Regular Grammatical Inference from Positive and Negative Samples by Genetic Search : the GIG method*

Pierre Dupont
France Télécom CNET/LAA/TSS/RCP
2, route de Trégastel BP40 - 22301 Lannion Cedex France
E-mail: dupont@lannion.cnet.fr

Abstract

We recall briefly in this paper the formal theory of regular grammatical inference from positive and negative samples of the language to be learned. We state this problem as a search toward an optimal element in a boolean lattice built from the positive information. We explain how a genetic search technique may be applied to this problem and we introduce a new set of genetic operators. In view of limiting the increasing complexity as the sample size grows, we propose a semi-incremental procedure. Finally, an experimental protocol to assess the performance of a regular inference technique is detailed and comparative results are given.

1 Introduction

Grammatical Inference is an instance of the *Inductive Learning* problem which can be formulated as the task of discovering common structures in examples which are supposed to be generated by the same process. In this particular case, the examples are sentences defined on a specific alphabet and the common structures are represented by a grammar or an equivalent machine. Once the grammar has been inferred from the learning data, the induced language is identified.

A set of negative information, i.e. a set of examples not respecting the common structures, may also help the language induction. This is theoritically motivated by the Gold's results of language identification in the limit [5]. He shows that the class of languages which are identifiable in the limit, given positive information only, is the class of *non superfinite*[1] languages. On the contrary, given a complete (positive and negative) information, the class of *admissible*[2] grammars are identifiable in the limit.

---

*Published in ICGI'94, Lectures Notes in Computer Science, No. 862, Grammatical Inference and Applications, Springer-Verlag, pp. 236-245, 1994.

[1] A superfinite class of languages denotes any class which contains all the languages of finite cardinality and at least one of infinite cardinality. Consequently, the regular class is superfinite.

[2] A class $\mathcal{C}$ of grammars is admissible if $\mathcal{C}$ is countable and for any grammar $\mathcal{G}$ in $\mathcal{C}$, the language membership problem is decidable. The context-sensitive grammars are admissible since the context-

In section 2, we review the formal theory of the regular inference problem and we state it as a search through a boolean lattice. In this framework, regular inference is viewed as the discovery of an optimal state merging operation in a canonical finite automaton built from the positive information. We show how the negative information may be used to control the grammar induction by means of an evaluation function of each potential solution. The proposed grammatical inference by genetic search is detailed in section 3. As the search space dramatically grows with the size of the positive sample, the correct inference becomes more difficult while getting a larger positive information on the language to be identified. This seemingly paradoxical situation motivates the semi-incremental procedure exposed in section 4. The experimental assessment is exposed in section 5 followed by comparative results with another inference method.

In the following sections of this paper, we assume the reader familiar with the usual notations of formal language theory (see, for instance, [1]).

2 The Regular Inference Problem

2.1 Definitions and notations

A regular language L may be referred to by $L(R)$, $L(A)$ or $L(G)$, depending on whether this language is respectively, but equivalently, specified by a regular expression R, a finite automaton A or a regular grammar G.

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of *states*, Σ is an *alphabet*, δ is a *transition function*, i.e. a mapping from QxΣ to 2^Q, q_0 is the *initial state* and F is a subset of Q identifying the *final* or *accepting states*. If for any q in Q and any a in Σ, $\delta(q, a)$ has at most one member, respectively exactly one member, the automaton A is said to be *deterministic*, respectively *complete*. Given a finite automaton $A = (Q, \Sigma, \delta, q_0, F)$, let $\Pi = (\pi_0, \ldots, \pi_{r-1})$ be a partition of the set Q. Let $A_\Pi = (\Pi, \Sigma, \gamma, \pi_0, F_\Pi)$ be the *derived automaton* with respect to the partition Π. It is defined as follows:

(1) $q_0 \in \pi_0$.

(2) $F_\Pi = \{\pi_i \in \Pi \mid \exists q_j \in F, q_j \in \pi_i\}$.

(3) $\gamma(\pi_i, a) = \{\pi_j \in \Pi \mid \exists a \in \Sigma, \exists q_k \in \pi_i, \exists q_l \in \pi_j \textit{ such that } q_l \in \delta(q_k, a)\}, 0 \leq i \leq r-1$.

Hence, the automaton A_Π is obtained by merging the states of A belonging to the same subset π_i, for each subset $\pi_i, 0 \leq i \leq r-1$. The finite automaton A_Π is said to be *derived from A with respect to the partition* Π. Note the important property that $L(A)$ is included in $L(A_\Pi)$ as a consequence of the construction of A_Π [4].

Let I_+ and I_- be two disjoint finite subsets of Σ^*. They respectively denote the positive and negative learning samples of the inference algorithm. A sample I is said to be *structurally complete* with respect to a finite automaton A if the two following conditions hold:

(1) every transition of A is used at least once when accepting the strings of I.

(2) every element of F (the final state set of A) is an accepting state of at least one string of I.

---

sensitive languages are accepted by nondeterministic linearly space-bounded Turing machines for which the language membership problem is decidable [8].

Let $MCA(I_+)$ denote the *maximal canonical automaton with respect to* I_+ [11]. It is the automaton having the largest number of states with respect to which I_+ is structurally complete, since each transition of $MCA(I_+)$ is used exactly once when accepting the strings of I_+. The $MCA(I_+)$ is generally non-deterministic. Let $PTA(I_+)$ denote the *prefix tree acceptor with respect to* I_+ [2]. It is necessarily deterministic and may be derived from the $MCA(I_+)$ by merging states sharing the same prefixes. Let UA denote the *universal automaton*. It accepts all the strings defined over the alphabet Σ, i.e. $L(UA) = \Sigma^*$, and it is the smallest automaton with respect to which every sample of Σ^* is structurally complete. Let $mDFA(L)$ denote the minimal deterministic automaton accepting the language L. It is unique up to a renumbering of its states.

2.2 Fundamental theorems of the regular inference

Theorem 1 *Let I_+ be a positive sample of any regular language L and let A be any automaton accepting exactly L. If I_+ is structurally complete with respect to A, then A may be derived from the $MCA(I_+)$ for some partition Π [11].*

Theorem 2 *Let I_+ be a positive sample of any regular language L and let A be an automaton isomorphic to the minimal DFA accepting L. If I_+ is structurally complete with respect to A, then A may be derived from the $PTA(I_+)$ for some partition Π [2].*

These theorems, revisited in [3], form the basis of a family of inference methods (see, for instance, [12, 2]) by state merging technique, since they amount either to heuristically discover the partition Π or to explicitly construct this partition for some specified subclass of regular languages. Our method aims at discovering the partition Π under the control of the negative sample I_-, like in [14], but here the problem is considered as an explicit search toward an optimal partition.

Let us mention that the first theorem is more general than the second one in the sense that it only suppose the structural completeness of I_+ with respect to any automaton accepting a language L, not necessarily with respect to the minimal DFA for L. In other respects, as the $PTA(I_+)$ generally contains less states than the $MCA(I_+)$, starting from the $PTA(I_+)$ allows for reducing the search space.

2.3 Statement of the regular inference problem

Given the positive and negative samples I_+ and I_-, a finite automaton A is a solution of the regular inference problem if the three following conditions are satisfied:
(1) I_+ is structurally complete with respect to A.
(2) $I_- \subseteq \Sigma^* - L(A)$.
(3) A is an automaton having the fewest number of states, which fulfils the two preceding conditions.

We may also restrict the possible solutions to deterministic automata. In that particular case, we search for the minimal DFA consistent with I_+ and I_-. That is, we take the simplicity of the inferred automaton as generality criterion but there is no guarantee that the identified language includes all the languages agreeing with I_+ and I_-. Unfortunately, the minimal DFA consistency problem is known to be NP-complete [6]. However we shall see that when looking for an optimal automaton as defined in section 2.4, we may correctly identify a set of typical regular languages, in practice.

2.4 Search for the optimal partition

Let Q denote the set of states of the $MCA(I_+)$, let N denote the cardinality of Q and let $P(I_+)$ be the set of all partitions of Q. Let $r(\Pi_i)$, or simply r_i, denote the number of elements, or *blocks*, of the partition Π_i. Let $\Pi_1 = \{\pi_{11}, \ldots, \pi_{1r_1}\}$ and Π_2 be two partitions of $P(I_+)$. We say that Π_2 *directly derives from* Π_1 if the partition Π_2 is constructed from Π_1 as follows: $\Pi_2 = \{\pi_{1j} \cup \pi_{1k}\} \cup \Pi_1 \setminus \{\pi_{1j}, \pi_{1k}\}$, for some j, k between 1 and r_1, $j \neq k$. Consequently, $r_2 = r_1 - 1$.

This derivation operation defines a partial order relation on $P(I_+)$, which we shall denote $\preceq$. In particular, we have $\Pi_1 \preceq \Pi_2$. Let $\ll$ denote its transitive closure. We already known the language inclusion property [4] which may be reformulated as follows: $\Pi_i \ll \Pi_j$ if $L(A_{\Pi_i}) \subseteq L(A_{\Pi_j})$.

Note that the set $P(I_+)$ ordered by the relation $\preceq$ is a boolean lattice of which $MCA(I_+)$ and UA are respectively the null and universal elements. Hence the regular inference problem may be stated as a search through this lattice where each potential solution may be evaluated by the function f_{I_+,I_-} defined as follows.
$f_{I_+,I_-} : P(I_+) \rightarrow \mathbf{R}^+, f_{I_+,I_-}(\Pi) = g(r_{I_+}(\Pi), k_{I_-}(\Pi))$, where g is a positive penalty function of two integer arguments:
$r_{I_+}(\Pi)$, the number of blocks of the partition Π, and
$k_{I_-}(\Pi)$, the number of strings from I_- being accepted by the derived automaton A_Π.
One possible definition of the function g is as follows:
$g(a, 0) = a, g(a, b) = N + 1 \text{ } if \text{ } b > 0, a \geq 1$.

We may now consider the minimal DFA problem as an optimization problem, that is, the search through the lattice $P(I_+)$ for the partition Π for which f_{I_+,I_-} is minimal. In other words, if $d(\Pi)$ denotes the *depth* of an automaton A_Π in the lattice, i.e. $d(\Pi) = N - r(\Pi)$, we are looking for an automaton at maximal depth and compatible with the positive and negative samples. In the same theoretical framework, Tomita proposed a heuristic search by hill-climbing [16] and already mentioned the potential advantage of adaptive search technique. Another search technique, called BRIG, which relies on a random pruning of the boolean lattice has been recently proposed by Miclet [13]. In the section 3, we detail the Grammatical Inference by Genetic search (GIG) method.

3 The GIG method

3.1 Overview of genetic algorithms

We follow here the presentation of Genetic Algorithms (GAs) given in [7]. A GA is a parallel search technique in which a set, namely the *population*, of potential solutions, called *individuals* is progressively updated through a *selection mechanism* and *genetic operations*, i.e. the *crossover* and the *mutation*. Each individual is coded as a fixed length bit string, called a *chromosome*.

Starting from a randomly generated initial population, i.e. at the first *generation*, each individual is evaluated by the *fitness function* to be optimized. The population of the next generation is obtained in three steps. Firstly, a random selection with repetitions is performed, usually on the whole population. The probability of a given individual to be selected, possibly several times, is obtained from the ratio between its fitness value and the average fitness value computed over the whole population. This is the fitness proportionate reproduction mechanism which promotes the fittest individuals to be reproduced. Secondly, a given percentage, i.e. at a given *crossover rate*, of this temporary population is selected by pairs. Each pairs of *parents* gives rise to two *children* which are constructed by taking alternative subparts from their parent chromosomes. After the crossover operation, the parents are replaced by their children in the temporary population. Thirdly, some individuals are selected at a given *mutation rate*. One position in the associated chromosome of each selected individual is randomly chosen and the corresponding bit value is swapped. The final population of the next generation is made of the individuals obtained after fitness proportionate reproduction, crossover and mutation. This process is iterated until some termination criterion is met[3].

The use of a GA for solving the regular grammatical inference problem is motivated by the three following considerations. Firstly, we would like to take advantage of an implicitly parallel search technique. Secondly, a GA offers a way to numerically optimize data represented in a symbolic form. Thirdly, since a GA aims at continuously improving potential solutions, it should offer the possibility to rapidly converge to a new solution after updating the learning samples.

A very preliminary work on the use of GA for grammatical inference has been recently proposed [17]. However, the theoretical framework was not stated, a lot of a priori knowledge was used (number of rules, maximal rule length, sets of terminals and nonterminals of the unknown grammar supposed to be known) and the experimental assessment was very limited.

In the sequel, we detail the representation scheme and the genetic operators of the GIG method. Besides, we consider the function f_{I_+,I_-} introduced in the section 2.4 as the fitness function to be optimized[4].

3.2 Classical partition representation scheme

Each individual in the population corresponds to a partition of the state set of the $MCA(I_+)$, that we represent by its *group-number encoding*. Formally, each partition Π is encoded as a string of N integers $(i_1, \ldots, i_N)$ where $i_j \in \{1, \ldots, r(\Pi)\}, 1 \leq j \leq N$. The $j-th$ integer i_j indicates the group number, or block index, assigned to the state j. For example, the encoding $(2,1,2,3,2)$ represents the partition $\{\{1,3,5\},\{2\},\{4\}\}$ of the set $\{1,2,3,4,5\}$.

The group-number encoding satisfies the closure property of the representation, that is, every chromosome may be interpreted as a valid partition. However, this coding is largely redundant since there are $r(\Pi)!$ equivalent representations. For this reason, a *left-to-right canonical group-number encoding* [9] has been introduced. Formally, the groups

[3] Usually the search is stopped when the number of distinct evaluated individuals reaches a maximal value or when the population has converged to a few distinct individuals.

[4] The value of f_{I_+,I_-} assigned to any derived automaton not compatible with I_-, i.e. accepting at least one string from I_-, is equal to $N+1$. As a consequence, such an automaton may be selected for reproduction although with a lower probability than a compatible automaton

are numbered in such a way that their are in the same left-to-right order as their state of minimal rank. For example, $(2,1,2,3,2)$ will be renumbered as $(1,2,1,3,1)$.

3.3 Genetic operators

We define here the genetic operators in the *phenotype space*, that is, the space of possible partitions. Note that these operations are actually performed on the associated representation, i.e. the *genotype*, in such a way that the resulting individuals remain canonically numbered.

3.3.1 Structural Mutation

The structural mutation consists of a random selection of a state in some block of a given partition followed by the random assignment of this state to a block. Consider, for instance, the partition $\{\{1,\mathbf{3},5\},\{2\},\{4\}\}$ and supposed that the state $\mathbf{3}$ is randomly selected for mutation. Suppose, moreover, that the randomly selected block is the second one. The resulting partition is $\{\{1,5\},\{2,\mathbf{3}\},\{4\}\}$. In that particular case, the number of blocks has not been changed. However, we may similarly obtain a partition with a cardinality increased or decreased by one. In other words, the depth of the derived automaton obtained after mutation differs at most by 1 from the original one. This is a particular feature of our problem, since classical GAs applied to partitioning problems often use repair function to insure that the fixed number of blocks remains constant during the search [10]. Here, we try to find the partitions having the smallest cardinality, i.e. the derived automaton at maximal depth, under the control of the negative sample[5].

3.3.2 Structural Crossover

The structural crossover consists of the union in both parent partitions of a randomly selected block. For example, let the two following partitions be selected for crossover, $\{\{\mathbf{1,4}\},\{2,3,5\}\}$ and $\{\{\mathbf{1,3}\},\{2\},\{4\},\{5\}\}$, and let the first block be randomly selected in both parents. The union of the selected blocks is $\{\mathbf{1,3}\}\cup\{\mathbf{1,4}\}=\{\mathbf{1,3,4}\}$ and the two resulting partitions are $\{\{\mathbf{1,3,4}\},\{2,5\}\}$ and $\{\{\mathbf{1,3,4}\},\{2\},\{5\}\}$. The derived automata obtained after crossover share a common block, i.e. a common subset of states merged together. Moreover, the depth of each new automaton is either equal to the original one or increased by 1.

4 A semi-incremental procedure

As mentioned in the section 1, we are faced with a paradoxical situation. Indeed, the search space size dramatically increases with the size of the $MCA(I_+)$, i.e. with the size of the positive sample, making the correct identification more difficult when we have a larger positive information on the language to be identified. For this reason, we propose a semi-incremental inference procedure. It is not purely incremental in the sense that it assumes that we dispose of some fixed positive and negative samples, but it makes use

[5] Remind that the derived automaton should not accept any strings from I_-.

of the positive information in an incremental way. We first sort the positive sample I_+ in lexicographical order. Consequently, the shortest strings are first taken into account. Starting with the first sentence of I_+, we construct the associated $MCA(I_1^+)$ and we search for the optimal partition of its state set under the control of the whole negative sample I_-. Let A_1 denote the derived automaton with respect to this optimal partition. Let s_{next} denote the next string in I_+. If s_{next} is already accepted by A_1, we skip it. Otherwise, the automaton A_1 is extended in such a way that $L(A_1^{ext}) = L(A_1) \cup s_{next}$. The search is then restarted from the extended automaton A_1^{ext} under the control of I_-. We pursue this incremental procedure till all the strings of I_+ had been considered.

Note that the proposed technique is a heuristic way of reducing the computational complexity of the inference problem, with no guarantee of getting the correct solution. It is not only useful for the GIG method but may be applied to other search algorithms.

5 Experimental Assessment

5.1 Protocol

The following experimental protocol has been proposed to evaluate regular inference methods. We test a set of typical regular languages [16, 13]. Associated to each language L, we automatically build the corresponding $mDFA(L)$. The learning samples are then generated as follows:
(1) I_+ consists of a set of strings randomly generated from the $mDFA(L)$, with the property that I_+ is structurally complete. Let $|I_+|_c$ be the sample size required to randomly get such a sample. Depending on the value of a parameter denoted M, we pursue the random generation until the sample size $|I_+|$ is equal to $M.|I_+|_c$.
(2) I_- is constructed in the same way but from the complementary automaton, i.e. the automaton accepting $\Sigma^* - L$.

Remark that there is no guarantee that the optimal solution, defined in section 2.4, exactly accepts the target language. In order to evaluate the quality of the proposed solutions, we compute the correct classification rate on Σ^l, the set of all the positive and negative strings up to a given length l, according to the target automaton[6].

The set of 15 test languages are defined hereafter. The 7 languages L_1 to L_7 are those defined by Tomita [16]:

L_1 : *a**
L_2 : *(ab)**
L_3 : any sentence without an odd number of consecutive *a's* after an odd number of consecutive *b's*.
L_4 : any sentence over the alphabet *a,b* without more than two consecutive *a's*.
L_5 : any sentence with an even number of *a's* and an even number of *b's*.
L_6 : any sentence such that the number of *a's* differs from the number of *b's* by 0 modulo 3.

[6] It is very important to distinguish the positive and negative classification rate. Otherwise, the empty automaton would have a nearly perfect score in the case of a very small proportion of positive strings in Σ^l. Similarly, the universal automaton would have a nearly perfect score in the case of a very small proportion of negative strings in Σ^l.

L_7 : $a^*b^*a^*b^*$
L_8 : a^*b
L_9 : $(a^* + c^*)b$
L_{10} : $(aa)^*(bbb)^*$
L_{11} : any sentence with an even number of a's and an odd number of b's.
L_{12} : $a(aa)^*b$
L_{13} : any sentence over the alphabet a,b with an even number of a's.
L_{14} : $(aa)^*ba^*$
L_{15} : $bc^*b + ac^*a$

5.2 Comparative results with the RPNI method

Grammatical inference evaluations often lack practical comparisons between alternative techniques. Here, the RPNI method has been chosen as "state of the art" technique, since it has been proved to identify any regular language in the limit [14]. Besides, there is a theoretical link with our method since the RPNI may be viewed as an ordered search in the boolean lattice following a single path of increasing depth with a limited possibility of backtracking.

The results presented here have been obtained according to the very same experimental protocol, learning and test data for both methods.

	st/tr	L	+class	-class	class	max	best
L_1	1/1	9	100.0	100.0	100.0	100.0	100.0
L_2	2/2	9	99.7	100.0	99.8	100.0	100.0
L_3	4/7	9	91.9	98.8	95.3	99.4	94.6
L_4	3/5	9	68.0	79.7	73.9	90.6	81.2
L_5	4/8	9	74.6	63.4	69.0	83.5	80.5
L_6	3/6	9	92.0	86.3	89.2	95.0	95.0
L_7	4/7	9	96.3	93.3	94.8	100.0	99.2
L_8	2/2	9	100.0	100.0	100.0	100.0	100.0
L_9	4/7	7	99.4	98.5	99.0	99.6	99.2
L_{10}	5/6	9	93.9	99.1	96.5	99.9	96.6
L_{11}	4/8	9	78.6	42.7	60.7	72.2	70.8
L_{12}	3/3	9	100.0	99.6	99.7	99.8	99.8
L_{13}	2/4	9	98.8	97.7	98.2	100.0	100.0
L_{14}	3/4	9	89.2	96.4	92.8	99.9	99.8
L_{15}	4/6	7	94.2	98.7	96.4	99.5	99.4
Mean	-	-	91.8	90.3	91.0	96.0	94.4

Table 1: Classification results of the GIG method with semi-incremental search

The parameters of the GIG method are not specifically tuned before each experiment but they have been chosen after a few tests. Their value is listed hereafter: Population size = 100, Max number of evaluations = 2000, Crossover rate = 0.2, Mutation rate/bit =

0.01. The initial population is constructed as follows: the first chromosome corresponds to the currently extended automaton (see section 4). Let N be the number of its states. Up to 50% of the initial population is made of partitions randomly selected from the $N.(N-1)/2$ partitions having $N-1$ elements. The remaining part of the population is randomly generated.

For each pair of samples, we performed 10 independent runs. The first column of table 1 refers to the number of states and transitions of the $mDFA(L)$ associated to each language, i.e. the target automaton. The test sets are made of all strings (except the strings contained in the learning samples) up to the maximal length[7] given in the second column. The third column refers to the correct classification rate of positive sample (according to the target automaton), averaged over 10 runs. The fourth column refers to the correct classification rate of negative sample, averaged over 10 runs. The fifth column gives the total classification rate computed as the arithmetic mean of the two previous features. The sixth column indicates the maximal total classification rate obtained over 10 runs and the last column indicates the total classification rate obtained with the solution automaton having the minimal number of states over the 10 runs[8]. All these results are averaged over the 10 independent (I_+, I_-) samples, randomly generated for each language with the M parameter being set equal to 3. They have been obtained with the semi-incremental procedure described in section 4.

M	RPNI	GIG	GIG_{inc}
1	80.5	82.3	82.2
3	93.9	85.4	94.4

Table 2: Comparative results

We show in table 2 the comparative results between the RPNI method and the GIG method (i.e. we refer here to the correct classification rate of the automaton having the minimal number of states over 10 runs). The two last columns refer to the results obtained without and with the semi-incremental procedure, respectively. These are averaged results over the 15 test languages and over 10 (I_+, I_-) samples for each language, with two different data set depending on the value of the M parameter. Note that for M being equal to 1, we observed that the minimal DFA agreeing with the samples did not always correspond to the target automaton. This explains the worse results obtained with these learning data. Besides, the semi-incremental procedure is very effective, in particular when the sample size grows (M being equal to 3 instead of 1).

[7] We may characterize the language of an $mDFA$ having N states by considering all the strings up to the length L, with L being at most $2N-1$ [15]. Seeing that the alphabet of the languages $L9$ and $L15$ contains 3 symbols instead of 2, we get a similar test set size by setting L equal to 7.

[8] Remark that this particular solution may be selected without knowing the target automaton.

6 Acknowledgments

I am indebted to L. Miclet for having taught me the topic of grammatical inference. I would like to thank E. Vidal for the many fruitful discussions we had about this work and for having performed the tests with the RPNI method. I also thank J. Oncina for having made his programs available.

References

[1] A. Aho and J. Ullman, *The Theory of Parsing, Translation and Compiling, Vol. 1: Parsing*, Series in Automatic Computation, Prentice-Hall, Englewood, Cliffs, 1972.

[2] D. Angluin, *Inference of Reversible Languages*, Journal of the ACM, Vol. 29, No. 3, pp. 741-765, 1982.

[3] P. Dupont, L. Miclet and E. Vidal, *What is the search space of the regular inference ?*, to be published in the Proc. of the 2nd ICGI, 1994.

[4] K.S. Fu and T.L. Booth, *Grammatical Inference: Introduction and Survey*, IEEE Transactions on SMC, Part 1: Vol. 5, pp. 85-111, Part 2: Vol. 5, pp. 409-423, 1975.

[5] E.M. Gold, *Language Identification in the Limit*, Information and Control, Vol. 10, No. 5, pp. 447-474, 1967.

[6] E.M. Gold, *Complexity of Automaton Identification from Given Data*, Information and Control, Vol. 37, pp. 302-320, 1978.

[7] D.E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1989.

[8] M.A. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, Reading, Massachusetts, 1978.

[9] D.R. Jones and M.A. Beltrano, *Solving Partitioning Problems with Genetic Algorithms*, Proc. of the 4th ICGA, Morgan Kaufmann Pub., CA, pp. 442-449, 1991.

[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, 1992.

[11] L. Miclet, *Inférence de Grammaires Régulières*, Thèse de Docteur-Ingénieur, E.N.S.T., Paris, France, 1979.

[12] L. Miclet, *Regular Inference with a Tail-Clustering Method*, IEEE Trans. on SMC, Vol. 10, pp. 737-743, 1980.

[13] L. Miclet and C. de Gentile, *Inférence Grammaticale à partir d'Exemples et de Contre-Exemples : deux Algorithmes Optimaux (BIG et RIG) et une Version Heuristique (BRIG)*, Actes des JFA-94, Strasbourg, France, pp. F1-F13, 1994.

[14] J. Oncina and P. Garcia, *Inferring Regular Languages in Polynomial Update Time*, Pattern Recognition and Image Analysis, N. Perrez de la Blanca, A. Sanfeliu and E. Vidal (editors), Series in Machine Perception and Artificial Intelligence, Vol. 1, pp. 49-61, World Scientific, 1992.

[15] B. Trakhtenbrot and Ya. Barzdin, *Finite Automata: Behavior and Synthesis*, North Holland Pub. Comp., Amsterdam, 1973.

[16] M. Tomita, *Dynamic Construction of Finite-Automata From Examples Using Hill-Climbing*, Proc. of the 4th Annual Cognitive Science Conference, USA, pp. 105-108, 1982.

[17] P. Wyard, *Context Free Grammar Induction Using Genetic Algorithms*, Proc. of the 4th ICGA, Morgan Kaufmann Pub., CA, pp. 514-518, 1991.