# Identifying Unrecognizable Regular Languages by Queries

Claudio Ferretti – Giancarlo Mauri
{ferretti,mauri}@imiucca.csi.unimi.it

Dipartimento di Scienze dell'Informazione
via Comelico 39, 20135 Milano – Università di Milano, ITALY

**Abstract.** We describe a new technique useful in identifying a subclass of regular trace languages (defined on a free partially commutative monoid). We extend an algorithm defined by Dana Angluin in 1987 for DFA's and using equivalence and membership queries. In trace languages the words are equivalence classes of strings, and we show how to extract, from a given class, a string that can drive the original learning algorithm. In this way we can identify a class of regular trace languages which includes languages which are not recognizable by any automaton.

## 1 Introduction

Considering the problem of learning formal languages from examples, and specifically regular languages, [An87] gave an efficient algorithm to learn deterministic finite automata by membership and equivalence queries with counterexamples. We call this algorithm DFAL, and we will show how to extend it to learn other classes of languages, even not representable by automata. Other extensions to DFAL are, e.g., in [BR87, Sa90].

Recently, researches on formal models for concurrent processes underlined the importance of trace languages [Ma85], defined as subsets of a free partially commutative monoid (f.p.c.m.), and a theory of trace languages has been developed [BMS89, AR86], parallel to that of classical languages on free non-commutative monoids (string languages).

A fundamental difference between trace and string languages is that regular trace languages are in general not recognized by a finite state automaton on the f.p.c.m., i.e. Kleene's theorem cannot be generalized to them. As known results about the identification of regular languages are based on automata, here we discuss some modifications to use them on regular trace languages.

## 2 Definitions and Notations

Given a finite alphabet $\Sigma$ and the free monoid $\Sigma^\star$, a *concurrence relation* $\theta$ is a subset of $\Sigma \times \Sigma$ and $\equiv_\theta$ denotes the congruence relation on $\Sigma^\star$ generated by the set $C_\theta = \{(ab, ba) \mid (a, b) \in \theta\}$. The quotient $M(\Sigma, \theta) = \Sigma^\star / \equiv_\theta$ is the *free partially commutative monoid* associated with the concurrence relation $\theta$.

An element of $M(\Sigma, \theta) = \Sigma^\star / \equiv_\theta$ is a *trace*, and can be seen as a set of strings. Given a string $s$, $[s]_\theta$ is the trace containing $s$; given a string language $L$, $[L]_\theta$ is the set of traces containing at least one string from $L$.

Any $T \subseteq M(\Sigma, \theta)$ is a *trace language.* As in the sequential case, the class $RTL_\theta$ of *regular* trace languages on $M(\Sigma, \theta)$ can be defined as the least class containing finite trace languages and closed with respect to set-theoretic union, concatenation and $(\cdot)^\star$ closure of languages, being the concatenation of two traces the equivalence class of the concatenations of their strings. I.e., these languages are defined by regular expressions on finite sets. Moreover, it can be shown that a trace language $T$ is regular if and only if there is a regular string language $L$ such that $T = [L]_\theta$ [BMS89].

We will consider only the case in which $\theta$ is a transitive relation. As a consequence, the maximal cliques of the graph associated to the concurrence relation induce a partition on $\Sigma$. Two letters will be in the same element of this partition if and only if they are nodes of the same maximal clique in the graph associated to $\theta$, i.e., if and only if they commute in $C_\theta$. So we can define the alphabet as the partition: $\Sigma = \bigcup_{i=1}^{n} c_i$, where $n$ is the number of maximal cliques in the graph of $\theta$. The term *clique* is from now on extended to the elements $c_i$ of the partition on $\Sigma$, when not explicitly referred to the graph of $\theta$. Given this, we can prove a useful result, where letters from $\Sigma$ are grouped as the variables in a usual algebraic monomial:

**Theorem 1.** *Each trace $t$ in $M(\Sigma, \theta)$ can be uniquely represented as a sequence of monomials $t_1 \ldots t_m$, where all the letters of each monomial $t_i$ are from the same clique, and any two adjacent monomials are for different cliques.*

*Proof.* (Sketch) Any string is divided in groups of letters that never mix together.

Given a monomial $t_i$, $|t_i|_{a_j}$ denotes the degree of $a_j$ in $t_i$, and $MCD(|t_i|_\Sigma)$ denotes the Maximum Common Divisor of the degrees of the letters in $t_i$.

## 3 Main Results

Our results apply to the restricted class of regular trace languages defined by a transitive $\theta$ and by regular expressions where, when an operation of the expression joins two different traces, the trailing letters of the first don't commute in $\theta$ with the leading letters of the second. This means that in such a regular expression the joining of different traces never mix letters, while this is still allowed when concatenating one or more copies of the same trace. We call them *isolating* regular expressions and *isolating* regular trace languages. This subclass of $RTL_\theta$ offers a way to extract strings with interesting properties from each trace. Given $\Sigma = \{a, b, x\}$ and $\theta = \{(a, b)\}$, isolating languages are: $[ax \cdot \{axb\}^\star]_\theta$, $[\{ab\}^\star]_\theta$.

### 3.1 Choosing a String

Given the monoid $M(\Sigma, \theta)$, with $\theta$ transitive, we can choose from any trace $t$, represented by the sequence of monomials $t_1 \ldots t_m$, the string $s = s_1 \ldots s_m$ made in the following way: for each $t_i$ write the string $s_i = a_1^{p_1} a_2^{p_2} \ldots a_1^{p_1} a_2^{p_2} \ldots$, where $a_i$ is a letter and $p_j = |t_i|_{a_j} / MCD(|t_i|_\Sigma)$. Let's call these strings *ordered* strings. E.g., given that $(a, b)$ is in $\theta$, the trace $[aaabbbbbb]_\theta$ can be represented by $a^3 b^6$, and the corresponding ordered string is *abbabbabb*.

The first key property of this rule is that the ordered string of a trace, obtained concatenating an unbounded number of times the same unknown trace, is the concatenation of the ordered strings of the repeated trace.

**Lemma 2.** *If the trace $t$ is represented by a single monomial, and $os(t)$ is the ordered string of $t$, then $os(t \cdot t) = os(t) \cdot os(t)$.*

*Proof.* The single monomial representing $t \cdot t$ will have each letter with twice the degree it has in $t$. Therefore also the $MCD$ is doubled, and the exponents $p_i$ in the resulting ordered string will be the same. Then this string will be the concatenation of two copies of the ordered string of $t$. □

When the concatenated trace is more complex we can state a weaker property: from any trace generated by the closure of a regular language, the ordered string we choose belongs to a slightly bigger regular language generating the same traces.

**Lemma 3.** *If $\theta$ is transitive, $s = s_1 s_2 s_3$ is a string on $\Sigma$, with strings $s_1$ and $s_3$ containing letters from the same clique, and $s_2$ an ordered string with trailing and leading letters from cliques different from that of $s_1$ and $s_3$:*

$$[\{L \cup \{s_1 s_2 s_3\}\}^\star]_\theta = [\{L \cup \{s_1 s_2 \{s' s_2\}^\star s_3\}\}^\star]_\theta,$$

*where $s'$ is the ordered string of $[s_3 s_1]_\theta$.*

*Proof.* Clearly, $[s']_\theta = [s_3 s_1]_\theta$, and the inner closure adds strings to the language between square brackets, but doesn't add new traces to the trace language. Any trace $[\ldots s_1 s_2 s_3 s_1 s_2 s_3 \ldots]_\theta$, generated by the first language, will contain also the string $\ldots s_1 s_2 s' s_2 s_3 \ldots$, which belong to the second language and that is its ordered string. □

Given any regular expression for $T$ we can find an equivalent, w.r.t. $\theta$, regular expression on $\Sigma^\star$ containing the ordered strings. This means that the ordered strings of traces of $T$ belong to a regular trace language $L$ such that $[L]_\theta = T$.

**Theorem 4.** *Given an isolating regular trace language $T$ over a transitive concurrence relation $\theta$, there exists a regular language $L$ on $\Sigma^\star$ such that $[L]_\theta = T$ and any ordered string extracted from $t \in M(\Sigma, \theta)$ belongs to $L$ if and only if $t$ belongs to $T$.*

*Proof.* (Sketch) Consider the regular expression that defines $T$ as being built from finite trace languages, applying to them many subsequent union, concatenation, and closure operations. We will build a regular expression on $\Sigma^\star$, that defines a language $L$ which satisfies our statement, by induction on the structure of the regular expression for $T$, using the properties stated for ordered strings over regular operations.

E.g., when $T = T'^\star$, being it a union of concatenations, we consider different cases of joined traces, and cover this unbounded operation using Lemma 3. □

### 3.2 Identifying Isolating Languages

We can identify an isolating target language $T$ using membership and equivalence queries with counterexamples using DFAL on $\Sigma^*$ [An87] to identify a regular language $L$ such that $[L]_\theta = T$. DFAL identifies an automaton, but in this way it represents $L$ and then an isolating regular trace language, and some of these have no finite automaton recognizing them, as it is for $[\{ab\}^*]_\theta$ (otherwise one could obtain by regular operations $\{a^n b^n \mid n \geq 0\}$, which is not regular).

Given any regular expression for $T$, by Theorem 4 we know that there exists an equivalent, w.r.t. $\theta$, regular expression made of ordered strings. This means that the ordered strings of traces of $T$ belong to a regular trace language $L$ such that $[L]_\theta = T$. This same $L$ is the real target of DFAL. The interactions between teacher and learner are filtered substituting counterexamples traces by their ordered strings, and strings from DFAL with the traces that are their equivalence classes in the f.p.c.m. This operation requires polynomial time in the relevant parameters, except when substituting a negative counterexample, where the best known method requires exponential time.

Given the regular expression for $T$, $n$ the number of states of the minimal DFA recognizing a regular string language $L$ such that $[L]_\theta = T$. Our operations enlarge the corresponding regular expression but this additions cannot require more than a polynomial, in $n$, number of new states in the automaton recognizing the new regular language.

We can then apply the results of [An87] on learning with DFAL a DFA of $p$ states, with $p$ polynomial in $n$ and, together with the observation on the time required to process a negative counterexample, we can state the following

**Theorem 5.** *Isolating regular trace languages with transitive concurrence relation, and generated by a language recognized by a DFA of $n$ states, can be exactly identified in polynomial time in $n$ and in the length of positive counter examples.*

It would be interesting to refine the algorithm making use of the information we have about the structure of the extracted regular language, trying to reduce the exponential dependence on the length of negative counterexamples.

## References

[AR86] IJ.Aalbersberg, G.Rozenberg. Theory of traces. *Theor. Comp. Sci.*, 60:1–, 1986.

[An87] D.Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–, 1987.

[BR87] P.Berman, R.Roos. Learning one-counter languages in polynomial time. In *Proc. of the Symp. on Found. of Comp. Sci.*, 61–, 1987.

[BMS89] A.Bertoni, G.Mauri, N.Sabadini. Membership problems for regular and context-free trace languages. *Information and Computation*, 82:135–, 1989.

[Ma85] A.Mazurkiewicz. Semantics of concurrent systems: A modular fixed point trace approach. *Lect. Notes in Comp. Sci.*, vol. 188, 353–, Springer-Verlag, 1985.

[Sa90] Y.Sakakibara. Inductive inference of logic programs based on algebraic semantics. *New Generation Computing*, 7:365–, 1990.