Message Sequence Graphs and Decision Problems on Mazurkiewicz Traces

Anca Muscholl[1]* and Doron Peled[2]

[1] Institut für Informatik, Universität Stuttgart,
Breitwiesenstr. 20-22, 70565 Stuttgart, Germany
[2] Department of Computer Science
Technion, Israel Institute of Technology
32000 Haifa, Israel
and
Bell Laboratories, 600 Mountain Ave.
Murray Hill, NJ 07974, USA

Abstract. Message sequence charts (MSC) are a graphical specification language widely used for designing communication protocols. Our starting point are two decision problems concerning the correctness and the consistency of a design based by MSC graphs. Both problems are shown to be undecidable, in general. Using a natural connectivity assumption from Mazurkiewicz trace theory we show both problems to be EXPSPACE-complete for locally synchronized graphs. The results are based on new complexity results for star-connected rational trace languages.

Keywords: message sequence graphs, Mazurkiewicz semi-traces, automata theory, universality problem

1 Introduction

A recent trend in formal methods is the use of tools and techniques that are based on visual notation. Another trend is the use of standard methods, allowing seemless transfer of technology. Message sequence charts (MSCs) is a notation that has a standard visual and textual presentation (ITU Recommendation Z.120, see figures below). This notation is frequently used for specifying the design of communication protocols. It abstracts away from e.g., the actual code, or the value of variables, and concentrates on the messages exchanged between the different participating processes.

Analogously to systems described using finite state automata, there are natural algorithmic problems which arise from debugging the design of communication systems using MSCs. Such problems are related to the correctness of the design with respect to the specification, and its internal consistency. It may initially seem that MSCs are easier to analyze than automata based finite state systems, since variables and values are abstracted away. It turns out that this

* The results were partly supported by Bell Labs and DIMACS.

is not the case: the semantics of MSCs is based on a partial order between its events (in comparison with the total order model in the traditional interleaving semantics). Further, it does not assume any bound on the capacity of its message queues. In its general form, the MSC notation allows infinite computations by using MSC graphs, in which each graph node includes an MSC.

In this paper we study two decision problems for MSC graphs, detecting race conditions and verifying confluence. For race conditions there is a quadratic algorithm for plain (finite) MSCs, which has been implemented e.g. in the tool uBET, [1,7]. A variant of the confluence problem has been considered in [2]. Both the specification and the execution sequences of MSC graphs are captured by the notion of rational trace language, which corresponds to the closure of regular languages under partial commutations. This easily yields the undecidability of both questions. However, we are interested in reasonable restrictions of MSC graphs which guarantee the decidability of verification tasks. A main result of trace theory [9,10,13] states that loop-connected automata (or star-connected regular expressions) are equivalent to regular languages closed under partial commutations. (For asymmetric partial commutations only the inclusion from left to right holds, [3]). This is a very natural restriction for protocols specified by MSC graphs, too. It simply means that we disallow global synchronization (needed for disconnected components) and unbounded message sequences in one direction, only (without acknowledgment). This directly leads to considering decision problems on rational trace languages specified by loop-connected automata. Surprisingly, this computational aspect of rational trace languages has deserved little attention until now. We show for example that the universality problem for this class is EXPSPACE-complete. The same complexity bound follows for both MSC problems. Furthermore, we show that the connectivity property for automata is co-NP-complete.

2 Preliminaries

We first recall the notion of Mazurkiewicz (semi-) traces, [5,8]. An *independence alphabet* is a pair (A, I), where A is an alphabet endowed by an irreflexive relation $I \subseteq A \times A$, called *independence relation* (or *commutation relation*). Note that we do not assume that I is symmetric. With a given independence alphabet (A, I) we associate a rewriting relation $\Rightarrow_I$ given as the reflexive, transitive closure of $\rightarrow_I$, where $xaby \rightarrow_I xbay$ for any contexts $x, y \in A^*$ and $(a, b) \in I$. Let $[u]_I \subseteq A^*$ be the set of all v with $u \Rightarrow_I v$, for $u \in A^*$, then $[u]_I$ is called a *semi-trace* over the independence alphabet (A, I). Let $D = A \times A \setminus I$ denote the complementary (dependence) relation. A semi-trace $[u_1 \cdots u_m]_I$, $u_i \in A$, can be also viewed as a poset $(\{1, \ldots m\}, \prec)$ with $i \prec j$ whenever there is some dependence path $i = i_1 < \cdots < i_l = j$, i.e., $(u_{i_k}, u_{i_l}) \in D$ for all $k < l$. The set of semi-traces defines with the concatenation $[u]_I[v]_I = [uv]_I$ the *monoid of semi-traces* $\mathbb{M}(A, I)$. For any set $L \subseteq A^*$ we define the *I-closure of L* by $[L]_I = \cup_{u \in L}[u]_I$. A language $[L]_I$ with L regular is called *rational semi-trace language*. For a word or trace t let $\mathrm{alph}(t)$ denote the set of letters occurring in

t. The *universality problem* for rational languages over the independence relation (A, I) is the question whether $[L]_I = A^*$, for regular languages $L \subseteq A^*$.

By EXPSPACE we mean the complexity class DSPACE($2^{n^{O(1)}}$).

Definition 1. *A* message sequence chart *(MSC)* $M = (E, <, \mathcal{P}, \ell, S, R, \mathcal{M})$ *is given by a poset* $(E, <)$ *of events, a set* $\mathcal{P}$ *of processes, and a mapping* $\ell : E \to \mathcal{P}$ *that associates each event with a process (location). For each process* P *the set* $\ell^{-1}(P)$ *is totally ordered by* $<_P$*. The event set is partitioned as* $E = S \cup R$*, where* S *(*R*, resp.) is the set of send (receive, resp.) events. Furthermore,* $\mathcal{M} \subseteq S \times R$ *is the graph of a bijective mapping, relating every send with a unique receive, and conversely.*

Let $e <_c f$ *for every pair* $(e, f) \in \mathcal{M}$*. It is required that the relation* $<_c \cup \bigcup_{P \in \mathcal{P}} <_P$ *is acyclic. Then* $<$ *is the partial order induced by* $<_c \cup \bigcup_{P \in \mathcal{P}} <_P$*.*

The partial order $<$ is called *visual order* and is defined according to the syntactical representation of the chart (e.g. represented according to the standard syntax ITU-Z 120).

In general, the visual order provides more ordering between its events than intended by the designer. For example, in the visual order, the events of each process (represented by a vertical line) are totally ordered, including messages received by a process from different processes. However, enforcing such linear ordering between receive events is in general not the intended semantics of the system. To make this distinction, we associate with every chart a causal structure by means of a given semantics, which depends on the system architecture. Formally, the *causal structure* associated with a chart M is given as $\mathrm{tr}(M) = (E, \prec, \mathcal{P}, \ell, S, R, \mathcal{M})$, where $\prec \subseteq E \times E$ is a partial order called *causal order*. The causal order $\prec$ is defined as the partial order induced by an acyclic relation $\prec$, denoted *precedence relation*. The precedence relation is defined by a set of rules that state which pairs of events ordered by the visual order also belong to the causal order. We give below the set of rules corresponding to an architecture where the communication is asynchronous and first-in-first-out (*fifo*). For this semantics we only consider charts where the visual order satisfies for any events e, f, e', f' and processes P, P':

$$e <_c f\,, \quad e' <_c f'\,, \quad e <_P e'\,, \quad \ell(f) = \ell(f') = P' \quad \Longrightarrow \quad f <_{P'} f'\,.$$

Let $M = (E, <, \mathcal{P}, \ell, S, R, \mathcal{M})$ be a chart and let $e \prec f$ according to the *fifo semantics* if one of the following conditions holds:

- A send and some event on the same process:

$$\{e, f\} \cap S \neq \emptyset \text{ and } e <_P f \text{ for some process } P\,.$$

- A message pair: $e <_c f$, i.e., $(e, f) \in \mathcal{M}$.
- Messages ordered by the fifo queue:

$$\{e, f\} \subseteq R\,,\ e <_P f \text{ for some process } P \text{ and}$$
$$\exists e',\ f'\ (e' <_c e,\ f' <_c f \text{ and } e' <_{P'} f' \text{ for some process } P')\,.$$

Infinite behaviours can be specified using MSC graphs (or alternatively, hierarchical MSC graphs, HMSC).

Definition 2. *An* MSC graph $M = (S, \to, s_0, c, \mathcal{P})$ *is given as a finite, directed graph* $(S, \to, s_0)$ *with source state* $s_0 \in S$ *and nodes labelled by the mapping* c*, assigning to each state* s *a finite chart* $c(s)$ *over* $\mathcal{P}$*.*

Given two charts $M_i = (E_i, \leq_i, \mathcal{P}, \ell_i, S_i, R_i, \mathcal{M}_i)$ over a common process set $\mathcal{P}$ let the concatenation of M_1, M_2 be the chart $M_1 M_2 = (E_1 \dot\cup E_2, <, \mathcal{P}, \ell_1 \dot\cup \ell_2, S_1 \dot\cup S_2, R_1 \dot\cup R_2, \mathcal{M}_1 \dot\cup \mathcal{M}_2)$, where $< = <_1 \cup <_2 \cup \bigcup_{P \in \mathcal{P}} \ell_1^{-1}(P) \times \ell_2^{-1}(P)$. The infinite concatenation $M_1 M_2 \cdots$ is defined correspondingly. The concatenation of the associated causal structures is defined as $\mathrm{tr}(M_1 M_2 \cdots) = \mathrm{tr}(M_1)\mathrm{tr}(M_2) \cdots$.

Each path $(s_1, s_2, \ldots)$ in an MSC graph M defines a (possibly infinite) MSC by concatenation, $c(s_1)c(s_2)\ldots$. A maximal path is simply a path starting with the source and having no proper extension in M. We denote the causal structure associated with a path χ by $\mathrm{tr}(\chi)$.

Both partial orders of MSCs, the visual and the causal order (under the fifo semantics), correspond exactly to semi-traces. With each set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of processes, we associate a set $A = \{s_{ij}, r_{ij} \mid 1 \leq i \neq j \leq n\}$ of actions. Letters in A express the type (send/receive) and the location of each event e, together with the location of the event f such that $(e, f) \in \mathcal{M}$ or $(f, e) \in \mathcal{M}$. Let $(e, f) \in \mathcal{M}$ be a message from P_i to P_j, i.e., $e \in S$, $f \in R$, $\ell(e) = P_i$ and $\ell(f) = P_j$. We define a labelling $\lambda : E \to A$ by letting $\lambda(e) = s_{ij}$ and $\lambda(f) = r_{ij}$, respectively. For a chart M with event set E let $\mathrm{msg}(M) = \{\lambda(e) \mid e \in E\}$.

Consider a chart M and its causal structure $\mathrm{tr}(M)$. The *visual order* is easily seen to be induced by the dependence alphabet (A, D_v) given by the dependence relation

$$D_v = \{(s_{ij}, r_{ij}) \mid i, j\} \cup \{(s_{ij}, s_{ik}), (s_{ij}, r_{ki}), (r_{ki}, s_{ij}), (r_{ik}, r_{jk}) \mid i, j, k\}\,.$$

The *causal order* under the fifo semantics is induced by the dependence alphabet (A, D_c):

$$D_c = \{(s_{ij}, r_{ij}) \mid i, j\} \cup \{(s_{ij}, s_{ik}), (s_{ij}, r_{ki}), (r_{ki}, s_{ij}), (r_{ij}, r_{ij}) \mid i, j, k\}\,.$$

Note that the reflexivity of the dependence relation D_c (for receives) is provided by the fifo rule. Moreover, we have $D_c \subset D_v$, reflecting that the causal order is less restrictive than the visual order. We denote in the following by I_c, I_v the complementary relations $A \times A \setminus D_c$, $A \times A \setminus D_v$.

In the trace setting, an MSC graph $M = (S, \to, s_0, c, \mathcal{P})$ is just a transition system with nodes labeled by some partial order over the set of events. An equivalent (non-deterministic) edge-labeled transition system over A can be easily defined. We choose for each state $s \in S$ some linearization $\alpha \in A^*$ of the (visual order) $c(s)$ such that $[\alpha]_{I_v} = \lambda(c(s))$, i.e., α is a representative of the semi-trace $\lambda(c(s))$. The language $L(\mathcal{A})$ in the proposition below is defined as the language of finite and infinite words accepted by $\mathcal{A}$ (i.e., all maximal paths in $\mathcal{A}$ due to $F = Q$).

Proposition 1. *Let $M = (S, \to, s_0, c, \mathcal{P})$ be an MSC graph over the process set $\mathcal{P} = \{P_1, \ldots, P_n\}$. Let $A = \{s_{ij}, r_{ij} \mid 1 \leq i \neq j \leq n\}$ and let $D_c, D_v \subseteq A \times A$ be defined as above. Then an automaton $\mathcal{A}_M = (Q, A, \delta, q_0, Q)$ with transition relation $\delta \subseteq Q \times A \times Q$ and initial state $q_0 \in Q$ can be constructed such that:*

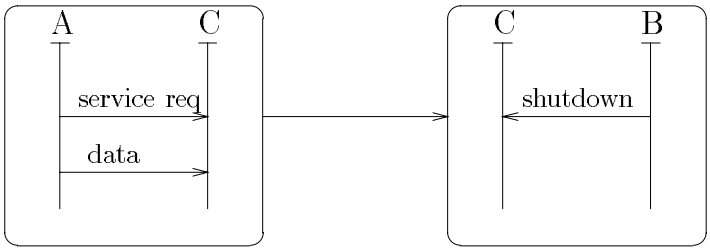
i) $[L(\mathcal{A}_M)]_{I_v} = \{\lambda(c(\xi)) \mid \xi$ *is a maximal path in* $M\}$.
ii) $[L(\mathcal{A}_M)]_{I_c} = \{\lambda(\mathrm{tr}(\xi)) \mid \xi$ *is a maximal path in* $M\}$.

Moreover, $|\mathcal{A}_M| \in O(m)$*, where* m *is the number of events and processes occurring in* M.

3 Detecting Race Conditions and Testing Confluence

3.1 Race conditions

An MSC M means just a specification of some scenario. By definition, its causal structure $\mathrm{tr}(M)$ possibly allows more executions than the specification. In this case we speak about *race conditions*. Clearly, race conditions should be avoided and verifying the absence of races belongs to the correctness check of the design. The next figure shows races on the process C (between the receive events from A resp. B):



Definition 3. *The set of all linearizations of the visual order of an MSC M is denoted by* $\mathrm{lin}(M)$*. The set of linearizations* $\mathrm{lin}(M) \subseteq E^\infty$ *of an MSC graph M is the collection of all linearizations of maximal paths in M.*

Definition 4. *The set of* executions *of an MSC M, denoted* $\mathrm{exec}(M)$*, is the set of all linearizations of the causal order of* $\mathrm{tr}(M)$*. The set of executions* $\mathrm{exec}(M) \subseteq E^\infty$ *of an MSC graph is the collection of all executions of maximal paths in M.*

By definition we have $\mathrm{lin}(M) \subseteq \mathrm{exec}(M)$ for every MSC M. If the inclusion is strict, then the specification is in some sense incomplete and it possibly allows some undesired behavior (*race conditions*).

Definition 5 (Race problem). *An MSC graph M contains race conditions if* $\mathrm{lin}(M)$ *is strictly included in the set of executions* $\mathrm{exec}(M)$ *of M.*

The problem of checking whether an MSC graph contains races is naturally related to a question on closures of regular languages. This closure problem will be shown below to be undecidable. Later, we will obtain that the race problem itself is undecidable.

Definition 6 (Closure problem over A, I_1, I_2). *Let $I_1, I_2 \subseteq A \times A$ be two commutation relations on A, with $I_1 \subseteq I_2$. Given a regular language $L \subseteq A^*$, we want to decide whether $[L]_{I_2} = [L_1]_{I_1}$.*

Proposition 2. *The closure problem over A, I_1, I_2, with $(A, I_1) = a \text{ —— } c \text{ —— } b$ and $I_2 = A^2 \setminus id_A$ is undecidable.*

Proof. By a reduction from the universality problem for rational trace languages over (A, I_1), which was shown to be undecidable in [14].

Let $L \subseteq A^*$ be regular. Then $[L]_{I_1} = A^*$ if and only if $[L]_{I_2} = A^*$ and $[L]_{I_1} = [L]_{I_2}$. Since I_2 is the total commutation relation, $[L]_{I_2} = A^*$ is an equality test between two semilinear sets, hence decidable by [6]. Thus, $[L]_{I_1} = [L]_{I_2}$ is undecidable.

Remark 1. We can state the above result more precisely. The universality problem for rational trace languages over (A, I) is decidable if and only if $I \cup \mathrm{id}_A$ is transitive, [14]. Moreover, rational sets over a transitive independence relation form a boolean algebra and are recognized by a particular kind of automata. Using this characterization we can show that the closure problem over A, I_1, I_2 with $I_2 = A^2 \setminus \mathrm{id}_A$ is decidable if and only if I_1 is transitive.

Theorem 1. *The race problem for MSC graphs under the fifo semantics is undecidable.*

Proof. (sketch) The proof is a simple application of the above result, using the messages depicted in Figure 1. Note that the receive events a, b are ordered in the visual order, but not in the causal order. Between a, c and b, c there is no visual order.

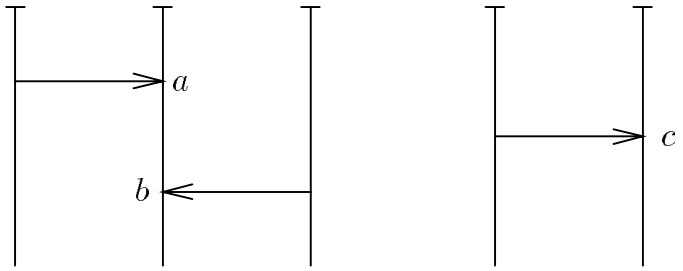
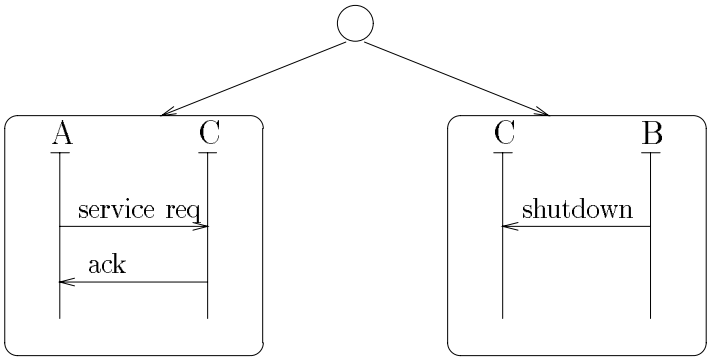


Fig. 1. Messages used in Thm. 1.

3.2 Checking confluence

An MSC graph uses non-deterministic branching to express alternative behaviors. This might be a problem for the implementation, since each computation should correspond to a single flow of control. A possible solution is to synchronize processes. However, global synchronization is not desirable and it can be avoided if we require that the execution of the MSC graph is confluent, as defined below. Intuitively, confluence corresponds to the following property: suppose that two finite prefixes of computations are consistent in the sense that they *can* be completed into a single computation. Then, there exists in the system's description a complete computation that indeed includes both prefixes. For example, consider the case where two different protocols are initiated by different processes, e.g., in the first protocol (but not in the second), process P_1 sends a message to P_2, and in the second protocol (but not in the first), process P_3 sends a message to P_4. Since the two protocols, so far, are compatible, then confluence imposes that there is an execution that contains both messages, possibly leading to the need to resolve the conflict between future behavior of the protocol. Failing to have the latter execution would mean that there is some additional way that the processes might have learned which of the two protocols to execute (perhaps by presetting some values). Thus, although this might not be a mistake, the confluence test that we discuss in this section is intended to alert the designer of such a possible problem. The next figure shows a confluent MSC graph: the possible executions have no upper bound. Omitting the acknowledgement from C to A would make the graph non-confluent, since there is no common extension of both executions.



We denote throughout this section the prefix order on causal structures (semi-traces, resp.) by $\sqsubseteq$. That is, let $\mathrm{tr}(M) \sqsubseteq \mathrm{tr}(N)$ if $\mathrm{tr}(N) = \mathrm{tr}(M)\mathrm{tr}(M')$ for some chart M'. The least upper bound of $\mathrm{tr}(M)$ and $\mathrm{tr}(N)$ with respect to the prefix order is denoted $\mathrm{tr}(M) \sqcup \mathrm{tr}(N)$ (if it exists).

Definition 7 (Confluence). *An MSC graph $M = (S, \to, s_0, c, \mathcal{P})$ is called* confluent *if for any maximal paths $\alpha, \beta \in S^\infty$ in M such that $\mathrm{tr}(\alpha) \sqcup \mathrm{tr}(\beta)$ exists, there is some maximal path $\gamma \in S^\infty$ in M such that $\mathrm{tr}(\alpha) \sqsubseteq \mathrm{tr}(\gamma)$ and $\mathrm{tr}(\beta) \sqsubseteq \mathrm{tr}(\gamma)$.*

Let us first formulate the confluence problem in terms of partial commutations. Consider a regular language $L \subseteq \Sigma^*$ and a commutation relation $I \subseteq \Sigma \times \Sigma$. Then we denote L as *confluent* over (Σ, I) if for any $x, y \in L$ such that $[x]_I \sqcup [y]_I$ exists, there is some $z \in L$ with $[x]_I \sqsubseteq [z]_I$, $[y]_I \sqsubseteq [z]_I$. The next proposition states that the confluence problem for partial commutations is in general undecidable.

Proposition 3. *Let (A, I) be defined by $A = \{a, b, c, d\}$ and the dependence relation $(A, A \times A \setminus I) = a \text{ —— } b \quad c \text{ —— } d$. Then it is undecidable whether a given regular language is confluent over (A, I).*

Proof. We use again the undecidability of the universality problem, i.e., the question whether $[L]_J = B^*$, where $B = \{a, b, c\}$, $L \subseteq B^*$ is regular and $J = \{(a, c), (c, a), (b, c), (c, b)\}$.

We first define an encoding $h : B^* \to B^*$ by $h(a) = ab$, $h(b) = ba$ and $h(c) = c$. Let $K = h(L)b^2d^2 + c^*d^2 + h(B^*)b^2d$. Suppose first that $[L]_J = B^*$, then $[K]_I = h(B^*)(b^2d^2 + b^2d) + c^*d^2$ is confluent. For the converse let K be confluent and consider $u \in (ab + ba)^*$, $v \in c^*$. Then $\{ub^2\, vd, vd^2\} \subseteq [K]_I$. Moreover, if $[u'b^2\, v'd^2]_I \in [h(L)b^2\, d^2]_I$ is an upper bound of both $[ub^2\, vd]_I$ and $[vd^2]_I$, where $u' \in \{a, b\}^*$, $v' \in c^*$, then $u = u'$ and $v = v'$. This implies $[h(L)]_I = [(ab + ba)^*c^*]_I$, hence $[L]_J = \{a, b, c\}^*$.

Similarly to Thm. 1 we also obtain:

Theorem 2. *The confluence problem for MSC graphs under the fifo semantics is undecidable.*

4 Loop-connected Automata and Restricted MSC Graphs

Basic decision problems for MSC graphs as detecting races, checking confluence or matching without gaps (a kind of model-checking) [12] are undecidable due to the connection to *rational semi-trace languages*, i.e., languages of the form $[L]_I$, where $L \subseteq A^*$ is regular and $I \subseteq A \times A$ is an independence relation. This class of languages strictly includes the class of regular, I-closed languages and contains also non-regular languages, in general (cf. Thm. 3 below). However, the expressiveness of rational semi-trace languages is based on the iteration of non-connected expressions, i.e., subexpressions K^* where $\mathrm{alph}(K) = \cup_{u \in K} \mathrm{alph}(u)$ is not a strongly connected subgraph of the dependence relation $(A, A \times A \setminus I)$. If we disallow this possibility, that is, if we restrict the Kleene iteration to strongly connected expressions, then rational semi-trace languages remain regular and enjoy all good (decidability) properties of regular languages. We consider in this section loop-connected automata $\mathcal{A}$ as defined below. We give an exponential upper bound on the size of a nondeterministic automaton for $[L(\mathcal{A})]_I$ and we show that the universality problem for connected rational languages is EXPSPACE-complete. Using similar arguments, we can also show that the problem of the

nonempty intersection of connected rational languages is PSPACE-complete, i.e., the question whether $[L(\mathcal{A})]_I \cap [L(\mathcal{B})]_I \neq \emptyset$, [11]. These results show that partial commutations in general yield an exponential blow-up in the complexity of finite automata.

Definition 8. *Let (A, I) be an independence alphabet, $D = A \times A \setminus I$ and consider a finite automaton $\mathcal{A} = (Q, A, \delta, J, F)$. We denote $\mathcal{A}$ as* loop-connected over (A, I) *if for every loop $(q_0, a_0, \ldots, a_{k-1}, q_k = q_0)$, $q_i \in Q$, $a_i \in A$, the set $alph(a_0 \cdots a_{k-1}) \subseteq A$ of letters occurring in the loop induces a strongly connected subgraph of (A, D).*

Theorem 3 ([4]). *Let (A, I) be an independence alphabet and $\mathcal{A}$ a loop-connected automaton over (A, I). Then $[L(\mathcal{A})]_I$ is a regular language.*

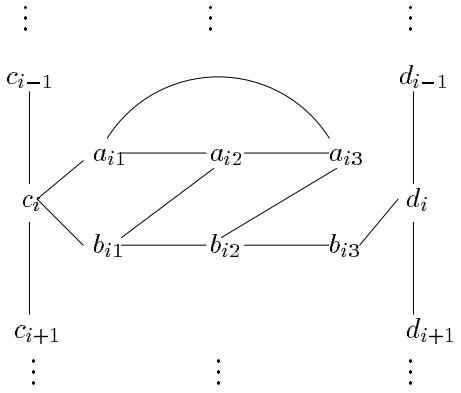
Before considering the complexity of testing whether an automaton is loop-connected (with (A, I) part of the input) let us note that it does not suffice to check the property on simple loops, only:

Example 1. Let $(\Sigma, I) = b \text{ —— } a \text{ —— } c$. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1, q_2, q_3\}$, $\delta = \{(q_0, a, q_1), (q_1, a, q_0), (q_1, b, q_2), (q_2, b, q_1), (q_0, c, q_3), (q_3, c, q_0)\}$, $F = \{q_0\}$. Every simple loop of $\mathcal{A}$ is connected. However, $[L(\mathcal{A})]_I$ is not regular. Note that the intersection of $[L(\mathcal{A})]_I$ with the regular language $[a^*(bbcc)^*]_I$ is the I-closure of $\{a^{2n}(bbcc)^m \mid n \geq m\}$, which is easily seen to be not regular.

Proposition 4. *The following problem is co-NP-complete:*
Instance: *An independence alphabet (Σ, I) and a finite automaton $\mathcal{A}$.*
Question: *Is $\mathcal{A}$ loop-connected over (Σ, I)?*

Proof. For the co-NP-hardness, assume that $F = C_1 \wedge \cdots \wedge C_m$ is a boolean formula in CNF over the variable set $\{x_1, \ldots, x_n\}$. Moreover, let $C_j = l_{j1} \vee l_{j2} \vee l_{j3}$, with l_{jk} literals.

Let $\Sigma = \{a_{jk}, b_{jk} \mid 1 \leq j \leq m, k = 1, 2, 3\} \cup \{c_j, d_j \mid 1 \leq j \leq m\}$. The symmetric *dependence relation* is given by the following picture:



There are two dependence chains above, $c_1 \text{ —— } c_2 \text{ —— } \cdots \text{ —— } c_m$ and $d_1 \text{ —— } d_2 \text{ —— } \cdots \text{ —— } d_m$, and the dependence relation restricted to any other subset $\{c_j, d_j, a_{jk}, b_{jk} \mid k = 1,2,3\}$ is analogous to the one depicted for $j = i$. The point is that the c-chain and the d-chain are connected only if we have all letters b_{j1}, b_{j2}, b_{j3} for some j.

The automaton $\mathcal{A}$ associated with F has for simplicity edges labeled by words. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, \{q_0\})$ with $Q = \{q_i \mid 0 \leq i \leq n\}$, and transition relation $\delta = \{(q_i, P_i, q_{i+1}), (q_i, N_i, q_{i+1}) \mid 0 \leq i \leq n\}$ (by convention, $q_{n+1} = q_0$). Hereby, $P_0 = N_0 = c_1 \cdots c_m d_1 \cdots d_m$ and $P_i = p_1 \cdots p_m$, $N_i = q_1 \cdots q_m$ where $p_j = a_{jk}$, if $x_i = l_{jk}$ and $q_j = b_{jk}$, if $\overline{x_i} = l_{jk}$, otherwise $p_j = q_j = \epsilon$. The idea is that the letters a_{jk} correspond to true literals, whereas b_{jk} correspond to false literals. Every (simple) loop from q_0 to q_n and back to q_0 corresponds to an assignment of the variables, the edge labeled P_i means that x_i is true, whereas N_i means that x_i is false. As noted previously, if some assignment σ of F is satisfying, then it can be easily checked that $\mathcal{A}$ is not loop-connected, using the loop corresponding to σ. Conversely, it can be easily checked that every non-satisfying assignment means that the corresponding loop is connected.

Remark 2. Our proof shows that even testing loop-connectedness for simple loops is co-NP-complete.

4.1 Decision problems for loop-connected automata

For our previous undecidability results on MSC graphs (Thms. 1, 2) we used the universality problem for rational trace languages, i.e., the question whether $[L]_I = A^*$, where $L \subseteq A^*$ is regular. If L is given by a loop-connected automaton $\mathcal{A}$ over (A, I), then by Thm. 3 we get decidability. In this section, we exhibit a precise complexity characterization by showing that the universality problem for rational semi-trace languages (i.e., I is not necessarily symmetric) is EXPSPACE-complete. This leads later to the same complexity bounds for detecting races and checking confluence.

The algorithm for the universality problem is based on an exponential bound on the size of the automaton accepting $[L(\mathcal{A})]_I$. The key lemma is a classical automaton construction. Note that an exponential increase of the number of states is unavoidable. Consider for example the finite language $L_n = (a\bar{a} + b\bar{b})^n$ over $A = \{a, b, \bar{a}, \bar{b}\}$, and $(A, A \times A \setminus I) = a \text{ —— } b \quad \bar{a} \text{ —— } \bar{b}$. Then $[L_n]_I \cap \{a,b\}^*\{\bar{a},\bar{b}\}^* = \{u\bar{u} \mid u \in \{a,b\}^n\}$. The latter language is known to require at least 2^n states for a minimal NFA recognizing it.

Lemma 1. *Let (A, I) be an independence alphabet and consider a loop-connected automaton $\mathcal{A} = (Q, A, \delta, q_0, F)$. Let $n = |Q|$ denote the number of states of $\mathcal{A}$. Consider some word $v \in A^*$ such that $\delta(q, v) \neq \emptyset$ for some state q. Moreover, assume that $v = t_0 u_1 \cdots t_{k-1} u_k t_k$ for some $u_i \neq 1$, $t_j \neq 1$, $1 \leq i \leq k$, $1 \leq j < k$ and $(t_i, u_j) \in I$ for all $i < j$. Then we have $k \leq (n-1)(|A|+1)$.*

Proof. Let A_j denote for each $0 \leq j < k$ the alphabet $A_j = I(u_{j+1} \cdots u_k)$. Then $A_0 \subseteq A_1 \cdots \subseteq A_{k-1} \subseteq A$. Suppose by contradiction that $k > (n-1)(|A|+1)$.

Then we obtain some indices $0 \leq i < j < k$ such that $A_i = A_{i+1} = \cdots = A_j$ and $j - i \geq n - 1$. Thus, we have $(t_i \cdots t_j, u_{i+1} \cdots u_{j+1}) \in I$. Let us also fix some computation ρ of $\mathcal{A}$ on v, i.e., a path from some state q to a state q', which is labeled by v. Let q_l denote the state reached on ρ after reading $t_0 u_1 \cdots t_{l-1} u_l$. With $j - i \geq n - 1$ we obtain some $i \leq l < m < j$ such that $q_l = q_m$. Therefore, $t_l u_{l+1} \cdots t_{m-1} u_m$ is the labelling of a loop of $\mathcal{A}$. However, $(t_l \cdots t_{m-1}, u_{l+1} \cdots u_m) \in I$, thus $\mathcal{A}$ is not loop-connected, contradiction.

Proposition 5. *Let (A, I) be an independence alphabet and $\mathcal{A} = (Q, A, \delta, q_0, F)$ a loop-connected automaton with $n = |Q|$. Then a finite automaton $\mathcal{B}$ with $(n^2 \cdot 2^{|A|})^{n|A|+n+1}$ states exists such that $L(\mathcal{B}) = [L(\mathcal{A})]_I$.*

Proof. Let $v \in L(\mathcal{A})$ be as in Lem. 1, i.e. $v = t_0 u_1 \cdots t_{k-1} u_k t_k$, with $(t_i, u_j) \in I$ for all $i < j$. Moreover, $u_i \neq 1$ for all i, $t_j \neq 1$ for all $0 < j < k$ and $k \leq (n-1)(|A|+1)$. We consider some input $v' \in [v]_I$ for $\mathcal{B}$ such that $v' = ww'$, where $w \in [u_1 \cdots u_k]_I$, $w' \in [t_0 \cdots t_k]_I$. Let us fix some computation ρ of $\mathcal{A}$ on v such that ρ starts in the initial state q_0. We denote by p_i (q_{i+1}, resp.), the state reached in ρ after reading $t_0 u_1 \cdots t_i$ ($t_0 u_1 \cdots t_i u_{i+1}$, resp.). Each state of $\mathcal{B}$ will encode the states $q_0, p_0, q_1, \ldots, q_k, p_k$ and the alphabets $I(u_1), \ldots, I(u_k)$. A state of this form is reachable in $\mathcal{B}$ by a prefix w of the input v' only if w satisfies $w \in [u'_1 \cdots u'_k]_I$ for some u'_i with $I(u'_i) = I(u_i)$ and $q_i \in \delta(p_{i-1}, u'_i)$.

Let $a \in A$ and assume that wa is a prefix of the input v'. Suppose that this occurrence of the letter a corresponds to some factorization $t_i = xay$ in v, where $x, y \in A^*$. Then we have $a \in I(u_{i+1} \cdots u_k)$. There are three possible cases. First, assume by symmetry that $x = 1$ and $y \neq 1$, hence we have for example $v = t_0 u_1 \cdots t_{i-1}(u_i\, a) y\, u_{i+1} \cdots u_k t_k$, if $i \neq 0$. In this case $\mathcal{B}$ replaces $I(u_i)$ by $I(u_i a)$ and q_i by some state $q'_i \in \delta(q_i, a)$, if $i \neq 0$. The case where $i = 0$ requires a bit more care. We guess a state q with $q \in \delta(q_0, a)$ and let the new list of states be $(q_0, q_0, q, p_0, \ldots, p_k)$. Second, suppose that $x \neq 1$ and $y \neq 1$, hence for $k < (n-1)(|A|+1)$ we have

$$v = t_0 u_1 \cdots t_{i-1} u_i\, xay\, u_{i+1} \cdots u_k t_k = t'_0 z'_1 \cdots t'_{i-1} z'_i \cdots z'_{k+1} t'_{k+1}\,,$$

where

$$t'_j = \begin{cases} t_j & \text{if } j < i \\ x & \text{if } j = i \\ y & \text{if } j = i+1 \\ t_{j-1} & \text{if } j > i+1 \end{cases}, \; z'_j = \begin{cases} u_j & \text{if } j \leq i \\ a & \text{if } j = i+1 \\ u_{j-1} & \text{if } j > i+1\,. \end{cases}$$

In this case $\mathcal{B}$ will guess the states $p, q \in Q$ such that p is reached after reading $t_0 u_1 \cdots u_i x$, whereas $q \in \delta(p, a)$. The new state of $\mathcal{B}$ will encode the list of states $(q_0, p_0, q_1, \ldots, q_i, p, q, p_i, \ldots, q_k, p_k)$ and the list of alphabets $(I(u_1), \ldots, I(u_i), I(a), \ldots, I(u_k))$.

Finally, let us consider $x = y = 1$. Then $p_i \in \delta(q_i, a)$. The new state of $\mathcal{B}$ encodes the states $(q_0, p_0, q_1, \ldots, p_{i-1}, q_{i+1}, \ldots, q_k, p_k)$, if $i \neq 0$, and the alphabets $(I(u_1), \ldots, I(u_{i-1}), I(u_i u_{i+1} a), \ldots, I(u_k))$. The case where $i = 0$ is dealt with similarly.

A final state of $\mathcal{B}$ is given by $k = 1$ and the states (q_0, q_0, p_1, p_1), with $p_1 \in F$. An initial state of $\mathcal{B}$ is given by the state list (q_0, q_0) and the alphabet list (A).

Remark 3. We stated the above upper bound only for finite strings. However, the same arguments also apply to nondeterministic Büchi automata.

Using Prop. 5 we obtain immediately:

Proposition 6. *The following problem is in EXPSPACE:*
Instance: *An independence alphabet (A, I) and a loop-connected automaton $\mathcal{A}$ over (A, I).*
Question: *Is $[L(\mathcal{A})]_I = A^*$?*

Proof. Given a loop-connected automaton $\mathcal{A}$, we guess some word $w \in A^*$ and verify that $w \notin [L(\mathcal{A})]_I$. By Prop. 5 there is some automaton $\mathcal{B}$ accepting $[L(\mathcal{A})]_I$ which has $2^{O(n|A|^2)}$ states. Moreover, we can construct the states and the transition relation of $\mathcal{B}$ using space which is polynomial in n and $|A|$. Hence it suffices to guess w on-line and store the subset of states of $\mathcal{B}$ reached so far, which requires $2^{O(n|A|^2)}$ space.

4.2 Lower Bounds

We have also a matching lower bound for all problems considered previously. The proof idea is based on a construction used by Walukiewicz [15] for proving lower bounds on the temporal logic LTrL.

Proposition 7. *The universality problem for loop-connected automata is hard for EXPSPACE. This holds even if the independence alphabet is fixed to (A, I), with $A = \{a, b, c, d, \$, \#\}$ and I the least symmetric relation containing $\{a, b\} \times \{c, d\} \cup \{a, b\} \times \{\#\} \cup \{c, d\} \times \{\$\}$.*

Proof. We express the set of invalid computations of a Turing machine M on an input w by means of regular expressions. Let M be a $(2^{n^k} - 1)$-space bounded Turing machine, where $k \geq 1$ is some constant, and let w be an input to M of length n. Assume that the head of M never moves to the left of the first position of w. We denote in the following $s = 2^{n^k} - 1$. A configuration α of M is of the form $\alpha = uqv$, where q denotes the state, uv the tape contents and the head scans the first symbol of v. Let Γ denote the alphabet of configurations, with $B \in \Gamma$ denoting the blank symbol. Without restriction we may assume that

i) The unique initial configuration is $q_0\, w \underbrace{B \cdots B}_{s-n}$, and the unique accepting configuration is $q_f\, \underbrace{B \cdots B}_{s}$.
ii) For every transition $\alpha \vdash \beta$ of M, $\alpha = a_0 \cdots a_s$, $\beta = b_0 \cdots b_s$ $(a_i, b_j \in \Gamma)$, we have that each symbol b_i is determined by a_{i-1}, a_i and a_{i+1}.
iii) Every accepting computation of M has odd length.

Let $a, b, c, d, \$, \#$ be letters not in Γ. For encoding computations of M we use the independence alphabet (A, I), where $A = \{x, \bar{x} \mid x \in \{a, b, c, d\} \cup \Gamma\} \cup \{\$, \#\}$. The independence relation $I \subseteq A \times A$ is the least symmetric relation satisfying $\{(x, \bar{y}), (\$, \bar{y}), (x, \#)\} \subseteq I$, for all $x, y \in \Gamma \cup \{a, b, c, d\}$.

Consider a computation $\rho = \alpha_0 \vdash \alpha_1 \vdash \cdots \vdash \alpha_t$ of M, where t is odd. We encode ρ as in Figure 2. Hereby, each A_i ($\bar{A}_i$, resp.) has the form

$$A_i = p(0, a, b)\, \alpha_{i,0}\, p(1, a, b)\, \alpha_{i,1} \cdots p(s, a, b)\, \alpha_{i,s}\,,$$

respectively

$$\bar{A}_i = p(0, \bar{a}, \bar{b})\, \bar{\alpha}_{i,0}\, p(1, \bar{a}, \bar{b})\, \bar{\alpha}_{i,1} \cdots p(s, \bar{a}, \bar{b})\, \bar{\alpha}_{i,s}\,,$$

where $p(l, a, b)$ ($p(l, \bar{a}, \bar{b})$, resp.) is the binary representation of l using a ($\bar{a}$, resp.) as 0 and b ($\bar{b}$, resp.) as 1. Moreover, least significant bits are leftmost. The strings C_j, $\bar{C}_j$ are defined analogously, with $a, b, \bar{a}, \bar{b}$ replaced by $c, d, \bar{c}, \bar{d}$.

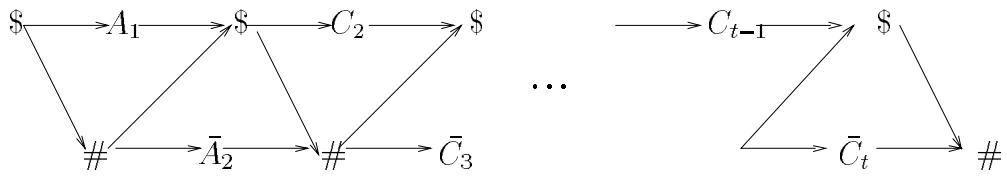


Fig. 2. Computations of M.

We define in the following a loop-connected automaton $\mathcal{A}$ over (A, I) satisfying $[L(\mathcal{A})]_I = A^*$ if and only if M has no accepting computation on w. The set $L(\mathcal{A})$ will be given as

$$L(\mathcal{A}) = N_s \cup N(a, \bar{a}) \cup N(c, \bar{c}) \cup N(\bar{a}, c) \cup N(\bar{c}, a) \cup N_{\text{init}} \cup N_{\text{fin}}\,,$$

where

i) $w \in [N_s]_I$ if and only if $[w]_I$ does not have the shape of Figure 2, with the required form for A_i, $\bar{A}_i$, C_j, $\bar{C}_j$.
ii) If $w \notin [N_s]_I$ then $w \in [N(a, \bar{a})]_I$ if and only if $\bar{A}_{i+1}$ is not the successor configuration of A_i, for some i. Similarly for $N(c, \bar{c})$ and C_j, $\bar{C}_{j+1}$.
iii) If $w \notin [N_s]_I$ then $w \in [N(\bar{a}, c)]_I$ if and only if $\bar{A}_i$ is not the same configuration as C_i, for some i. Similarly for $N(\bar{c}, a)$ and $\bar{C}_i$, A_i.
iv) If $w \notin [N_s]_I$ then $w \in [N_{\text{init}}]_I$ if and only if $[w]_I$ has no prefix from $\$\#\, p(0, a, b)\, q_0\, p(1, a, b) w_1 \cdots p(n, a, b)\, w_n\, (\{a, b\}^*\, B)^*\, \$$.
v) If $w \notin [N_s]_I$ then $w \in [N_{\text{fin}}]_I$ if and only if $[w]_I$ has no suffix from $\#p(0, \bar{c}, \bar{d})\, q_f\, (\{\bar{c}, \bar{d}\}^*\, B)^*\, \$\#$.

Let us sketch how to define loop-connected regular expressions for the above languages. First, for i) it is not difficult to express that $[w]_I$ does not have the

shape of Figure 2, for alphabetical reasons. Hence, it suffices to express that some A_i ($\bar{A}_i$, C_j, $\bar{C}_j$, resp.) has not the required form w.r.t. the counters. This is precisely the case when

i) A_i does not start with $a^{n^k}\, g$ for some $g \in \Gamma$.
ii) A_i does not end with $b^{n^k}\, g$ for some $g \in \Gamma$.
iii) $A_i \notin (\{a,b\}^{n^k}\Gamma)^*$.
iv) A_i contains a factor $g_1\, u\, g_2\, v\, g_3$, with $g_i \in \Gamma$, $u, v \in \{a,b\}^{n^k}$, but $v \neq u+1$. This condition can be expressed by guessing some position $0 \leq i < n^k$ such that either u starts with b^{i-1}, but the i-th symbol of v is the same as the i-th symbol of a, or dually, u starts with some prefix in $\{a,b\}^{i-1} \setminus b^{i-1}$, but the i-th symbol of v differs from the i-th symbol of u.

For the languages described in ii), iii) we use the counters for identifying corresponding positions. For example, we require for $N(\bar{a}, c)$ that a factor from $(a\bar{a} + b\bar{b})^{n^k}\, g_1 \bar{g}_2$ occurs, with $g_1, g_2 \in \Gamma$, $g_1 \neq g_2$. Note that the construction can be realized by iterating only connected expressions.

Using encodings we also obtain the last claim of the theorem.

Theorem 4. *The universality problem for loop-connected automata is complete for EXPSPACE (even if the independence relation is fixed).*

4.3 The race problem and the confluence problem revisited

The classes of MSC graphs considered in this section are defined by restricting loops to be strongly connected w.r.t. the visual (causal, resp.) order. This restriction is natural for two reasons. First, it means that we disallow global synchronization. (That is, iterating in parallel disjoint sequences of messages would require some global counting mechanism which is not natural in a concurrent setting). Second, it corresponds to finite state systems. That is, we disallow e.g. unbounded computations where P repeats sending a message to P' (without waiting for an answer).

Recall that $A = \{s_{ij}, r_{ij} \mid 1 \leq i \neq j \leq n\}$ denotes the set of actions, and that the visual and causal order, respectively, are induced by the independence relations I_v, I_c (dependence relations D_v, D_c).

Definition 9. *Let $M = (S, \rightarrow, s_0, c, \mathcal{P})$ be an MSC graph. We say that M is* locally synchronized *with respect to the visual order (causal order, resp.) if for every loop $(s_1, s_2, \cdots, s_k = s_1)$ in M, the set $\cup_{i=1}^{k} \mathrm{msg}(c(s_i)) \subseteq A$ induces a strongly connected subgraph of (A, D_v) ((A, D_c), resp.).*

Using the connection between MSC graphs and rational semi-trace languages (Prop. 1) we obtain also:

Proposition 8. *The race problem under the fifo asynchronous semantics over MSC graphs which are locally synchronized w.r.t. the visual order is decidable in EXPSPACE.*

Proof. By Prop. 1 it suffices to consider two independence alphabets (A, I_1), (A, I_2) with $I_1 \subseteq I_2$, and a loop-connected automaton $\mathcal{A}$ (w.r.t. (A, I_1)). It is not hard to see that $[L(\mathcal{A})]_{I_1} \neq [L(\mathcal{A})]_{I_2}$ if and only if some words $u, v \in A^*$ and $(a, b) \in I_2$ exist such that $uabv \in [L(\mathcal{A})]_{I_1}$, but $ubav \notin [L(\mathcal{A})]_{I_1}$. By Prop. 5 the automaton $\mathcal{B}$ accepting $[L(\mathcal{A})]_{I_1}$ has $2^{O(n|A|^2)}$ states, hence it is possible to guess u, v and simultaneously store the sets of states of $\mathcal{B}$ reached on $uabv$, resp. $ubav$, using exponential space.

Proposition 9. *The confluence problem under the fifo asynchronous semantics over MSC graphs which are locally synchronized w.r.t. the causal order is in EXPSPACE.*

Proof. By Prop. 1 it suffices to consider an independence alphabet (A, I) and a loop-connected automaton $\mathcal{A}$ (w.r.t. (A, I)). Then $\mathcal{A}$ is not confluent if and only if some words $p, q, r \in A^*$ exist such that $\{pq, pr\} \subseteq [L(\mathcal{A})]_I$, $\mathrm{alph}(q) \times \mathrm{alph}(r) \subseteq I$, but $pqrs \notin [L(\mathcal{A})]_I$, for any $s \in A^*$. Clearly, these conditions can be checked in exponential space using the automaton accepting $[L(\mathcal{A})]_I$.

Theorem 5. *Detecting races for MSC graphs which are locally synchronized w.r.t. the visual order is EXPSPACE-complete.*

Proof. For the hardness we use Prop. 7 and a reduction from the universality problem as given in Prop. 2. Note that the language L defined in Prop. 7 over the alphabet $A = \{a, b, c, d, \bar{a}, \bar{b}, \bar{c}, \bar{d}, \$, \#\}$ is such that $\pi(L) = A^*$ is always satisfied. Moreover, this property still holds if we encode $\{a, b, \bar{a}, \bar{b}\}$ into $\{a, b\}$ (resp. $\{c, d, \bar{c}, \bar{d}\}$ into $\{c, d\}$). Thus, $[L]_I = A^*$ is equivalent to $[L]_I = \pi(L)$, without restriction $A = \{a, b, c, d, \$, \#\}$.

We encode the alphabet A into MSCs as follows. Consider processes P_1, P_2, P_3. Let a, b be messages from processes P_a, P_b, respectively, to P_1. The messages c, d are defined similarly, from P_c, P_d to P_2. Let $\$_1$ ($\#_1$, resp.) be a message from $P_\$$ ($P_\#$, resp.) to P_3. Let $\$_2$ be a message from $P_\$$ to P_1, and $\#_2$ from $P_\#$ to P_2. Note that the receive events of $a, b, \$_2$ are visually ordered, but not causally ordered. The same holds for $c, d, \#_2$, resp. $\$_1$, $\#_1$. The details of the proof are straightforward.

Theorem 6. *The confluence problem for MSC graphs which are locally synchronized w.r.t. the causal order is EXPSPACE-complete.*

Proof. We use a modified reduction from the universality problem $[L]I = A^*$, with L, $A = \{a, b, c, d, \$, \#\}$ and I as in Prop. 7 Let e be a new letter which commutes with every letter in A. Let $h : A^* \to A^*$ be an encoding of A, defined as $h(x) = xx$. Then it is easy to check that $[L]_I = A^*$ if and only if the language

$$K = e + h(A^*)\, abcd\$\# + e\, h(L)\, abcd\$\#$$

is confluent. The alphabet A is encoded into messages as in Thm. 5, whereas e is encoded as a message between a new pair of processes.

Acknowledgment: The authors thank Volker Diekert for an improvement of the proof of Prop. 2.

References

1. R. Alur, G. H. Holzmann, and D. A. Peled. An analyzer for message sequence charts. *Software Concepts and Tools*, 17(2):70–77, 1996.
2. H. Ben-Abdallah and S. Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems (TACAS'97)*, LNCS 1217, pp. 259–274.
3. M. Clerbout. *Commutations Partielles et Familles de Langages*. Thèse, Université des Sciences et Technologies de Lille (France), 1984.
4. M. Clerbout and M. Latteux. Semi-Commutations. *Information and Computation*, 73:59–74, 1987.
5. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.
6. S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.
7. G. Holzmann, D. Peled, and M. Redberg. Design tools for requirements engineering. *Bell Labs Technical Journal - Software*, 2(1):86–95, 1997.
8. A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.
9. Y. Métivier. Une condition suffisante de reconnaissabilité dans un monoïde partiellement commutatif. *R.A.I.R.O. — Informatique Théorique et Applications*, 20:121–127, 1986.
10. Y. Métivier. On recognizable subsets of free partially comutative monoids. *Theoretical Computer Science*, 58:201–208, 1988.
11. A. Muscholl. *Decision and complexity issues on concurrent systems*. Habilitationsschrift (postdoctoral thesis), Universität Stuttgart, Jan. 1999. Submitted.
12. A. Muscholl, D. Peled, and Z. Su. Deciding properties of message sequence charts. In *Proc. of the 1st Int. Conference on Foundations of Software Science and Computation Structures (FoSSaCS'98)*, LNCS 1378, pp. 226–242, 1998.
13. E. Ochmański. Regular behaviour of concurrent systems. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 27:56–67, Oct. 1985.
14. J. Sakarovitch. The "last" decision problem for rational trace languages. Report LITP 91.77, Univ. Paris 6, 1991. Abstract presented at the 1st Int. Symp. of Latin American Theor. Informatics (LATIN'92), LNCS 583 (1992), pp. 460–473.
15. I. Walukiewicz. Difficult configurations – on the complexity of LTrL. In *Proc. of the 25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, LNCS 1443, pp. 140–151, 1998.