RECENT DEVELOPMENTS IN TRACE THEORY [1]

VOLKER DIEKERT
Institut für Informatik, Universität Stuttgart
Breitwiesenstr. 20-22, D-70565 Stuttgart
e-mail: `Volker.Diekert@informatik.uni-stuttgart.de`

PAUL GASTIN
LITP, Université Paris 7
2, place Jussieu, F-75251 Paris Cedex 05
e-mail: `Paul.Gastin@litp.ibp.fr`

ANTOINE PETIT
LIFAC, ENS Cachan
61, av. du Prés. Wilson, F-94235 Cachan Cedex
e-mail: `Antoine.Petit@lifac.ens-cachan.fr`

ABSTRACT

In this paper we give a survey on some active research in the theory of Mazurkiewicz traces. We restrict our attention to some few topics: recognizable languages, asynchronous automata including infinite traces, trace codings, and trace rewriting.

Introduction

The theory of free partially commutative monoids has its origins in the combinatorial study of rearrangements, [8]. In computer science partial commutation became an important concept to study parallelism. It was Mazurkiewicz who introduced this viewpoint in a systematic way, and trace theory was born, [26].

Nearly two decades passed and the theory of traces is still an active area of research, as it is documented in *The Book of Traces,* [16]. In this survey we would like to focus on some recent developments around the themes: recognizable languages, asynchronous automata for finite and infinite traces, trace codings, and trace rewriting. Each theme leads to open research problems and we give references where more details can be found.

We shall use the following notations. A *dependence alphabet* is a pair (Σ, D), where Σ is a finite alphabet and $D \subseteq \Sigma \times \Sigma$ is a reflexive and symmetric relation, called the *dependence relation.* The complement $I = \Sigma \times \Sigma \setminus D$ is called the *independence relation.* Both, (Σ, D) and (Σ, I) are viewed as undirected graphs, for convenience we omit self-loops when drawing (Σ, D). The trace monoid $\mathbb{M}(\Sigma, D)$ is the quotient monoid $\Sigma^*/\{ab = ba \mid (a,b) \in I\}$. Due to its algebraic properties it is also called a *free partially commutative monoid.* The canonical surjection from Σ^* to $\mathbb{M}(\Sigma, D)$ which associate with each word w its equivalence class $[w]$ is denoted by φ. An element

[1] This research has been supported by the ESPRIT Basic Research Action No. 6317 ASMICS II and the French-German research programme PROCOPE.

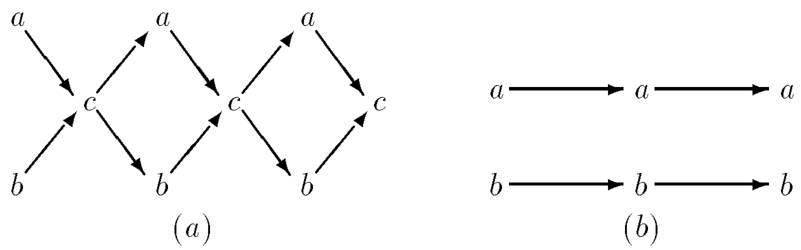

Figure 1: Hasse diagram of $[(abc)^3]$ and $[(ab)^3]$ over $(\Sigma, D) = a - c - b$

$t \in \mathbb{M}(\Sigma, D)$ is called a *trace*, it is given by a representing word $w \in \Sigma^*$ such that $[w] = t$. We can use any representing word to define the length $|t|$ and the alphabet $\mathrm{alph}(t)$ by $|t| = |w|$, $\mathrm{alph}(t) = \mathrm{alph}(w)$, if $[w] = t$.
The most convenient way to represent a trace is however by its *dependence graph*. Let $w = a_1 \cdots a_n \in \Sigma^*, a_i \in \Sigma, 1 \leq i \leq n$ be a non empty word. We associate with w a labeled directed graph $[V, E, \lambda]$. We choose a set V of n nodes, $V = \{x_1, \ldots, x_n\}$. Each node x_i is labeled with $\lambda(x_i) = a_i \in \Sigma$. The set of edges E is defined by $E = \{(x_i, x_j) \mid i < j \text{ and } (\lambda(x_i), \lambda(x_j)) \in D\}$. It turns out that two words w, w' define isomorphic labeled graphs if and only if they are equivalent, i.e., $[w] = [w']$. Therefore, the trace $t = [w] = [w']$ can be identified with (the isomorphism class of) the labeled directed graph $[V, E, \lambda]$, its dependence graph. Note that this graph is acyclic.
Replacing E by its transitive closure E^*, we obtain a labeled partial order (or pomset) $[V, E^*, \lambda]$, which is another way to represent a trace, once (Σ, D) is fixed.

Example *Let $(\Sigma, D) = a - c - b$ with $a < b < c$. The trace $[(bac)^3]$ is connected. The Hasse diagram of its dependence graph is depicted in Figure 1 (a). The trace $[(ab)^3]$ is not connected and the Hasse diagram of its dependence graph is depicted in Figure 1 (b).*

Given a total order $<$ on Σ we can choose the lexicographical first word representing a trace $t \in \mathbb{M}(\Sigma, D)$. This is called the lexicographical normal form of t. For instance, if $(\Sigma, D) = a - c - b$, the lexicographic normal form of $[(bac)^3]$ is $abcabcabc$, of $[(ab)^3]$ it is the word a^3b^3. A word is the lexicographical normal form of some trace if and only if it contains no factor of the form bua where $a, b \in \Sigma, u \in \Sigma^*$ such that $a < b$ and $\{a\} \times \mathrm{alph}(bu) \subseteq I$ (see [2]). Therefore the set of all lexicographical normal forms, denoted by LexNF, is a regular set. Moreover, due to the description above, it is obvious that LexNF is star-free. (The definition of star-freeness is recalled in the next section).
The notion of finite traces generalizes to infinite traces by starting with an infinite sequence $a_1a_2a_3 \cdots \in \Sigma^\omega$. The dependence graphs which are images of the finite and infinite words form the set $\mathbb{R}(\Sigma, D)$ of real traces. The surjection from Σ^∞ to $\mathbb{R}(\Sigma, D)$, associating with a finite or infinite word its dependence graph, is denoted

by φ. Note that $\varphi(\Sigma^*) = \mathbb{M}(\Sigma, D)$. The alphabet at infinity of a dependence graph t is the set $\mathrm{alphinf}(t)$ of letters which label an infinite number of nodes. A trace t is *connected*, if its dependence graph is connected. That is, $\mathrm{alph}(t)$ induces a connected subgraph of (Σ, D).
Let $(g_i = [V_i, E_i, \lambda_i])_i$ be any sequence of real traces. We may assume without restriction that the sets of vertices $(V_i)_i$ are pairwise disjoint. The product $g_0 g_1 g_2 \cdots$ is defined only if $\mathrm{alphinf}(g_i) \times \mathrm{alph}(g_j) \subseteq I$ for all $i < j$. Then it is equal to the dependence graph

$$[\cup_i V_i, \cup_i E_i \cup \{(x,y) \mid \exists i < j : x \in V_i, y \in V_j, (\lambda_i(x), \lambda_j(y)) \in D\}, \cup_i \lambda_i].$$

1 Recognizable Languages

1.1 Finite Traces

A trace language $L \subseteq \mathbb{M}(\Sigma, D)$ is *recognizable* if and only if there exists a homomorphism $h : \mathbb{M}(\Sigma, D) \to N$ to some finite monoid N such that $L = h^{-1}(h(L))$. It turns out that a trace language $L \subseteq \mathbb{M}(\Sigma, D)$ is recognizable if and only if the word language $\varphi^{-1}(L)$ is a recognizable word language. The characterization of recognizable trace languages by finite asynchronous automata is stated in Sect. 1.3.
Recall that a language is *rational* if and only if it can be built up from finite languages using the operation $\cup$ (union), $\cdot$ (concatenation), and $*$ (Kleene-star). If we allow the complementation operation instead of Kleene-star, we obtain the family of *star-free* languages. The family of recognizable languages is a Boolean algebra. Since trace monoids are finitely generated, it is easy to see that recognizable languages are rational. The first non-trivial result implies that all star-free languages are recognizable [18, 10]. A trace language is called *connected* if all its elements are connected (i.e., connected dependence graphs). Métivier and Ochmański have independently shown that the star of a connected recognizable trace language is recognizable. One obtains the family of *c-rational* languages, if in the definition of rational the use of the $*$-operation is restricted to connected languages, only. Ochmański's Theorem states that a language is recognizable if and only if it is c-rational, [32].
As for words, recognizability can be directly related to logic. First-order formulae and monadic second-order formulae are based on first order variables $x, y, \ldots$ ranging over V and second-order variables $X, Y, \ldots$ ranging over subsets of V. Atomic formulae are of the form $x < y, (x,y) \in E, \lambda(x) = a$ for $a \in \Sigma$, and $x \in X$. We then allow the usual logical connectives $\wedge, \vee, \neg$ and quantification $\forall, \exists$ for the first and second order variables. A first-order formula is a formula without any second order variable.
A closed formula ψ has an interpretation over a trace by identifying $t \in \mathbb{M}(\Sigma, D)$ with its dependence graph $t = [V, E, \lambda]$. The truth value of $t \models \psi$ is therefore well-defined and we can speak of first-order and monadic second-order definability of trace languages. Due to the definition of a dependence graph the transitive closure $(x,y) \in E^*$ of the edge relation can be specified by a first-order formula with at most

$|\Sigma|-2$ additional variables. The connection with logic leads to the following theorem [32, 36], see also [16, chap. 10].

Theorem 1 *Let $L \subseteq \mathbb{M}(\Sigma, D)$ be a trace language, then the following assertions are equivalent:*
1. L is recognizable,
2. L is c-rational,
3. L is definable in monadic second-order logic.

This theorem has a first-order counterpart stating the equivalence between star-free languages, languages definable in first-order logic and languages recognizable by some aperiodic finite monoid [17, 22].
Using the bijection between traces and their representing words in lexicographical normal form we obtain a canonical one-to-one correspondence between recognizable (star-free) trace languages and recognizable (star-free) word languages in lexicographical normal form. In fact, this correspondence (together with classical results) can be viewed as the heart of the theorems above. This viewpoint has been stressed in [11, 16, Chapt. 10]. There is an amazingly simple proof for this logical correspondence, which is worth to be sketched.
Assume we have a monadic second-order (first-order, resp.) formula ψ defining a recognizable trace language L. Since we know that $(x, y) \in E$ is equivalent to $x < y$ and $(\lambda(x), \lambda(y)) \in D$ for any representing word, we may replace the atomic formula $(x, y) \in E$ by the conjunction above. This gives us the formula defining all representing words, i.e., the language $\varphi^{-1}(L) \subseteq \Sigma^*$. The language LexNF $\subseteq \Sigma^*$ of words in lexicographical normal form is first-order. Hence, we obtain a monadic second-order (first-order, resp.) formula for $\varphi^{-1}(L) \cap$ LexNF. The converse is a little more subtle. Let $t = [V, E, \lambda]$ be a trace and let $\hat{t} = x_1 \cdots x_n \in \Sigma^*$ be its representation in lexicographical normal form. Every node $x \in V$ corresponds to some x_i in $\hat{t}$. We write $\mathrm{lex}(x, y)$, if x corresponds to x_i, y corresponds to x_j, and if we have $i < j$. If this happens, then either $(x, y) \in E^+$ or there is some minimal index k such that $i < k \leq j$, where x_k corresponds to some node $z \in V$ such that $(z, y) \in E^*$. Since $\hat{t}$ is in lexicographical normal form, we have $\lambda(x) < \lambda(z)$ with respect to the ordering of the alphabet. Based on this observation we obtain the following relation:

$$\begin{array}{rcl} \mathrm{lex}(x, y) & = & (x, y) \in E^+ \text{ or} \\ & & \lambda(x) < \lambda(y) \text{ and } \neg\mathrm{lex}(y, x) \text{ or} \\ & & \lambda(y) < \lambda(x) \text{ and } \exists z : \lambda(x) < \lambda(z) \wedge \mathrm{lex}(x, z) \wedge (z, y) \in E^* \end{array}$$

Since the alphabet is finite, it is enough to unfold the recursion $(2 \cdot |\Sigma| - 1)$ times. Thus, $\mathrm{lex}(x, y)$ can be replaced by a first-order formula which has an interpretation over traces. The result can be now deduced as follows: Given a formula for a word language $K \subseteq$ LexNF. Replace all atomic formulae $x < y$ (used for the linear ordering of words) by $\mathrm{lex}(x, y)$. As explained above the new formula defines the trace language $\varphi(K)$.

1.2 The Star Problem

One of the main open problems about recognizable trace languages is a decidability question related to the Kleene-star : the star problem. It is not known whether or not one can decide the recognizability of L^*, if $L \subseteq \mathbb{M}(\Sigma, D)$ is recognizable (finite, resp.). The existence of a semi-algorithm to solve this problem can be derived from the following well-known fact:

Proposition 2 *Let $L \subseteq \mathbb{M}(\Sigma, D)$ be a recognizable trace language. Then L^* is recognizable if and only if there exists a recognizable language $K \subseteq \mathbb{M}(\Sigma, D) \setminus \{1\}$ such that both $K^2 \subseteq K$ and $(K \setminus K^2) \subseteq (L \setminus \{1\}) \subseteq K$. Moreover, if such a K exists, then $K = L^* \setminus \{1\}$ and $(K \setminus K^2)$ is a basis of L^*.*

Many attempts have been made to solve the star problem, see [33] and [16, Chapt. 5]. Recently, there has been some progress [19, 27, 28, 29, 35] showing a close relation between the star problem and the finite power property (FPP). A language $L \subseteq \mathbb{M}(\Sigma, D)$ is said to satisfy the FPP, if $L^* = (L \cup \{1\})^k$ for some $k \geq 0$. If a recognizable language L satisfies the FPP, then L^* is recognizable, too. The converse need not be true, e.g., let $\emptyset \neq \Gamma \subseteq \Sigma$. Then $\Gamma^* \subseteq \mathbb{M}(\Sigma, D)$ is recognizable, it is even star-free, but it does not satisfy the FPP. Richomme has shown the following result.

Theorem 3 ([35]) *If the independence alphabet (Σ, I) does not contain a C_4 (cycle of four letters) as an induced subgraph, then both, the star problem and the finite power problem are decidable.*

1.3 Infinite Traces and Asynchronous Automata

A convenient way to define the notion of recognizable language of finite or infinite traces uses the canonical surjection φ from Σ^∞ into $\mathbb{R}(\Sigma, D)$. A real trace language $L \subseteq \mathbb{R}(\Sigma, D)$ is recognizable if the word language $\varphi^{-1}(L)$ is recognizable. The family of recognizable languages of real traces can be characterized in many equivalent ways assessing the robustness of this definition. For instance, a finite monoid S recognizes a trace language $L \subseteq \mathbb{R}(\Sigma, D)$ if there exists some morphism η from $\mathbb{M}(\Sigma, D)$ to S such that for all finite or infinite sequences $(r_i)_i, (s_i)_i \subseteq \mathbb{M}(\Sigma, D)$ with $\eta(r_i) = \eta(s_i)$ for all i, it holds

$$r_0 r_1 r_2 \ldots \in L \Longleftrightarrow s_0 s_1 s_2 \ldots \in L$$

The finite and infinite products defined on real traces induce natural operations on real trace languages. For instance, the concatenation of $K, L \subseteq \mathbb{R}(\Sigma, D)$ is

$$K \cdot L = \{g \cdot h \mid g \in K, h \in L, \mathrm{alphinf}(g) \times \mathrm{alph}(h) \subseteq I\}$$

The $*-$iteration and the $\omega-$iteration of a real trace language L are defined similarly using, respectively, finite and infinite products of elements of L.
The family of *c-rational* languages of real traces is defined as in the case of finite traces by just adding an $\omega-$iteration restricted to connected languages. The notion

of real trace languages defined by a first-order or monadic second-order formula is a straightforward generalization of the finite trace case.
The following theorem gives various characterizations of recognizable real trace languages. See [16, Chapt. 11] or the original papers [17, 21]. It can be seen as a generalization of Kleene's and Büchi's theorems on words and of Theorem 1 on finite traces. As in the case of finite traces, this theorem has a first-order counterpart which states the equivalence between first-order languages, star-free languages and aperiodic languages [16, 17].

Theorem 4 *Let $L \subseteq \mathbb{R}(\Sigma, D)$ be a real trace language, then the following assertions are equivalent:*
1. L is recognizable,
2. L is recognized by some finite monoid,
3. L is c-rational,
4. L is definable in monadic second-order logic.

In the sequel, we are concerned with the relation between recognizable real trace languages and asynchronous (cellular) automata. Asynchronous (cellular) automata, introduced by Zielonka [39, 40], have a decentralized control allowing the concurrent execution of independent actions. This feature makes them fundamental for the recognition of trace languages. We deal with the cellular version only, since one can transform an asynchronous cellular automaton directly into an asynchronous automaton, as defined originally in [39]. For more details about asynchronous automata see also [16, Chapt. 6,7].
An asynchronous cellular automaton is a tuple $\mathcal{A} = ((Q_a)_{a\in\Sigma}, (\delta_a)_{a\in\Sigma}, S, F, \mathcal{R})$ where

- for each letter a, Q_a is a finite set of local states for a,
- for each letter a, $\delta_a : \prod_{b\in D(a)} Q_b \longrightarrow \mathcal{P}(Q_a)$ is the (non deterministic) local transition function for a,
- $S \subseteq \prod_{a\in\Sigma} Q_a$ is the set of initial global states,
- $F \subseteq \prod_{a\in\Sigma} Q_a$ is the set of final global states,
- $\mathcal{R} \subseteq \prod_{a\in\Sigma} \mathcal{P}(Q_a)$ is the acceptance table for infinite runs.

The automaton is deterministic if there is only one initial state and if the transition functions are deterministic. The behaviour of asynchronous cellular automata was originally described by Zielonka using a global transition function. In order to underline its decentralized control and its concurrent behaviour, we prefer to define its runs on dependence graphs.
A run of the asynchronous cellular automaton $\mathcal{A}$ on the real trace $g = [V, E, \lambda]$ is a pair (s, ρ) where $s \in S$ is a global initial state and $\rho : V \longrightarrow \cup_{a\in\Sigma} Q_a$ is a labeling of the vertices which satisfies :

1. for all $a \in \Sigma$, $\rho(\lambda^{-1}(a)) \subseteq Q_a$
2. for all $v \in \lambda^{-1}(a)$, $\rho(v) \in \delta_a\left(\rho\left(\partial(s,v,b)\right)_{b \in D(a)}\right)$

where $\partial(s,v,b)$ is s_b if there is no vertex labeled by b in the past of v and $\partial(s,v,b)$ is the greatest vertex labeled by b in the past of v otherwise.
In order to define when a run is accepted, we need to introduce its final information : for all $a \in \Sigma$ we let $f(s,\rho) = (f_a(s,\rho))_{a \in \Sigma}$ where

$$f_a(s,\rho) = \begin{cases} \{s_a\} & \text{if } a \notin \mathrm{alph}(g) \\ \{\rho(v_a)\} & \text{if } a \in \mathrm{alph}(g) \setminus \mathrm{alphinf}(g) \text{ and } v_a \text{ is} \\ & \text{the greatest vertex of } g \text{ labeled by } a \\ \{q \in Q_a \mid \rho^{-1}(q) \text{ is infinite}\} & \text{if } a \in \mathrm{alphinf}(g) \end{cases}$$

A finite run is accepted if $f(s,\rho) \in F$. Now we can state Zielonka's Theorem [39, 40]:

Theorem 5 *A language of finite traces is recognizable if and only if it is accepted by some deterministic asynchronous cellular automaton.*

An infinite run is Büchi accepted if there exists some $(R_a)_{a \in \Sigma} \in \mathcal{R}$ such that $R_a \subseteq f_a(s,\rho)$ for all $a \in \Sigma$. The connection between asynchronous cellular automata and recognizable real trace languages was first given in [20] :

Theorem 6 *A real trace language is recognizable if and only if it is accepted by some non-deterministic Büchi asynchronous cellular automaton.*

Indeed, this theorem does not hold if we restrict to deterministic Büchi asynchronous cellular automata. As for words, we have to consider more powerful acceptance conditions in order to use deterministic automata. An infinite run (s,ρ) is *Muller-accepted*, if $f(s,\rho) \in \mathcal{R}$. McNaughton's Theorem was extended to infinite traces in [14] :

Theorem 7 *A real trace language is recognizable if and only if it is accepted by some deterministic Muller asynchronous cellular automaton.*

The original proof [14] contains, in an implicit way, a double exponential algorithm to compute a deterministic Muller asynchronous cellular automaton from a non-deterministic Büchi one. This complexity has been improved in [30] where, as intermediary results, determinization procedures for asynchronous cellular automata and asynchronous automata are proposed. For the later family, a determinization procedure is also presented in [25]. Using the results of Pighizzini [34] to construct non-deterministic asynchronous automata from any c-rational languages, these determinization procedures yield a new proof of Zielonka's theorem.
Concerning recognizable real trace languages, the main open question is to define a suitable family of deterministic languages. As in the infinite word case, this family should be characterized in terms of deterministic Büchi automata and limits of

recognizable finite trace languages. Moreover, its Boolean closure should coincide with the whole family of recognizable real trace languages. Note that this later property is verified by a family introduced in [14] which corresponds to the deterministic recognizable word languages when the dependence relation is full, i.e., in the word case.

2 Trace Codings

Codes play a fundamental role in computer science and codes over words are widely studied [3]. There are well-known algorithms to test whether a finite (recognizable, resp.) set $X \subseteq \Sigma^*$ is a code, i.e., whether it generates a free submonoid with basis X. By a reduction of the Post-Correspondence Problem, it has been shown that for $M = \{a,b\}^* \times \{c,d\}^*$ the code property is undecidable. That is, given a finite set $X \subseteq M$, it is undecidable whether X^* is a free submonoid [24, page 102], see also [9]. In the setting of partial commutation the following notion of a code seems to be more natural. A (finite) set $X \subseteq \mathbb{M}(\Sigma, D)$ is called a *trace-code*, if X is the basis of a free partially commutative submonoid of $\mathbb{M}(\Sigma, D)$. Rephrasing this in another way, one can associate with a subset X an independence relation $I_X \subseteq X \times X$ by letting $(x,y) \in I_X$, if $xy = yx$ in $\mathbb{M}(\Sigma, D)$. Let $D_X = X \times X \setminus I_X$ be the complement. Then the set-inclusion $X \subseteq \mathbb{M}(\Sigma, D)$ induces a canonical homomorphism $h_X : \mathbb{M}(X, D_X) \longrightarrow \mathbb{M}(\Sigma, D)$. The subset X is a trace-code if and only if the homomorphism h_X is injective. For example any subset $X \subseteq \Sigma$ is a trace-code of $\mathbb{M}(\Sigma, D)$. The undecidability result for $\{a,b\}^* \times \{c,d\}^*$ holds for trace-codes, too. In fact, since a code is simply a trace-code such that I_X is empty, we can easily reduce the decision of the code property to the trace-code property.
The decidability result below is stated for the code property in [9]. Using decidability results from [1] for the intersection problem of rational trace languages one can show the following slightly stronger assertion.

Theorem 8 *Let the independence alphabet be a transitive forest. Then it is decidable whether a finite set $X \subseteq \mathbb{M}(\Sigma, D)$ is a trace-code.*

Transitive forests can be defined by forbidden structures. An independence alphabet is a transitive forest if and only if it does not contain neither C_4 nor P_4 as an induced subgraph. By C_4 we mean the cycle and by P_4 the chordless path with 4 letters. An undecidability result for P_4 would have settled the problem. However, very recently Hoogeboom and Muscholl obtained the following positive result, (which is therefore negative for completing the picture). It has been independently observed by Matiyasevich (personal communication), too, who also proposed the notion of trace-code.

Theorem 9 ([23]) *If (Σ, I) is a P_4, then the trace-code property can be decided in polynomial time.*

It is easy to see that if the trace-code property can be decided for (Σ, D), then it can be decided for all induced subgraphs of (Σ, D) and for all dependence alphabets which are obtained by edge contraction. Unfortunately, this is not enough to conclude that the undecidability of the trace-code property can be characterized by some finite set of forbidden induced subgraphs. The classification of all these forbidden structures is therefore perhaps an infinite program.
A closely related question is how to decide whether for two given trace monoids $\mathbb{M}_i = \mathbb{M}(\Sigma_i, D_i)$, $i = 1, 2$, there exists an injective homomorphism (also called a coding) $h : \mathbb{M}_1 \longrightarrow \mathbb{M}_2$. In general, this question is open and only some partial results and conjectures are known [6, 7, 15]. A natural subclass of homomorphisms are those which map independent letters to independent traces. Bruyère et al. called a homomorphism $h : \mathbb{M}_1 \longrightarrow \mathbb{M}_2$ a *strong coding*, if it is injective and $\mathrm{alph}(h(a)) \times \mathrm{alph}(h(b)) \subseteq I_2$ for all $(a, b) \in I_1$. In [7] the question is raised, whether the existence of a strong coding can be decided. The decision result below implies the NP-completeness of this problem.

Proposition 10 ([15]) *There is some strong coding $h : \mathbb{M}(\Sigma_1, D_1) \longrightarrow \mathbb{M}(\Sigma_2, D_2)$ if and only if there is a relation $H \subseteq \Sigma_2 \times \Sigma_1$ with the following properties:*

1. *for all $a \in \Sigma_1$ there exists $b \in \Sigma_2$ with $a \in H(b)$*
2. *for all $(b, b') \in D_2$ we have $H(b) \times H(b') \subseteq D_1$*
3. *for all $(a, a') \in D_1$, $a \neq a'$, there exists $(b, b') \in D_2$, $b \neq b'$, such that $a \in H(b)$ and $a' \in H(b')$.*

Furthermore, if $h : \mathbb{M}(\Sigma_1, D_1) \longrightarrow \mathbb{M}(\Sigma_2, D_2)$ is a strong coding, then such a relation $H \subseteq \Sigma_2 \times \Sigma_1$ can be defined by $\bigcup_{a \in \Sigma_1} \mathrm{alph}(h(a)) \times \{a\}$. Conversely, if $H \subseteq \Sigma_2 \times \Sigma_1$ is a relation satisfying the conditions above then we find a strong coding h such that $(b, a) \in H \Leftrightarrow b \in \mathrm{alph}(h(a))$.

3 Trace Rewriting

A trace rewriting system (TRS) is a finite set of rules $S \subseteq \mathbb{M} \times \mathbb{M}$ with $\mathbb{M} = \mathbb{M}(\Sigma, D)$. It is called *length-reducing*, if we have $|l| > |r|$, for all $(l, r) \in S$. The one-step reduction $t \underset{S}{\Longrightarrow} t'$ is defined, if $t = ulv$, $t' = urv$ for some $u, v \in \mathbb{M}$, $(l, r) \in S$. The reduction relation is the transitive closure $\overset{*}{\underset{S}{\Longrightarrow}}$. The system is called *confluent*, if for all $p, t, q \in \mathbb{M}$ with $p \overset{*}{\Longleftarrow} t \overset{*}{\Longrightarrow} q$ there exists some $z \in \mathbb{M}$ such that $p \overset{*}{\Longrightarrow} z \overset{*}{\Longleftarrow} q$. If $\mathbb{M} = \Sigma^*$ is free, then a TRS is a semi-Thue system; if $\mathbb{M}$ is commutative, then a TRS is a vector addition system (or Petri net). In both cases the confluence of length-reducing systems is decidable by considering finitely many critical pairs. Over traces confluence becomes undecidable, in general.

Theorem 11 ([31]) *The confluence of a length-reducing TRS is undecidable, in general, even if $\mathbb{M}(\Sigma, D)$ is defined by a single commutation rule.*

The undecidability of Thm. 11 holds already for four letters, but the status is unclear for the dependence alphabet $a - c - b$. For one-rule systems the situation is simpler:

Proposition 12 ([38]) *The confluence of a one-rule TRS is decidable if the left-hand side is non-empty and connected.*

Another interesting question related to length-reducing TRS is the (non-uniform) complexity of computing irreducible descendants. (Non-uniform means that the TRS is fixed, the input is therefore a single trace.) In the case of semi-Thue or vector addition systems linear time algorithms are well-known. It is not difficult to exhibit a square time bound, and nothing better is known yet for the general case. So far, better algorithms have been developed for special cases, only.

Proposition 13 ([12]) *For length-reducing TRS there is a linear time algorithm which computes irreducible descendants, if all left-hand sides are cones or blocks.*

A cone is a trace t with a unique minimal element labeled by $a \in \Sigma$ (hence a pyramid in the terminology of Viennot [37]) such that $(a, b) \in I \Leftrightarrow \mathrm{alph}(t) \times \{b\} \subseteq I$, for all $b \in \Sigma$. A block is a trace t such that $tb = bt \Leftrightarrow ab = ba$ for all $b \in \Sigma$ and all minimal elements a of t. The above proposition generalizes already the word and vector case. It has been generalized to less restrictive conditions, see [12], and Bertol proved the following linear time result.

Theorem 14 ([4]) *Let (Σ, D) be a cograph. For every length-reducing TRS over $\mathbb{M}(\Sigma, D)$ there is a linear time algorithm, which computes irreducible descendants.*

The proof of this result is an induction on the cograph structure of the underlying dependence alphabet. We doubt that linear time algorithms are possible, in general. The following result about the possibility of an $\mathcal{O}(n \log n)$-algorithm has been shown in [5].

Proposition 15 *For every length-reducing TRS, where all left-hand sides are connected, there is an algorithm which computes an irreducible descendant of a trace of length n in $\mathcal{O}(n \log n)$ steps.*

It is still open whether we can remove the hypothesis of connectedness in the proposition above. Other open problems and some nice relations between trace rewriting and Möbius functions are surveyed in [13].

References

[1] IJ. J. Aalbersberg and H. J. Hoogeboom. Characterizations of the decidability of some problems for regular trace languages. *Math. Syst. Theory*, 22:1–19, 1989.

[2] A. V. Anisimov and D. E. Knuth. Inhomogeneous sorting. *Int. J. of Comp. and Inf. Sci.*, 8:255–260, 1979.

[3] J. Berstel and D. Perrin. *Theory of Codes.* Pure and Applied Mathematics; 117. Academic Press, Orlando, Florida, 1985.

[4] M. Bertol. Efficient rewriting in cograph trace monoids. In H. Reichel, editor, *Proc. of the 10th FCT '95, Dresden (Germany) 1995*, number 965 in LNCS, pages 146–155, Springer, 1995. Springer.

[5] M. Bertol and V. Diekert. Trace rewriting: Computing normal forms in time $\mathcal{O}(n \log n)$. In C. Puech and R. Reischuk, editors, *Proc. of the 13th STACS, Grenoble (France) 1996*, LNCS. Springer, 1996.

[6] V. Bruyère and C. De Felice. Trace codings. In E. W. Mayr and C. Puech, editors, *Proc. of the 12th STACS 1995*, number 900 in LNCS, pages 373–384. Springer, 1995.

[7] V. Bruyère, C. De Felice, and G. Guaiana. Coding with traces. In P. Enjalbert, E.W. Mayr, and K. W. Wagner, editors, *Proc. of the 11th STACS 1994*, number 775 in LNCS, pages 353–364. Springer, 1994.

[8] P. Cartier and D. Foata. *Problèmes combinatoires de commutation et réarrangements.* Number 85 in LN in math. Springer, 1969.

[9] M. Chrobak and W. Rytter. Unique decipherability for partially commutative alphabets. *Fundamenta Informaticae*, X:323–336, 1987.

[10] R. Cori and D. Perrin. Automates et commutations partielles. *R.A.I.R.O.*, 19:21–32, 1985.

[11] B. Courcelle. The monadic second-order logic of graphs X: linear orderings. *Theoret. Comp. Sci.*, 1996. To appear.

[12] V. Diekert. *Combinatorics on Traces.* Number 454 in LNCS. Springer, 1990.

[13] V. Diekert. Rewriting, semi-commutations, and Möbius functions. In Z. Ésik, editor, *Proc. of the 9th FCT 93, Szeged (Hungary) 1993*, number 710 in LNCS, pages 1–15. Springer, 1993. Invited Lecture.

[14] V. Diekert and A. Muscholl. Deterministic asynchronous automata for infinite traces. *Acta Informatica*, 31:379–397, 1994.

[15] V. Diekert, A. Muscholl, and K. Reinhardt. On codings of traces. In E.W. Mayr and C. Puech, editors, *Proc. of the 12th STACS 1995*, number 900 in LNCS, pages 385–396. Springer, 1995.

[16] V. Diekert and G. Rozenberg, editors. *The Book of Traces.* World Scientific, Singapore, 1995.

[17] W. Ebinger and A. Muscholl. Logical definability on infinite traces. In Andrzej Lingas, Rolf Karlsson, and Svante Carlsson, editors, *Proc. of the 20th ICALP Lund (Sweden) 1993*, number 700 in LNCS, pages 335–346. Springer, 1993.

[18] M. Fliess. Matrices de Hankel. *J. Math. Pures et Appl.*, 53:197–224, 1974.

[19] P. Gastin, E. Ochmański, A. Petit, and B. Rozoy. Decidability of the Star problem in $A^*\times\{b\}^*$. *Inform. Proc. Let.*, 44:65–71, 1992.

[20] P. Gastin and A. Petit. Asynchronous automata for infinite traces. In W. Kuich, editor, *Proc. of the 19th ICALP Vienna (Austria) 1992*, number 623 in LNCS, pages 583–594. Springer, 1992.

[21] P. Gastin, A. Petit, and W. Zielonka. An extension of Kleene's and Ochmański's theorems to infinite traces. *Theoret. Comp. Sci.*, 125:167–204, 1994. A preliminary version was presented at ICALP'91, LNCS 510 (1991).

[22] G. Guaiana, A. Restivo, and S. Salemi. Star-free trace languages. *Theoret. Comp. Sci.*, 97:301–311, 1992. A preliminary version was presented at STACS'91, LNCS 480 (1991).

[23] H. J. Hoogeboom and A. Muscholl. The code problem for traces – improving the boundaries. Submitted for publication., 1995.

[24] G. Hotz and V. Claus. *Automatentheorie und Formale Sprachen, Band III.* Bibliographisches Institut, Mannheim, 1972.

[25] N. Klarlund, M. Mukund, and M. Sohoni. Determinizing asynchronous automata. In Serge Abiteboul and Eli Shamir, editors, *Proc. of the 21st ICALP Jerusalem (Israel) 1994*, number 820 in LNCS, pages 130–141, 1994.

[26] A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

[27] Y. Métivier and G. Richomme. On the star operation and the finite power property in free partially commutative monoid. In P. Enjalbert, E. W. Mayr, and K. W. Wagner, editors, *Proc. of the 11th STACS, 1994*, number 775 in LNCS, pages 341–352. Springer, 1994.

[28] Y. Métivier, G. Richomme, and P. A. Wacrenier. Computing the closure of words under partial commutation. In Zoltán Fülöp and F. Gécseg, editors, *Proc. of the 22nd ICALP Szeged (Hungary) 1995*, number 944 in LNCS, pages 75–86, 1995 LNCS. Springer, 1995.

[29] Y. Métivier and B. Rozoy. On the star operation in free partially commutative monoids. *International Journal of Foundations of Computer Science*, 2:257–265, 1991.

[30] A. Muscholl. On the complementation of Büchi asynchronous cellular automata. In Serge Abiteboul and Eli Shamir, editors, *Proc. of the 21st ICALP Jerusalem (Israel) 1994*, number 820 in LNCS, pages 142–153. Springer, 1994.

[31] P. Narendran and F. Otto. Preperfectness is undecidable for Thue systems containing only length-reducing rules and a single commutation rule. *Inform. Proc. Let.*, 29:125–130, 1988.

[32] E. Ochmański. Regular behaviour of concurrent systems. *Bull. of the EATCS*, 27:56–67, Oct 1985.

[33] E. Ochmański. Notes on a star mystery. *Bull. of the EATCS*, 40:252–257, 1990.

[34] G. Pighizzini. Synthesis of nondeterministic asynchronous automata. In M. Droste and Y. Gurevich, editors, *Semantics of programming languages and model theory*, number 5 in Algebra, Logic and Applications, pages 109–126. Gordon and Breach Science Publ., 1993.

[35] G. Richomme. Some trace monoids where both the Star Problem and the Finite Power Property Problem are decidable. In I. Privara et al., editors, *Proc. of the 19th MFCS, Košice (Slovakia) 1994*, number 841 in LNCS, pages 577–586. Springer, 1994.

[36] W. Thomas. On logical definability of trace languages. In V. Diekert, editor, *Proc. an ASMICS workshop, Kochel am See 1989*, Report TUM-I9002, Technical University of Munich, pages 172–182, 1990.

[37] X. G. Viennot. Heaps of pieces I: Basic definitions and combinatorial lemmas. In G. Labelle et al., editors, *Proceedings Combinatoire énumerative, Montréal, Québec (Canada) 1985*, number 1234 in LN in math., pages 321–350. Springer, 1986.

[38] C. Wrathall and V. Diekert. On confluence of one-rule trace-rewriting systems. *Math. Syst. Theory*, 28:341–361, 1995.

[39] W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O.*, 21:99–135, 1987.

[40] W. Zielonka. Safe executions of recognizable trace languages by asynchronous automata. In A. R. Mayer et al., editors, *Proc. of the Symp. on LFCS, Logic at Botik '89 (USSR), 1989*, number 363 in LNCS, pages 278–289. Springer, 1989.