INFORMATION AND COMPUTATION **82**, 135–150 (1989)

Membership Problems for Regular and Context-Free Trace Languages*

A. BERTONI, G. MAURI, AND N. SABADINI

Dipartimento di Scienze dell'Informazione, Università di Milano, Milan, Italy

Trace languages have been introduced in order to describe the behaviour of concurrent systems in the same way as usual formal languages do for sequential system. They can be defined as subsets of a free partially commutative monoid and a theory of trace languages can be developed, generalizing the usual formal languages theory. In this paper, the time complexity of membership problems for regular and context-free trace languages is investigated. It is proved that the membership problem for context free trace languages can be solved in time $O(BM(n^\alpha))$, where α is the dimension of the greatest clique of the concurrency relation C and $BM(n)$ is the time required for multiplying two arbitrary $n \times n$ boolean matrices. For regular trace languages, our method gives an algorithm which requires $O(n^\alpha)$ time. Finally, the uniform membership problem is shown to be NP-complete. © 1989 Academic Press, Inc.

1. INTRODUCTION

A *trace*, as introduced by Mazurkiewicz (1977), can be considered to be a model of a process in a concurrent system, at the same abstraction level as a string is a model of a process in a sequential system. The main difference is that a string is a linearly (totally) ordered set of symbol occurrences, each symbol representing an event that can take place in the system, while a trace is a partially ordered set of symbol occurrences. In fact, as pointed out by Petri (1977), imposing a linear (temporal) ordering on occurrences of concurrent events is an arbitrary restriction, since they can happen in any order, or even at the same time. For this reason, partial orders have been proposed to model concurrent processes by several authors (e.g., Nielsen, Plotkin, and Winskel (1981), Degano and Montanari (1985), Winskel (1986), Gischer (1984), Pratt (1986)).

In Mazurkiewicz (1977), a different aspect of traces is stressed, by defining a trace as an equivalence class of words over a given alphabet that differ only in that some symbols, representing concurrent actions, are commuted. The main advantage of such a point of view is the fact that it

* This research has been supported by Ministero della Pubblica Istruzione, under Project "Theory of Algorithms."

naturally leads to a further abstraction step, since traces can be abstractly defined as elements of a suitable free partially commutative monoid (fpcm, for short); such elements admit some different concrete representations, among which partial orders and equivalence classes. This allows us, therefore, to deal with traces in the framework of the theory of fpcm's, introduced and developed by Cartier and Foata (1969) in order to give an algebraic interpretation of MacMahon's master theorem (see also Lallement, 1979).

Furthermore, the definition of trace languages as subsets of a fpcm gives a complete analogy with the usual definition of languages as subsets of a free (non-commutative) monoid, hence, giving a sound mathematical basis for developing a theory of trace languages, which generalizes the classical theory of formal languages. In particular, a Chomsky-like classification for trace languages may be given, and the usual problems (closure properties, equivalence problems, membership problems, algebraic characterization and so on) for the classes so obtained can be studied.

An extensive analysis of regular trace languages and their subclasses has been carried out by Szijarto (1981), Knuth (1978), Bertoni, Mauri, and Sabadini (1981, 1982a, 1982b), Ochmanski (1985), Mazurkiewicz (1977, 1985), Janicki (1978), and Zielonka (1987). A comprehensive review of the state of the art about trace languages can be found in Aalbersberg and Rozenberg (1986).

In this paper, we are interested in the computational complexity of the (uniform and nonuniform) membership problem for regular and context-free trace languages. In this direction, Rytter (1984) has proved, with an application of the theory of multihead nondeterministic automata, that (nonuniform) membership problem for context-free trace languages can be solved in time $O(n^{3\alpha}/\log n)$, where α is the dimension of the greatest clique of the concurrency relation C, and in space $O(\log^2 n)$. Here, we are interested in time complexity. With a different approach, in Section 6 we exhibit a "fast" reduction of the membership problem for context-free trace languages to the transitive closure of a matrix on a non associative algebra. Hence, generalizing the well-known result of Valiant (1975) on general context-free recognition in less than cubic time, we prove that (nonuniform) membership problem for context-free trace languages can be solved in time $O(BM(n^\alpha))$, where α is as above and $BM(n)$ is the time required for multiplying two arbitrary $n \times n$ boolean matrices. For regular trace languages, this method gives an algorithm which requires $O(n^\alpha)$ time. In order to obtain such results, it is required an efficient representation of prefixes and intervals of a trace described in Section 3 and 4. Sections 2 and 5 contain the basic definitions about trace languages and their Chomsky-like classification. Finally, in Section 7 it is shown that the uniform membership problem is NP complete.

2. Free Partially Commutative Monoids and Traces

In this section, we will recall the basic definitions and facts about trace languages and the algebraic structure supporting them, i.e., free partially commutative monoids. In the following, the reader will be supposed to be acquainted with the standard notions on automata and formal languages (Hopcroft and Ullman, 1969).

Definition 2.1. A *concurrent alphabet* is a pair $\langle \Sigma, C\rangle$, where:

(a) $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ is a finite alphabet;

(b) $C \subseteq \Sigma \times \Sigma$ is a symmetric and irreflexive relation, the *concurrency* relation.

An important characteristic of the concurrency relation we will use in the following is its clique number.

Definition 2.2. Given a concurrency relation C on a set Σ, the *clique number* of C is the maximal cardinality of the cliques of C:

$$\mathrm{cn}(C) = \mathrm{Max}\{|S| \mid S \subseteq \Sigma \text{ and } S \text{ is a clique of } C\}.$$

We remember that a clique in a graph is complete subgraph.

Now, the following definition allows us to consider as being equivalent sequences of actions which differ only for the order in which concurrent actions are executed.

Definition 2.3. The *free partially commutative monoid* (fpcm, for short) generated by a concurrent alphabet $\langle \Sigma, C\rangle$ is the initial object $F(\Sigma, C)$ of the category of monoids generated by the elements of Σ and satisfying, besides the monoid equations, the set of "commutativity laws" :

$$\{ab = ba \mid a, b \in \Sigma \text{ and } (a, b) \in C\}.$$

Definition 2.4. A *trace* on a concurrent alphabet $\langle \Sigma, C\rangle$ is any element $t \in F(\Sigma, C)$; a *trace language* is any subset $T \subseteq F(\Sigma, C)$.

By such a definition, a trace is an abstract object, which can be represented in many different ways. In the usual representation, $F(\Sigma, C)$ is the quotient structure $F(\Sigma, C) = \Sigma^*/\equiv_C$, where $\equiv_C$ is the least congruence on Σ^* which extends the set of commutativity laws, and a trace is a congruence class of words. As usual, the congruence class of the word $w \in \Sigma^*$ is denoted by $[w]_C$, the composition on $F(\Sigma, C)$ is defined by $[w]_C \cdot [v]_C = [w \cdot v]_C$ and the identity is the class $[\varepsilon]_C = \{\varepsilon\}$, where ε denotes the empty string.

The representation of a trace as a poset can be obtained as follows.

Let $\langle \Sigma, C \rangle$ be a concurrent alphabet and $x = x_1 x_2 \cdots x_n \in \Sigma^*$ ($x_k \in \Sigma$, $1 \leqslant k \leqslant n$); x is intended as a representative element of the trace $t = [x]_C \in F(\Sigma, C)$. To such a trace we can associate the partial order $\text{ord}(t) = \langle P_t, \leqslant \rangle$, where:

(1) $P_t = \{(x_1, k_1), ..., (x_n, k_n)\}$, where k_s denotes the number of symbols equal to x_s in the string $x_1 x_2 \cdots x_s$;

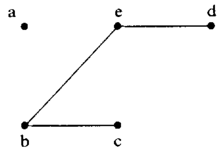
(2) $\leqslant$ is the transitive closure of the relation L defined by:

$$(x_i, k_i)\, L\, (x_j, k_j) \qquad \text{iff} \quad i \leqslant j \qquad \text{and} \qquad \text{not}(x_i\, C\, x_j).$$

$\langle P_t, \leqslant \rangle$ represents the trace $[x]_C$ in the sense that equivalent strings determine the same ordering; on the other hand, given the ordering $\langle P_t, \leqslant \rangle$, the equivalence class $[x]_C$ can be obtained considering all the total orderings compatible with it.

Since the algorithms presented in the following receive as input the string $x_1 x_2 \cdots x_n$, it is useful to denote the element (x_i, k_i) of P_t by $\langle x_i, i \rangle$, when we need to put into evidence that the symbol x_i is in position i in the string.

EXAMPLE. Let $\Sigma = \{a, b, c, d, e\}$ and C defined by the graph:



The trace $[abcdeac]_C$ is described by:

$$\begin{tikzcd}
 & \boxed{c} \arrow[r] \arrow[rd] & \boxed{e} \arrow[rd] & & \\
\boxed{a} \arrow[ru] \arrow[rd] & & & \boxed{a} \arrow[r] & \boxed{c} \\
 & \boxed{b} \arrow[r] & \boxed{d} \arrow[ru] & &
\end{tikzcd}$$

The representation of a trace as a poset allows us to give an algebraic counterpart for concepts related to ordering relations and vice versa. For instance, the concept of order ideal of a poset $\langle E, \leqslant \rangle$, i.e., a set $A \subseteq E$ such that, if $x \in A$ and $y \leqslant x$, then $y \in A$, can be identified with the algebraic notion of prefix of a trace.

DEFINITION 2.5. Given a fpcm $F(\Sigma, C)$, a trace x is said to be a *prefix* of a trace y, denoted by $x \angle y$, iff there is a z such that $y = x \cdot z$.

The following theorem gives a strong relation among prefixes and order ideals:

THEOREM 2.1. *The set* $\langle \mathrm{Pre}_t, \angle \rangle$ *of prefixes of a trace* t, *with the prefix relation* $\angle$, *is isomorphic to the lattice of the order ideals of* $\langle P_t, \leqslant \rangle = \mathrm{ord}(t)$.

Proof. By induction on the length of t, remembering that in $F(\Sigma, C)$ the left cancellation law holds.

As a consequence, the notions of prefix and order ideal can be "identified," and all the well-known operations on order ideals (in particular the set theoretic union $\cup$) can be transferred to prefixes.

3. On Representing Prefixes of a Trace

In this section, we will discuss the possibility of representing the prefixes of a given trace by means of arrays. Since many procedures on traces can be easily defined by recursion on their prefix structure, an efficient implementation of such procedures relies on the possibility of efficiently representing the prefixes of a given trace. Our main result is an algorithm that, for every string $x \in \Sigma^*$ interpreted as the representative of the trace $t = [x]_C \in F(\Sigma, C)$, computes the set of the representations of the prefixes of t in time $O(|x|^\alpha)$, being α the clique number of C.

In order to design and analyze the algorithm we will give in the following, we recall some definitions and combinatorial results (for more information on the subject, see Aigner, 1979, pp. 30–40).

DEFINITION 3.1. Let $\langle \Sigma, C \rangle$ be a concurrent alphabet, $x = x_1 x_2 \cdots x_n$ an element of Σ^*, $t = [x]_C \in F(\Sigma, C)$ and $\mathrm{ord}(t) = \langle (x_1, k_1), ..., (x_n, k_n)\}$, $\leqslant \rangle$ the corresponding partial order. Given $(x_j, k_j) \in \mathrm{ord}(t)$, let $I_j = \{(x_i, k_i) \mid (x_i, k_i) \leqslant (x_j, k_j)\}$; I_j is said to be the *principal order ideal* generated by (x_j, k_j).

FACT 3.1 (Dilworth, 1950). *The set of order ideals of* $\mathrm{ord}(t)$ *is isomorphic to the set of antichains of* $\mathrm{ord}(t)$, *by associating with every antichain A the ideal which is the union of the principal ideals generated by the elements of A. Vice versa, the antichain corresponding to a given ideal I is the set of maximal elements of I.*

Now, we introduce the notion of representation.

DEFINITION 3.2. Given a trace t, the *representation* of the prefixes of t is the function

$$R_t\colon \mathrm{Pre}_t \to N^{|\Sigma|}$$

defined by

$$R_t(s) = (|s|_{\sigma_1}, ..., |s|_{\sigma_h}) \qquad (\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_h\}, s \in \mathrm{Pre}_t)$$

where $|s|_\sigma$ denotes the number of symbols $\sigma \in \Sigma$ in s.

Let now $\mathbf{v} = (v_1, ..., v_h)$, $\mathbf{v}' = (v'_1, ..., v'_h)$ be two vectors with integer non-negative components. We define:

(1) $\mathbf{v} \sqcup \mathbf{v}' = (j_1, ..., j_h)$, where $j_k = \mathrm{Max}\{v_k, v'_k\}$ $(1 \leqslant k \leqslant h)$

(2) $\mathbf{v} \leqslant \mathbf{v}'$ iff $\forall k$ $(v_k \leqslant v'_k)$

(3) $\|\mathbf{v}\| = \sum_{k=1}^h v_k$.

It follows that:

FACT 3.2. (a) $R_t\colon \mathrm{Pre}_t \to N^{|\Sigma|}$ *is an injective function*;

(b) *for every pair* I, I' *of ideals of* $\mathrm{ord}(t)$, $R_t(I \cup I') = R_t(I) \sqcup R_t(I')$;

(c) $\|R_t(I)\| = |I|$.

From Fact 3.1, every ideal I can be obtained as union of principal ideals, i.e., $I = \bigcup_{k \in \xi} I_k$ for a suitable set of indices ξ.

From Fact 3.2 the following result holds:

$$R_t(I) = R_t\left(\bigcup_{k \in \xi} I_k\right) = \bigsqcup_{k \in \xi} R_t(I_k).$$

We are now ready to construct an algorithm which, when receiving as input the string $x = x_1 x_2 \cdots x_n \in \Sigma^*$, denoting the trace $t = [x]_C$, outputs the sets:

$\mathbb{P}^{(k)}$ = set of the representations of prefixes of length k $(1 \leqslant k \leqslant n)$.

Such an algorithm consists of two subalgorithms, performing the following steps:

(1) the representations of the principal ideals $I_1, ..., I_n$ of $\mathrm{ord}(t)$ are generated;

(2) the sets $\mathbb{P}^{(k)}$ $(1 \leqslant k \leqslant n)$ are constructed by using the above obtained representations.

ALGORITHM 1.

Input. An array $(x_1, x_2, ..., x_n)$, with $x_i \in \Sigma$, which denotes the trace $t = [x_1, x_2, ..., x_n]_C$
Output. An array of vectors $(\mathbf{v}(1), ..., \mathbf{v}(n))$, where $\mathbf{v}(k) \in N^{|\Sigma|}$ is the representation of the principal ideal I_k.

```
begin
  for $\sigma, \rho \in \Sigma$ do $N_{\sigma,\rho}(0) := 0$;
  for $0 \leqslant k < n$ do
    for $\sigma, \rho \in \Sigma$ do
      begin
        if $x_{k+1} \neq \sigma$ then $N_{\sigma,\rho}(k+1) := N_{\sigma,\rho}(k)$;
        if $x_{k+1} = \sigma = \rho$ then $N_{\sigma,\rho}(k+1) := N_{\sigma,\rho}(k) + 1$;
        if $x_{k+1} = \sigma \neq \rho$ then $N_{\sigma,\rho}(k+1) := \mathrm{Max}\{N_{\mu,\rho}(k) \mid \mathrm{not}(\mu C \sigma)\}$
      end;
  for $1 \leqslant k \leqslant n$ do if $x_k = \sigma$ then $\mathbf{v}(k) := (N_{\sigma,\sigma'_1}(k), ..., N_{\sigma,\sigma'_h}(k))$
end.
```

The correctness of Algorithm 1 is proved by showing, by induction, that $N_{\sigma,\rho}(k)$ is the number of symbols ρ in the prefix of t associated with the order ideal generated by $\langle \sigma, j \rangle$, with $x_j = \sigma$ and $x_i \neq \sigma$ $(j < i \leqslant k)$; if $x_i \neq \sigma$ for every $1 \leqslant i \leqslant k$, then $N_{\sigma,\rho}(k) = 0$. The computation time is $O(n)$.

ALGORITHM 2.

Input. Two arrays $(x_1, x_2, ..., x_n)$, $(\mathbf{v}(1), ..., \mathbf{v}(n))$ which represent the input and the output of Algorithm 1, respectively;
Output. The class $\{\mathbb{P}^{(0)}, ..., \mathbb{P}^{(n)}\}$, where $\mathbb{P}^{(k)}$ is the set of the representations of k-element prefixes of the trace $t = [x_1 x_2 \cdots x_n]_C$.

```
begin
  $A$ := all the subsets of $\{1, 2, ..., n\}$ with at most $\alpha$ elements, where $\alpha$ is the clique number of
       $C$, different from $\varnothing$;
  $X^{(1)} := X^{(2)} := \cdots := X^{(n)} := \varnothing$; $X^{(0)} := \{(0, 0, ..., 0)\}$;
  for $S \in A$ do
  *$S \equiv \{k_1, ..., k_\beta\}$ with $k_i \in \{1, 2, ..., n\}$ $(1 \leqslant i \leqslant \beta)$ and $\beta \leqslant \alpha$*
    begin
      $V := \varnothing$;
      for $k \in S$ do $V := V \cup \{\mathbf{v}(k)\}$;
      *$V$ contains the representations of the principal ideals $I_{k1}, ..., I_{k\beta}$*
      if $V$ is an antichain of $\langle N^{|\Sigma|}, \leqslant \rangle$ then
        begin
          $\mathbf{v} := \bigsqcup_{k \in S} \mathbf{v}(k)$;
          *$\mathbf{v}$ is the representation of the ideal associated with the antichain
           $\{\langle x_k, k \rangle \mid k \in S\}$*
          $j := \|\mathbf{v}\|$;
          *$j$ is the number of elements in the ideal represented by $\mathbf{v}$*
          $X^{(j)} := X^{(j)} \cup \{\mathbf{v}\}$
        end
    end;
  output $(X^{(0)}, X^{(1)}, ..., X^{(n)})$
end.
```

The correctness of Algorithm 2 can be easily proved on the basis of the comments included in the text, taking into account that every ideal is generated by an antichain and that every antichain contains at most α elements. In order to find the complexity of the algorithm, we observe that the critical steps, which depend on the length n of the input, are the generation of subsets of $\{1, 2, ..., n\}$ of cardinality α at most and the **for** cycle on such subsets. The computation time is then bounded, up to a constant, by the number of such subsets, i.e.:

$$\binom{n}{1}+\binom{n}{2}+\cdots+\binom{n}{\alpha}=O(n^{\alpha})$$

We can conclude that the algorithm requires $O(n^{\alpha})$ time.

4. On Representing Intervals in a Trace

The representation of prefixes given in the previous section can be easily used for representing intervals. In fact, given a trace t, let us consider the decomposition $t=\alpha \cdot x \cdot \beta$. Since in the monoid $F(\Sigma, C)$ the right cancellation law holds, the pair (α, x), where α and αx are prefixes of t, univocally determines the above decomposition.

Definition 4.1. An *interval* in the trace t is a pair (α, x) such that both α and αx are prefixes of t.

The set Int_t of the intervals of t can be equipped with a partial composition operation $\circ$ as follows:

Definition 4.2. If for (α, x), $(\beta, y) \in \text{Int}_t$ we have $\beta = \alpha x$, then (α, x) and (β, y) are said to be *composable*, and their composition is $(\alpha, x) \circ (\beta, y) = (\alpha, xy)$.

Let us consider now the set $N^{|\Sigma|} \times N^{|\Sigma|}$ with the partial composition operation $\diamond$, where:

$(\mathbf{a}, \mathbf{b}), (\mathbf{c}, \mathbf{d}) \in N^{|\Sigma|} \times N^{|\Sigma|}$ are composable iff $\mathbf{b} = \mathbf{c}$, and their composition is $(\mathbf{a}, \mathbf{b}) \diamond (\mathbf{c}, \mathbf{d}) = (\mathbf{a}, \mathbf{d})$.

The representation of the intervals of t is the function

$$S_t \colon \text{Int}_t \to N^{|\Sigma|} \times N^{|\Sigma|}$$

defined by

$$S_t(\alpha, x) = (R_t(\alpha), R_t(\alpha x)),$$

where R_t is the representation of prefixes defined in the previous section. We have:

THEOREM 4.1. S_t: $\langle \mathrm{Int}_t, \circ \rangle \rightarrow \langle N^{|\Sigma|} \times N^{|\Sigma|}, \diamond \rangle$ *is a one-to-one morphism.*

It follows that S_t is an isomorphism on its homomorphic image; the inverse function will be denoted by S_t^{-1}.

5. Regular and Context-free Trace Languages

In the theory of formal languages on a monoid M, relevant subclasses of languages can be characterized in terms of closure properties with respect to operations on languages. More precisely, given a monoid M, any subset $A \subseteq M$ will be said to be a language on M. Here, we will study the particular case where M is the fpcm $F(\Sigma, C)$.

Let us define on the set $2^{F(\Sigma, C)}$ of trace languages the operations $+$, $\circ$, and $(-)^*$, where $+$ denotes the set-theoretic union, $\circ$ the concatenation of languages (i.e., $L_1 \circ L_2 = \{t \mid t = xy, x \in L_1, y \in L_2\}$), and $(-)^*$ the closure operation, which associates with a language L the least submonoid of $F(\Sigma, C)$ which contains L. It is known that $L^* = \bigcup_{k \geqslant 0} L^k$.

As in the sequential case, the class $\mathrm{Reg}(\Sigma, C)$ of regular trace languages on $F(\Sigma, C)$ can be defined as the least class containing finite trace languages and closed with respect to the above operations. Furthermore, Mazurkiewicz (1977) extended to trace languages the algebraic characterization of regular languages as solutions of right linear equations on language variables, such as

$$X_i = \sum_{k=1}^{n} A_{ik} \cdot X_k + B_i \qquad (1 \leqslant i \leqslant n)$$

with A_{ik} and B_i finite trace languages and $A_{ik} \neq [\varepsilon]$.

Here, we give a similar definition of the set of context free or algebraic trace languages as the set of languages which are solution of an algebraic system of the form

$$X_i = \sum_{k,h} A_{ikh} \cdot X_k \cdot X_h + \sum_{\sigma \in \Sigma} B_{i\sigma} \cdot \sigma \qquad (1 \leqslant i \leqslant n)$$

with A_{ikh}, $B_{i\sigma} \in \{\{\varepsilon\}, \varnothing\}$ and $\sigma \in \Sigma$.

Regular and algebraic languages have been studied for some classes of monoids which are particular cases of fpcm's. Besides the well-known case of the free monoid Σ^*, corresponding to $C = \varnothing$, we can recall Parikh languages (Parikh, 1966), which are defined on totally commutative monoids, i.e., $C = \Sigma^2$, and rational relations, which correspond to regular languages in the (partially commutative) monoid $\Sigma_1^* \times \Sigma_2^*$ (Eilenberg, 1974).

Now, given a language $L \subseteq \Sigma^*$, it is possible to associate with it a trace language as follows:

DEFINITION 5.1. Given a language L on a finite alphabet Σ and a concurrency relation C on Σ, the set $[L]_C = \{[w]_C \mid w \in L\}$ is the *trace language generated by L under C*.

The following easy result relates the above classes of trace languages to the usual classes of languages:

THEOREM 5.1. *A trace language T is regular (context-free) iff there is a regular (context-free) language L on Σ^* such that $T = [L]_C$.*

6. THE MEMBERSHIP PROBLEM FOR TRACE LANGUAGES

The membership problem is a classical problem on languages that is known to be solvable in polynomial time both for regular and context free languages. In the case of trace languages, we will suppose to have a concurrent alphabet $\langle \Sigma, C \rangle$ and a grammar $G = \langle V_N, \Sigma, S, \mathbb{P} \rangle$, where V_N is the set of nonterminal symbols, Σ the terminal alphabet, S the axiom, and $\mathbb{P}$ the set of productions; we denote with T_G the trace language $[L_G]_C$, where L_G is the language generated by G. Then the membership problem for trace languages can be defined as follows:

PROBLEM (MPTL).

Instance. A word $x \in \Sigma^*$
Question. $[x]_C \in T_G$?

The following theorem show that MPTL can be solved in polynomial time in the case where G is a context free grammar, which we suppose to be given in Chomsky normal form.

THEOREM 6.1. *Given a concurrent alphabet $\langle \Sigma, C \rangle$ and a context free trace language T, the membership problem for T can be solved in time $O(BM(|x|^\alpha))$, where $|x|$ is the length of the input word x, α is the clique number of C, and $BM(n)$ is the time required for multiplying two arbitrary $n \times n$ boolean matrices.*

To prove this statement, which is a nontrivial generalization of a well-known result of Valiant (1975) on general context-free recognition in less than cubic time, we associate with every trace t the set:

$$Q(t)=\{A \mid A \in V_N \wedge \exists y([y]_C=t \wedge A \Rightarrow^* y)\}.$$

Hence, $t \in T$ iff the axiom $S \in Q(t)$. Furthermore, $Q(t)$ can be recursively defined as

$$\begin{aligned} Q(t)= & \ \textbf{if } |t|=1 \textbf{ then } \{A \mid A \in V_N, A \rightarrow t \in \mathbb{P}\} \\ & \ \textbf{else } \{A \mid A \in V_N, A \rightarrow A'A'' \in \mathbb{P}, A' \in Q(t'), A'' \in Q(t''), t't''=t\}. \end{aligned}$$

We can observe that, in order to compute $Q(t)$, the recursion requires the computation of $Q(p)$ for every interval (α, x) of t. Using this observation, we will be able to transform the problem of computing $Q(t)$ in the computation of the transitive closure of a suitable matrix. To clarify this reduction, we recall some definitions and results from Valiant (1975).

Let X be a finite set and $\diamond : 2^X \times 2^X \rightarrow 2^X$ a binary operation, left and right distributive with respect to the union operation. Now, let us consider the set M_n of $n \times n$ matrices with subsets of X as elements, with the operations $+$ (sum), $\circ$ (product), and $^+$ (transitive closure):

DEFINITION 6.1. Given $A, B \in M_n$, then:

$$(A+B)_{ij}=A_{ij} \cup B_{ij}$$

$$(A \circ B)_{ij}=\bigcup_k A_{ik} \diamond B_{kj}$$

$$A^+=A^{(1)}+A^{(2)}+\cdots+A^{(n)}+\cdots,$$

where

$$A^{(1)}=A \qquad \textit{and} \qquad A^{(i)}=\bigcup_{j=1}^{i-1} A^{(j)}A^{(i-j)}.$$

We remark that, in general, the product is not associative. As proved in Valiant (1975), the following holds:

FACT 6.1. *Given an upper triangular matrix $A \in M_n$, its transitive closure A^+ can be computed in time $O(BM(n))$, where $BM(n)$ is the time required for the multiplication of arbitrary $n \times n$ boolean matrices.*

Let us go back to our subject; given the representation S_t of the intervals of t, we can reduce the problem MPTL for context-free languages to the transitive closure of a matrix in two steps, as follows:

(1) Given a context free grammar $G = \langle V_N, \Sigma, S, \mathbb{P} \rangle$, we associate with every $x \in \Sigma^*$, representative of the trace $t = [x]_C$, a matrix B in the following way:

(a) the "indexes" of the elements of B are the representations of the prefixes of t. With a slight modification of the algorithm given in the previous section, it is not difficult to construct a vector $[\mathbf{v}_1, ..., \mathbf{v}_s]$ whose elements are these indexes, such that the component $\mathbf{v}_i$ follows $\mathbf{v}_k$ in the vector if $\mathbf{v}_i \geqslant \mathbf{v}_k$, $\geqslant$ being the order relation introduced in $N^{|\Sigma|}$.

(b) The components of B are subsets of V_N. In particular, we define:

$$B_{\mathbf{v}_i, \mathbf{v}_j} = \begin{cases} A_k = \{X \mid X \in V_N \text{ and } X \to \sigma_k \in \mathbb{P}\} \text{ if } \mathbf{v}_i < \mathbf{v}_i \text{ and } (\mathbf{v}_i, \mathbf{v}_j) \text{ is} \\ \qquad \text{the representation of the interval } \langle \alpha, \alpha\sigma_k \rangle \qquad (\sigma_k \in \Sigma) \\ \varnothing \qquad \text{otherwise.} \end{cases}$$

We observe that B is an $m \times m$ upper triangular matrix, with $m = O(|x|^\alpha)$. Furthermore, it can be constructed in time $O(|x|^{2\alpha})$.

(2) We introduce in 2^{V_N} the binary operation $\diamond$:

$$A \diamond B = \{X \mid X \in V_N,\ X \to YZ \in \mathbb{P},\ Y \in A, Z \in B\} \qquad (A, B \subseteq V_N)$$

and compute the transitive closure of B, $B^+ = \bigcup B^{(k)}$. By induction, it is possible to prove that:

$$(B^{(k)})_{\mathbf{v}_i, \mathbf{v}_j} = \begin{cases} Q(p) & \text{if } \mathbf{v}_i < \mathbf{v}_j \text{ and } (\mathbf{v}_i, \mathbf{v}_j) \text{ is the representation} \\ & \quad p \text{ of } t \text{ of length } k \\ \varnothing & \text{otherwise.} \end{cases}$$

So, we can conclude:

$$[x]_C \in T_G \qquad \text{iff } S \in B_{\mathbf{v}_1, \mathbf{v}_s}, \text{ where } \mathbf{v}_1 \text{ is the representation of the empty ideal } \varepsilon \text{ and } \mathbf{v}_s \text{ is the representation of } t.$$

Remembering the result by Valiant (1975) on the transitive closure of matrices, Theorem 6.1 follows.

If the operation $\diamond$ is associative, which corresponds to the case of regular languages, a different algorithm gives a stronger result. In fact, if G is a regular grammar, represented in right linear form, the set $Q(t)$ defined as above can be expressed in the following recursive form:

$$Q(t) = \textbf{if } |t| = 1 \textbf{ then } \{A \mid A \to t \in \mathbb{P}\}$$
$$\textbf{else} \bigcup_{t = t'a, a \in \Sigma} \{A \mid A \to Ba \in \mathbb{P} \wedge B \in Q(t')\}.$$

In fact, by induction on the length of t:

(a) for $|t| = 1$ the assertion obviously holds;

(b) let $|t| > 1$. Then:

$A \in Q(t)$ iff there are a, B such that $A \to Ba \in \mathbb{P}$, $B \in Q(t')$, $t'a = t$.

Since $|t'| < |t|$, by induction hypothesis we have that:

$$A \in Q(t) \qquad \text{iff there are } a, B, y \text{ such that:}$$
$$A \to Ba, \qquad B \Rightarrow^* y, \qquad [y]_C = t', \qquad t'a = t.$$

Since $A \Rightarrow^* ya$ and $[ya]_C = [y]_C a = t$ hold, we can conclude:

$$A \in Q(t) \qquad \text{iff} \quad \exists y (A \Rightarrow^* y \wedge [y]_C = t).$$

Then, our problem is reduced to efficiently computing such $Q(t)$. Given the grammar G, let us define the functions $\lambda: \Sigma \to 2^{V_N}$ and $\delta: \Sigma \times 2^{V_N} \to 2^{V_N}$ by:

$$\lambda(\sigma) = \{A \mid A \to \sigma \in \mathbb{P}\}$$
$$\delta(\sigma, \mathcal{A}) = \{A \mid A \to B\sigma \in \mathbb{P}, B \in \mathcal{A}\} \qquad (\sigma \in \Sigma, \mathcal{A} \subseteq V_N).$$

Now, we can easily modify Algorithm 2 in Section 3 so as to compute, for every ideal p of $\text{ord}([x_1 \cdots x_n]_C$, not only its representation $\mathbf{v}$, but also the set $\text{MAX}(\mathbf{v})$ of its maximal elements. For $\langle x_k, k \rangle \in \text{MAX}(\mathbf{v})$, let $\mathbf{v} \backslash x_k$ denote the representation of the ideal obtained from p by erasing the element $\langle x_k, k \rangle$. Our algorithm works as follows:

```
for $\mathbf{v} \in \mathbb{P}^{(1)}$ do if $\mathbf{v}$ represents $\{\langle x_k, k \rangle\}$ and $x_k = \sigma$ then $X(\mathbf{v}) := \lambda(\sigma)$;
for $2 \leqslant k \leqslant n$ do
  for $\mathbf{v} \in \mathbb{P}^{(k)}$ do
    begin
      $S := \text{MAX}(\mathbf{v})$;
      $X(\mathbf{v}) := \bigcup \{\delta(x_k, \mathbf{v} \backslash x_k) \mid \langle x_k, k \rangle \in S\}$
    end
```

For $k = n$, the only element of $\mathbb{P}^{(n)}$ is the representation $\mathbf{w}$ of the whole poset $\text{ord}(t)$, and $X(\mathbf{w}) = Q(t)$. It is easy to prove that the algorithm works in time $O(|x|^\alpha)$, so we can conclude:

THEOREM 6.2. *Given a concurrent alphabet $\langle \Sigma, C \rangle$ and a regular trace language T, the membership problem for T can be solved in time $O(|x|^{\alpha})$.*

7. THE UNIFORM MEMBERSHIP PROBLEM FOR TRACE LANGUAGES

For the membership problem, the concurrent alphabet and the language are given a priori, and the instance of the problem consists only of the word we want to test for its membership to the language. In the case where even the alphabet and the language are given as variable parameters of the problem, we have the uniform memebership problem for trace languages:

PROBLEM (UMPTL).

Instance. A concurrent alphabet $\langle \Sigma, C \rangle$, a grammar G and word $x \in \Sigma^*$.
Question. $[x]_C \in T_G$?

This problem is a difficult one, both for context-free and regular trace languages, as stated by the following:

THEOREM 7.1. *The UMP for context free and regular trace languages is NP-complete.*

Proof. First of all, let us show that the UMPTL belongs to the class NP (Garey and Johnson, 1979). In fact, we have

$$[x]_C \in [L_G]_C = T_G \qquad \text{iff} \quad \exists z([z]_C = [x]_C \text{ and } z \in L_G).$$

Hence, we can proceed as follows. First, we can nondeterministically select, in polynomial time, a string y belonging to the class $[x]_C$. Then, a test is carried out to verify whether $y \in L_G$; but this amounts to solving an instance of the MP for context free (regular) languages in the usual sense, a polynomial time task.

We have now to prove that the UMP for context free (regular) trace languages is NP-hard; the proof is obtained by showing that a well-known NP-complete problem, the Hamiltonian path problem, can be polynomially reduced to it. The Hamiltonian path problem is defined as follows:

PROBLEM (HPP).

Instance. A graph $G = \langle V, E \rangle$.
Question. Is there a Hamiltonian path in G?

Now, given an instance $\langle \{v_1, v_2, ..., v_n\}, E \rangle = G$ of the HPP, we can (polynomially) construct a regular grammar R_G for the set of all the paths in G. Furthermore, we define the concurrency relation on V by $C = V \times V - I$ (I the identity relation). It is easy to see that:

$$G \text{ has a Hamiltonian path iff } [v_1 v_2 \cdots v_n]_C \in [L(R_G)]_C,$$

so completing the reduction.

RECEIVED January 23, 1983; ACCEPTED May 5, 1988

REFERENCES

AALBERSBERG, IJ., AND ROZENBERG, G. (1986), Theory of Traces, *Theoret. Comput. Sci.* **60**, 1–83.

AIGNER, M. (1979), "Combinatorial Theory," Springer-Verlag, Berlin.

BERTONI, A., BRAMBILLA, M., MAURI, G., AND SABADINI, N. (1981), An application of the theory of partially commutative monoids: Asymptotic densities of trace languages, *in* "Lecture Notes in Comput. Sci." Vol. 118 (J. Gruska and M. Chytil, Eds.), pp. 205–215, Springer-Verlag, Berlin.

BERTONI, A., MAURI, G., AND SABADINI, N. (1982a), A hierarchy of regular trace languages and some combinatorial applications, *in* "Proceedings, Second World Conference on Mathematics at the Service of Men" (A. Ballester, D. Cardus, and E. Trillos, Eds.), Las Palmas, pp. 146–153.

BERTONI, A., MAURI, G., AND SABADINI, N. (1982b), Equivalence and membership problems for regular trace languages, *in* "Proceedings, 9th ICALP," Lecture Notes in Comput. Sci. Vol. 140 (M. Nielsen and E. M. Schmidt, Eds.), pp. 61–71, Springer-Verlag, Berlin.

CARTIER, P., AND FOATA, D. (1969), Problèmes combinatoires de commutation et réarrangements, *in* "Lecture Notes in Math. Vol. 85," Springer-Verlag, Berlin.

DEGANO, P., AND MONTANARI, U. (1985), Distributed systems, partial orderings of events and event structures, *in* "Control Flow and Data Flow: Concepts of Distributed Programming" (M. Broy, Ed.), NATO ASI Series F, Vol. 14, pp. 7–106, Springer-Verlag, Heidelberg.

DILWORTH, R. P. (1950), A decomposition theorem for partially ordered sets, *Ann. of Math.* **51**, 161–166.

EILENBERG, S. (1974), "Automata, Languages and Machines," Vol. A, Academic Press, New York.

GAREY, M., AND JOHNSON, D. J. (1979), "Computers and Intractability," Freeman, San Francisco.

GISCHER, J. L. (1984), "Partial Orders and the Axiomatic Theory of Shuffle," Rep. STAN-CS-84-1033, Department of Computer Science, Stanford University.

HOPCROFT, J. E., AND ULLMAN, J. D. (1969), "Formal Languages and Their Relations to Automata," Addison–Wesley, Reading, MA.

KNUTH, E. (1978), "Petri nets and regular trace languages," Comput. Lab., ASM/47, University of Newcastle-upon-Tyne.

JANICKI, R. (1978), Synthesis of concurrent schemes, *in* "Lecture Notes in Comput. Sci. Vol. 64," pp. 298–307, Springer-Verlag, Berlin.

LALLEMENT, G. (1979), "Semigroups and Combinatorial applications," Wiley, New York.

MAZURKIEWICZ, A. (1977), "Concurrent Program Schemes and Their Interpretations," DAIMI, PB 78, Aarhus University.

MAZURKIEWICZ, A. (1985), Semantics of concurrent systems: A modular fixed point trace approach, *in* "Lecture Notes in Comput. Sci." Vol. 188, pp. 353–375, Springer-Verlag, Berlin.

NIELSEN, M., PLOTKIN, G. D., WINSKEL G. (1981), Petri nets, event structures and domains, Part 1, *Theoret. Comput. Sci.* **13**, 85–108.

OCHMANSKI, E. (1985), Regular behaviour of concurrent systems, *EATCS Bull.* **27**, 56–67.

PARIKH, R. J. (1966), On context free languages, *J. Assoc. Comput. Mach.* **13**, 570–581.

PETRI, C. A. (1977), "Nonsequential Processes," ISF Rep. 77/01, GMD, Bonn.

PRATT, V. (1986), Modeling concurrency with partial orders, *Int. J. Parallel Programming* **15**, No. 1, 33–71.

RYTTER, W. (1984), Some properties of trace languages, *Fund. Inform.* **7**, 117–127.

SZIJARTO, M. (1981), Trace languages and closure operations, *Fund. Inform.* **4**, 531–549.

VALIANT, L. (1975), General context free recognition in less than cubic time, *J. Comput. System Sci.* **10**, 308–315.

WINSKEL, G. (1986), Event structures, *in* "Proceedings Advanced Course on Petri nets," Lecture Notes in Comput. Sci. Vol. 255, Springer-Verlag, Berlin.

ZIELONKA, W. (1987), Notes an asynchronous automata, *RAIRO* **21**, 99–135.