Learning Behaviors of Automata from Multiplicity and Equivalence Queries

F. Bergadano and S. Varricchio

Dipartimento di Matematica
Università di Catania
via A. Doria 6/A, 95100 Catania, Italy

Abstract

We consider the problem of identifying the behavior of an unknown automaton with multiplicity in the field $\mathcal{Q}$ of rational numbers ($\mathcal{Q}$-automaton) from multiplicity and equivalence queries. We provide an algorithm which is polynomial in the size of the $\mathcal{Q}$-automaton and of the maximum length of the given counterexamples. As a consequence, we have that $\mathcal{Q}$-automata are PAC-learnable in polynomial time when multiplicity queries are allowed. A corollary of this result is that regular languages are polynomially predictable using membership queries w.r.t. the representation of unambiguous non-deterministic automata. This is important, as there are unambiguous automata such that the equivalent deterministic automaton has an exponentially larger number of states.

1 Introduction

Learning automata from examples and from queries has been extensively investigated in the past, and important results have been obtained recently [10,2,13,12]. Besides positive examples, an important input information comes from asking an oracle whether a particular string belongs to the target language. Such questions are called *membership queries*. Angluin [3] proves that, if we start from a set of strings that lead to every reachable state in the target automaton, a polynomial number of membership queries is sufficient for exact identification. However, if such a set of strings is not available, even if we know the size n of the target automaton, the needed number of queries is exponential in n.

Another possibility is found in *equivalence queries*: asking an oracle whether a guess is correct, and obtaining a counterexample if it is not. We shall also assume that the counterexamples have a maximum length m. It may be shown [5] that there is no polynomial time algorithm to exactly identify automata from equivalence queries only. However, there is a polynomial algorithm if both equivalence and membership queries are used [4].

Equivalence and membership queries then seem to be a necessary requirement for learning deterministic finite state automata. It remains to be seen if stronger formalisms may be learned under the same framework. This paper

gives a positive answer in the direction of behaviors of non-deterministic finite state automata, i.e. functions assigning to every string the number of its accepting paths in a non-deterministic finite state acceptor. Such functions may be described with formal series, as defined in the next section, in the more general framework of automata with multiplicity.

2 Formal Series and Automata

Formal series and automata with multiplicity are the most important generalizations of the classical automata theory. In the last years their significant development helped in solving old problems in automata theory. In [11], using formal series, the decidability of the equivalence problem for deterministic multitape automata has been solved; in [16] a similar result has been shown for unambiguous regular languages in a free partially commutative monoid.

Let M be a non deterministic finite automaton, one can consider the so called *behavior* of M which is the map that associates to any word the number of its different accepting paths. More generally one can assign a multiplicity to the initial states, the final states and the edges of the automaton, so that the corresponding behavior must take into account the assigned multiplicities. In this way one constructs a theory which is general enough to contain, as particular objects, classical and probabilistic automata. In this context the theory of formal power series has been developed, giving a powerful algebraic tool for generalizing the concept of a formal language. Formal power series in non commuting variables have been extensively studied in Theoretical Computer Science and we refer to [8], [9], [14]; here we recall some notations and definitions.

Let K be a semiring and A^* the free monoid over a finite alphabet A; we consider the set $K\langle\langle A\rangle\rangle$ of all the applications $S : A^* \to K$. An element $S \in K\langle\langle A\rangle\rangle$ will also be denoted by $\sum_{u\in A^*} S(u)u$ or by $\sum_{u\in A^*}(S,u)u$ where (S,u) denotes $S(u)$. $K\langle\langle A\rangle\rangle$ is called the set of the *formal power series* over A^* with coefficients in K. In the sequel we shall consider formal series having coefficients in the rational field $\mathcal{Q}$. We denote by $\mathcal{Q}^{n\times n}$ the set of all square matrices $n \times n$ with entries in $\mathcal{Q}$. We shall consider $\mathcal{Q}^{n\times n}$ as having the structure of a monoid with respect to the row by column product; the identity matrix is denoted by Id. A map $\mu : A^* \to \mathcal{Q}^{n\times n}$ is a monoid morphism if $\mu(\epsilon) = Id$ and for any $w \in A^+$, $w = a_1a_2\dots a_n$, $a_i \in A$, one has $\mu(w) = \mu(a_1)\mu(a_2)\dots\mu(a_n)$. A formal power series $S \in \mathcal{Q}\langle\langle A\rangle\rangle$ is called *recognizable* or also *representable* if there exist a positive integer n, λ, $\gamma \in \mathcal{Q}^n$ and a monoid morphism $\mu : A^* \to \mathcal{Q}^{n\times n}$, such that for any $w \in A^*$

$$(S,w) = \lambda\mu(w)\gamma.$$

where λ and γ are to be considered row and column vectors, respectively. The triple (λ,μ,γ) is called a *linear representation* of S of dimension n, or also a *$\mathcal{Q}$-automaton* for S.

A *non deterministic automaton* is a 5-tuple $M = (A,Q,E,I,F)$ where A is the input alphabet, Q is a finite set of *states*, $E \subseteq Q \times A \times Q$ is a set of *edges* and $I, F \subseteq Q$ are respectively the sets of the *initial* and *final* states. Let $w = a_1a_2\dots a_n \in A^*$, an *accepting path* for w is any sequence $\pi = (p_1,a_1,p_2)(p_2,a_2,p_3)\dots(p_n,a_n,p_{n+1})$, with $p_1 \in I$, $p_{n+1} \in F$ and $(p_i,a_i,p_{i+1}) \in$

E, for $1 \leq i \leq n$. The language accepted by M is the set $L(M)$ of all the words which have at least one accepting path; M is *unambiguous* if for any word $w \in L(M)$ there exists only one accepting path for w. We can associate to M a formal series $S_M \in \mathcal{Q}\langle\langle A \rangle\rangle$, also called the *behavior* of M, defined as follows: for any $w \in A^*$, (S_M, w) is the number of different paths which are accepting for w. Let $Q = 1, 2, \ldots, n$ and let $\lambda, \gamma \in \mathcal{Q}^n$ be the characteristic vectors respectively of I and F; consider the monoid morphism $\mu : A^* \rightarrow \mathcal{Q}^{n \times n}$, defined by $\mu(a)_{i,j} = 1$ if $(i, a, j) \in E$, and $\mu(a)_{i,j} = 0$ otherwise. Then one can easily prove (cf. [9]) that for any $w \in A^*$,

$$(S_M, w) = \lambda \mu(w) \gamma.$$

In particular S_M is representable and, when M is unambiguous, S_M corresponds to the characteristic function of $L(M)$. Finally, we shall need the following definitions:

Definition 1 *For any string $u \in A^*$, and a formal series S, the series S_u is defined by*
$(\forall w \in A^*)$ $(S_u, w) = (S, uw)$.

Definition 2 *For any set of strings $E \subseteq A^*$,*
$S \equiv_E T$ iff $(\forall w \in E)$ $(S,w)=(T,w)$.
$S \equiv T$ stands for $S \equiv_{A^*} T$.

We shall use an oracle for answering *multiplicity queries* for any string w, i.e. for providing the value of (S, w), where S is the target formal series.

3 Observation Tables

Based on previous work by Angluin on deterministic finite state automata [4], we now introduce the concept of an observation table for a series S.

Definition 3 *Let $S \in \mathcal{Q}\langle\langle A \rangle\rangle$; an observation table is a triple $\mathcal{T}=(P,E,T)$ where P and E are sets of strings, P is prefix-closed, E is suffix-closed, and $T:(P \cup PA)E \rightarrow \mathcal{Q}$ gives observed values of S, i.e., for all strings $w \in (P \cup PA)E$, $T(w)=(S,w)$.*

Definition 4 *An observation table (P,E,T) is closed iff $\forall u \in P$, $\forall a \in A$, there are coefficients $\alpha_v \in \mathcal{Q}$, $v \in P$, such that*

$$S_{ua} \equiv_E \sum_{v \in P} \alpha_v S_v. \quad \text{S\_ua is a linear combination of S\_v's} \qquad (1)$$

Definition 5 *An observation table (P,E,T) is consistent iff, for any choice of coefficients $\beta_v \in \mathcal{Q}$, $v \in P$,*

$$\sum_{v \in P} \beta_v S_v \equiv_E 0 \Rightarrow (\forall a \in A) \sum_{v \in P} \beta_v S_{va} \equiv_E 0. \qquad (2)$$

Definition 6 *P is a complete set of strings for S iff $\forall u \in P, \forall a \in A$, there are coefficients $\lambda_v \in \mathcal{Q}$, $v \in P$, such that*

$$S_{ua} \equiv \sum_{v \in P} \lambda_v S_v. \tag{3}$$

When a table (P, E, T) is consistent and P is complete, the linear dependencies that are observed on E are valid for any string in A^*, as proved in the following:

Theorem 1 *Let (P,E,T) be a consistent observation table, where P is a complete set of strings for S, then*

$$\sum_{v \in P} \beta_v S_v \equiv_E 0 \Rightarrow \sum_{v \in P} \beta_v S_v \equiv 0. \tag{4}$$

As a consequence, the linear dependencies showing that the table is closed are also valid in A^*:

Corollary 1 *Let (P,E,T) be a consistent observation table, where P is a complete set of strings for S, then*

$$S_{ua} \equiv_E \sum_{v \in P} \lambda_v S_v \Rightarrow S_{ua} \equiv \sum_{v \in P} \lambda_v S_v. \tag{5}$$

4 The Learning Algorithm

The above results suggest that at any stage of the learning process when the table is closed and consistent, we may guess a series $M(P, E, T)$ by basing its representation upon the existing linear dependencies:

- define $\{S^1, ..., S^k\} = \{S_u | u \in P\}$, with $S^1 = S_\epsilon = S$.
- for all $a \in A^*$, compute $\mu(a)$ satisfying

$$S_a^i \equiv_E \sum_j \mu(a)_{i,j} S^j. \tag{6}$$

  Such a matrix exists because the table is closed.

- Construct the series M so as to make it depend from $\{S^1, ..., S^k\}$ with the same coefficients as $S{=}S^1$: let $\lambda = (1, 0, 0, ..., 0)$ and $\gamma_j = (S^j, \epsilon)$. The value of (S^j, ϵ) is found in the table since $S^j = S_u$ for some $u \in P$ and $\epsilon \in E$. Define the constructed series M with $(M, w) = \lambda\mu(w)\gamma$.

Theorem 2 *If P is a complete set of strings for S and (P,E,T) is a closed and consistent table, then $M(P,E,T) \equiv S$.*

We are now left with three problems: (1) closing a table (2) making a table consistent (3) making P complete. However, we will obtain completeness only indirectly and will return to it later.

4.1 Closing a table

Given a table (P, E, T), suppose S_{ua} is linearly independent from $\{S_v | v \in P\}$ with respect to E, in the sense that there are no coefficients $\lambda_{u,v}$ such that $S_{ua} \equiv_E \sum_{v \in P} \lambda_{u,v} S_v$. In this case ua is added to P, and the table is again checked for closure.

This procedure must terminate; more precisely, if the correct series S is representable with $(S, x) = \lambda\mu(x)\gamma$, where $\lambda, \gamma \in \mathbb{Q}^n$ and $\mu : A^* \to \mathbb{Q}^{n \times n}$ is a morphism, then at most n strings can be added to P when closing the table.

In fact, it should be noted that, when ua is added to P as indicated above, the dimension of $\{\lambda\mu(v) | v \in P\}$, as subset of the vectorial space $\mathbb{Q}^n$, is increased by one. Otherwise, $\lambda\mu(ua)$ would be equal to $\sum_{v \in P} \beta_v \lambda\mu(v)$ for some coefficients β_v and

$$(S_{ua}, x) = (S, uax) = \lambda\mu(ua)\mu(x)\gamma = \sum \beta_v \lambda\mu(v)\mu(x)\gamma = \sum \beta_v (S_v, x)$$

i.e., S_{ua} would depend linearly on $\{S_v | v \in P\}$. Since the dimension of $\{\lambda\mu(v) | v \in P\}$ is less than n, we cannot close the table more than n times. The above discussion does not depend on E.

4.2 Making tables consistent

Given a table (P, E, T) and a symbol $a \in A$, consider the two systems of linear equations:

$$(1) \sum_{v \in P} \beta_v S_v \equiv_E 0 \quad (2) \sum_{v \in P} \beta_v S_{va} \equiv_E 0.$$

with β_v as unknowns. Check if every solution of system (1) is also a solution of system (2). In this case the table is consistent. Otherwise, let β'_v, $v \in P$, be some solutions of (1) that are not solutions of (2) and $x \in E$ such that $\sum_{v \in P} \beta_v (S_{va}, x) \neq 0$. Add ax to E.

Suppose that S has a linear representation (λ, μ, γ) of dimension n; there cannot be more than n such additions to E, because every time a new string ax is added, the dimension of $\{\mu(w)\gamma | w \in E\}$ is increased by one. In fact, if $\mu(ax)\gamma = \sum_{w \in E} \delta_w \mu(w)\gamma$, then

$$\begin{aligned}
\sum_{v \in P} \beta_v (S_{va}, x) = \sum_{v \in P} \beta_v \lambda\mu(v)\mu(ax)\gamma = \\
\sum_{v \in P} \beta_v \lambda\mu(v) \sum_{w \in E} \delta_w \mu(w)\gamma = \\
\sum_{w \in E} \delta_w \sum_{v \in P} \beta_v \lambda\mu(vw)\gamma = \\
\sum_{w \in E} \delta_w \sum_{v \in P} \beta_v (S_v, w) = 0
\end{aligned}$$

i.e., ax would not have been added to E.

4.3 The algorithm

We main now describe the procedure for exactly identifying S from multiplicity queries and counterexamples:

$\mathcal{T} \leftarrow (\{\epsilon\},\{\epsilon\},T)$, where $(T,\epsilon) = (S,\epsilon)$.
Repeat

- make the table closed and consistent (P and E are extended and the entries of T are filled in by multiplicity queries).
- ask for a counterexample t to $M(P,E,T)$ by means of an equivalence query.
- add t and its prefixes to P

until M is correct.

The main loop makes the table closed and consistent as described in the two previous subsections, and then constructs a guess M, which is based on the observed linear dependencies. We shall now prove that, if S has a linear representation (λ,μ,γ) of dimension n, after at most n equivalence queries, well have a correct guess, i.e. $M \equiv S$. We need the following:

Lemma 1 *Let $u \in P$ and $t \in uA^*$, and suppose that, for every x such that t=uxz, S_{ux} depends linearly from $\{S_v | v \in P\}$. Then, for every prefix ux of t,*

$$S_{ux} \equiv_E \sum_{v \in P} \mu(x)_{u,v} S_v. \tag{7}$$

where $\mu : A^* \rightarrow \mathbb{Q}^{k \times k}$ is the morphism corresponding to the observed linear dependencies as computed in (6), obtained when closing the table.

Theorem 3 *Let $(S,t) \neq (M,t)$. Then there is a prefix t_0 of t such that S_{t_0} is linearly independent from $\{S_v | v \in P\}$.*

Corollary 2 *If S has a linear representation of dimension n, then after at most n iterations, the algorithm stops*

4.4 Complexity analysis

All we need to do is reorganize some of the previous results and determine the complexity of computing the linear dependencies. Let n be the dimension of a linear representation of S, one has:

- The main loop in the algorithm is repeated at most n times (corollary 2).
- $|\mathrm{E}| \leq n$ (section 4.2).

- For the cardinality of P, the discussion is slightly more involved. From the discussions of sections 4.1 and 4.3, we see that every time either (1) a string is added to P while closing the table or (2) a counterexample is processed, the dimension of $\{\lambda\mu(v)|v \in P\}$ is increased by at least one. The worst case is when this always happens with the counterexamples and the main loop in the algorithm is repeated exactly n times, because also the prefixes of the counterexamples need to be added to P. If m is the maximum length of a counterexample, then, $|P| \leq nm$.

- For every $a \in A$, the table needs to be closed. This amounts to solving a system of $|E|$ equations in $|P|$ unknowns; in the worst case, nm simultaneous equations in nm unknowns. This is an operation of complexity $O(n^3m^3)$ [1]. If $|A|=k$, the complexity of closing the table is $O(kn^3m^3)$.

- Checking for consistency was described in section 4.2, and requires the algorithm to confront the two systems of equations

$$(1) \sum_{v \in P} \beta_v S_v \equiv_E 0 \quad (2) \sum_{v \in P} \beta_v S_{va} \equiv_E 0.$$

  with β_v as unknowns. The table is consistent if every solution of (1) is also a solution of (2). This is the same as checking whether the $|E|$ equations of system (2) do not add additional constraints, i.e. if the corresponding vectors of coefficients depend linearly from the ones of system (1). This operation, in the worst case, requires solving n systems of nm equations in nm unknowns, with complexity $O(n^4m^3)$.

Therefore the complexity of the algorithm is $O(kn^4m^3 + n^5m^3)$. This, together with the fact that, when the main loop terminates, $M \equiv S$, establishes our main result:

Theorem 4 *Representable series may be exactly identified in polynomial time from queries and counterexamples.*

This may be seen as a generalization of Angluin's result for finite state automata [4]. It should be noted that theorem 2 was the inspiring idea behind the algorithm, but has not been used to obtain the above result. In particular, completeness of P was not directly verified in the algorithm nor used in the proofs. However, when the algorithm terminates, the set P must be complete:

Theorem 5 *When* $M \equiv S$,

$$S_{ua} \equiv \sum_{v \in P} \mu(a)_{u,v} S_v \tag{8}$$

As a consequence, when the guess M is correct, P must be complete.

Therefore completeness of P may be seen as a characterization of success when learning S. Moreover, it may be noted that, if we start with a set P which is already complete, by virtue of theorem 2, S may be exactly identified in polynomial time by means of multiplicity queries only. The algorithm is as follows:

$\mathcal{T} \leftarrow (P, \{\epsilon\}, T)$, where T is filled in with multiplicity queries.
make the table consistent, and fill missing values with queries
output $M(\mathcal{T})$.

The table will certainly be closed, as P is complete, and, by theorem 2, the final guess M must be correct. This result should be confronted with [3], where a similar framework is described for finite state automata: a regular language may be exactly identified in polynomial time, from membership queries only, if we are given a complete set of representants for Nerode's equivalence classes.

5 PAC-learnability

If we are not interested in the exact identification of $\mathcal{Q}$-automata, but only in a probably approximately correct (PAC) series, equivalence queries are not necessary, as they can be substituted by sampling of example strings with their correct multiplicity. The technique is analogous to the one used for DFAs in [4]. As a consequence, $\mathcal{Q}$-automata are PAC-learnable, when multiplicity queries are allowed.

If a non-deterministic automaton is unambiguous, the corresponding formal series S will be such that, for any string w, (S, w) is wither 0 or 1. In this case, then, multiplicity queries reduce to ordinary membership queries. Suppose now that a regular language L is recognized by an unambiguous NFA M, with a corresponding formal series S. The present paper gives an algorithm for PAC-learning a representation of S in polynomial time w.r.t. the number of states of M, if membership queries are allowed. In other words, following the usual terminology (see, e.g., [12]), regular languages are polynomially *predictable* using membership queries w.r.t. the representation of unambiguous non-deterministic automata. The importance of this lies in the fact that there are unambiguous NFAs such that the equivalent DFA has an exponentially larger number of states [15]. We then have a substantial improvement over previous results establishing predictability w.r.t. to a deterministic representation.

References

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms* Addison-Wesley, 1974.

[2] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39:337–350, 1978.

[3] D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51:76–87, 1981.

[4] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[5] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[6] F. Bergadano and A. Giordana and L. Saitta. *Machine Learning: an Integrated Framework and its Applications* Ellis Horwood, 1991.

[7] F. Bergadano and D. Gunetti. *An Interactive System to Learn Functional Logic Programs. Proc. Int. Joint Conf. on Artificial Intelligence*, Morgan Kaufmann, 1993.

[8] J. Berstel and C. Reutenauer. *Rational series and their languages* Springer-Verlag, Berlin, 1988.

[9] S. Eilenberg *Automata, Languages and Machines* Vol. A, Academic Press, New York, 1974.

[10] M. E. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[11] T. Harju and J. Karhumaki. Decidability of the multiplicity equivalence problem of multitape finite automata *Proc. 22nd STOC*, 477-481, (1990).

[12] B. K. Natarajan. *Machine Learning: a Theoretical Approach* Morgan Kaufmann, 1991.

[13] L. Pitt and M. K. Warmuth. The Minimum Consistent DFA Problem Cannot be Approximated within any Polynomial. *Journal of the ACM*, 40:95-142, 1993.

[14] A. Salomaa and M. Soittola. *Automata theoretic aspects of formal power series* Springer-Verlag, New York, 1978.

[15] R. E. Stearns and H. B. Hunt. On the Equivalence and Containment Problems for Unambiguous Regular Expressions, Regular Grammars and Finite Automata *SIAM J. Comput.*, 14:3,598-611, 1985.

[16] S. Varricchio. On the decidability of the equivalence problem for partially commutative rational power series *Theoretical Computer Science*, 99: 291-299, 1992.