Learning Commutative Regular Languages*

Antonio Cano Gómez and Gloria I. Álvarez

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia. Valencia (Spain)
Departamento de Ciencias e Ingeniería de la Computación,
Pontificia Universidad Javeriana Cali, calle 18 No. 118-250 Cali, Colombia
{acano,galvarez}@dsic.upv.es

Abstract. In this article we study the inference of commutative regular languages. We first show that commutative regular languages are not inferable from positive samples, and then we study the possible improvement of inference from positive and negative samples. We propose a polynomial algorithm to infer commutative regular languages from positive and negative samples, and we show, from experimental results, that far from being a theoretical algorithm, it produces very high recognition rates in comparison with classical inference algorithms.

1 Introduction

Regular languages are not inferable from positive samples is a well known result from Angluin [2]. This means that only inference from positive and negative samples is allowed. Nevertheless, the most useful algorithms for learning languages from positive and negative samples are not enough efficient to be applied in practice when looking at the recognition rate, mainly *RPNI* [11,10] and *red-blue* [4,9]. In [1] has been introduced the idea of trying to learn some subclasses of regular languages from positive and negative samples with the aim of improving the efficiency of the learning process, either if the considered subclass is not inferable from positive samples, the inference of those classes from positive and negative samples can be improved. In this article we study the class of commutative regular languages proposing a new algorithm called CRPNI and showing experimentally that CRPNI outperforms significantly recognition rates obtained with classical inference algorithms like red-blue and RPNI. Despite the algorithm given in this article follows the ideas of [1], actually both algorithms are very different in concepts and implementation.

In Section 2 we give the most important definitions about words and languages. We define in this sections the concepts of deterministic finite automata and Moore machine, that would be slightly modified for the commutative case in section 3.

Section 3 is devoted to the inference of regular commutative languages. First, we prove that commutative language are not inferable from positive samples,

* Work partially supported by Ministerio de Educación y Ciencia under project TIN2007-60760.

A. Clark, F. Coste, and L. Miclet (Eds.): ICGI 2008, LNAI 5278, pp. 71–83, 2008.

and then we give an algorithm for the inference of commutative regular languages from positive and negative samples. For this algorithm, first, we modify the definition of deterministic finite automata and Moore machine defining commutative deterministic finite automata and commutative Moore machine, these models consider the fact that languages with which we are working are commutative. We also prove that any commutative language can be recognised by a commutative deterministic finite automata. Nevertheless, the minimal deterministic finite automata is not equivalent to the minimal commutative deterministic finite automata. Finally we give our algorithm *CRPNI*, that is inspired in the RPNI [11] algorithm and we show its convergence in the limit.

In section 4 we analyse experimental results obtained from our implementation of the algorithm. We study two cases $|\Sigma| = 2$ and $|\Sigma| = 3$. In both cases we can see a great improvement of the CRPNI recognition rates with respect to two of the most used algorithms, namely redblue and RPNI.

2 Preliminaries

In this section we give the most important concepts used in this article. For further details about these concepts, the reader is referred to [8], [11] and [12]. In this paper we will use Moore machines in order to define the algorithm but also other machines like unbiased finite state automata [3] or the finite state classifiers [6] could be used. It could be interesting whether an implementation with those machines could improve the inference process.

2.1 Formal Languages and Automata

Throughout this paper if Σ denotes a finite alphabet, Σ^* denotes the free monoid generated by Σ with the concatenation as the internal law and λ as the neutral element. A *language* L over Σ is a subset of Σ^* and the elements of Σ^* are called *words*. Given $x \in \Sigma^*$, if $x = uv$ with $u, v \in \Sigma^*$, then u (resp. v) is called *prefix* (resp. *suffix*) of x. Given a language $L \in \Sigma^*$ we denote by $Pre(L)$ ($Suf(L)$) the set of prefixes (resp. suffixes) of all words in the language L. Given a total order $<$ on Σ we can define an order $<_{lex}$ on Σ^* by setting for two words $u, v \in \Sigma^*$, $u <_{lex} v$ if $|u| < |v|$ or $|u| = |v|$ and there exists $x, y_1, y_2 \in \Sigma^*$ such that $u = xay_1$ and xby_2 with $a < b$. Given a word w on an alphabet Σ and one letter $a \in \Sigma$, we denote by $\pi_a(w)$ the projection of w in a, for instance, if $w = abbcaa$, then $\pi_a(w) = aaa$. Given a language L we define the *syntactic left congruence* as the left-congruence on Σ^* defined as $u \sim_L v$ if and only if for any $x \in \Sigma^*, ux \in L \Leftrightarrow vx \in L$.

A *deterministic finite automaton* (DFA) is a 5-tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ if the set of final states and δ is a partial function that maps $Q \times \Sigma$ in Q, which can be easily extended to words. A word x is accepted by an automaton $\mathcal{A}$ if $\delta(q_0, x) \in F$. The set of words accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$.

Given a language L, the *minimal deterministic finite automaton* of L is the automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where $Q = \{[x]_{\sim_L} \mid x \in \Sigma^*\}$, $\delta([x]_{\sim_L}, y) = [xy]_{\sim_L}$ for any $x, y \in \Sigma^*$, $q_0 = [\lambda]_{\sim_L}$, $F = \{[x]_{\sim_L} \mid x \in L\}$.

A Moore machine is a 6-tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \Phi)$ where Σ (resp. Γ) is an input (resp. output) alphabet, δ is a partial function that maps $Q \times \Sigma$ in Q, and Φ is a function that maps Q in Γ called *output function*. The behaviour of $\mathcal{M}$ is given by the function $t_{\mathcal{M}} : \Sigma^* \to \Gamma$ defined as $t_{\mathcal{M}}(x) = \Phi(\delta(q_0, x))$ for every $x \in \Sigma^*$ such that $\delta(q_0, x)$ is defined.

Given a Moore machine $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \Phi)$ with $\Gamma = \{0, 1, \uparrow\}$ we can associate an automaton $\mathcal{A}_{\mathcal{M}} = (Q, \Sigma, \delta, q_0, F)$ where $F = \{q \in Q \mid \Phi(q) = 1\}$.

Given two finite sets of words D_+ and D_-, we define the (D_+, D_-)-*prefix tree machine* $(PTM(D_+, D_-))$ as the Moore machine having $\Gamma = \{0, 1, \uparrow\}$, $Q = Pre(D_+ \cup D_-)$, $q_0 = \lambda$ and $\delta(u, a) = ua$ if $u, ua \in Q$ and $a \in \Sigma$. For every state u, the value of the output function Φ associated to u is 1, 0 and $\uparrow$ (undefined) depending whether u belongs to D_+, to D_- or to the complementary set of $D_+ \cup D_-$.

A Moore Machine $\mathcal{M} = (Q, \Sigma, \Gamma, \cdot, q_0, \Phi)$ is consistent with (D_+, D_-) if for every $x \in D_+$ we have $\Phi(\delta(q_0, x)) = 1$ and for every $x \in D_-$ we have $\Phi(\delta(q_0, x)) = 0$.

2.2 Commutative Languages

Given two words u and v we say the they are commutatively equivalent if $u = a_1 a_2 \cdots a_n$ with $a_i \in \Sigma$ for $1 \leq i \leq$, and there exist a permutation σ on $\{1, 2, \ldots, n\}$ such that $a_{\sigma(1)} a_{\sigma(2)} \cdots a_{\sigma(n)} = v$. We denote it by $u \sim_{com} v$. For instance, $abca \sim_{com} cbaa$.

Given an alphabet Σ, a language L is commutative if and only if it is the union of some $\sim_{com}$-classes.

It seems to be some relation between *planar languages* and commutative languages [5] and their inference. An interesting work would be to compare the inference algorithm described in [5] and the CRPNI described here.

In this article we are interested in commutative regular languages. In order to describe the expressively of commutative regular languages we recall here a result from [12].

Proposition 1. (Pin) *For every alphabet Σ, the class of commutative languages of Σ is the boolean algebra generated by the languages of the form $K(a, r) = u \in \Sigma^* \mid |u|_a = r$, where $r > 0$ and $a \in \Sigma$, or $L(a, k, p^n) = \{u \in \Sigma^* \mid |u|_a \equiv k \mod p^n\}$, where $0 \leq k < p^n$, p is prime, $n > 0$ and $a \in \Sigma$*

Note that Proposition 1 show that the idea of inferring a language for each letter and trying to infer the whole language from those using boolean operation is not possible. For instance, $L = \{x \in \Sigma^* \mid |x|_a \equiv |x|_b \mod 2\}$ is a commutative language that can not be inferred in that way.

More equivalent definitions and proofs about commutative regular languages can be found in [12] and [13].

We now introduce the concept of commutative deterministic finite automata. Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. We define a *commutative deterministic finite automaton* (CDFA) on Σ as $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where $Q = Q_{a_1} \times Q_{a_2} \times \cdots \times Q_{a_n}$, $q_0 \in Q$, $F \subseteq Q$, and $\delta((q1, \ldots, q_i, \ldots, q_n), a_i) = (q1, \ldots, \delta_{a_i}(q_i, a_i), \ldots, q_n)$ where δ_{a_i} is a function from Q_{a_i} onto Q_{a_i} for $1 \leq i \leq n$.

Let $\Sigma = \{a_1, a_3, \ldots, a_n\}$ be an alphabet and let L be a commutative language on Σ. We define define the *minimal commutative automaton* of L as $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where $Q = Q_{a_1} \times \ldots \times Q_{a_n}$ being $Q_{a_i} = \bigcup_{m \leq 0} [a_i^m]_{\sim L}$ (note that this union is finite) for $1 \leq i \leq n$, $q_0 = ([\lambda]_{\sim_L}, \ldots, [\lambda]_{\sim_L})$, $F = \{(\pi_{a_1}(x), \ldots, \pi_{a_n}(x)) \mid x \in L\}$ and $\delta_{a_i}([a_i^m]_{\sim_L}, a_i) = [a_i^{m+1}]_{\sim_L}$ for $1 \leq i \leq n$.

Proposition 2. *For any regular commutative language L, the minimal commutative deterministic automaton is well defined and accepts L.*

Proof. Let $\Sigma = \{a_1, \ldots, a_n\}$ and let L be a commutative regular language. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be the minimal commutative finite automaton L. First, let $x \in L$, since L is commutative we have that $\pi_{a_1}(x)\ \pi_{a_2}(x) \cdots \pi_{a_n}(x) \in L$, then we have, by the definition of δ that $\delta(q_0, x) = \delta(([\lambda]_{\sim_L}, [\lambda]_{\sim_L}, \ldots, [\lambda]_{\sim_L}), x)$ $= ([\pi_{a_1}(x)]_{\sim_L}, [\pi_{a_2}(x)]_{\sim_L}, \cdots, [\pi_{a_n}(x)]_{\sim_L})$ and then by definition $([\pi_{a_1}(x)]_{\sim_L}$, $[\pi_{a_2}(x)]_{\sim_L}, \cdots, [\pi_{a_n}(x)]_{\sim_L})$ belongs to F, that is, x is accepted by A. Now, let x be a word accepted by A, then by definition we have that $\delta(q_0, x) = \delta(([\lambda]_{\sim_L}$, $[\lambda]_{\sim_L}, \ldots, [\lambda]_{\sim_L}), x) = ([\pi_{a_1}(x)]_{\sim_L}, [\pi_{a_2}(x)]_{\sim_L}, \cdots, [\pi_{a_n}(x)]_{\sim_L}) \in F$. Now, by the definition of F we have necessarily $\pi_{a_1}(x)\pi_{a_2}(x) \cdots \pi_{a_n}(x) \in L$ and since L is commutative we also have $x \in L$.

Note that this proposition does not assure that the minimal automaton of a given language L is a commutative deterministic finite automata. For instance, for the language $L = \{x \in \Sigma^* \mid |x|_a = 0 \text{ or } |x|_b > 0\}$ we have that the minimal DFA and the minimal CDFA are not the same as shown in Figure 1.

The relationship between the size of the minimal DFA and the size of the minimal CDFA for a given commutative regular language is a interesting question

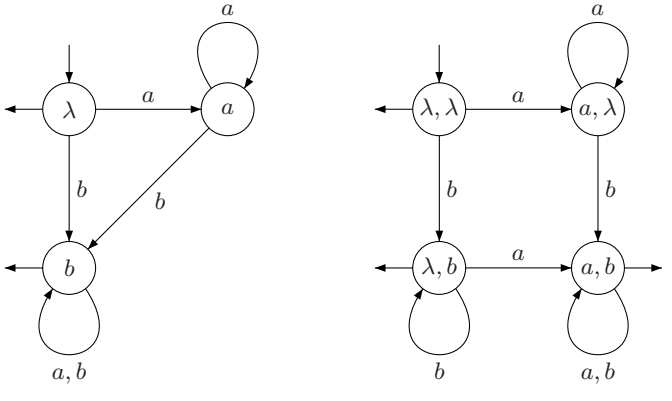

Fig. 1. The minimal DFA (on the left) and minimal CDFA (on the right) of the language $\{x \in \Sigma^* \mid |x|_a = 0 \text{ or } |x|_b > 0\}$

that would require an special study. The size of the alphabet seems to be a factor that would increase considerably the difference between both sizes.

We can also adapt the definition of Moore machine to the commutative case.

A *commutative Moore machine* is a 6-tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \Phi)$ where Σ (resp. Γ) is an input (resp. output) alphabet, $Q = Q_{a_1} \times Q_{a_2} \times \cdots \times Q_{a_n}$, (where all Q_{a_i} for $1 \leq i \leq n$ are finite sets of states), $q_0 \in Q$, $\delta((q1, \ldots, q_i, \ldots, q_n), a_i) = (q1, \ldots, \delta_{a_i}(q_i, a_i), \ldots, q_n)$ where δ_{a_i} is a function from Q_{a_i} onto Q_{a_i} for any $1 \leq i \leq n$ and Φ is a function that maps Q in Γ called *output function*. The behaviour of $\mathcal{M}$ is given by a partial function $t_{\mathcal{M}} : \Sigma^* \to \Gamma$ defined as $t_{\mathcal{M}}(x) =$ for every $x \in \Sigma^*$ such that $\Phi(\delta(q_0, x))$ is defined.

Given a commutative Moore machine $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \Phi)$ with $\Gamma = \{0, 1, \uparrow\}$ we can associate a commutative automaton $\mathcal{A}_{\mathcal{M}} = (Q, \Sigma, \delta, q_0, F)$ where $F = \{q \in Q \mid \Phi(q) = 1\}$.

Given two finite sets of words D_+ and D_- with $D_+ \cap D_- = \emptyset$, we define the (D_+, D_-)-*commutative prefix acceptor machine* $(CPAM(D_+, D_-))$ as the commutative Moore machine having $\Gamma = \{0, 1, \uparrow\}$, $Q = (\pi_{a_1}(x), \pi_{a_2}(x), \ldots, \pi_{a_n}(x))$ with $x \in Pre(D_+ \cup D_-)$, $q_0 = (\lambda, \ldots, \lambda)$ and where $\delta((u_1, \ldots, u_i, \ldots, u_n), a_i) = (u_1, \ldots, u_i a_i, \ldots, u_n)$ if $(u_1, \ldots, u_i, \ \ldots \ , u_n)$, $(u_1, \ \ldots \ , u_i a_i, \ \ldots \ , u_n) \in Q$ and $a \in \Sigma$. For every state $(u_1, u_2, \ldots, u_n)$, the value of the output function Φ associated to $(u_1, u_2, \ldots, u_n)$ is 1, 0 depending whether there exist x such that $\pi_{a_1}(x) = u_1, \pi_{a_2}(x) = u_2$, $\ldots$, $\pi_{a_n}(x) = u_n$ and belonging to D_+, or to D_-. In other case the value is $\uparrow$ (undefined).

A Moore machine $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \Phi)$ is consistent with (D_+, D_-) if for every $x \in D_+$ and any $y \sim_{com} x$ we have $\Phi(\delta(q_0, y)) = 1$ and for every $x \in D_-$ and any $y \sim_{com} x$ we have $\Phi(\delta(q_0, y)) = 0$.

3 Inference of Commutative Regular Languages

In this section we study the inference of commutative regular languages from positive samples and from positive and negative samples.

Firstly, we show that commutative regular languages are not inferable from positive samples, and so, no algorithm could be given for the inference.

Proposition 3. *Commutative regular languages are not inferable from positive samples.*

Proof. In order to show that the family of regular commutative languages is not inferable from positive samples, we take the family of languages $F = \bigcup_{n \geq 0} \{a^i \mid i \leq n\} \cup a^*$ where all these languages are commutative, and then F also is. Now F is superfinite, then by a standard result of [7] we conclude that the class of commutative languages is not inferable from positive samples.

Now, since any regular language is inferable from positive and negative samples, we describe here a new algorithm which allows us to infer faster commutative regular languages from positive and negatives samples.

3.1 Description of the Algorithm *CRPNI*

In this section we describe an inference algorithm from negative and positive samples (see Algorithm 1) for commutative regular languages called *commutative regular positive negative inference (CRPNI)*.

The definition of this algorithm is quite close to the description of the RPNI algorithm, being the main difference the sort of states we are working with. The main differences between them resides in the definition of $CPAM(D_+, D_-)$ and $PTM(D_+, D_-)$ and how the merge operation works as consequence of the definition of the states. For instance, in algorithm 1 any couple of states (p_a, q_a) belong necessarily Σ_a for some $a \in \Sigma$.

Algorithm 1. $CRPNI(D_+, D_-)$

1. $M = CPAM(D_+ \cup D_-)$
2. $\Sigma = alphabet(D_+ \cup D_-)$
3. $listcomp = generateComparisonOrder(M, \Sigma)$
4. **while** $listcomp \neq \emptyset$ **do**
5. $(p_a, q_a) = first(listComp)$ (with $p_a, q_a \in Q_a$ for some $a \in \Sigma$)
6. $listcomp = listcomp \backslash (p_a, q_a)$
7. $push(queue, (p_a, q_a))$
8. $M' = M$;
9. $finish = false$;
10. **while** $\neg emptyset(queue)$ and $\neg finish$ **do**
11. $(p_a, q_a) = pop(queue)$
12. **if** $\neg merge(M, p_a, q_a)$ **then**
13. $M = M'$
14. $finish = true$
15. **else**
16. $queue = detectNoDeterminism(queue, M)$
17. **end if**
18. **end while**
19. **end while**
20. Return M

The algorithm 1 considers all couples of elements in the order given by subroutine *generateComparisonOrder*, that is, considering the pairs in ascending order on the numerical consecutive and one different symbol each time; except possibly at the end if just one symbol remains with pairs unprocessed. Once a couple is chosen, the algorithm saves a copy of the current machine M and tries to merge these elements into the states of the machine. Subroutine $merge(M, p_a, q_a)$ (with $p_a, q_a \in Q_a$ for some $a \in \Sigma$) merges p_a, q_a in Q_a and consequently any two states $(p_1, \ldots, p_a, \ldots, p_n)$ and $(q_1, \ldots, q_a, \ldots, q_n)$ in Q. If the merge is successful, M changes and subroutine *detectNoDeterminism* looks for couples of elements that should be merged in order to maintain determinism, these new pairs are added to the *queue*. Merges continue until *queue* becomes empty, but if any merge fails, the complete chain of merges is undone. When all the pairs are considered algorithm finishes and the automaton associated with M is the answer proposed.

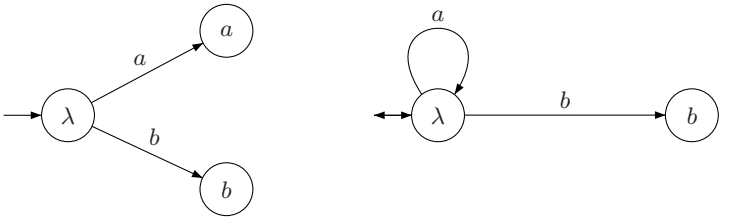

Fig. 2. $PTM(D_+ \cup D_-)$ with $\Phi(\lambda) = \uparrow$, $\Phi(a) = 1$ and $\Phi(b) = 0$ (on the left), and the resulting automaton from the RPNI algorithm for $D_+ = \{a\}$ and $D_- = \{b\}$ (on the right)

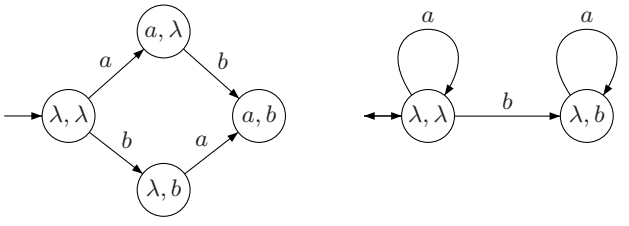

Fig. 3. $CPAM(D_+ \cup D_-)$ with $\Phi(\lambda, \lambda) = \uparrow$, $\Phi(a, \lambda) = 1$, $\Phi(\lambda, b) = 0$ and $\Phi(a, b) = \uparrow$, (on the left) and the resulting automaton from the CRPNI algorithm for $D_+ = \{a\}$ and $D_- = \{b\}$ (on the right)

Note that the order for merging states plays a crucial role in the algorithm. In our case, the order is always we compare any state with its predecessor by $<_{lex}$ and anyone of this elements one by one, in other words, if we have $\Sigma = \{a, b\}$, the merging order would be aa with a, bb with b, aaa with a, ...

Example 3.1. We show in Figures 2 and 3 , $CPAM(D_+ \cup D_-)$, $PTM(D_+ \cup D_-)$ and the final result of the algorithms RPNI and CRPNI when we use them with samples $D_+ = a$ and $D_- = b$.

3.2 Convergence of *CRPNI*

Let $\Sigma = \{a_1, a_3, \ldots, a_n\}$ be an alphabet, let L be a commutative language on Σ, and $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ the minimal commutative automaton of L.

Now, in order to show that the algorithm converges, it suffices to give a characteristic sample for the inference algorithm, and show than for it CRPNI algorithm infer the minimal deterministic automaton.

For the definition of the characteristic sample we define for $1 \leq i \leq n$ and $q^{a_i} \in Q_{a_i}$, the index $I(q^{a_i}) \geq 0$ such that $[a_i^{I(q^{a_i})}]_{\sim_L} = q^{a_i}$ and for any $j \leq 0$ with $[a^j]_{\sim_L} = q^{a_1}$, $I(q^{a_i}) \leq j$.

We define the characteristic sample $D = D_+ \cup D_-$ as a sample satisfying the following condition:

1. for $1 \leq i \leq n$, for any, $q_1^{a_i}, q_2^{a_i} \in Q_{a_1}$ with $q_1^{a_i} \neq q_2^{a_i}$, there exists $x \in D_+$ and $x \in D_-$ (or $x \in D_-$ and $x \in D_+$) such that $\pi_{a_i}(x) \geq I(q_1^{a_i})$, $\pi_{a_i}(y) \geq I(q_2^{a_i})$, $|\pi_{a_i}(x)| - I(q_1^{a_i}) = |\pi_{a_i}(y)| - I(q_2^{a_i})$ and for any $1 \leq j \leq n$ with $i \neq j$, $\pi_{a_j}(x) = \pi_{a_j}(y)$.
2. for $1 \leq i \leq n$, for any, $q_1^{a_i}, q_2^{a_i} \in Q_{a_1}$ such that $q_2^{a_i} \neq \delta(q_1^{a_i}, a_i)$, there exists $x \in D_+$ and $y \in D_-$ (or $x \in D_-$ and $y \in D_+$) such that $\pi_{a_i}(x) \geq I(q_1^{a_i})$, $\pi_{a_i}(y) \geq I(q_2^{a_i})$, $|\pi_{a_i}(x)| - (I(q_1^{a_i}) + 1) = |\pi_{a_i}(y)| - I(q_2^{a_i})$ and for any $1 \leq j \leq n$ with $i \neq j$, $\pi_{a_j}(x) = \pi_{a_j}(y)$.
3. for any $(q^{a_1}, \ldots, q^{a_n}) \in F$ there exists $x \in D_+$ such that $(q^{a_1}, \ldots, q^{a_n}) = ([\pi_{a_1}(x)]_{\sim_L}, \ldots, [\pi_{a_n}(x)]_{\sim_L})$.

Proposition 4. *CRPNI converges for the characteristic sample.*

Proof. Let again $\Sigma = \{a_1, a_3, \ldots, a_n\}$ be an alphabet, let L be a commutative language on Σ, and $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ the minimal commutative automaton of L.

In order to show that the algorithm converges we show that the automaton we build at any moment is a subautomaton of the minimal commutative automaton. As the automaton that we are building is determined by Q_{a_i} and δ_{a^i} for $1 \leq i \leq n$ and $F \subseteq Q_{a_1} \times \ldots \times Q_{a_n}$, it suffices to see that these elements are as the ones of the minimal commutative automaton in the part already learnt. We denote by $M = (Q', \Sigma, \Gamma, \delta', q_0', \Phi)$ the Moore machine that initially will be equal to $CPAM(D_+ \cup D_-) = (Q'', \Sigma, \Gamma, \delta'', q_0'', \Phi')$ and on which we will apply the algorithm. For any $l \geq 0$ and $1 \leq i \leq n$ such that $a^l \in Q''$ we will denote by $[a^l]_M$ the state of Q' corresponding to all the states merged with a^l in M.

For $1 \leq i \leq n$, we will denote by $K_{a_i} \subseteq Q_{a_1}$ (Kernel of a_i) to the set of states that have compared with all precedent states, and for $B_{a_i} \subseteq Q_{a_1}$ (Border of a_i) we will denote the set $\{q^{a_i} \in Q_{a_i} \mid$ there exists $q_2^{a_i}$ such that $\delta_{a_i}(q_2^{a_i}, a_i) = q^{a_i}$ and $q_2^{a_i} \in K\}$.

Now, it suffices to show that states and transition in K_{a_i} belong to the minimal commutative automaton, and that the final states of the learnt automaton corresponds with the states of $\mathcal{A}$.

For the states, let $[a_i^j]_M, [a_i^k]_M \in K_{a_i}$ choosing j and k in such a way that $I([a_i^j]_{\sim_L}) = j$ and $I([a_i^k]_{\sim_L}) = k$ with $[a_i^j]_{\sim_L} \neq [a_i^k]_{\sim_L}$, then by condition 1 of the definition of the characteristic sample there exists $x \in D_+$ and $y \in D_-$ (or $x \in D_-$ and $y \in D_+$) such that $\delta(([\lambda]_M, \ldots, [\lambda]_M), x) = ([a_1^{m_1}]_M, \ldots, [a_i^j a_i^{m_i}]_M, \ldots, [a_n^{m_n}]_M)$, $\delta(([\lambda]_M, \ldots, [\lambda]_M), y) = ([a_1^{m_1}]_M, \ldots, [a_i^k a_i^{m_i}]_M, \ldots, [a_n^{m_n}]_M)$ and then necessarily $merge(M, [a_i^j]_M, [a_i^k]_K)$ fails. So, by the order of merging we have that all states in K_{a_i} are equivalent to some state of $\mathcal{A}$.

For the transition, let $[a_i^j]_M, [a_i^k]_M \in K_{a_i}$ choosing j and k in such a way that $I([a_i^j]_{\sim_L}) = j$ and $I([a_i^k]_{\sim_L}) = k$ with $[a_i^k]_{\sim_L} \neq \delta([a_i^j]_{\sim_L}, a_i)$, then by condition 2 of the definition of the characteristic sample there exists $x \in D_+$ and $y \in D_-$ (or $x \in D_+$ and $y \in D_-$) such that $\delta(([\lambda]_M, \ldots, [\lambda]_M), x) = ([a_1^{m_1}]_M, \ldots, [a_i^j a_i^{m_i}]_M, \ldots, [a_n^{m_n}]_M)$, and $\delta(([\lambda]_M, \ldots, [\lambda]_M), y) = ([a_1^{m_1}]_M, \ldots, [a_i^k a_i^{m_i}]_M, \ldots, [a_n^{m_n}]_M)$ and then necessarily $merge(M, [a_i^j]_M, [a_i^k]_K)$ fails. Again by the order of merging we have that all transitions in K_{a_i} are equivalent transition of $\mathcal{A}$.

Finally, since all states and transition in K forms a subautomaton of $\mathcal{A}$, it suffices to see that at the end of the algorithm by condition 3 of the definition of the characteristic sample we have that for any $(q^{a_1}, \ldots, q^{a_n}) \in F$ there exists $x \in D_+$ such that for $(\pi_1(x), \ldots, \pi_n(x)) = ([q^{a_1}]_M, \ldots, [q^{a_n}]_M)$ which is equivalent to $([q^{a_1}]_{\sim_L}, \ldots, [q^{a_n}]]_{\sim_L})$ with $\Phi([q^{a_1}]_M, \ldots, [q^{a_n}]_M) = 1$ and then the algorithm identifies correctly F.

4 Experimental Results

The aim of this experimentation is to evaluate the performance of the *CRPNI* algorithm strategy with respect to classical inference methods such as redblue and RPNI when they are applied to the learning of regular commutative languages. We do not do an experiment of learning regular languages in general because CRPNI-algorithm is not able to learn general regular languages, and even the samples could be inconsistent, i.e. $D_+ = \{ab\}$ and $D_- = \{ba\}$ is an inconsistent sample, and in those cases the algorithms should not work. The criteria of comparison are the recognition rate on test samples and the average size of the hypothesis generated, the size of a model is its number of states.

The target languages are generated randomly, taking care of guarantee their complete commutativity between the symbols. The generation strategy consists on choose random transitions from each state previously generated and with each symbol. To control the difficulty degree of target languages is possible to change the number of different states available to build the target automaton. The experimentation consists of target languages on 2 and 3 symbols alphabet.

The corpus consist of several incremental sets of different samples, tagged for each target language. The size of the training set varies from 10 samples to 500 samples. The test set consists of 1000 samples different from training ones.

Table 1 shows the results of the first experiment. In this case, three algorithms are compared: RPNI, redblue and *CRPNI*. For each one were trained 200 regular commutative regular target languages which states number range between 4 and 30 states, their average size is 11.52 states. The corpus contains incremental training sets of 10, 20, 30, 40, 50, 100, 200, 300, 400 and 500 samples. The identification of each experiment shows the size of the training set, the percentage presented for each algorithm is the average of the recognition rate on the test set for the 200 target languages.

With respect to the size of the hypothesis proposed by the algorithms, Table 2 shows the behaviour of the three algorithms. The average of the states number for each of the 200 languages learned is presented.

The results on Table 1 and Table 2 shows a notorious improvement in recognition rate using the *CRPNI* with respect to the reference algorithms RPNI and redblue. With 100 training samples the new algorithm reaches more than 99% of accuracy and the size of its hypothesis is like the competitors or even smaller.

Table 3 shows the results of the second experiment, which trains bigger commutative regular target languages with two symbols alphabet. The states number of the target languages varies between 4 and 60 states, the average size is 57.46 states. Three algorithms are compared: RPNI, redblue and our proposal. The

Table 1. Recognition rates of *RPNI*, *redblue* and new algorithm in the first experiment

id	RPNI	redblue	*CRPNI*
t10	69.74%	67.77%	83.74%
t20	75.00%	70.22%	90.43%
t30	78.18%	75.36%	93.30%
t40	79.62%	77.60%	95.75%
t50	82.20%	80.30%	98.12%
100	87.71%	86.46%	99.69%
t200	91.49%	91.25%	99.89%
t300	94.38%	93.66%	99.95%
t400	95.23%	94.74%	99.96%
t500	95.70%	96.07%	99.97%

Table 2. Average states number of *RPNI*, *redblue* and new algorithm in first experiment

id	RPNI	redblue	*CRPNI*
t10	4.09	4.20	4.25
t20	5.65	5.97	6.59
t30	6.65	6.99	8.70
t40	7.69	7.85	9.53
t50	8.25	8.19	9.83
t100	10.21	9.87	10.34
t200	12.58	11.78	10.48
t300	12.69	12.61	10.55
t400	13.55	13.36	10.57
t500	14.79	13.52	10.58

Table 3. Recognition rates of *RPNI*, *redblue* and new algorithm in second experiment

id	RPNI	redblue	*CRPNI*
t10	70.88%	70.92%	78.55%
t50	71.73%	71.12%	90.73%
t100	73.09%	72.99%	96.72%
t200	74.19%	74.64%	99.01%
t300	75.94%	76.48%	99.52%
t400	78.30%	78.96%	99.72%
t500	80.23%	81.05%	99.78%

corpus contains incremental training sets of 10, 50, 100, 200, 300, 400 and 500 samples. The identification of each experiment shows the size of the training set, the percentage presented for each algorithm is the average of the recognition rate on the test set for the 200 target languages.

With respect to the size of the hypothesis proposed by the algorithms, Table 4 shows the behaviour of the three algorithms. The average of the states number for each of the 200 languages learned is presented.

Table 4. Average states number of *RPNI*, *redblue* and new algorithm in third experiment

id	RPNI	redblue	*CRPNI*
t10	4.12	4.16	5.59
t50	11.16	11.15	29.51
t100	17.35	16.56	40.7
t200	28.10	26.51	47.05
t300	37.26	34.21	49.15
t400	43.90	40.08	50.03
t500	50.14	45.11	50.45

Table 5. Recognition rates of *RPNI* and new algorithm in third experiment

id	RPNI	*CRPNI*
t10	52.38%	61.67%
t20	52.41%	69.29%
t30	52.85%	77.18%
t40	52.28%	82.96%
t50	52.97%	87.32%
t100	54.06%	96.38%
t200	57.58%	98.84%
t300	58.87%	99.48%
t400	59.80%	99.66%
t500	60.86%	99.77%

Second experiment shows, again, a clear superiority of *CRPNI* with respect to the reference ones, almost 20 points are the quantitative difference between them in recognition rate. Beside, the size of the hypothesis proposed becomes smaller than those of the reference algorithms as the size of the training set grows.

Table 5 shows the results of the third experiment. In this case, a three symbols alphabet is used. Two algorithms are compared: RPNI and our proposal; redBlue is not reported because previous experiments showed similar behaviour between RPNI and redblue. We trained 200 regular commutative regular target languages which states number range between 6 and 90 states, the average size is 35.23 states. The corpus contains incremental training sets of 10, 20, 30, 40, 50, 100, 200, 300, 400 and 500 samples. The identification of each experiment shows the size of the training set, the percentage presented for each algorithm is the average of the recognition rate on the test set for the 200 target languages.

With respect to the size of the hypothesis proposed by the algorithms, Table 6 shows the behaviour of the algorithms. The average of the states number for each of the 200 languages learned is presented.

In the third experiment the reference algorithm gets stick on a recognition rate of 60% while CRPNI reaches a recognition rate higher than 99% with 300 training samples. The size of the hypothesis of the new algorithm stabilises from 50 training samples on while RPNI ones grows until the last training set.

Table 6. Average states number of *RPNI* and new algorithm in third experiment

id	RPNI	*CRPNI*
t10	4.60	8.09
t20	6.56	17.47
t30	8.31	26.24
t40	9.98	33.01
t50	11.41	36.14
t100	18.24	35.37
t200	28.57	35.69
t300	37.99	34.51
t400	46.11	34.58
t500	54.29	34.62

Comparing results with $|\Sigma| = 2$ and $|\Sigma| = 3$ it is noticeable the increase in performance of CRPNI as soon as alphabet size grows. These results lead us to think that CRPNI could behave even better with bigger alphabets.

Although this experiments are suitable to be improved to make them even harder, this preliminary test is very promising about the utility of this new algorithm for the inference of commutative regular languages, not only because of the excellent recognition rates achieved but also because of the reasonable size of the hypothesis obtained.

5 Conclusions

In this work we study the problem of inferring the class of regular commutative languages. After showing that they are not inferable from positive data we show that some improvement in its inferring from positive and negative samples can be done. For this purpose we give the CRPNI (commutative regular positive negative inference) algorithm. These properties lead us to define commutative deterministic automata and commutative Moore machines for the algorithm. Finally, by an experimentation we also show that CRPNI algorithm has an excellent behaviour in practice. This shows that this kind of works can be useful for real problems.

References

1. Ruiz, J., Cano, A., García, P.: Inferring subclasses of regular languages faster using RPNI and forbidden configurations. In: Adriaans, P.W., Fernau, H., van Zaanen, M. (eds.) ICGI 2002. LNCS (LNAI), vol. 2484, pp. 28–36. Springer, Heidelberg (2002)
2. Agluin, D.: Inductive inference of formal languages from positive data. Information and Control 45(2), 117–135 (1980)

3. Alquézar, R., Sanfeliu, A.: Incremental grammatical inference from positive and negative data using unbiased finite state automata. In: Shape, Structure and Pattern Recogniton, Proc. Int. Workshop on Structural and Syntactic Pattern Recognition, SSPR 1994, pp. 291–300. World Scientific, Singopore (1995)
4. Cichelo, O., Kremer, S.C.: Inducing grammars from sparce data sets: A survey of algorithms and results. Journal of Machine Learning Research 4, 603–632 (2003)
5. Clark, A., Florêncio, C.C., Watkins, C., Serayet, M.: Planar languages and learnability. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS (LNAI), vol. 4201, pp. 148–160. Springer, Heidelberg (2006)
6. Coste, F., Fredouille, D.: Efficient ambiguity detection in C-NFA. In: Oliveira, A.L. (ed.) ICGI 2000. LNCS (LNAI), vol. 1891, pp. 25–38. Springer, Heidelberg (2000)
7. Gold, E.M.: Language identification in the limit. Information and Control 10, 447–474 (1967)
8. Hopcroft, J., Ullman, J.: Introduction to automata theory, languages and computation. Addison-Wesley, Reading (1980)
9. Pearlmutter, B.A., Lang, K.J., Price, R.A.: Results of the abbadingo one DFA learning competition and a new evidence state merging algorithm. In: Honavar, V.G., Slutzki, G. (eds.) ICGI 1998. LNCS (LNAI), vol. 1433, pp. 1–12. Springer, Heidelberg (1998)
10. Lang, K.J.: Random dfa's can be approximately learned from sparse uniform. In: Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, pp. 45–52 (1992)
11. Oncina, J., Garcia, P.: Inferring regular languages in polynomial updated time. In: Pattern Recognition and Image Analysis (1992)
12. Pin, J.-E.: Varieties of formal languages. North Oxford, London (1986) (Traduction of Variétés de langages formels)
13. Pin, J.-É.: Syntactic semigroups. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of formal languages, ch. 10, vol. 1, pp. 679–746. Springer, Heidelberg (1997)