

Non Deterministic RPNI *

Pedro García, José Ruiz and Antonio Cano

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

email: {pgarcia, jruiz, acano}@dsic.upv.es

Abstract

A Residual Finite State Automaton (*RFSA*) [2],[3],[4] is a Nondeterministic Finite Automaton (*NFA*) in which every state corresponds to a residual language of the language it recognizes. We describe in this paper an algorithm that, for a given sample D , infers a *RFSA* consistent with D that converges to the canonical *RFSA* of the target language using a generalization of *RPNI* algorithm. As *NFA*'s are sometimes smaller than their equivalent deterministic minimal automaton, the characteristic sample required for inference may also be smaller, so this algorithm may be useful. We also describe how to use this algorithm to infer the universal automaton for a language.

1 Introduction

A main topic in Grammatical Inference, used in Pattern Recognition, is the identification of Finite automata from samples of their behavior, that is, from words such that the membership to the target language is known.

Although the general problem of determining the existence of a deterministic finite automaton (*DFA*) with a number of states less than a given number n consistent with a given sample D has proved to be *NP*-complete by Gold [6], under certain conditions of representativeness, two algorithms solve the problem in deterministic polynomial time. A unified description and a comparison between both algorithms can be found in [5].

The first algorithm was proposed by Trakhtenbrot and Barzdin [15] as a contraction procedure in a finite tree that represents a set that contains all the words up to a certain length. If this set is somehow characteristic for an unknown regular language L , the algorithm finds the smallest *DFA* that recognizes L . This algorithm was rediscovered and used for grammatical inference by Gold [6]. Then, characteristic samples are not required as input, and consistency is

*Work partially supported by Ministerio de Ciencia y Tecnología under project TIC2003-09319-C03-02

guaranteed in this case at the cost of outputting the prefix tree acceptor (*PTA*) of the positive sample.

The second is due to Oncina and García [13] by the one hand and to Lang [8] by the other. It is known in the literature as the *RPNI* (Regular Positive and Negative Inference) algorithm and, on input of a sample, outputs a *DFA* consistent with the input in polynomial time. It builds the prefix tree acceptor of the sample, merges states in lexicographical order and propagates the merges to keep a deterministic automaton under the condition that it does not accept a negative sample.

Both algorithms are somehow based in Nerode's theorem, but experiments done [5] show that *RPNI* algorithm raises convergence more rapidly than Gold's algorithm as it makes better use of the information it has.

Variations of the *RPNI* algorithm (modifications in the way the states are merged) have been proposed [9] in order to gain efficiency, although theoretical aspects, like convergence, of those algorithms have not been reported.

F. Denis et al. [2], [3], [4] define the concept of Residual Finite State Automata (RFSAs) as a nondeterministic automata such that every state is associated to a residual language of the language it recognizes. They propose the algorithm DeLeTe2 such that for a sample of the target language L , converges to the canonical RFSAs that accepts L . The only difficulty is that the output of the algorithm may not be consistent with the input sample if the sample is not characteristic.

The use of a RFSAs representation for target languages may be interesting as the size of a RFSAs is sometimes smaller than the minimal DFA for the same language.

By the other hand, the concept of *universal automaton of a language* was implicitly introduced by Conway [1]. Sakarovitch and Lombardy [11], [12] have used this concept for the construction of reversible automata for reversible languages and also in the study of star height of rational languages.

Pólak [14] studies the universal automaton of a language in connection with the syntactic semiring of a language and with conjunctive varieties.

The universal automaton of a language, as we will see, is closely related to concepts like the canonical and the saturated RFSAs.

We propose in this paper three different algorithms. The first one is a nondeterministic version of Gold's algorithm. If we emit a new hypothesis every time we add a new state to the table we obtain the algorithm DeLeTe2 [4]. The second is a nondeterministic version of RPNI (NRPNI). It outputs a RFSAs which is always consistent with the sample and converges to the canonical RFSAs that accepts the target language. Thirdly, we adapt RPNI algorithm (NRPNI'), it has the same scheme that RPNI, but it uses a function that realizes nondeterministic merging of states. If we compute all the residuals it can be used to infer the universal automaton of the target language.

The leading idea of this paper was the conjecture that the advantage of the performance (both consistency and number of samples needed to converge) of RPNI with respect to Gold's could be translated to the inference of RFSAs. Facts like consistency and convergence of some of the proposed algorithms are

straight forward.

The rest of the paper is structured as follows, Section 2 contains the basic definitions and notation used. Section 3 contains a description of Gold's algorithm as well as a nondeterministic version of it and of how it can be modified to obtain the algorithm DeLeTe2, an example is given for better understanding. Section 4 contains a nondeterministic version of RPNI (NRPNI) algorithm. Section 5 contains a description of the modifications to be made in the NRPNI algorithm to be used to infer the universal automaton of a language. A section with the conclusions ends the paper.

2 Definitions and notation.

We suppose that the reader is familiar with the rudiments of formal languages. For further details the reader is referred to [7]. Definitions and previous works concerning RFSA's can be found in [2], [3] and [4].

2.1 Formal languages

Let A be a finite alphabet and let A^* be the free monoid generated by A with concatenation as the internal law and ε as neutral element. A *language* L over A is a subset of A^* . The elements of L are called *words*. Given $x \in A^*$, if $x = uv$ with $u, v \in A^*$, then u (resp. v) is called *prefix* (resp. *suffix*) of x . $\text{Pr}(L)$ (resp. $\text{Suf}(L)$) denotes the set of prefixes (suffixes) of L .

A *nondeterministic finite automaton* (*NFA*) is a 5-tuple $\mathcal{A} = (Q, A, \delta, Q_0, F)$ where Q is the (finite) set of states, A is a finite alphabet, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states and δ is a partial function that maps $Q \times A$ in 2^Q . The extension of this function to words is also denoted as δ . A word x is accepted by \mathcal{A} if $\delta(Q_0, x) \cap F \neq \emptyset$. The set of words accepted by \mathcal{A} is denoted by $L(\mathcal{A})$. A *deterministic finite automaton* (*DFA*) is a *NFA* such that $|Q_0| = 1$ and $|\delta(q, a)| \leq 1, \forall q \in Q, \forall a \in A$. Two automata are *equivalent* if they recognize the same language.

Given a finite set of words D_+ , the *prefix tree acceptor of S* is defined as the automaton $\mathcal{A} = (Q, A, \delta, q_0, F)$ where $Q = \text{Pr}(S)$, $q_0 = \varepsilon$, $F = D_+$ and $\delta(u, a) = ua, \forall u, ua \in Q$.

A Moore machine is a 6-tuple $M = (Q, A, B, \delta, q_0, \Phi)$, where A (resp. B) is the input (resp. output) alphabet, δ is a partial function that maps $Q \times A$ in Q and Φ is a function that maps Q in B called *output function*. A nondeterministic Moore machine is defined in a similar way except the fact that δ maps $Q \times A$ in 2^Q .

The behavior of M is given by the partial function $t_M : A^* \rightarrow B$ defined as $t_M(x) = \Phi(\delta(q_0, x))$, for every $x \in A^*$ such that $\delta(q_0, x)$ is defined.

Given two finite sets of words D_+ and D_- , we define the (D_+, D_-) -*prefix Moore machine* ($PTM(D_+, D_-)$) as the Moore machine having $B = \{0, 1, ?\}$, $Q = \text{Pr}(D_+ \cup D_-)$, $q_0 = \varepsilon$ and $\delta(u, a) = ua$ if $u, ua \in Q$ and $a \in A$. For every

state u , the value of the output function associated to u is 1, 0 or ? (undefined) depending whether u belongs to D_+ , to D_- or to $Q - (D_+ \cup D_-)$.

A Moore machine $M = (Q, A, \{0, 1, ?\}, \delta, q_0, \Phi)$ is *consistent* with (D_+, D_-) if $\forall x \in D_+$ we have $\Phi(x) = 1$ and $\forall x \in D_-$ we have $\Phi(x) = 0$.

2.2 Canonical RFSA and universal automaton

The *left quotient* of a language L by a word u , also called *residual language* of L associated to u is $u^{-1}L = \{v \in A^* : uv \in L\}$. The same definition for languages is $L_1^{-1}L_2 = \{v \in A^* : \exists u \in L_1, uv \in L_2\}$. A residual language $u^{-1}L$ is *composite* if $u^{-1}L = \cup\{v^{-1}L : v^{-1}L \subsetneq u^{-1}L\}$. A residual language is *prime* if it is not composite.

A Residual Finite State Automaton (RFSA) \mathcal{A} is a NFA such that every state defines a residual language of $L(\mathcal{A})$.

Given a language $L \subseteq A^*$ the *canonical* RFSA of L is the automaton $\mathcal{A} = (Q, A, \delta, Q_0, F)$ where:

- $Q = \{u^{-1}L : u^{-1}L \text{ is prime}, u \in A^*\}$
- $Q_0 = L$
- $F = \{u^{-1}L \in Q : \varepsilon \in u^{-1}L\}$
- $\delta(u^{-1}L, a) = \{v^{-1}L \in Q : v^{-1}L \subseteq (ua)^{-1}L\}$

Given a language $L \subseteq A^*$ we define the set of all possible intersections of its residuals, that is, $U = \{u_1^{-1}L \cap \dots \cap u_k^{-1}L : k \geq 1, u_1, \dots, u_k \in A^*\}$. Note that U contains the set all the residuals of L .

The *universal automaton* for L is a NFA defined as $\mathcal{U} = (U, A, \delta, Q_0, F)$ where:

- $Q_0 = \{P \in U : P \subseteq L\}$
- $F = \{P \in U : \varepsilon \in P\}$
- $\delta(P, a) = \{P' \in U : P' \subseteq (a)^{-1}P\}$

Automaton \mathcal{U} recognizes L and is such that any nondeterministic automaton that accepts a subset of \mathcal{U} can be projected to \mathcal{U} by means of a homomorphism [10]. Note that if \mathcal{A} is the canonical RFSA of L , the identity (that maps every residual to itself) is an homomorphism of automata, that is, the canonical RFSA is the subautomaton of \mathcal{U} induced by the prime residuals of L .

Two relations between the states of an automaton link RFSA's with grammatical inference. Let $D = (D_+, D_-)$ a sample, let $u, v \in Pref(D_+)$. We say that $u \prec v$ if no word w exists such that $uw \in D_+$ and $vw \in D_-$. We say that $u \simeq v^1$ if $u \prec v$ and $v \prec u$.

¹This relation is known in the terminology set up by Gold as *not obviously different states*.

3 Gold's algorithm. A nondeterministic version

We will briefly describe the algorithm proposed by Gold [6], based in the so called states characterization matrix. A complete description of it, both in terms of matrixes and a merging states version can be found in [5].

Definition 1 A states characterization matrix over an alphabet A is a triple (S, E, T) where S, E are finite subsets of A^* and $T : (S \cup SA)E \rightarrow \{0, 1, ?\}$. The elements of S are called states, while those of E are called experiments. If there exists $u \in (S \cup SA)$ and $v \in E$, $x = uv$ and $x \in D_+$ (resp. D_-), then $T(uv) = 1$ (resp. $T(uv) = 0$). For the rest of the words we set $T(uv) = ?$.

Every element u of $(S \cup SA)$ defines a row which will be called $row(u)$. Given $u, v \in (S \cup SA)$, we say that $row(u)$ is *obviously different* from $row(v)$, and we write $row(u) \not\approx row(v)$, if there exists an experiment $e \in E$ such that $T(ue), T(ve) \in \{0, 1\}$ and $T(ue) \neq T(ve)$. Otherwise we write $row(u) \approx row(v)$. We say that $row(u) \prec row(v)$ if there does not exist $e \in E$ such that $T(ue) = 1$ and $T(ve) = 0$.

A state characterization matrix is called *closed* if neither row belonging to $SA - S$ is obviously different from all the rows of S .

Algorithm 1 Gold (D_+, D_-)

Begin

$S = \{\varepsilon\};$

$E = \text{Suf}(A^{-1}(D_+ \cup D_-));$

Build the table $(S, E, T);$

While there exists $s' \in SA - S$ such that $\forall s \in S, row(s) \not\approx row(s')$

 Choose s' ; //The first one in lexicographical order//

$S = S \cup \{s'\};$

 Update (S, E, T)

End While;

$Q = S; q_0 = \varepsilon; (*)$

For $s \in S$

$s \in F$ iff $T(s) = 1;$

For $a \in A$

If $sa \in S$ **Then** $\delta(s, a) = sa$

Else $\delta(s, a) =$ any s' such that $row(sa) \approx row(s')$

 //The first one in lexicographical order//

End For

End For;

$\mathcal{A} = (Q, A, \delta, q_0, F);$

If M is consistent with (D_+, D_-) **Then** Return (\mathcal{A})

Else Return $(PTA(D_+, D_-))$

End.

The initial states characterization matrix is established as the triplet $(S, E, T) = (\{\varepsilon\}, E = \text{Suf}(A^{-1}(D_+ \cup D_-)), T)$.²

The algorithm begins by setting $S = \{\varepsilon\}$. We have to maintain two sets of words (each word defines a row), the set S and the set $SA - S$. At every step we have to move from the set $SA - S$ to the set S one of the rows which is obviously different from the row of every word of S . We have then to update the set $SA - S$ and this operation has to be done until neither word can be moved from $S\Sigma - S$. The output of the algorithm is an automaton obtained from the state characterization matrix. If this automaton is not consistent with the data, the algorithm outputs the automaton $PTA(D_+, D_-)$.

3.1 Example

Let $D = (D_+, D_-)$, were $D_+ = \{\varepsilon, 00, 11, 010, 10\}$ and $D_- = \{0, 1, 001\}$. The initial states characterization matrix is the table of Figure 1 (left).

As $0 \not\prec \varepsilon$, we add 0 to the S part of the table and obtain $SA - S$. Then we obtain the table of Figure 1 (right).

		ε	0	1	01	10
S	ε	1	0	0	?	1
$SA - S$	0	0	1	?	0	1
	1	0	1	1	?	?

		ε	0	1	01	10
S	ε	1	0	0	?	1
	0	0	1	?	0	1
$SA - S$	1	0	1	1	?	?
	00	1	?	0	?	?
	01	?	1	?	?	?

Figure 1: Initial (left) and final (right) states characterization matrixes used by Gold's algorithm on input $D_+ = \{\varepsilon, 00, 11, 010, 10\}$ and $D_- = \{0, 1, 001\}$

The matrix is now closed as $0 \simeq 1$, $00 \simeq \varepsilon$ and $01 \simeq 0$. The algorithm would try to output the automaton depicted in Figure 2 (left). As it is not consistent with the sample (it accepts 11), outputs the prefix tree acceptor of the sample instead.

3.2 Non Deterministic Gold and DeLeTe2 algorithms

Some modifications have to be done in Gold's algorithm to obtain RFSA's. They affect to the set of initial states and to the establishment of the transitions between states.

The following changes should be made in **Gold** (D_+, D_-) algorithm:

- The part marked as (*) in the original algorithm has to be changed to $Q_0 = \{u \in S : u \prec \varepsilon\}$.

²In [6], it is only established that E must be *suffix-complete* ($\forall x \in E, \text{Suf}(x) \subseteq E$) and that it must be large enough to contain every word belonging to $D_+ \cup D_-$.

- $\delta(u, a) = \{v \in S : v \prec ua\}$.

With the same input as before, the output is depicted in Figure 2 (right). We see again that this output is not consistent with the input, as it accepts the negative sample 001.

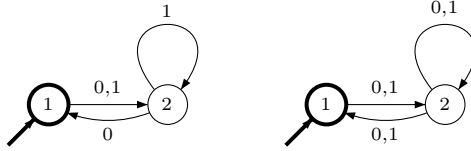


Figure 2: Output automata given by Gold’s algorithm (left) and by nondeterministic Gold’s algorithm (right) before checking for consistency.

If in Nondeterministic Gold’s algorithm we emit a hypothesis and check for consistency every time we obtain a new table, we obtain the algorithm DeLeTe2 proposed by Denis et al. [4].

4 NRPNI algorithm

The algorithm RPNI is based in deterministic merging of states (see the function *detmerge* [5]). States in the PTM are lexicographically ordered and in the current automaton we try to merge every state (following this order) with the previous ones (also in order) using the function *detmerge*. The process continues until there are no remaining states. The function *detmerge* guarantees the consistency of the intermediate automata and thus, of the output automaton.

In the algorithm we propose, the deterministic merging of states is replaced by a nondeterministic merging using the function *ndetmerge* we will next describe. During the process, in the sub-automaton of the explored states two types of transitions may appear

- Type *A* transitions, which are the ones that RPNI would create.
- Type *B* transitions, which are determined by the relation \prec .

In the Figures, type *B* transitions are depicted as dashed lines, whereas for type *A* transitions we will use continuous ones.

When we apply function *ndetmerge* to a pair of states, consecutive merges (folding) of states may appear only on type *A* transitions. New transitions produced by this folding, are also of type *A*.

The following algorithm describes the behavior of the function *ndetmerge*. State *va* (the first state which has not been explored) is such that *v* belongs to the explored part of the automaton. *P* is the set of explored states. It has been written in a way that we think it can be better understood. The algorithm does not reflect the situation where there is more than one state in *P* not distinguishable from state *va*. If this is the case the rest of the equivalent states give type *B* transitions -see the example of run and the automaton of Figure 4 (right).

Algorithm 2 *ndetmerge* (va, P, M)

Begin
 $List = \{\}$; $equiv = False$; $initial = False$;
For $u \in P$
 If $(u \prec va \wedge va \not\prec u)$
 Then $\delta = \delta \cup (v, a, u)$ //new type B transition//
 End If
 If $(u \not\prec va \wedge va \prec u)$
 Then Append u to $List$
 If $u = \varepsilon$ **Then** $initial = True$;
 End If
 If $(u \prec va \wedge va \prec u)$ // u and va are not distinguishable //
 Then $equiv = True$;
 $q = u$;
 End If
End For
If $equiv$
 Then $M = detmerge(q, va, M)$
 Else For $r \in List$
 For $(p, b, r) \in \delta$
 $\delta = \delta \cup (p, b, va)$ //new type B transition//
 End For
 End For
 If $initial$ **Then** $Q_0 = Q_0 \cup \{va\}$
End For
End.

Note also that when a state is input to function *ndetmerge* it is only needed one comparison with every previous state to decide the different conditions. Those conditions can be established making a virtual folding as function *detmerge* would do. When states u and va are compared, and the automaton is folded using type A transitions, either the pair (u, va) or their successors (let us denote a pair of successors by (p, q)) are such that if (p, q) represents the pair (u, va) or any of its followers when the folding is produced, then:

1. $u \prec va$ and $va \not\prec u$ if and only if there does not exist (p, q) with $\Phi(p) = 1$ and $\Phi(q) = 0$ and exists (p, q) with $\Phi(p) = 0$ and $\Phi(q) = 1$.
2. $va \prec u$ and $u \not\prec va$ if and only if there does not exist (p, q) with $\Phi(q) = 1$ and $\Phi(p) = 0$ and there exist (p, q) with $\Phi(q) = 0$ and $\Phi(p) = 1$
3. $u \prec va$ and $va \prec u$ if and only if there does not exist (p, q) with $\Phi(p) = 1$ and $\Phi(q) = 0$ and neither exists (p, q) with $\Phi(p) = 0$ and $\Phi(q) = 1$

Figures 3 to 5 show examples of application of function *ndetmerge*.

In order to design algorithm NRPNI it should be enough to change the function *detmerge* by *ndetmerge* in former RPNI. The algorithm obtained this way would converge to the saturated RFSA and not to the canonical RFSA. To do so it would be necessary to eliminate composite states. This also would slow the convergence as samples would be needed to distinguish every state of the minimal automaton of the target language.

This scheme will be used in section 6 to infer the universal automaton for a given sample and will be referred to as NRPNI. To obtain the canonical RFSA, the algorithm will have to be modified as follows:

Algorithm 3 NRPNI (D_+, D_-)

Begin

$M_0 = PTM(D_+, D_-)$; // $M_0 = (Q, A, \{0, 1, ?\}, \delta, \{\varepsilon\}, \Phi_0)$ //

$S = \{\varepsilon\}$; $M = M_0$;

Repeat

For $v \in SA - S$

$M = ndetmerge(v, S, M)$;

End For

If M is consistent with (D_+, D_-)

Then Return (M);

Else $S = S \cup \{w\}$ // w is the first state in $SA - S$ which is distinguishable from every state in S //

End If

End.

4.1 Example of run of function *ndetmerge*

Let us suppose that M is the Moore Machine of Figure 3. The stable part of the automaton is the set $\{1, 2, 3\}$.

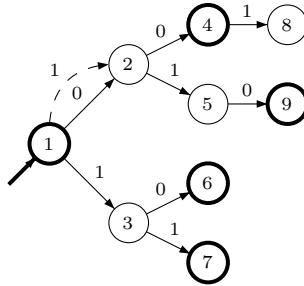


Figure 3:

The function $ndetmerge(4, \{1, 2, 3\}, M)$ is called. It finds the relations $1 \prec 4 \wedge 4 \prec 1$ and $2 \prec 4 \wedge 4 \not\prec 2$, so $q = 1$ and loop of type B around state 2 is

added. As $3 \not\prec 4 \wedge 4 \not\prec 3$ we add the type *A* transition from state 2 to state 1 and thus, we obtain automaton of Figure 4 (left).

Now the function $ndetmerge(5, \{1, 2, 3\}, M)$ is called. It finds the relations $1 \not\prec 5 \wedge 5 \prec 1$, so $list = \{1\}$ and $2 \prec 5 \wedge 5 \prec 2$ and $3 \prec 5 \wedge 5 \prec 3$. Note that this is the special situation that we have not described in the algorithm for better understanding of it. It does $q = 2$ and states 2 and 5 are merged. The transition from state 2 to state 3 is of type *B*. We obtain automaton of Figure 4 (right).

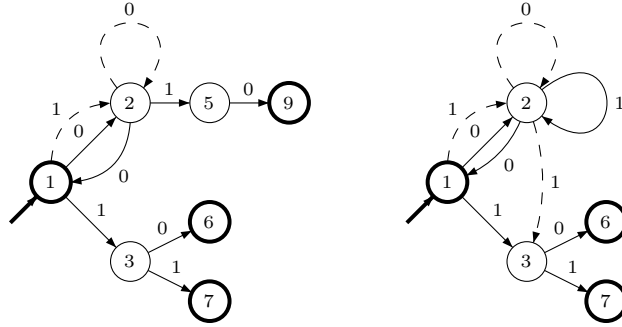


Figure 4:

At this point, the function $ndetmerge(6, \{1, 2, 3\}, M)$ is called. It finds the relations $1 \not\prec 6 \wedge 6 \prec 1$, so $q = 1$, then $2 \prec 6 \wedge 6 \not\prec 2$ so it adds type *B* transition from state 3 to state 2 labelled with 0. Then it finds $3 \prec 6 \wedge 6 \not\prec 3$, so it adds a loop around state 3 labelled with 0. The resulting automaton after this step is depicted in Figure 5 (left). The situation after $detmerge(1, 6, \mathcal{A})$ is depicted in 5 (right).

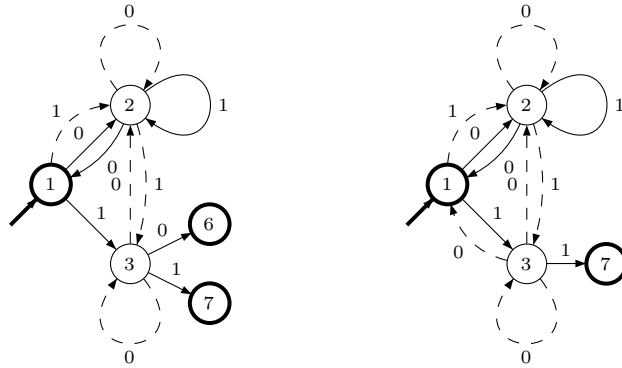


Figure 5:

4.2 Example of run of NRPNI

Let $D = (D_+, D_-)$, where $D_+ = \{\varepsilon, 00, 11, 010, 10\}$ and $D_- = \{0, 1, 001\}$. The prefix tree acceptor for that sample can be seen in Figure 6.

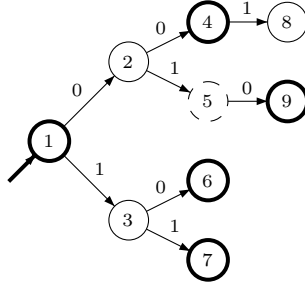


Figure 6: Prefix tree acceptor for the sample $D_+ = \{\varepsilon, 00, 11, 010, 10\}$ and $D_- = \{0, 1, 001\}$.

We use thick lines to design final states, thin lines for the non final and dashed lines for the non defined.

We first have $S = \{1\}$ and $S\Sigma - S = \{2, 3\}$. The merges $(1, 2)$ and $(1, 3)$ can not be fulfilled, as state 1 is final, whereas 2 and 3 are not, then we output automaton of Figure 7 (left) which is not consistent. Thus, we have now $S = \{1, 2\}$ and $S\Sigma - S = \{3, 4, 5\}$. States 3 and 2 are equivalent and they are merged giving automaton of Figure 7 (right). Note that the output function of state 5 in the Moore machine is now positive.

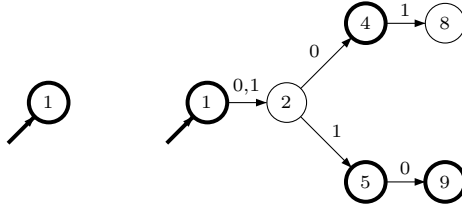


Figure 7: Output automata when $S = \{1\}$ (left) and after merging states 2 and 3 (right).

We find now that 4 and 1 are equivalent, while 4 and 2 are not. The merging of them outputs the automaton of Figure 8 (left).

The relations $1 \prec 5$ and $2 \prec 5$ add new transitions (the ones with dashed lines) to the former automaton, giving the automaton of Figure 8 (right).

Now the algorithm checks for consistency with states 1 and 2 and outputs automaton of Figure 9.

4.3 Convergence of NRPNI

The convergence of NRPNI algorithm relies in the assertion that any characteristic sample for RPNI makes NRPNI to converge also. In fact, for a target automaton, if we call S_P (short prefixes) to the set of the smallest words in lexicographical order that raises every state from the initial one, and K (kernel)

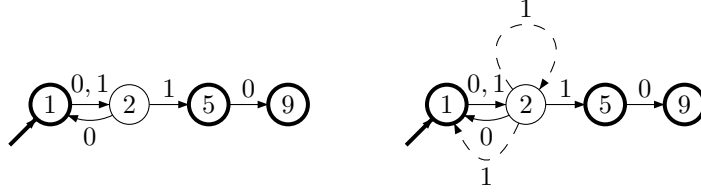


Figure 8: Output automata after merging states 4 and 1 (left) and after detecting the relations $1 \prec 5$ and $2 \prec 5$ (right).

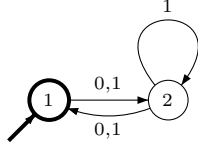


Figure 9: Output automata for NRPNI algorithm after checking for consistency.

to the set $S_P \cup S_P A$, where A is the alphabet, the characteristic sample must contain the words that:

- For every element of K there must be a completion in the target language (in the case that the word raises a final state, this completion must be ε).
- For every pair $(u, v) \in S_P \times K$ there must be a word w such that $(uw, vw) \in (D_+ \times D_-) \cup (D_- \times D_+)$.

To distinguish states u, v such that $u \prec v$ ($L_u \subseteq L_v$) we must have a word w such that $vw \in D_+$ and $uw \in D_-$ and we must never have w such that $uw \in D_+$ and $vw \in D_-$. So if two states are distinguished with respect to equivalence, they are also distinguished with respect to inclusion.

The worst case is the one in which there are not composite states. In this case it will be necessary to distinguish all the states. The convergence of RPNI guarantees this fact.

5 Inference of the universal automaton

The identification of regular languages through the universal automaton can be done either using nondeterministic Gold's algorithm or using NRPNI' both algorithm converge to the saturated RFSA of the language (the subautomaton induced by states 1, 2, 3, 4 and 5 of the Figure 11) and thus, the only thing to do is to compute the intersection of the residuals. The following example shows how to obtain the universal automaton from a given sample.

5.1 Example

Let $D_+ = \{\varepsilon, 00, 10, 11, 000, 010, 100, 101, 110, 0010, 1001, 1010\}$ and $D_- = \{0, 1, 01, 001, 0001\}$. This sample is a characteristic sample for RPNI and for automa-

ton depicted in [4], page 272, Fig. 1 (left). To visualize the intersection of residuals we can dispose them in a table similar to the states characterization matrix:

	ε	0	1	00	01	10	001	010
ε	1	0	0	1	0	1	0	1
0	0	1	0	1	0	1	0	1
1	0	1	1	1	1	1	1	1
00	1	1	0	1	0	1	0	1
10	1	1	1	1	1	1	1	1
$\varepsilon \cap 0$	0	0	0	1	0	1	0	1

If we compute the intersections of the residuals (we just calculate the conjunction of the rows, element by element) we obtain: $\varepsilon \cap 1 = \varepsilon \cup 0$, $\varepsilon \cap 00 = \varepsilon$, $\varepsilon \cap 10 = \varepsilon$, $0 \cap 1 = 0$, $0 \cap 00 = 0$, $0 \cap 10 = 0$, $0 \cap 10 = 0$, $1 \cap 00 = 0$, $00 \cap 10 = 00$, $1 \cap 10 = 1$.

Thus, the only new state obtained (denoted as $\varepsilon \cap 0$) is then $\varepsilon^{-1}L \cap 0^{-1}L$.

The partial order relation scheme between the states of the universal automaton for the inferred language is depicted in Figure 10. Every transition that enters a state has to enter all the previous states with this partial order.

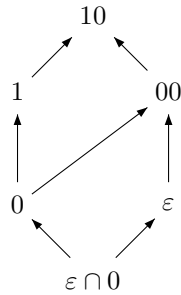


Figure 10: Scheme of the order relation between states of the universal automaton

Following this scheme with mentioned sample, we obtain automaton of Figure 11. The transitions added which are related to the new state ($\varepsilon \cap 0$) are depicted in Figure 12

6 Conclusions

Two algorithms solve the problem of identifying regular languages from a complete presentation of it, Gold and RPNI algorithms. While the later is always consistent, the former may not be (in this case it outputs the PTA of the positive sample). We have extended both algorithms to the nondeterministic case. In this case they infer RFSAs. The extension of Gold's algorithm modified in

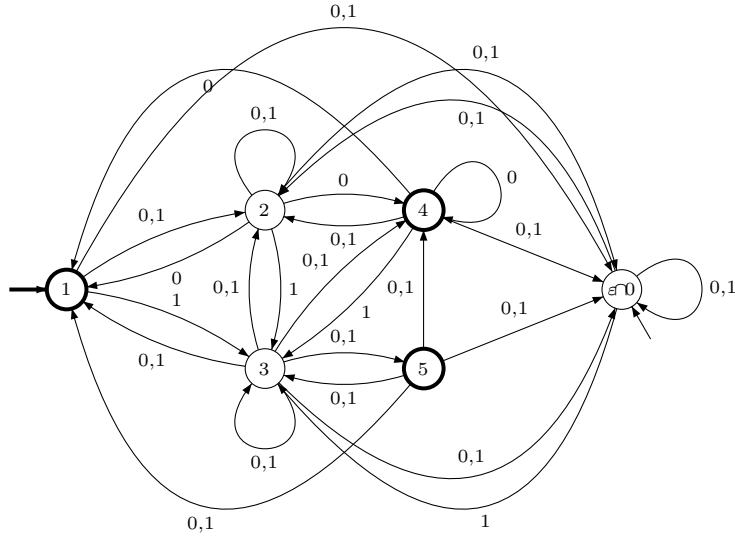


Figure 11: Universal automaton for the language obtained on input of sample $D_+ = \{\varepsilon, 00, 10, 11, 000, 010, 100, 101, 110, 0010, 1001, 1010\}$ and $D_- = \{0, 1, 01, 001, 0001\}$

the way that it emits a hypothesis with every new state and checks for consistency is the algorithm DeLeTe2 [4] and thus, may not be consistent, while the later always is. Denis et al. reported a consistent version of DeLeTe2, although they don't explain how they do it. Experiments should be done to compare the behavior of both algorithms. By the other hand, the main advantage of the use of RFSAs is the smaller size of them in comparison with DFAs.

References

- [1] Conway, J.H. *Regular Algebra and Finite Machines*. Chapman and Hall Ltd.(1971).
- [2] Denis, F. Lemay, A. and Terlutte, A. *Learning regular languages using nondeterministic finite automata*. A.L. Oliveira (Ed.), ICGI 2000, LNAI 1891, pp 39-50 (2000).
- [3] Denis, F. Lemay, A. and Terlutte, A. *Residual finite state automata*. In STACS 2001, LNAI 2010, pp 144-157 (2001).
- [4] Denis, F. Lemay, A. and Terlutte, A. *Learning regular languages using RFSAs*. Theoretical Computer Science 313, pp 267-294 (2004).
- [5] García, P. Cano, A. and Ruiz, J. *A comparative study of two algorithms for Automata Identification*. A.L. Oliveira (Ed.), ICGI 2000, LNAI 1891, pp 115-126 (2000).
- [6] Gold, M. *Complexity of Automaton Identification from Given Data*. Information and Control 37, pp 302-320 (1978).

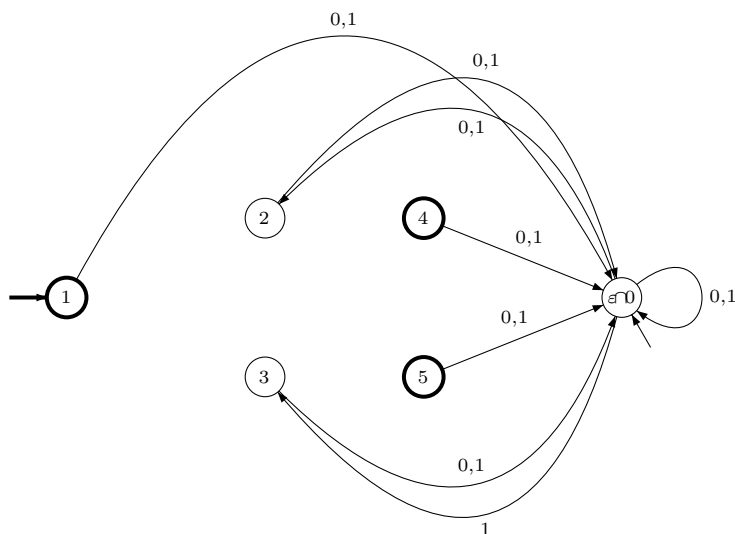


Figure 12: Transitions added in the universal automaton related to state $\varepsilon \cap 0$.

- [7] Hopcroft, J. and Ullman, J. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley (1979).
- [8] Lang, K.J. *Random DFA's can be Approximately Learned from Sparse Uniform Examples*. In Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, pp 45-52 (1992).
- [9] Lang, K.J. Pearlmutt, B.A. Price, R.A. *Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm*. In Proceedings of the ICGI 98, LNAI 1433 pp 1-12 (1998).
- [10] Lombardy, S. *Approche structurelle de quelques problèmes de la théorie des automates*. Ph D. Thesis. École nationale Supérieure des Télécommunications, Paris (2001).
- [11] Lombardy, S. *On the construction of reversible automata for reversible languages*. In Proceedings of ICALP. LNAI 2380 pp 170-182 (2002).
- [12] Lombardy, S. and Sakarovitch, J. *On the star height of rational languages. A new presentation for two old results*. Proc. 3rd International Colloquium on Words, Languages and Combinatorics, World Scientific (To appear).
- [13] 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).
- [14] Pólak, L. *Syntactic Semiring and Universal Automaton*. Z. Ésik and Z. Fülöp (Eds.), DLT 2003, LNCS 2710, pp 411-422 (2003).
- [15] Trakhtenbrot B. and Barzdin Ya. *Finite Automata: Behavior and Synthesis*. North Holland Publishing Company (1973).