35,5
nfa_50	64.8%	14.3	69.3%	71.26	74,76%	39,3
nfa_100	68.25%	21.83	74.4%	149.13	76.46%	79,83
nfa_150	71.21%	28.13	76.7%	218.26	77.27%	121,1
nfa_200	71.74%	33.43	78.9%	271.3	81.16%	148,13

second uses the same algorithm with an inverse order (the word  $x_i$  corresponds to state  $q_{i_0}$  and  $\lambda$  to state  $q_{i_m}$ ). Once the whole set  $D_+$  has been processed, the algorithm outputs the automata with less number of states.

Table 1 resumes the recognition rates and the average size of the automata obtained by the three algorithms to be compared, that is, *RPNI*, *DeLeTe2* and *WASRI*. As it can be seen, the recognition rates obtained by *WASRI* are higher than those obtained by the other algorithms. It can also be seen that the number of states of the automata obtained by *WASRI* is smaller than those obtained by *DeLeTe2*.

## 7 Conclusions

We describe in this paper a family of algorithms that, each one of them, infers the class of regular languages in the limit. We have made the same experiments as in [3] to compare the error rate and the size of the output automata of one of the algorithms of the family, the *WASRI*, with the results obtained by the algorithm *DeLeTe2*. The results obtained by *WASRI* are better both in error rate as in smaller size of the output.

Some work needs to be done to complete the comparisons. It has been reported in [3] that *RPNI* behaves better than *DeLeTe2* when the source of the target language is generated in a deterministic way. The comparisons between *RPNI* and *WASRI* remain to be done with this type of source. Also, some effort to characterize the type of the output produced by our algorithm would probably lead us to determine some languages for which its performance might not be so good.

## References

1. Arnold, A. Dicky, A. Nivat, M. *A note about minimal non-deterministic automata*. Bull. EATCS 47, pp 166-169, 1970.
2. Carrez, C. *On the minimalization of non-deterministic automata*. Laboratoire de Calcul de la Faculté des Sciences de L'Université de Lille, 1970.
3. Denis, F. Lemay, A. and Terlutte, A. *Learning regular languages using RFSA's*. Theoretical Computer Science 313(2), pp 267-294 (2004).
4. García, P. and Vazquez de Parga, M. *A Note about mergible states in large NFA*. Bull. of the EATCS 87 pp. 181-184 (2005).
5. Gold, E.M. *Language identification in the limit*. Information and Control 10, pp 447-474 (1967).
6. Gold, E.M. *Complexity of Automaton Identification from Given Data*. Information and Control 37, pp 302-320 (1978).
7. Lombardy, S. *Approche structurelle de quelques problèmes de la théorie des automates*. Ph.D. Thesis, Ecole N.S. des Télécommunications. (2001).
8. Oncina, J. and García, P. *Inferring Regular Languages in Polynomial Updated Time*. In Pattern Recognition and Image Analysis. Pérez de la Blanca, Sanfeliú and Vidal (Eds.) World Scientific (1992).
9. Polák, L. *Minimalizations of NFA using the universal automaton*. LNCS 3317 pp 325-326 (2005).