A New Regular Language Learning Algorithm From Lexicographically Ordered Complete Samples*

Jose M. Sempere Pedro García

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46071 Valencia (Spain)
email:jsempere@dsic.upv.es email:pgarcia@dsic.upv.es

Abstract

A new regular language learning algorithm is presented to obtain descriptions which consist of Deterministic Finite Automata (DFAs). The process is an identification in the limit process. The main characteristic is that the DFAs are conjectured using a constructive strategy which does not use a large data space. The total time used is polynomial in the size of the minimum-state DFA and the data seen so far. In the course of learning, the algorithm uses deterministic hypotheses which are bounded in space with the minimal state DFA consistent with the sample and the single sample string used as input. The algorithm works on lexicographically ordered examples and this order is shown to be transcendental for learning.

1 Introduction.

An algorithm has recently been proposed for identifying in the Limit the Regular Language Family from lexicographically ordered complete samples using Non-Deterministic Finite Automata as current hypotheses which in the Limit converged to a DFA which is isomorphic to the unknown regular language canonical acceptor [3]. We propose an alternative algorithm which works within the same paradigm and framework, that is, an ordered and completed samples presentation. The proposed algorithm has Porat's algorithm same computational cost order but it works using DFAs as current hypotheses. The proposed algorithm uses a constructive technique for state grouping which is inspired from another regular language learning algorithm developed in [2].

The present article is divided into the following sections: In part 2, we present the basic concepts and notation used in the task we want to carry out. In part 3, we formally present the proposed algorithm and we explain how it works as well

*Work partially supported by the Spanish CICYT under grant TIC-1026/92-CO2

as prove the algorithm convergence. In part 4, we calculate the space and time complexity of the algorithm.

2 Basic concepts and notation.

The notation used is discussed in [1].

Definition 1. A Deterministic Finite Automaton (DFA) is defined as the tuple (Q,Σ,δ,q_0,F),where Q is a finite set of states, Σ is an alphabet of input symbols, δ : Q$\times\Sigma \rightarrow$ Q is the transition function between states, $q_0 \in$ Q is the initial state and F $\subseteq$ Q is the acceptance state set.

Given a DFA A, we define the following set as the language accepted by A and we denote it by $L(A)$:

$$L(A)= \{\ w \in \Sigma^* \mid \delta(q_0,w) \in \mathrm{F}\ \}$$

Definition 2. Given $L \subseteq \Sigma^*$, we denote by $x^{-1}L$ the right quotient of L by x, that is, $x^{-1}L=\{\ w \in \Sigma^* \mid xw \in \mathrm{L}\ \}$.

So, $\equiv_L$ denotes the right congruence defined in the following way: $\forall\ x,y\in\Sigma^*$, $x \equiv_L y$ *iff* $x^{-1}L=y^{-1}L$. If L is a regular language, then $\equiv_L$ has a finite index and is the least fine right congruence which covers L. So, the L canonical acceptor, its minimal state DFA, is defined as $A(L)$=(Q,Σ,δ,q_0,F), where Q=$\{u^{-1}\ L \mid u \in \Sigma^*\}$, $q_0=\lambda^{-1}\ L$, F=$\{u^{-1}L \mid u \in L\ \}$ and $\delta\ (u^{-1}L,a)=(ua)^{-1}L$.

Definition 3. Given a language $L \in \Sigma^*$, a completed presentation for L is the infinite sequence $\{(x_1,\mathrm{I}(x_1)),...,(x_n,\mathrm{I}(x_n)),...\}$, where every word in Σ^* appears classified according to its inclusion in L, that is, $I(x_i)$ can take the following values

$$I(x_i) = \begin{cases} + & \text{if } x_i \in L \\ - & \text{otherwise} \end{cases}$$

In the task we are working on, we will use a lexicographically ordered completed presentation for the language to be inferred. We define a sample for the language to be inferred as every initial finite sequence of its presentation as defined previously. We denote by M_x the ordered sample for the language where x is the last string which appears in the sequence.

Definition 4. Given M_x as a sample for the language L, we define the sample language L as the set of every string which appears in M_x and is included in L and we denote it by L_{x+}. We denote by L_{x-} the set of every string which appears in M_x and is not included in L.

Definition 5. Given the sample M_x, two strings w and y are distinguishable under M_x if $(w^{-1}L_{x+} \cap y^{-1}L_{x-}) \cup (w^{-1}L_{x-} \cap y^{-1}L_{x+}) \neq \emptyset$ is true. Two strings u and v have a relationship under M_x and we will denote it by $u \equiv_{M_x} v$ *iff* they are not distinguishable under the sample M_x.

Definition 6. Let the DFA A=(Q,Σ,δ,q_0,F). We define the first lexicographically ordered string $w\in \Sigma^*$ that carries out the following predicate $\delta(q_0, w) = q$ is the shortest prefix of a state $q \in Q$ and we denote it by $Sp(q)$. Two states, q and p, will keep the following order: $q < p$ *iff* $Sp(q) < Sp(p)$ under the lexicographical order.

3 Algorithm and properties.

The algorithm we propose obtains completed DFAs which have two kinds of transitions which we call *fixed* and *variable*. The *fixed* transitions have a transcendental value, given that they distinguish every automaton state from its shortest prefix.

The algorithm constructs new hypotheses using refinements over the current hypotheses. The refinement of the hypotheses is made through a sequence of well-defined actions from a new single sample string (w,I(w)) which is inconsistent with the current hypotheses. These actions are:

- To delete the first variable transition which makes the current hypotheses inconsistent with the input data.
- To add new states, which we call expanded states, in order to construct the needed transitions in order to accept the input string (w,I(w)).
- To try to relate the new expanded states with old unexpanded states using the relationship which we define later.

In Figure 1, we show the proposed algorithm for identifying the regular language class in the Limit as an alternative to the algorithm proposed in [3]. We will explain every operation that we use in the algorithm and we will prove that the algorithm actually converges to a minimal state deterministic hypothesis. We will explain every operation separately and we will explain its theoretical foundation as follows:

deltransition(A_x,w).

Given the input string w, this operation deletes the first variable transition which is used in the automaton A_x for incorrectly accepting or rejecting the input sample string w. So, the string w can be factored , given a state p, an input symbol em a and a string y as w=$Sp(p)ay$, under the assumption that $\delta(p,a)$ is the first variable transition used for analyzing the string w. As output for this operation we obtain a new incomplete DFA which has no transitions affecting the input string w or affecting any string which used the deleted transition, that is, any string which has $Sp(p)a$ as prefix.

expstate(A_w,w,I(w),A_x).

This operation constructs a set of new expanded states and transitions to accept or reject the input string w. Given a state p, remember that w=$Sp(p)ay$. This operation also constructs the transitions needed to accept or reject the prefixed strings of w which are lexicographically greater than $Sp(p)$. Every new expanded state parity is calculated as follows: If the state is the last one used to analyze w, then this is an acceptance state *iff* I(w)=+. If the state is not the last one used to analyze w, then, given a string z, we can relate the state to a shortest prefix $Sp(p)z$ and its parity will be the $Sp(p)z$ parity obtained through the automaton A_x. The new expanded states will have $Sp(p)z$ as shortest prefix on the assumption that z is the sequence of transitions needed for acceding from state p to the new state.

Initially the free monoid or the empty set is constructed depending on whether I(λ) value is positive or negative.
Input : A_x (w,I(w)) w=succ(x).
Output : A_w M_w={(λ,I(λ)),...,(x,I(x)),(w,I(w))}-*sample consistent*
Method :
if A_x *is not consistent with* (w,I(w)) *then*
A_w := *deltransition*(A_x,w);
A_w := *expstate*(A_w,w,I(w),A_x);
A_w := *joinstate*(A_w,w,I(w),A_x);
else
A_w := A_x;
endif
endmethod.

Figure 1: The proposed algorithm for Regular Language Identification in the Limit.

joinstate(A_w,w,I(w),A_x).

This operation attempts to relate the new expanded states with other old ones. In every intermediate hypothesis, the state set can be partitioned into two expanded and unexpanded state subsets, and every subset can be ordered by applying the lexicographic order to the shortest prefix of the states. Let's consider the sets Q_e={q_{1e},q_{2e},...,q_{me}} and Q_f={q_{1f}, q_{2f},...,q_{nf}} as the previous ordered subsets. The operation attempts to join the expanded states, while maintaining the established order. So, in the first place, it will try to join q_{1e} with q_{1f}. If this is not successfully done, then it will try to join q_{1e} with q_{2f} and so on until it is successfully done or there are no more unexpanded states as candidates for joining. Subsequently, the operation attempts to join q_{2e} in the same manner and so on until the set Q_e is empty. In the case that the state did not join with another unexpanded one, then the algorithm deletes the state from Q_e and adds it to Q_f and proceeds to reorder this last set.

The rule used for joining states is the following: A state q_{ie} is joined with another q_{jf}, if the consistence with the complete sample M_w={(λ,I(λ)),...,(x,I(x)), (w,I(w))} is preserved and this happens *iff* $Sp(q_{ie}) \equiv_{M_w} \mathrm{Sp}(q_{jf})$. In such a case, the transition which acceded to the state q_{ie} will accede to q_{jf} and then it belongs to the variable transition set. On the other hand, the state q_{ie} is deleted and every one of its expanded succesors is also deleted. In the case that the new state q_{ie} cannot be joined with any of Q_f, the transition which acceded to q_{ie} is converted to a fixed one and the algorithm proceeds to construct every variable transition with all the alphabet symbols from the nonjoined state to any state q_{jf}, including q_{ie}, in keeping with the next rule : $\forall a \in \Sigma$, a transition from q_{ie} to q_{jf} is constructed under the symbol a only if the hypothesis preserves the consistence with the sample M_w= {(λ,I(λ)),...,(x,I(x)) ,(w,I(w))} and this will happen *iff* $Sp(q_{ie})a \equiv_{M_w} Sp(q_{jf})$.

In Figure 2, we show an example of the elaboration of current hypotheses up to the convergence to a minimal state DFA. In Figure 2, every hypothesis has: the input string which makes the previous hypotheses inconsistent with the input data; the expanded DFA showing the states and transitions expanded with discontinuos arrows; and the DFA obtained as the output of the algorithm. The fixed transitions have been drawn with bold arrows and the variable transitions have been drawn

with normal arrows.

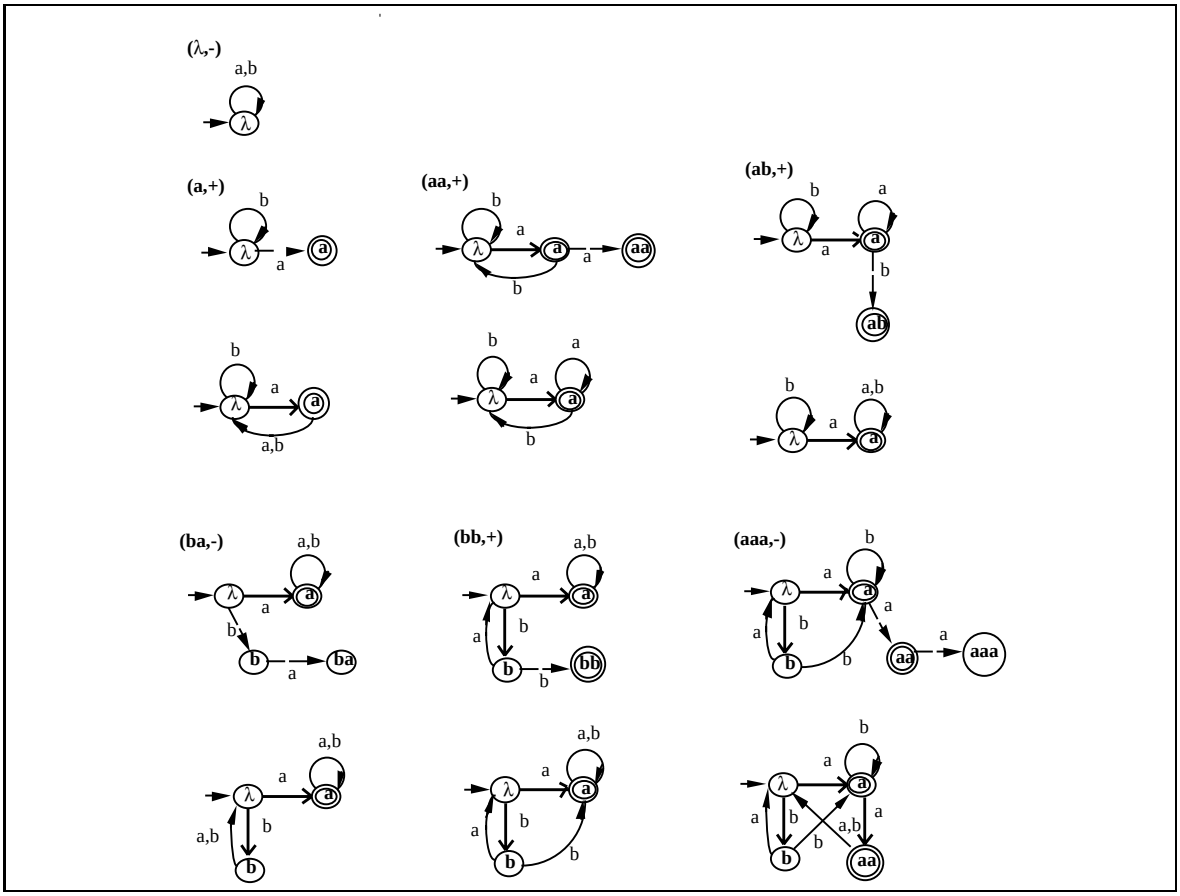

Figure 2: Learning a language and elaborating the intermediate hypotheses.

We will prove that the proposed algorithm identifies the regular language class in the Limit. This is, that given an unknown regular language L, the algorithm obtains a deterministic, completed and minimal state hypotheses $A(L)$ in the Limit. This result will be proved through three theorems that we will enunciate and prove as follows.

Theorem 1 *Given the lexicographically ordered completed sample* M_x={(λ,I(λ))... (x,I(x))} *and the DFA* A_x *obtained by the proposed algorithm as current hypotheses output for the input sample* M_x. *Then given* A_x *and* (w,I(w)), w=succ(x), *as input to the proposed algorithm, the algorithm outputs a DFA which is consistent with the sample* M_w={(λ,I(λ)),...,(x,I(x)),(w,I(w))}.

Proof

Let's study two different situations with the present input.

- A_x is consistent with $(w,\mathrm{I}(w))$ In such a case, $A_w = A_x$ and A_w is consistent with the sample M_w

- A_x is inconsistent with $(w,\mathrm{I}(w))$

  Three different actions will be taken :

  - A_w:=*deltransition*(A_x,w) which deletes the first variable transition that makes A_x inconsistent.
  - A_w:=*expstate*$(A_w,w,\mathrm{I}(w),A_x)$ which constructs a new path in the transition diagram for the string $(w,\mathrm{I}(w)$.
  - A_w:=*joinstate*$(A_w,w,\mathrm{I}(w),A_x)$ which is consistent with M_w given that for joining two states or making new transitions the hypothesis keeps on preserving the relationship $\equiv_{M_w}$ and this is the condition for making the hypothesis consistent with the sample M_w.

Theorem 2 *Let the completed lexicographically ordered sample* M_x=$\{(\lambda,\mathrm{I}(\lambda))\ldots(\mathrm{x},\mathrm{I}(\mathrm{x}))\}$ *for the language* L. *The proposed algorithm obtains as current hypothesis output a DFA* A_x *that has no more states than A*(L) *(the language canonical acceptor).*

Proof

We will prove this theorem by induction on the size of the sample M_w=$\{(\lambda,\mathrm{I}(\lambda))\ldots(x,\mathrm{I}(x)),(w,\mathrm{I}(w))\}$

Our induction base has the single string M_λ=$\{(\lambda,\mathrm{I}(\lambda))\}$ as input sample. Therefore, by definition, the algorithm constructs a DFA that has a unique state and consequently it has no more states than the DFA A(L).

Let's suppose as induction hypothesis that given the sample M_x=$\{(\lambda,\mathrm{I}(\lambda))\ldots(x,\mathrm{I}(x))\}$, the algorithm constructs a DFA A_x that has no more states than the DFA A(L).

Given a new single sample string input $(w,\mathrm{I}(w))$, w=*succ*(x), then we can study two different cases:

- A_x is consistent with $(w,\mathrm{I}(w))$.

  In such a case A_w=A_x and A_w has no more states than the DFA A(L).

- A_x is not consistent with $(w,\mathrm{I}(w))$.

  The following actions will be taken:

  - A_w:=*deltransition*(A_x,w) that does not add a new state to the DFA A_x.
  - A_w:=*expstate*$(A_w,w,\mathrm{I}(w),A_x)$. This operation can add a number of states that makes A_w have more states than A(L).
  - A_w:=*joinstate*$(A_w,w,\mathrm{I}(w),A_x)$. As a result of this operation there is not a number of states greater than in A(L) given that under the relation that defines the DFA A_w, $\equiv_{M_w}$, the algorithm cannot join two different states, therefore according to the relation that defines the DFA $A(L)$, $\equiv_L$, these two states cannot be joined because the relation $\equiv_L$ is a refinenment over $\equiv_{M_w}$.

Theorem 3 *The proposed algorithm identifies every regular language L in the Limit from its lexicographically ordered completed presentation.*

Proof

Let's suppose that the proposed algorithm does not converge with a lexicographic presentation. Let L be the source language, let $A(L)$ be the canonical acceptor and let n be the number of states of $A(L)$. The number of different DFAs with a number of states equal or smaller than n is finite. If the proposed algorithm obtains a hypothesis that is different to $A(L)$ as output, then this hypothesis will be inconsistent with any input data. Every hypothesis that is obtained as output has a number of states smaller or equal to n and Theorem 1 implies that if a hypothesis has been rejected, then it cannot be obtained as output again. So if the number of possible hypotheses obtained as output is finite, then it implies that after a finite time the DFA $A(L)$ will be obtained as output and then the proposed algorithm will converge to $A(L)$.

4 Complexity.

Let's consider an input to the algorithm made up by A_x and $(w,\mathrm{I}(w))$ with the sizes $\mid Q_x \mid = n$, $\mid w \mid = m$ and $\mid \Sigma \mid = p$. The size of the input data seen so far, $M_x=\{(\lambda,\mathrm{I}(\lambda))\ldots (x,\mathrm{I}(x))\}$, is bounded as follows

$$\sum_{i=1}^{m-1} p^i < \mid M_x \mid \leq \sum_{i=1}^{m} p^i$$

and it bounds the number of algorithm runs that have been made until the present input.

4.1 Time complexity.

We are going to analyze the time required for every action in the proposed algorithm.

- *Checking the A_x consistence with* $(w,\mathrm{I}(w))$

  Given that A_x is a DFA, the consistence checking is equal to testing the only path in A_x for the input string w. This path has a size of m. So the algorithm needs as many computation steps as input string symbols and in accordance with this, this operation is $\mathrm{O}(m)$ time complexity.

- *deltransition*(A_x,w)

  This operation will delete the first variable transition for the input string w in the DFA A_x as in the previous operation and it is $\mathrm{O}(m)$ time complexity.

- *expstate*$(A_w,w,\mathrm{I}(w),A_x)$

  In the best case, this operation makes a single state and in the worst case it will make as many states as the w size, that is m. Every elaboration of a state implies establishing its parity, and this is made by checking the DFA A_x with every w prefix which produces the new states. So the total time required is

$$m + \sum_{i=1}^{m} i = m + \frac{m \cdot (m+1)}{2}$$

But, given that the algorithm can establish the prefix parity by checking the completed string w and storing the prefix parity only one time, the total time required is $O(m)$ time complexity.

- *joinstate*(A_w,w,I(w),A_x)

  This operation attempts to join every new expanded state with an older one. The maximum number of expanded states is bounded by m and the maximum number of unexpanded states is bounded by n. In every joining attempt, the operation checks the consistence with the completed sample, so the total time is bounded by

$$n \cdot m \cdot \sum_{i=1}^{m} p^i$$

  but there are a number of states that will never join with other ones and they will be transformed into unexpanded ones. In such cases, the operation makes variable transitions to unexpanded states and it will have a time complexity cost bounded by

$$p \cdot n \cdot m \cdot \sum_{i=1}^{m} p^i$$

Considering that for a language L the number of states of $A(L)$ is k, then the definitive total time required is

$$k \cdot m \cdot \sum_{i=1}^{m} p^i \cdot (p+1)$$

and this cost will be considered polynomial time in an *amortized* sense as is expressed in [3].

4.2 Space complexity.

Supposing that, for the language L, $A(L)$ has a number of states equal to k, then the total space used in the Limit will be $k(p+1)$. The only operation that needs additional space is *expstate*(A_w,w,I(w),A_x) and it needs to make a number of states bounded by m, so in such a case, the total space required is $m + k(p+1)$.

References

[1] J. Hopcroft, J. Ullman *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Company, 1979.

[2] J. Oncina, P. García Inferring regular languages in polinomial update time. *Pattern Recognition and Image Analysis. Selected Papers from the IVth Spanish Symposium.* Series in Machine Perception Artificial Intelligence. Vol. 1. World Scientific. 1992.

[3] S. Porat, J. Feldman Learning Automata from Ordered Examples. *Machine Learning*, 7. pp 109-138. Ed. Kluwer Academic Publishers, 1991.