INFORMATIQUE THÉORIQUE ET APPLICATIONS

IJSBRAND JAN AALBERSBERG
EMO WELZL

Trace languages defined by regular string languages

Informatique théorique et applications, tome 20, n° 2 (1986), p. 103-119.

<http://www.numdam.org/item?id=ITA_1986__20_2_103_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 20, n° 2, 1986, p. 103 à 119)

TRACE LANGUAGES DEFINED BY REGULAR STRING LANGUAGES (*)

by IJsbrand Jan AALBERSBERG ([1]) and Emo WELZL ([2])

Abstract. – *A* concurrent alphabet *is a pair* $\mathscr{C} = \langle \Sigma, C \rangle$, *where* Σ *is an alphabet and* C *is a relation over* Σ, *called the* concurrency relation. *Two words over* Σ *are called* C-equivalent, *if they can be obtained from each other by successively interchanging adjacent symbols which are related by C. A* trace (over $\mathscr{C}$) *is now simply an equivalence class with respect to C-equivalence.*

This paper considers trace languages (*i. e., sets of traces*) *as they are defined by regular string languages in the following ways:* (i) existentially regular trace languages (*the trace language defined existentially by a regular string language L consists of all traces which have a representative in L*), (ii) universally regular trace languages (*the trace language defined universally by a regular string language L consists of all traces which have all representatives in L*), *and* (iii) consistently regular trace languages (*a regular string language L defines a consistently regular trace language T if and only if L is the union of all the traces in T*).

In particular, the main result of this paper characterizes those concurrent alphabets for which the family of existentially regular trace languages equals the family of universally regular trace languages. Furthermore, using this result, a number of decidability results and characterizations of closure properties for the three above mentioned families of trace languages are derived.

Résumé. – *Un* alphabet concurrent *est un couple* $\mathscr{C} = \langle \Sigma, C \rangle$, *où* Σ *est un alphabet et C est une relation sur* Σ, *appelée* relation de concurrence. *Deux mots sur l'alphabet* Σ *sont dits* C-équivalents, *s'ils peuvent se déduire l'un de l'autre par interversions successives de symboles adjacents en relation par C. Une* trace (sur $\mathscr{C}$) *est une classe d'équivalence par rapport à la C-équivalence.*

Dans cet article, on considère des langages de trace comme étant des langages réguliers définis de diverses façons : (i) langages de trace réguliers existentiels (*le langage de trace défini « existentiellement » par un langage régulier L est constitué de toutes les traces ayant un représentant dans L*), (ii) langages de trace réguliers universels (*le langage de trace défini « universellement » par un langage régulier est constitué de toutes les traces qui ont tous leurs représentants dans L*), (iii) langages de trace réguliers consistent (*un langage régulier L définit un langage de trace régulier et consistent T si et seulement si L est l'union de toutes les traces dans T*).

En particulier, le résultat principal de cet article caractérise les alphabets concurrents pour lesquels la famille des langages de trace réguliers existentiels est égal à la famille des langages de trace réguliers universels. De plus, en utilisant ce résultat, on obtient divers résultats de décidabilité et des caractérisations des propriétés de clôtures relatives aux trois familles mentionnées ci-dessus.

(*) Received in October 1984, revised in June 1985.

([1]) Institute of Applied Mathematics and Computer Science, University of Leiden, Wassenaarseweg 80, Leiden, The Netherlands.

([2]) Institute of Applied Mathematics and Computer Science, University of Leiden, Wassenaarseweg 80, Leiden, The Netherlands.

On leave from: Institutes for Information Processing, IIG, Technical University of Graz and Austrian Computer Society, Schiessstattgasse 4a, A-8010 Graz, Austria.

0296-1598/ 86/02 103 17/$3.70/

0. INTRODUCTION

The theory of *traces* has been introduced in [12] and became quite popular as an approach to the theory of concurrent events, *see*, e. g., [1, 2, 3, 13, 14, 18]. In this approach strings (corresponding to observations by sequential observers) are divided into equivalence classes according to an equivalence relation which is induced by a *concurrency relation* (called also *independence relation*) describing concurrency of events within a system.

More precisely, this can be described as follows. A *concurrent alphabet* is a pair $\mathscr{C} = \langle \Sigma, C \rangle$, where Σ is a finite alphabet (set of events) and C is a symmetric and irreflexive relation over Σ, called the *concurrency relation.* Two words over Σ (sequences of events from Σ) are called *C-equivalent,* if they can be obtained from each other by successively interchanging adjacent (occurrences of) symbols which are related by C. A *trace* (*over* $\mathscr{C}$) is now simply an equivalence class with respect to C-equivalence and a *trace language* (*over* $\mathscr{C}$) is a set of traces (over $\mathscr{C}$).

To specify trace languages over a concurrent alphabet $\mathscr{C} = \langle \Sigma, C \rangle$, one may use string languages (over Σ) in various ways. Let L be a string language over Σ and let T be a trace language over $\mathscr{C}$. (i) T is *existentially* defined by L if T is the set of all traces which have a representative in L. This way of claiming is the one used in [12]. Moreover, if L is regular, then we say that T is an *existentially regular trace language*, *see*, e. g., [12, 18]. (ii) T is *universally* defined by L if T is the set of all traces which have all representatives in L. This way of claiming is introduced in [1]. Moreover, if L is regular, then we say that T is a *universally regular trace language.* (iii) T is defined *consistently by* L if and only if L is the union of all the traces of T. Moreover, if L is regular, then we say that T is a *consistently regular trace language.* Consistently regular trace languages have been investigated in [4], where they are called recognizable trace languages.

It is known that the family of consistently regular trace languages is strictly included in the intersection of existentially and universally regular trace languages, unless the concurrency relation considered is empty (in which case all three classes coincide, *see* [1]). The main goal of this paper is the investigation of the relation between existentially regular and universally regular trace languages over a concurrent alphabet. This question is closely related to the problem over which concurrent alphabets existentially regular trace languages are closed under complement.

The paper is organized as follows. First, in Section 1, we recall some basic definitions from trace theory. In Section 2 we show that the classes of existentially and universally regular trace languages over a concurrent alpha-

bet $\mathscr{C} = \langle \Sigma, C \rangle$ coincide if and only if C is "transitive" (we put transitive in quotation, because we mean a restricted kind of transitivity that still ensures irreflexivity). In Section 3 we apply this result to a number of decidability problems: emptiness, inclusion, equality and emptiness of intersection. Moreover, we characterize basic closure properties of trace languages (union, intersection, complement, concatenation and Kleene star) for all three types of regular trace languages. In particular, we show that existentially regular trace languages over a concurrent alphabet $\mathscr{C} = \langle \Sigma, C \rangle$ are closed under complement if and only if C is "transitive". Finally, in Section 4, we discuss briefly the interpretations of our results in terms of "free partially commutative" monoids.

Remark: The main result of this paper (Theorem 2.7) was obtained independently in [15]. A referee pointed out that the "if-part" of the main result has also been stated in [6], however, without an explicit proof.

A first version of this paper contained also a proof of Proposition 3.5. However, a referee indicated that it has already been proved in [8] and in [7]; another referee pointed out that it has also been stated in [19], however, without an explicit proof. □

1. PRELIMINARIES AND DEFINITIONS

We assume the reader to be familiar with basic formal string language theory (*see*, e. g., [10, 17]). We mostly use standard notation and terminology; perhaps only the following points require some additional attention.

For sets A and B, $A - B$ denotes their difference; $\varnothing$ denotes the empty set and for a set A, 2^A denotes the set of all subsets of A.

We use the the notation of a finite automaton (consistent with [10]) as a 5-tuple $A = (Q, \Sigma, \delta, q, F)$, where Q is the set of states, Σ is the input alphabet, δ is the transition function (from $Q \times \Sigma$ to 2^Q if A is nondeterministic and from $Q \times \Sigma$ into Q if A is complete and deterministic), q is the initial state and F is the set of final states.

The shuffle of two string languages K and L is denoted by $K \| L$.

Finally, λ denotes the empty word.

We need the following notions from the theory of "traces".

A *concurrent alphabet* is a pair $\mathscr{C} = \langle \Sigma, C \rangle$, where Σ is a finite nonempty alphabet and C is a symmetric and irreflexive relation over Σ, called *concurrency relation.* If two symbols a and b are related by a concurrency relation C, then we say that a and b are *concurrent in* C. Since C is symmetric and

irreflexive, we may (and will) consider C as a set of two-element subsets of Σ. We call C *complete* if, for every $a, b \in \Sigma$ with $a \neq b$, $\{a, b\} \in C$ and we call C *transitive* if, for every $a, b, c \in \Sigma$, ($\{a, b\} \in C$, $\{b, c\} \in C$ and $a \neq c$) implies that $\{a, c\} \in C$.

For the following definitions, let $\mathscr{C} = \langle \Sigma, C \rangle$ be an arbitrary (but fixed) concurrent alphabet.

Let $v, w \in \Sigma^*$. We write $v =_C w$ if there exist words $x, y \in \Sigma^*$ and symbols $a, b \in \Sigma$, such that $\{a, b\} \in C$, $v = xaby$ and $w = xbay$. The least equivalence relation containing $=_C$ is denoted by $\equiv_C$ (hence $\equiv_C$ is the transitive and reflexive closure of $=_C$). If $v \equiv_C w$, then we say that v and w are *C-equivalent.* A *trace* (*over* $\mathscr{C}$) is an equivalence class of $\equiv_C$. The set of all traces over $\mathscr{C}$ is denoted by $T^{\mathscr{C}}$. A *trace language* (*over* $\mathscr{C}$) is a set of traces (over $\mathscr{C}$) - hence a subset of $T^{\mathscr{C}}$.

For a word w over Σ, $[w]^C$ denotes the trace containing w. For x and y in Σ^*, the *trace-concatenation* of $[x]^C$ and $[y]^C$, denoted by $[x]^C \circ [y]^C$, is the trace $[xy]^C$. (Note that $[x]^C \circ [y]^C$ does not equal the string-concatenation:

$$[x]^C [y]^C = \{ x' y' \mid x' \in [x]^C \text{and} y' \in [y]^C \}.)$$

Let T and T' be trace languages over $\mathscr{C}$. The *trace-concatenation* of T and T', denoted by $T \circ T'$, is the trace language $\{ t \circ t' \mid t \in T \text{ and } t' \in T' \}$. The *Kleene trace star* of T, denoted by $T^{(*)}$, is the least trace language containing $T \cup \{[\lambda]^C\}$ which is closed under trace-concatenation.

Let L be a string language over Σ. The trace language (over $\mathscr{C}$) *existentially* defined by L, denoted by $[L]^C_\exists$, is the set $\{ t \in T^{\mathscr{C}} \mid t \cap L \neq \varnothing \}$. The trace language (over $\mathscr{C}$) *universally* defined by L, denoted by $[L]^C_\forall$, is the set $\{ t \in T^{\mathscr{C}} \mid t \subseteq L \}$. Furthermore, if $[L]^C_\exists = [L]^C_\forall = T$, then we call T the trace language (over $\mathscr{C}$) *consistently* defined by L, denoted by $[L]^C$. Clearly, in this case $L = \bigcup_{t \in T} t$. (Note that, if $[L]^C_\exists \neq [L]^C_\forall$, then $[L]^C$ is undefined.)

Finally, if T is a trace language existentially (universally, consistently) defined by a regular string language, then we call T an *existentially* (*universally*, *consistently*, respectively) *regular trace language.* $\mathscr{T}^{\mathscr{C}}_\exists(\text{REG})$ ($\mathscr{T}^{\mathscr{C}}_\forall(\text{REG})$, $\mathscr{T}^{\mathscr{C}}(\text{REG})$) will denote the class of all existentially (universally, consistently, respectively) regular trace languages over $\mathscr{C}$.

The following observations, which are crucial throughout the paper, are easy to prove.

OBSERVATION 1.1: Let $\mathscr{C} = \langle \Sigma, C \rangle$ be a concurrent alphabet and let L_1 and L_2 be string languages over Σ.

Then:

$$[L_1]^C_\exists \cup [L_2]^C_\exists = [L_1 \cup L_2]^C_\exists, \tag{1}$$

$$[L_1]^C_\forall \cap [L_2]^C_\forall = [L_1 \cap L_2]^C_\forall, \tag{2}$$

$$T^{\mathscr{C}} - [L_1]^C_\exists = [\Sigma^* - L_1]^C_\forall, \tag{3a}$$

$$\bar{T}^{\mathscr{C}} - [L_1]^C_\forall = [\bar{\Sigma}^* - L_1]^C_\exists, \tag{3b}$$

$$[L_1]^C_\forall - [L_2]^C_\exists = [L_1 - L_2]^C_\forall, \tag{3c}$$

$$[L_1]^C_\exists \circ [L_2]^C_\exists = [L_1 L_2]^C_\exists, \tag{4}$$

and

$$([L_1]^C_\exists)^{(*)} = [L_1^*]^C_\exists. \tag{5}$$

If both $[L_1]^C$ and $[L_2]^C$ are defined, then:

$$[L_1]^C \cup [L_2]^C = [L_1 \cup L_2]^C, \tag{6}$$

$$[L_1]^C \cap [L_2]^C = [L_1 \cap L_2]^C, \tag{7}$$

and

$$[L_1]^C - [L_2]^C = [L_1 - L_2]^C. \quad \square \tag{8}$$

Remark: Whenever we consider in this paper a decision problem and a regular string language, then it is implicitly assumed that the language is specified by a finite automaton. $\square$

2. MAIN RESULT

If, for a concurrent alphabet $\mathscr{C}$, $\mathscr{T}^{\mathscr{C}}_\exists(\mathrm{REG}) = \mathscr{T}^{\mathscr{C}}_\forall(\mathrm{REG})$, then we write briefly that $\mathscr{C}$ is *of type* $\exists = \forall$.

In this section we characterize those concurrent alphabets which are of type $\exists = \forall$ (Theorem 2.7).

We start with a lemma which shows that there are concurrent alphabets which are not of type $\exists = \forall$.

LEMMA 2.1: *Let* $\mathscr{C} = \langle \Sigma, C \rangle$ *be a concurrent alphabet, where* $\Sigma = \{a, b, c\}$ *and* $C = \{\{a, b\}, \{b, c\}\}$, *and let* $L_0 = L_1 \cup L_2 \cup L_3$, *where* $L_1 = (ab + c)^*$, $L_2 = b(a + b + cb)^*$ *and* $L_3 = (a + c + cb)^* c (a + c + cb)^*$. *Then* $[L_0]^C_\exists \notin \mathscr{T}^{\mathscr{C}}_\forall(\mathrm{REG})$.

Proof: Intuitively, the given C means that the a's and c's are "rigid", while the b's can move freely through the a's and c's.

For every $w \in \Sigma^*$, (i) $[w]^C \in [L_1]_\exists^C$ if and only if $\#_a(w) = \#_b(w)$, (ii) $[w]^C \in [L_2]_\exists^C$ if and only if $\#_b(w) > \#_c(w)$, and (iii) $[w]^C \in [L_3]_\exists^C$ if and only if $\#_b(w) < \#_c(w)$. These three facts together mean that for every $w \in \Sigma^*$, $[w]^C \in [L_0]_\exists^C$ if and only if $\#_a(w) = \#_b(w)$ or $\#_b(w) \neq \#_c(w)$. Consequently, $[w]^C \in T^{\mathscr{C}} - [L_0]_\exists^C$ if and only if $\#_a(w) \neq \#_b(w)$ and $\#_b(w) = \#_c(w)$ (which entails that $\#_a(w) \neq \#_c(w)$).

Assume now that there exists a regular string language K over Σ, such that $[L_0]_\exists^C = [K]_\forall^C$. Hence, Observation 1.1.(3*b*) implies that for the string language $M = \Sigma^* - K$, we have $T^{\mathscr{C}} - [L_0]_\exists^C = T^{\mathscr{C}} - [K]_\forall^C = [M]_\exists^C$. Let h be the homomorphism from Σ^* to $\{a, c\}^*$, defined by: $h(a) = a$, $h(b) = \lambda$ and $h(c) = c$. Then it is easily seen that $h(M) = \{ w \in \{a, c\}^* \mid \#_a(w) \neq \#_c(w) \}$, which is not a regular string language. Thus, M is not a regular string language, which contradicts the fact that $K = \Sigma^* \quad M$ is a regular string language. Consequently, there exists no regular string language K over Σ such that $[L_0]_\exists^C = [K]_\forall^C$, which proves the lemma. □

From Lemma 2.1 it easily follows that a concurrent alphabet $\mathscr{C} = \langle \Sigma, C \rangle$ is not of type $\exists = \forall$, when C is not transitive.

The main theorem (Theorem 2.7) is now proved by first showing that a concurrent alphabet is of type $\exists = \forall$, when the concurrency relation considered is complete. Then we show that the disjoint union of two concurrent alphabets of type $\exists = \forall$ is again of type $\exists = \forall$. This actually shows that a transitive concurrency relation gives rise to a concurrent alphabet of type $\exists = \forall$.

We continue with a lemma which shows that whenever we want to prove that a concurrent alphabet $\mathscr{C}$ is of type $\exists = \forall$, then it is sufficient to prove that $\mathscr{T}_\exists^{\mathscr{C}}(\mathrm{REG}) \subseteq \mathscr{T}_\forall^{\mathscr{C}}(\mathrm{REG})$ (this fact will be used implicitly in the forthcoming proofs). Moreover, this lemma gives also evidence of the close relationship between the question whether a concurrent alphabet is of type $\exists = \forall$ on the one side and closure properties of $\mathscr{T}_\exists^{\mathscr{C}}(\mathrm{REG})$ and $\mathscr{T}_\forall^{\mathscr{C}}(\mathrm{REG})$ on the other side (this will be treated more extensively in Section 3).

LEMMA 2.2: *For a concurrent alphabet $\mathscr{C} = \langle \Sigma, C \rangle$, the following three statements are equivalent.*

(1) *$\mathscr{C}$ is of type $\exists = \forall$.*

(2) $\mathscr{T}_\exists^{\mathscr{C}}(\mathrm{REG}) \subseteq \mathscr{T}_\forall^{\mathscr{C}}(\mathrm{REG})$.

(3) $\mathscr{T}_\exists^{\mathscr{C}}(\mathrm{REG})$ *is closed under complement.*

Proof: Trivially (2) follows from (1).

Observe that $\mathscr{T}_\forall^{\mathscr{C}}(\mathrm{REG}) = \{ T^{\mathscr{C}} - T \mid T \in \mathscr{T}_\exists^{\mathscr{C}}(\mathrm{REG}) \}$, i.e., $\mathscr{T}_\exists^{\mathscr{C}}(\mathrm{REG})$ and $\mathscr{T}_\forall^{\mathscr{C}}(\mathrm{REG})$ are "dual" in the sense that one family contains exactly the complements of the other family and vice versa. Hence it follows from (2) that $\mathscr{T}_\forall^{\mathscr{C}}(\mathrm{REG})$ is closed under complement. This implies, also by the above

duality, that $\mathscr{T}_{\exists}^{\mathscr{C}}(\mathrm{REG})$ is closed under complement. Consequently (3) follows from (2).

Moreover, the above duality shows that (1) follows from (3). □

In the following lemma we will see that the fact that a concurrent alphabet $\mathscr{C}=\langle \Sigma, C\rangle$ with complete C is of type $\exists=\forall$ can be easily derived from the fact that semilinear sets are closed under complement.

LEMMA 2.3: *A concurrent alphabet $\mathscr{C}=\langle \Sigma, C\rangle$, where C is complete, is of type $\exists=\forall$.*

Proof: Let L be a regular string language over Σ. Then the Parikh image $\psi(L)$ of L is a semilinear set (*see* [9] for the definitions of the Parikh mapping ψ and of semilinear sets). The complement of a semilinear set is again a semilinear set, i.e., $\psi(\Sigma^*)-\psi(L)$ is a semilinear set (which can be effectively given from $\psi(L)$, *see* Theorem 5.6.2 in [9]). Since there is a regular string language M with $\psi(M)=\psi(\Sigma^*)-\psi(L)$, the regular string language $K=\Sigma^*-M$ is such that $\psi(\Sigma^*-K)=\psi(\Sigma^*)-\psi(L)$.

We claim now that $[L]_{\exists}^{C}=[K]_{\forall}^{C}$, which can be seen as follows.

From the fact that C is complete, it follows that, for every x and y in Σ^*, $\psi(x)=\psi(y)$ if and only if $x\equiv_C y$. Hence, for every x in Σ^*, $[x]^C\in[L]_{\exists}^{C}$ if and only if $\psi(x)\in\psi(L)$ if and only if $\psi(x)\notin\psi(\Sigma^*)-\psi(L)$ if and only if $\psi(x)\notin\psi(\Sigma^*-K)$ if and only if $[x]^C\in[K]_{\forall}^{C}$.

Thus, indeed $[L]_{\exists}^{C}=[K]_{\forall}^{C}$ and consequently, $[L]_{\exists}^{C}\in\mathscr{T}_{\forall}^{\mathscr{C}}(\mathrm{REG})$ – this yields the lemma (recall Lemma 2.2). □

The following two lemmas prepare now the proof of the fact that the disjoint union of two concurrent alphabets of type $\exists=\forall$ is again a concurrent alphabet of type $\exists=\forall$.

LEMMA 2.4: *Let $\mathscr{C}=\langle \Sigma, C\rangle$ be a concurrent alphabet of type $\exists=\forall$ and let $L_1, \ldots, L_n$ be regular string languages over Σ $(n\geqq 0)$. Then there exist regular string languages $K_1, \ldots, K_n$ over Σ, such that:*

$$(*)\qquad \Big[\bigcup_{i\in I} L_i\Big]_{\exists}^{C}=\Big[\bigcup_{i\in I} K_i\Big]_{\forall}^{C}, \quad \textit{for every } I\subseteq\{1, \ldots, n\}.$$

Proof: Since the finite union of regular string languages is again a regular string language, there exists, for every $I\subseteq\{1, \ldots, n\}$, a regular string language K_I', such that $\big[\bigcup_{i\in I} L_i\big]_{\exists}^{C}=[K_I']_{\forall}^{C}$. Let, for every $1\leqq i\leqq n$, $K_i=\bigcap_{I\ni i} K_I'$. We claim that, for every $I\subseteq\{1, \ldots, n\}$, $\big[\bigcup_{i\in I} L_i\big]_{\exists}^{C}=\big[\bigcup_{i\in I} K_i\big]_{\forall}^{C}$, which can be seen as follows.

Clearly, for every $1 \leqq i \leqq n$, $[L_i]^C_\exists = [K_i]^C_\forall$, because (i) $K_i \subseteq K'_{\{i\}}$ implies that $[K_i]^C_\forall \subseteq [K'_{\{i\}}]^C_\forall = [L_i]^C_\exists$, and (ii) if, for x in Σ^*, $[x]^C \in [L_i]^C_\exists$, then, for every I with $i \in I$, $[x]^C \subseteq K'_I$ and hence $[x]^C \subseteq K_i$, which implies $[x]^C \in [K_i]^C_\forall$.

Let $I \subseteq \{1, \ldots, n\}$. From above it directly follows that:

$$[\bigcup_{i \in I} L_i]^C_\exists \subseteq [\bigcup_{i \in I} K_i]^C_\forall.$$

To show the reverse inclusion, let x in Σ^* be such that:

$$[x]^C \in [\bigcup_{i \in I} K_i]^C_\forall.$$

By definition:

$$\bigcup_{i \in I} K_i = \bigcup_{i \in I} (K_i \cap K'_I) = (\bigcup_{i \in I} K_i) \cap K'_I,$$

and so:

$$[x]^C \in [K'_I]^C_\forall = [\bigcup_{i \in I} L_i]^C_\exists.$$

The lemma follows from the two inclusions. □

If string languages $K_1, \ldots, K_n$ satisfy condition (*) formulated in Lemma 2.4 for a concurrent alphabet $\mathscr{C} = \langle \Sigma, C \rangle$ and string languages $L_1, \ldots, L_n$ over Σ ($n \geqq 0$), then we say that $(K_1, \ldots, K_n)$ *universally* $\mathscr{C}$*-represents* $(L_1, \ldots, L_n)$.

LEMMA 2.5: *Let L be a regular string language over an alphabet Σ and let (Σ_1, Σ_2) be a partition of Σ (i.e., $\Sigma_1 \cup \Sigma_2 = \Sigma$ and $\Sigma_1 \cap \Sigma_2 = \varnothing$). Then there exists a regular string language K over an alphabet Δ and a regular substitution η from Δ^* to 2^{Σ^*}, such that:*

(1) $\eta(K) = L$, and

(2) there exists a partition (Δ_1, Δ_2) of Δ, such that:

(2*a*) $\eta(\Delta_1) \subseteq \Sigma_1^+$ and $\eta(\Delta_2) \subseteq \Sigma_2^+$, and

(2*b*) $K \cap \Delta^* \Delta_1 \Delta_1 \Delta^* = \varnothing$ and $K \cap \Delta^* \Delta_2 \Delta_2 \Delta^* = \varnothing$ (i.e., two adjacent symbols of a word in K do not belong to the same part of the partition of Δ).

Proof: The proof uses a standard construction from automata theory.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a complete deterministic finite automaton with $L(A) = L$. Let $\Delta_1 = Q \times Q \times \{1\}$, let $\Delta_2 = Q \times Q \times \{2\}$ and let $\Delta = \Delta_1 \cup \Delta_2$. Let $K' = \{(s_1, s_2, i_1)\ (s_2, s_3, i_2) \ldots (s_n, s_{n+1}, i_n) \in \Delta^+ \mid n \geqq 1,\ s_1 = q_0,\ s_{n+1} \in F,$

and, for every $1 \leqq j \leqq n-1$, $i_j \neq i_{j+1}\}$. If $\lambda \in L$, then we set $K = K' \cup \{\lambda\}$, otherwise we set $K = K'$. We define the substitution η from Δ^* to 2^{Σ^*} by: for every s and s' in Q and every i in $\{1, 2\}$, $\eta((s, s', i)) = \{ w \in \Sigma_i^+ \mid \delta(s, w) = s' \}$.

Obviously, K is a regular string language and η is a regular substitution. Moreover, conditions $(2a)$ and $(2b)$ hold for K and η. Thus it is left to show that $\eta(K) = L$.

The inclusion $\eta(K) \subseteq L$ can be seen by a standard argument. To show the reverse inclusion, let x in L. If $x = \lambda$, then $x \in \eta(K)$. If $x \neq \lambda$, then we can write x in the from $x = w_1 \ldots . w_n$, where $n \geqq 1$, and, for every $1 \leqq j \leqq n$, $w_j \in \Sigma_{i_j}$, for some $i_j \in \{1, 2\}$, such that, for every $1 \leqq j \leqq n-1$, $i_j \neq i_{j+1}$. Let $s_0 = q_0$ and let, for every $1 \leqq j \leqq n$, $s_j = \delta(q_0, w_1 \ldots . w_j)$. Clearly, $s_n \in F$ and, for every $1 \leqq j \leqq n$:

$$\delta(s_{j-1}, w_j) = s_j \text{ (i. e., } w_j \in \eta((s_{j-1}, s_j, i_j))),$$

which implies that $x \in \eta(y)$, for:

$$y = (s_0, s_1, i_1)(s_1, s_2, i_2) \ldots (s_{n-1}, s_n, i_n) \in K. \quad \square$$

LEMMA 2.6: *Let $\mathscr{C}_1 = \langle \Sigma_1, C_1 \rangle$ and $\mathscr{C}_2 = \langle \Sigma_2, C_2 \rangle$ be two concurrent alphabets of type $\exists = \forall$, where Σ_1 and Σ_2 are disjoint. Then the concurrent alphabet $\mathscr{C} = \langle \Sigma_1 \cup \Sigma_2, C_1 \cup C_2 \rangle$ is of type $\exists = \forall$.*

Proof: Let $\Sigma = \Sigma_1 \cup \Sigma_2$, $C = C_1 \cup C_2$, and let L be a regular string language over Σ. For the partition (Σ_1, Σ_2) of Σ we can find a regular string language K over an alphabet Δ, a partition (Δ_1, Δ_2) of Δ and a regular substitution η from Δ^* to 2^{Σ^*} as described in Lemma 2.5. Let $\Delta_1 = \{a_1, \ldots, a_n\}$ and $\Delta_2 = \{b_1, \ldots, b_m\}$, where $m, n \geqq 0$.

Consider now a regular substitution ρ from Δ^* to 2^{Σ^*}, such that $(\rho(a_1), \ldots, \rho(a_n))$ universally $\mathscr{C}_1$-represents $(\eta(a_1), \ldots, \eta(a_n))$ and $(\rho(b_1), \ldots, \rho(b_m))$ universally $\mathscr{C}_2$-represents $(\eta(b_1), \ldots, \eta(b_m))$. The existence of such a substitution follows from Lemma 2.4. It is easily seen that $(\rho(a_1), \ldots, \rho(a_n))$ universally $\mathscr{C}$-represents $(\eta(a_1), \ldots, \eta(a_n))$ and that $(\rho(b_1), \ldots, \rho(b_m))$ universally $\mathscr{C}$-represents $(\eta(b_1), \ldots, \eta(b_m))$.

We claim that $[L]_\exists^C = [\rho(K)]_\forall^C$. This can be seen as follows.

First we show that $[L]_\exists^C \subseteq [\rho(K)]_\forall^C$. Consider a word x in L. Then, for some $d_1 \ldots d_r \in K$, where $d_1, \ldots, d_r \in \Delta$ and $r \geqq 0$, $x \in \eta(d_1) \ldots \eta(d_r)$. Hence, for every $1 \leqq i \leqq r$, there is a word $w_i \in \eta(d_i)$, such that $x = w_1 \ldots w_r$. By condition $(2a)$ from the statement of Lemma 2.5 imposed on η and by the fact that Σ_1 and Σ_2 are disjoint, it follows that every word x' in $[x]^C$ can be written in the form $x' = w_1' \ldots w_r'$, where, for every $1 \leqq i \leqq r$, $w_i' \in [w_i]^C$. Moreover, for

every $1 \leqq i \leqq r$, $[\eta(d_i)]^C_\exists = [\rho(d_i)]^C_\forall$ and so $[w_i]^C \subseteq \rho(d_i)$. This implies that $[x]^C \subseteq \rho(d_1) \ldots \rho(d_r) \subseteq \rho(K)$ and, consequently, $[x]^C \in [\rho(K)]^C_\forall$.

Secondly we prove the reverse inclusion $[\rho(K)]^C_\forall \subseteq [L]^C_\exists$. Consider $[x]^C \in [\rho(K)]^C_\forall$ for some x in Σ^*. Then x can be written as $x = w_1 \ldots w_r$, where $r \geqq 0$ and, for every $1 \leqq i \leqq r$, $w_i \in \Sigma^+_{l_i}$ and $l_i \in \{1, 2\}$, such that, for every $1 \leqq i \leqq r-1$, $l_i \neq l_{i+1}$. Hence, for some $p \geqq 1$, there exist words $y_j = d_{j,1} \ldots d_{j,r} \in K$, where $1 \leqq j \leqq p$ and $d_{j,i} \in \Delta$ for every $1 \leqq j \leqq p$ and every $1 \leqq i \leqq r$, such that $[x]^C \subseteq \bigcup_{i \leqq j \leqq p} \rho(y_j)$ and $\rho(d_{j,i}) \subseteq \Sigma^+_{l_i}$ for every $1 \leqq j \leqq p$ and every $1 \leqq i \leqq r$.

We make the following crucial observation. If, for some $1 \leqq i \leqq r$ and some $J \subseteq \{1, \ldots, p\}$, $[w_i]^C \subseteq \bigcup_{j \in J} \rho(d_{j,i})_i$, then there exists a $j_0 \in J$ such that $[w_i]^C \subseteq \rho(d_{j_0,i})$. This stems simply from the choice of ρ. Otherwise we would have $[w_i]^C \notin [\eta(d_{j,i})]^C_\exists$ for every $j \in J$, i.e., $[w_i]^C \notin [\bigcup_{j \in J} \eta(d_{j,i})]^C_\exists$, while, on the other hand, we would have $[w_i]^C \in [\bigcup_{j \in J} \rho(d_{j,i})]^C_\forall$.

Hence, if we set, for every $1 \leqq i \leqq r$:

$$J_i = \{j \in \{1, \ldots, p\} \mid [w_i]^C \not\subseteq \rho(d_{j,i})\},$$

then $[w_i]^C \not\subseteq \bigcup_{j \in J_i} \rho(d_{j,i})$.

Choose now, for every $1 \leqq i \leqq r$, a word $w'_i \in [w_i]^C$ such that $w'_i \notin \bigcup_{j \in J_i} \rho(d_{j,i})$. Then $w'_1 \ldots w'_r \in [x]^C$ and so $w'_1 \ldots w'_r \in \rho(y_{j_0})$ for some $1 \leqq j_0 \leqq p$. Thus $w'_i \in \rho(d_{j_0,i})$ for every $1 \leqq i \leqq r$ and consequently $j_0 \notin J_i$. This implies that, for every $1 \leqq i \leqq r$, $[w_i]^C \subseteq \rho(d_{j_0,i})$. Consequently, for every $1 \leqq i \leqq r$, there exists a word $w''_i \in [w_i]^C$ such that $w''_i \in \eta(d_{j_0,i})$ and hence $w''_1 \ldots w''_r \in [x]^C \cap \eta(y_{j_0}) \subseteq [x]^C \cap L$. This proves that $[x]^C \in [L]^C_\exists$.

Thus, $[L]^C_\exists = [\rho(K)]^C_\forall$ and consequently $[L]^C_\exists \in \mathscr{T}^{\mathscr{C}}_\forall(\mathrm{REG})$, which immediately proves the lemma (recall Lemma 2.2). □

Now we are ready to prove the main theorem of the paper.

THEOREM 2.7: *A concurrent alphabet $\mathscr{C} = \langle \Sigma, C \rangle$ is of type $\exists = \forall$ if and only if C is transitive.*

Proof: From Lemmas 2.3 and 2.6 it follows that if C is transitive, then $\mathscr{C}$ is of type $\exists = \forall$ (note that if C is transitive, then C is the disjoint union of a finite number of complete concurrency relations).

If C is not transivitive, then there exist (pairwise different) symbols a, b and c in Σ, such that $\{\{a, b\}, \{b, c\}\} \subseteq C$, but $\{a, c\} \notin C$. Using Lemma 2.1, it is easily seen that $\mathscr{C}$ is not of type $\exists = \forall$. □

3. APPLICATIONS: DECIDABILITY AND CLOSURE PROPERTIES

In this section we consider a number of applications of our main result.

First, it turns out that using the result from the previous section we can show that a number of basic decidability questions concerning trace languages (such as emptiness, inclusion, equality and emptiness of intersection) are decidable for both existentially and universally regular trace languages, provided that the considered concurrency relation is transitive. (Note that for consistently regular trace languages all these decidability questions are obviously decidable for every concurrency relation.)

Secondly, we consider closure properties of trace languages with respect to the classical language operations, like union, intersection, complement, trace-concatenation and Kleene trace star. For each of these operations and for all three types of regular trace languages we characterize those concurrency relations which guarantee the closure under the given operations.

We start with a theorem that states that our main result (Theorem 2.7) holds "effectively".

THEOREM 3.1: *Let $\mathscr{C} = \langle \Sigma, C \rangle$ be a concurrent alphabet, where C is transitive. For every regular string language L over Σ, regular string languages K_1 with $[L]_\exists^C = [K_1]_\forall^C$ and K_2 with $[L]_\forall^C = [K_2]_\exists^C$ can be effectively given from L.*

Proof: This can be easily seen, since all constructions in Lemmas 2.3 through 2.6 are effective. □

We move now to decidability results. (Note that the problem (3 *a*) from the statement of the next theorem has been proved already in [5], while the problems (1 *a*) and (2 *a*) from the statement of the next theorem have been proved already in [6]).

THEOREM 3.2: *Let $\mathscr{C} = \langle \Sigma, C \rangle$ be a concurrent alphabet, where C is transitive. For regular string languages L_1 and L_2 over Σ, the following problems are decidable:*

(1 *a*) $[L_1]_\exists^C = \varnothing$?

(1 *b*) $[L_1]_\forall^C = \varnothing$?

(2 *a*) $[L_1]_\exists^C \subseteq [L_2]_\exists^C$?

(2 *b*) $[L_1]_\forall^C \subseteq [L_2]_\forall^C$?

(3 *a*) $[L_1]_\exists^C = [L_2]_\exists^C$?

(3 *b*) $[L_1]_\forall^C = [L_2]_\forall^C$?

(4 *a*) $[L_1]_\exists^C \cap [L_2]_\exists^C = \varnothing$?

and

$$(4\,b) \qquad [L_1]_\forall^{\mathscr{C}} \cap [L_2]_\forall^{\mathscr{C}} = \varnothing\,?$$

Proof: $[L_1]_\exists^{\mathscr{C}} = \varnothing$ holds if and only if $L_1 = \varnothing$, which is decidable and so $(1\,a)$ holds.

By Theorem 3.1 a regular string language K with $[L_1]_\forall^{\mathscr{C}} = [K]_\exists^{\mathscr{C}}$ can be effectively given. This reduces $(1\,b)$ to $(1\,a)$.

$[L_1]_\exists^{\mathscr{C}} \subseteq [L_2]_\exists^{\mathscr{C}}$ holds if and only if $[L_1]_\exists^{\mathscr{C}} - [L_2]_\exists^{\mathscr{C}} = \varnothing$. By Theorem 3.1 a regular string language K with $[L_1]_\exists^{\mathscr{C}} = [K]_\forall^{\mathscr{C}}$ can be effectively given. By Observation 1.1.$(3\,c)$, $[K]_\forall^{\mathscr{C}} - [L_2]_\exists^{\mathscr{C}} = [K - L_2]_\forall^{\mathscr{C}}$ holds, i.e., $[L_1]_\exists^{\mathscr{C}} - [L_2]_\exists^{\mathscr{C}} = \varnothing$ if and only if $[K - L_2]_\forall^{\mathscr{C}} = \varnothing$. Since $K - L_2$ is a regular string language, this reduces $(2\,a)$ to $(1\,b)$.

By Theorem 3.1 $(2\,b)$ follows from $(2\,a)$.

The remaining statements follow now easily from the above and Observation 1.1. □

To put the above results into a better perspective, we mention here that the equality problem (and hence the inclusion problem) becomes undecidable for existentially regular trace languages and the concurrent alphabet:

$$\mathscr{C} = \langle \{a, b, c, d\}, \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\} \rangle,$$

as it has been shown in [5]. Of course, this means also that these problems become undecidable for universally regular trace languages and this concurrent alphabet (*see* also [1]).

We will consider now the closure properties of existentially, universally and consistently regular trace languages. This will be done as follows.

First we state a number of very basic, easily obtainable, closure properties. Secondly we consider four (counter-) examples which will be used to show some negative closure properties. Finally, we settle the (closure under) trace-concatenation problem for consistently regular trace languages. Then the remaining closure properties will follow from Theorem 2.7 and some simple observations.

LEMME 3.3: *Let $\mathscr{C}$ be a concurrent alphabet. Then:*

(1) $\mathscr{T}_\exists^{\mathscr{C}}(\mathrm{REG})$ *is closed under union, trace-concatenation and Kleene trace star,*

(2) $\mathscr{T}_\forall^{\mathscr{C}}(\mathrm{REG})$ *is closed under intersection, and*

(3) $\mathscr{T}^{\mathscr{C}}(\mathrm{REG})$ *is closed under union, intersection and complement.*

Proof: Follows directly from Observation 1.1 and the closure of regular string languages under union, intersection, complement, concatenation and Kleene star. □

In the following two lemmas we consider the concurrent alphabet $\mathscr{D} = \langle \Delta, D \rangle$, where $\Delta = \{a, b, c\}$ and $D = \{\{a, b\}, \{b, c\}\}$. Note that D is the minimal non-transitive concurrency relation; moreover, D appears as a "subpattern" of every non-transitive concurrency relation, i. e., a concurrency relation C is non-transitive if and only if there are letters a, b and c such that $C \cap \{\{a, b\}, \{b, c\}, \{c, a\}\} = D$.

LEMME 3.4: *Let:*

$$L_1 = (((acac(a^2+c^2)^+) \,\|\, b^*) - acac(a^2 b^2 + c^2)^*) + \lambda,$$
$$L_2 = (((acac(a^2+c^2)^+) \,\|\, b^*) - acac(a^2 + c^2 b^2)^*) + \lambda,$$

and

$$L_1' = (((ac(a^2+c^2)^+) \,\|\, b^*) - ac(a^2 b^2 + c^2)^*).$$

Then:

(1) $$[\Delta^* - L_1]_\exists^D \cap [\Delta^* - L_2]_\exists^D \notin \mathscr{T}_\exists^{\mathscr{D}}(\mathrm{REG}),$$

(2) $$[L_1]_\forall^D \cup [L_2]_\forall^D \notin \mathscr{T}_\forall^{\mathscr{D}}(\mathrm{REG}),$$

(3) $$[L_1]_\forall^D \circ [L_2]_\forall^D \notin \mathscr{T}_\forall^{\mathscr{D}}(\mathrm{REG}),$$

and

(4) $$([L_1' \cup L_2 \cup \{ac\}]_\forall^D)(*) \notin \mathscr{T}_\forall^{\mathscr{D}}(\mathrm{REG}).$$

Proof: Throughout the whole proof, let h be the homomorphism from Δ^* to $\{a, c\}^*$, defined by $h(a) = a$, $h(c) = c$ and $h(b) = \lambda$.

(1) Let $w \in \Delta^*$ be such that $h(w) \in acac(a^2+c^2)^+$. Then (i) $[w]^D \in [\Delta^* - L_1]_\exists^D$ if and only if $\sharp_b(w) = \sharp_a(w) - 2$, and (ii) $[w]^D \in [\Delta^* - L_2]_\exists^D$ if and only if $\sharp_b(w) = \sharp_c(w) - 2$. Consider now a string language L such that:

$$[L]_\exists^D = [\Delta^* - L_1]_\exists^D \cap [\Delta^* - L_2]_\exists^D.$$

Then it is easily seen by (i) and (ii) above, that:

$$h(L \cap ((acac(a^2+c^2)^+) \,\|\, b^*)) = \{ w \in acac(a^2+c^2)^+ \mid \sharp_a(w) = \sharp_c(w) \},$$

which is not a regular string language. This implies that L is not a regular string language and proves assertion (1) of the lemma.

(2) A standard set-theoretical argument and Observation 1.1 show that, for a string language L over Δ, $[L]_{\forall}^{D}=[L_1]_{\forall}^{D}\cup[L_2]_{\forall}^{D}$ if and only if:

$$[\Delta^*-L]_{\exists}^{D}=[\Delta^*-L_1]_{\exists}^{D}\cap[\Delta^*-L_2]_{\exists}^{D}.$$

This and assertion (1) yield assertion (2) of the lemma.

(3) Observe that, (i) for $w\in\Delta^*$, $[w]^D\in[L_1]_{\forall}^{D}$ implies ($w=\lambda$ or $h(w)\in acac(a^2+c^2)^+$) and (ii) for $w\in\Delta^*$, $[w]^D\in[L_2]_{\forall}^{D}$ implies ($w=\lambda$ or $h(w)\in acac(a^2+c^2)^+$). Let $t_1\in[L_1]_{\forall}^{D}$, $t_2\in[L_2]_{\forall}^{D}$ and $t\in[(acac(a^2+c^2)^+)\,\|\,b^*]_{\forall}^{D}$. If $t=t_1\circ t_2$, then $t_1=[\lambda]^D$ or $t_2=[\lambda]^D$, which implies that $t\in[L_1]_{\forall}^{D}$ or $t\in[L_2]_{\forall}^{D}$. Consider the trace language $T=[L_1]_{\forall}^{D}\circ[L_2]_{\forall}^{D}$. Now it is easily seen (recall (1) and (2) above) that:

$$T\cap[((acac(a^2+c^2)^+)\,\|\,b^*)+\lambda]_{\forall}^{D}=[L_1]_{\forall}^{D}\cup[L_2]_{\forall}^{D}.$$

By assertion (2) and Lemma 3.3. (2) it follows now directly that $T\notin\mathscr{T}_{\forall}^{\mathscr{D}}(\mathrm{REG})$.

(4) Note that the choice of L_1' and L_2 implies that:

$$[L_1'\cup L_2\cup\{ac\}]_{\forall}^{D}=[L_1']_{\forall}^{D}\cup[L_2]_{\forall}^{D}\cup[\{ac\}]_{\forall}^{D}$$

(which is not the case in general).

Let $T=([L_1'\cup L_2\cup\{ac\}]_{\forall}^{D})^{(*)}$. Since it is easily seen that $[L_1]_{\forall}^{D}-[\{\lambda\}]_{\forall}^{D}=[\{ac\}]_{\forall}^{D}\circ[L_1']_{\forall}^{D}$, we have $[L_1]_{\forall}^{D}\cup[L_2]_{\forall}^{D}\subseteq T$. Let now $u\in\Delta^*$ be such that $h(u)\in acac(a^2+c^2)^+$ and $[u]^D\in T$. Then $[u]^D=[w_1]^D\circ\ldots\circ[w_n]^D$ for some $n\geqq 1$ and $[w_1]^D,\ldots,[w_n]^D\in[L_1']_{\forall}^{D}\cup[L_2-\{\lambda\}]_{\forall}^{D}\cup[\{ac\}]_{\forall}^{D}$. It is easy to see that $h(u)=h(w_1)\ldots h(w_n)$, which implies that $n\leqq 2$ (*otherwise* $h(u)\in(ac(a^2+c^2+ac)^*)^3$ which contradicts $h(u)\in acac(a^2+c^2)^+$). More precisely, either $[w_1]^D\in[L_2]_{\forall}^{D}$ and $n=1$, or $[w_1]^D=[ac]^D$, $[w_2]^D\in[L_1']_{\forall}^{D}$ and $n=2$. Consequently, either:

$$[u]^D\in[L_2]_{\forall}^{D}\qquad\text{or}\qquad[u]^D\in[\{ac\}]_{\forall}^{D}\circ[L_1']_{\forall}^{D}=[L_1]_{\forall}^{D}-[\{\lambda\}]_{\forall}^{D}.$$

Hence, we can conclude that:

$$T\cap[((acac(a^2+c^2)^+)\,\|\,b^*)+\lambda]_{\forall}^{D}=[L_1]_{\forall}^{D}\cup[L_2]_{\forall}^{D},$$

which (by assertion (2) and Lemma 3.3. (2)) implies that $T\notin\mathscr{T}_{\forall}^{\mathscr{D}}(\mathrm{REG})$. □

It has been observed already in [18], that in general $\mathscr{T}_{\exists}^{\mathscr{C}}(\mathrm{REG})$ is not closed under intersection. Moreover, the reader might realize that Δ^*-L_1 and Δ^*-L_2 (L_1 and L_2), where L_1 and L_2 are the string languages from the statement of the above lemma, constitute by no means the simplest example

to prove that $\mathscr{T}_{\exists}^{\mathscr{D}}(\mathrm{REG})$ ($\mathscr{T}_{\forall}^{\mathscr{D}}(\mathrm{REG})$, respectively) is not closed under intersection (union, respectively). For example, one may consider $K_1=(ab+c)^*$ and $K_2=(a+bc)^*$ – then $[K_1]_{\exists}^{D} \cap [K_2]_{\exists}^{D} \notin \mathscr{T}_{\exists}^{\mathscr{D}}(\mathrm{REG})$. However, our choice of L_1 and L_2 was motivated by the proofs of (3) and (4) of the above lemma.

The following closure property for consistently regular trace languages has been proved independently in [8] and in [7].

PROPOSITION 3.5: *Let $\mathscr{C}$ be a concurrent alphabet and let T_1 and T_2 be consistenly regular trace languages over $\mathscr{C}$. Then $T_1 \circ T_2$ is a consistently regular language over $\mathscr{C}$.* □

Now we are ready for the theorem about the closure properties of (all three types of) regular trace languages.

THEOREM 3.6: *The closure properties indicated in Table 3.1 hold (the entry "Yes" means that the given class of trace languages is closed under the given operation independently of the nature of the concurrent alphabet $\mathscr{C}$).*

Proof: The "Yes"-entries of the table follow from Lemma 3.3 and Proposition 3.5.

The fact that $\mathscr{T}_{\exists}^{\mathscr{C}}(\mathrm{REG})$ and $\mathscr{T}_{\forall}^{\mathscr{C}}(\mathrm{REG})$ are closed under complement if and only if C is transitive follows from Theorem 2.7 and Lemma 2.2.

Concerning the other "if and only if C is transitive"-entries, the "if"-parts follow directly from Theorem 2.7 and the "only if"-parts follow from Lemma 3.4.

TABLE 3.1

Closure properties for $\mathscr{T}_{\exists}^{\mathscr{C}}(\mathrm{REG})$, $\mathscr{T}_{\forall}^{\mathscr{C}}(\mathrm{REG})$ and $\mathscr{T}^{\mathscr{C}}(\mathrm{REG})$, where $\mathscr{C}=\langle \Sigma, C \rangle$ is a concurrent alphabet.

	$\mathscr{T}_{\exists}^{\mathscr{C}}(\mathrm{REG})$	$\mathscr{T}_{\forall}^{\mathscr{C}}(\mathrm{REG})$	$\mathscr{T}^{\mathscr{C}}(\mathrm{REG})$
Union	Yes	If and only if C is transitive	Yes
Intersection	If and only if C is transitive	Yes	Yes
Complement	If and only if C is transitive	If and only if C is transitive	Yes
Trace-concatenation	Yes	If and only if C is transitive	Yes
Kleene trace star	Yes	If and only if C is transitive	If and only if C is empty

Finally, that $\mathscr{T}^{\mathscr{C}}(\mathrm{REG})$ is closed under Kleene trace star when C is empty follows from the correspondence with regular string languages. Furthermore, if Σ contains two different symbols a and b such that $\{a, b\} \in C$, then $([\{ab, ba\}]^C)^{(*)} \notin \mathscr{T}^{\mathscr{C}}(\mathrm{REG})$, which proves that in this case $\mathscr{T}^{\mathscr{C}}(\mathrm{REG})$ is not closed under Kleene trace star. □

4. DISCUSSION

We have characterized those concurrent alphabets, for which a trace language obtained from a regular string language by "existential claiming" can be obtained also from a (possibly different) regular string language by "universal claiming". It turns out that exactly those concurrent alphabets with a transitive concurrency relation have this property. This has implications for decidability questions and closure properties as demonstrated in Section 3.

On the one hand, this result can be considered within the framework of the *theory of concurrent events* (*see* [12]). On the other hand, it represents a result for "*partially commutative*" *languages* (subsets of a free "partially commutative" monoid, *see*, e. g., [11]): obviously a trace language over a concurrent alphabet $\mathscr{C} = \langle \Sigma, C \rangle$ is "isomorphic" to a subset of the free monoid Σ^* if C is empty, and it is "isomorphic" to a subset of the free commutative monoid $\Sigma^{(*)}$, if C is complete (*see*, e. g., [16]). As we have seen, for C empty, our result corresponds to the fact that regular string languages are closed under complement, and, for C complete, it corresponds to the fact that the complement of a semilinear set is a semilinear set. (In case the reader skipped the technical parts of this paper, we mention here explicitely that our proofs are built upon those "boundary results" and that we do not want to sell them here as "easy corrollaries".)

Existentially regular trace languages over $\langle \Sigma, C \rangle$ can be regarded as those subsets of the quotient monoid $\Sigma^*/\equiv_C$ which can be obtained from finite sets by a finite sequence of operations union, product and Kleene star (*see* [12]). Thus we have demonstrated that these subsets of $\Sigma^*/\equiv_C$ defined by "regular expressions" are closed under complement (and intersection) if and only if C is transitive. It is also easily seen that universally regular trace languages represent exactly the complements of these "regular expressions".

While we have been able to settle the closure properties for the basic operations (such as union, intersection, trace-concatenation, complement and Kleene trace star), we could give sufficient conditions for which the decidability problems like emptiness, inclusion, equality and intersection emptiness are decidable for existentially (and universally) regular trace languages over a concurrent alphabet $\langle \Sigma, C \rangle$: the problems mentioned are decidable, if C

is transitive (as solved already in [5] for equality and in [6] for emptiness and inclusion). Although the problem is known to be undecidable for arbitrary concurrent alphabets (*see* again [5]), a characterization of "decidable" concurrent alphabets is still missing and is a topic of (con-) current research.

ACKNOWLEDGMENTS

We thank Joost Engelfriet, Grzegorz Rozenberg and the referees for their comments on a first manuscript of this paper and for providing us with references.

REFERENCES

1. IJ. J. AALBERSBERG and G. ROZENBERG, *Traces—a Survey*, Techn. Rep. 85-16, Inst. of Appl. Math. and Comput. Sc., Univ. of Leiden, Leiden, 1985.
2. IJ. J. AALBERSBERG and G. ROZENBERG, *Traces, Dependency Graphs and DNLC Grammars*, Discrete Appl. Math, Vol. 11, 1985, pp. 299-306.
3. A. BERTONI, M. BRAMBILLA, G. MAURI and N. SABADINI, *An Application of the Theory of Free Partially Commutative Monoids: Asymptotic Densities of Trace Languages*, Lecture Notes in Computer Science, Vol. 118, 1981, pp. 205-215.
4. A. BERTONI, G. MAURI and N. SABADINI, *A Hierarchy of Regular Trace Languages and Some Combinatorial Applications*, Proc. 2nd. World Conf. on Math. at the Service of Men, Las Palmas, 1982, pp. 146-153.
5. A. BERTONI, G. MAURI and N. SABADINI, *Equivalence and Membership Problems for Regular Trace Languages*, Lecture Notes in Computer Science, Vol. 140, 1982, pp. 61-71.
6. A. BERTONI, G. MAURI and N. SABADINI, *Unambiguous Regular Trace Languages*, to appear in Algebra, Combinatorics and Logic in Comput. Sc. (to appear), Colloquia Math. Soc. J. Bolay.
7. R. CORI and D. PERRIN, *Automates et Commutations Partielles*, R.A.I.R.O., Inform. Théor., Vol. 19, 1985, pp. 21-32.
8. M. FLEISS, *Matrices de Hankel*, J. Math. Pures Appl., Vol. 53, 1974, pp. 197-222.
9. S. GINSBURG, *The Mathematical Theory of Context Free Languages*, Mc-Graw-Hill Book Company, New York, London, 1966.
10. J. E. HOPCROFT and J. D. ULLMAN, *Introduction to automata theory, languages and computation*, Addison—Wesley, Reading, Mass, 1979.
11. G. LALLEMENT, *Semigroups and combinatorial applications*, J. Wiley and Sons, New York, 1979.
12. A. MAZURKIEWICZ, *Concurrent Program Schemes and Their Interpretations*, DAIMI Rep. PB-78, Aarhus Univ., Aarhus, 1977.
13. A. MAZURKIEWICZ, *Traces, Histories, Graphs: Instances of a Process Monoid*, Lecture Notes in Computer Science, Vol. 176, 1984, pp. 115-133.
14. A. MAZURKIEWICZ, *Semantics of Concurrent Systems: a Modular Fixed-Point Trace Approach*, Lecture Notes in Computer Science, Vol. 188, 1985, pp. 353-375.
15. J. SAKAROVITCH, *On Regular Trace Languages*, R.A.I.R.O., Inform. Théor. (to appear).
16. A. SALOMAA, *Theory of automata*, Pergamon Press, Oxford—New York, 1969.
17. A. SALOMAA, *Formal languages*, Academic Press, New York, 1973.
18. M. SZIJARTO, *A Classification and Closure Properties of Languages for Describing Concurrent System Behaviours*, Fund. Inform., Vol. 4, 1981, pp. 531-549.
19. A. TARLECKI, *Notes on the Implementability of Formal Languages by Concurrent Systems*, ICS PAS Rep. 481, Inst. of Comput. Sc., Polish Acad. of Sc., Warshaw, 1982.