On the synchronization of semi-traces*

Klaus Reinhardt

Wilhelm-Schickhard Institut für Informatik, Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
e-mail: reinhard@informatik.uni-tuebingen.de

Abstract. The synchronization of two or more semi-traces describes the possible evaluation of a concurrent system, which consists of two or more concurrent subsystems in a modular way, where communication between the subsystems restricts the order of the actions. In this paper we give criteria, which tell us for given semi-traces in given semi-commutation systems, whether they are synchronizable and whether the synchronization is again a semi-trace; and criteria, which tell us for given semi-commutation systems, whether all semi-traces have this property. We prove that deciding these criteria is **NLOGSPACE**-complete for given semi-traces. The same holds for the synchronizability of all semi-traces for given semi-commutation systems. On the other hand the question, whether for given semi-commutation systems the synchronization of synchronizable semi-traces is a semi-trace is **co-NP**-complete. Furthermore we give a **co-NP**-complete condition for being able to decide synchronizability locally in $\mathbf{TC}^0$.

1 Introduction

Traces were introduced by Mazurkiewicz in [Maz87] to describe the behavior of concurrent systems. Traces are equivalence classes of words under partial commutations, which allow only to describe symmetric dependencies between actions. In order to enhance the possibilities of descriptions, M.Clerbout introduced the notion of semi-commutation in her thesis [Cle84] as a generalization of partial commutation, see also M. Clerbout and M. Latteux [CL87]. In a broader context it was observed in [HK89], [Och90] and [Och92], that semi-commutations are very useful for modeling behaviors of Petri-nets. For instance we are able to describe a producer/consumer system. For more information see [CGL+92], [OW93], [DR95].
If we want to describe a distributed system and we have described the single components by semi-dependence alphabets, where the alphabets can partially overlap at those actions, where the components interact, we need the operation of *synchronization* to get the consistent evaluations of the entire system.
The *partial traces* in [Die94] are closed under synchronization; but this notion gives up the efficient description by just one representing word. Semi-traces are

* this research has been supported by the EBRA working group No. 3166 ASMICS.

not closed under synchronization and hence we have to distinguish between *stable* and *unstable* synchronizations of semi-traces.
In this paper we classify the complexity of problems and operations on semi-commutation systems, which are semantically relevant for such concurrent systems. Since it is important for practical applications to be able to design algorithms with low complexity and particularly efficient parallel algorithms, it is a good news, that some of the problems are in such low complexity classes as **NLOGSPACE** and $\mathbf{TC}^0$.

2 Preliminaries

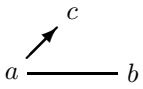
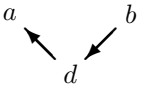
Let $\mathbf{TC}^0$ be the class of problems being recognized by a uniform circuit family of constant depth and polynomial size with threshold-gates or computable in constant time for the enlarged PRAM model in [Par90], respectively. $\mathbf{NC}^k$ ($\mathbf{AC}^k$) is the class of problems being recognized by a circuit family of $O(log^k n)$ depth and polynomial size with bounded (unbounded) fan-in gates [Ruz81]. The problems in $\mathbf{NC} = \bigcup_k \mathbf{NC}^k$ are regarded as the efficiently parallelizable problems. **NLOGSPACE** (**NP**) is the class of problems being recognized by a logarithmic space-bounded (polynomial time-bounded) nondeterministic Turing machine. For completeness we use **LOGSPACE**-reducibility. **co-NP** is the set of the complements of problems in **NP**. The following inclusions are know [Joh90]: $\mathbf{AC}^0 \subset \mathbf{TC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{LOGSPACE} \subseteq \mathbf{NLOGSPACE} \subseteq \mathbf{AC}^1 \subseteq \mathbf{TC}^1 \subseteq \mathbf{NC}^2 \subseteq \mathbf{NC} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{\Sigma}_2^P \subseteq \mathbf{PSPACE}$. Although nearly no separations are known (e.g., $\mathbf{TC}^0 \neq \mathbf{NP}$? is not known), proper inclusions are conjectured, which motivates the difference for local checking in $\mathbf{TC}^0$ or **NLOGSPACE** regarded in this paper.
To describe semi-commutation, we use *semi-dependence alphabets*, of the form (A, SD) where $SD \subseteq A \times A$ is reflexive but possibly asymmetric. This defines the associated *semi-commutation system* $SC = \{\, ab \Longrightarrow ba \mid (a,b) \notin SD \,\}$. A *semi-trace* over (A, SD) is by definition the set of words $[u\rangle = \{\, w \in A^* \mid u \overset{*}{\underset{SC}{\Longrightarrow}} w \,\}$, which can be derived from a word $u \in A^*$ by applying semi-commutation rules from SC. This means that we can use one possible evaluation of the concurrent system expressed by the word u to describe all possible evaluations. For several semi-dependence alphabets (A_i, SD_i) the corresponding semi-traces $[u\rangle_i$ get the same index.

2.1 Graph representation

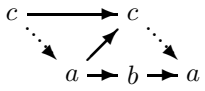
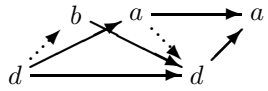
We represent a semi-trace $[a_1 ... a_n\rangle$ over (A, SD) by a graph with nodes labeled by $a_1 ... a_n$ and two kinds of edges: The *hard arcs* $a_i \rightarrow a_j$ with $i < j$ and $(a_i, a_j) \in SD$ and the *soft arcs*[2] $a_i \dashrightarrow a_j$ with $i < j$ and $(a_i, a_j) \in SD^{-1} \setminus SD$, which emphasize the semi-dependence structure (see also [DOR94]). The graph examples are restricted to the Hasse diagram, that means we do not show arcs,

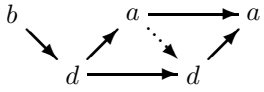
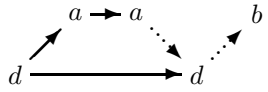
[2] An asymmetric commutation can change the order described by a soft arc but then the outcome is not representing the complete semi-trace. $(SD^{-1} = \{(a,b) \mid (b,a) \in SD\})$

which are in the transitive closure of shown arcs.
Example: Consider the following semi-dependence alphabets

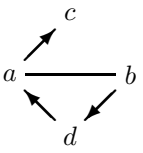
$(A_1, SD_1) =$  and $(A_2, SD_2) =$ 

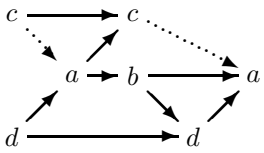
and the following semi-traces together with their graph representation:

$[cacba\rangle_1 =$  $[dbada\rangle_2 =$ 

$[bdada\rangle_2 =$  $[daadb\rangle_2 =$ 

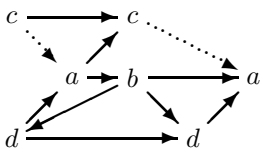
2.2 Synchronization

In order to construct complex systems in a modular way, we need a notion of synchronization for semi-commutation as it was introduced by Mazurkiewicz in [Maz87] and [DV89] for the symmetric case of partial commutation.
The synchronization $(A_1, SD_1) \parallel ... \parallel (A_m, SD_m)$ of semi-dependence alphabets is simply their union $(A, SD) = (\bigcup_{i \le m} A_i, \bigcup_{i \le m} SD_i)$. The synchronization[3] of semi-traces is $[u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m = \{w \in A^* \mid \Pi^A_{A_i}(w) \in [u_i\rangle_i \, \forall 0 < i \le m\}$, where $\Pi^A_{A_i}(w)$ is the *projection* of the word w to the letters in A_i with $\Pi^A_{A_i}(a) = a$ for all $a \in A_i$ and $\Pi^A_{A_i}(a) = \lambda$ otherwise. We call semi-traces *synchronizable*, if their synchronization is not the empty set and *stably synchronizable* if their synchronization is a semi-trace.
On the graph representation level the synchronization is the union of the graphs of the semi-traces, where we have to identify nodes with the same labeling according to their order preserving all arcs describing dependencies. Of course $\forall i, j \le m \; \forall a \in A_i \cap A_j \mid u_i \mid_a = \mid u_j \mid_a$ is a necessary condition for synchronizability; together with the non-existence of a cycle of hard arcs, the condition becomes also sufficient. If there is a soft arc (in either direction) in a cycle with hard arcs having the same direction, the soft arc can be deleted.
Continuing the above example we get the new semi-dependence alphabet $(A, SD) = (A_1 \cup A_2, SD_1 \cup SD_2) =$  and for $[cacba\rangle_1 \parallel [dbada\rangle_2$ we get the stable synchronization

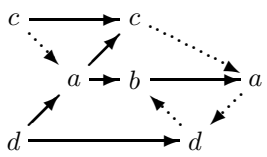
$[cdacbda\rangle =$  over (A, SD).

[3] It is easy to see that the synchronization on an identical alphabet is simply the intersection: $[u\rangle_i \parallel [v\rangle_i = [u\rangle_i \cap [v\rangle_i$

On the other hand we get $[cacba\rangle_1 \parallel [bdada\rangle_2 =$ which has a cycle of hard arcs and therefore $[cacba\rangle_1 \parallel [bdada\rangle_2 = \emptyset$.



Another example is $[cacba\rangle_1 \parallel [daadb\rangle_2$. Here we get the graph



having a directed cycle containing two soft arcs, which means, that either the soft arc $a \dashrightarrow d$ or the soft arc $d \dashrightarrow b$ must be turned around to get a word describing a possible evaluation. Therefore $[cacba\rangle_1 \parallel [daadb\rangle_2 = [cdacbad\rangle_1 \cup [cdacdba\rangle_2$ is not a stable synchronization.

2.3 Deciding the synchronizability of semi-traces

The above observations lead to the following theorem:

Theorem 1. *The following problems are* **NLOGSPACE***-complete:*

i) Given m semi-dependence alphabets (A_i, SD_i) with $0 < i \leq m$ and the semi-traces $[u_1\rangle_1, [u_2\rangle_2, \ldots [u_m\rangle_m$.
Are the traces synchronizable, that means $[u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m \neq \emptyset$?

ii) Given a semi-dependence alphabet (A, SD) and the semi-traces $[u\rangle, [v\rangle$ with $\forall a \in A\ |u_i|_a = |u_j|_a$.
Are the two traces synchronizable, that means $[u\rangle \cap [v\rangle \neq \emptyset$?

iii) Given m semi-dependence alphabets (A_i, SD_i) with $0 < i \leq m$ and the semi-traces $[u_1\rangle_1, [u_2\rangle_2, \ldots [u_m\rangle_m$.
Are the traces stably synchronizable, that means is there a word $w \in A^*$ with $[u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m = [w\rangle$?

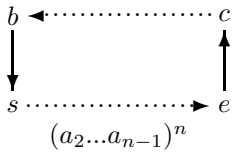
iv) Given a semi-dependence alphabet (A, SD) and the semi-traces $[u\rangle, [v\rangle$ with $[u\rangle \cap [v\rangle \neq \emptyset$.
Are the two traces stably synchronizable, that means is there a word $w \in A^*$ with $[u\rangle \cap [v\rangle = [w\rangle$?

Proof. i): A nondeterministic logarithmic space-bounded Turing machine first deterministically checks $\forall i, j \leq m\ \forall a \in A_i \cap A_j \mid u_i \mid_a = \mid u_j \mid_a$, then it guesses a cycle and tests each connection by testing precedence and dependence of the pair in each semi-trace. According to [Imm88] and [Sze88] the non-existence of the cycle can also be tested in **NLOGSPACE**. For hardness see *ii*).
ii): The problem is in **NLOGSPACE** because of *i*). The monotone graph reachability problem for directed graphs is well known to be **NLOGSPACE**-complete; for a given graph

$$G = (\{s = a_1, a_2, \ldots a_n = e\}, R)$$

with the property $(a_i, a_j) \in R \Rightarrow i < j$ the question is whether there exists a directed path from s to e. This can be reduced to the complementary of

the synchronizability problem by adding the edge (e, s) to the graph, which is now the dependence graph $(\{s, a_2, a_3 \dots a_{n-1}, e\}, R)$ $(SD := R \cup \{(e, s)\})$ and consider the synchronizability of $[esa_2a_3 \dots a_{n-1}\rangle$ and $[a_2a_3 \dots a_{n-1}es\rangle$. There exists a directed path from s to e, iff the semi-traces are not synchronizable, because of a cycle of hard arcs.
iii): A nondeterministic logarithmic space-bounded Turing machine first checks the synchronizability like in *i*), then is uses the [Imm88] and [Sze88] technique to do the opposite of the following: It guesses and tests a cycle with two soft arcs. Then again using [Imm88] and [Sze88] it tests, whether both soft arcs can be turned around without producing a cycle. For hardness see *iv*).
iv): The problem is in **NLOGSPACE** because of *iii*). The monotone graph reachability problem for directed graphs can be reduced to the problem by adding the additional letters b and c and the edges $(b, s), (b, c), (e, c)$ and (e, s) to the graph, which is now the dependence graph $(\{b, c, s, a_2, a_3, \dots a_{n-1}, e\}, SD)$ and consider, whether the synchronization of the semi-traces
$[cbs(a_2a_3 \dots a_{n-1})^n e\rangle$ and $[s(a_2a_3 \dots a_{n-1})^n ecb\rangle$ is a semi-trace.



The synchronization is the semi-trace $[bs(a_2a_3 \dots a_{n-1})^n ec\rangle$, iff there is a path of hard arcs from s to e. □

For the existence of a non-confluent situation we get the same complexity:

Theorem 2. *[Rei94] The following problem is* **NLOGSPACE***-complete:*
Given a semi-dependence alphabet (A, SD) and the semi-trace $[u\rangle$.
Are there words $v, w \in [u\rangle$ with $[v\rangle \parallel [w\rangle = \emptyset$?

3 The synchronizability in semi-commutation systems

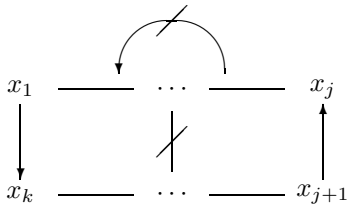
It is easy to see that only a cycle of hard arcs from at least two semi-traces can be responsible for non-synchronizability of semi-traces, where $\forall i, j \leq m\ \forall a \in A_i \cap A_j \mid u_i \mid_a = \mid u_j \mid_a$. (The cycle may even consist of a symmetric dependence.) Of course this cycle must be also in the dependence graph and must be composed of edges from at least two dependence alphabets.

Theorem 3. *Given m semi-dependence alphabets (A_i, SD_i) with $0 < i \leq m$. The following assertions are equivalent:*

i) $\forall u_1 \in A_1^*, \dots u_m \in A_m^*, (\forall i, j \leq m\ \forall a \in A_i \cap A_j \mid u_i \mid_a = \mid u_j \mid_a) \Rightarrow [u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \dots [u_m\rangle_m \neq \emptyset$
ii) $\neg\exists 1 < k, i \neq j, C = \{(x_1, x_2), \dots (x_{k-1}, x_k), (x_k, x_1)\}$ *with* $C \subseteq SD \wedge C \cap SD_i \neq \emptyset \wedge C \cap SD_j \neq \emptyset$

Again this property is **NLOGSPACE**-complete.
The following theorem says that semi-dependence alphabets have unstable synchronization (that means it can happen that the synchronization is neither empty nor a semi-trace), iff there exists a cycle in the union of the semi-dependence alphabets where

- all nodes are different,
- the two directed arcs (x_1, x_k) and (x_{j+1}, x_j) have reverse direction,
- there is no chord which separates them,
- there is no chord in backward direction and
- in each semi-dependence alphabet the cycle is either interrupted or has a directed arc in the opposite direction at some place, but then there is a another semi-dependence alphabet, having an arc at this place and is either interrupted or has a directed arc in the opposite direction at another place.



Theorem 4. *Given m semi-dependence alphabets (A_i, SD_i) with $0 < i \leq m$. The following assertions are equivalent:*

i) $\exists u_1 \in A_1^*, \ldots u_m \in A_m^* \ \forall w \in A^* \ \ [w\rangle \neq [u_1\rangle_1 \parallel [u_2\rangle_2 \parallel \ldots [u_m\rangle_m \neq \emptyset$

ii) $\exists j \in \{1, \ldots k-1\}, (x_1, x_2), \ldots, (x_{k-1}, x_k) \in SD \cup SD^{-1}$ *with*
$\forall n \neq p \ \ x_n \neq x_p \wedge$
$(x_1, x_k), (x_{j+1}, x_j) \in SD \setminus SD^{-1} \wedge$
$\forall (n,p) \in \{1, \ldots j\} \times \{j+1, \ldots k\} \setminus \{(1,k), (j, j+1)\} \ (x_n, x_p) \notin SD \cup SD^{-1} \wedge$
$\forall n \in \{1, \ldots k\} \ \forall p \in \{n+2, \ldots k\} \ \ (x_p, x_n) \notin SD \wedge$
$\forall i \leq m \ \exists n \leq k, \big((x_{n+1}, x_n) \notin SD_i \wedge$
$((x_n, x_{n+1}) \notin SD_i \vee$
$\exists l \in \{1, \ldots m\} \setminus \{i\} (x_n, x_{n+1}) \in SD_l \cup SD_l^{-1} \wedge$
$\exists p \in \{1, \ldots k\} \setminus \{n\} \ (x_{p+1}, x_p) \notin SD_l) \big)$

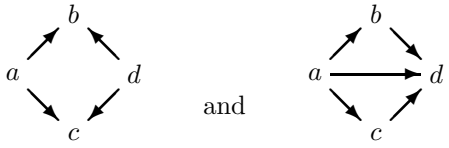
The simplest example for this is $(A_1, SD_1) = (\{a, b\}, \{(a, b)\})$ and $(A_2, SD_2) = (\{a, b\}, \{(b, a)\})$ then $[ba\rangle_1 \parallel [ab\rangle_2 = \{ab, ba\} \neq [u\rangle$ for any u.
If we only regard semi-traces over one semi-dependence alphabet, then it follows, that it has unstable synchronization, iff there exists a cycle of different nodes with two directed arcs in both directions and only directed chords from the part of the cycle where two of these arcs start (in different direction) to the part of the cycle where the other two arcs end.

Corollary 5. *Given a semi-dependence alphabet (A, SD). The following assertions are equivalent:*

i) $\exists u, v \in A^*\ \forall w \in A^*\ \ [w\rangle \neq [u\rangle \parallel [v\rangle \neq \emptyset$

ii) $\exists j \neq q \neq r \neq j \in \{1, ...k-1\}, (x_1, x_2), \dots (x_{k-1}, x_k) \in SD \cup SD^{-1}$ *with*
$\forall n \neq p\ \ x_n \neq x_p \wedge$
$(x_1, x_k), (x_{j+1}, x_j) \in SD \setminus SD^{-1} \wedge$
$(x_{q+1}, x_q), (x_{r+1}, x_r) \in SD^{-1} \setminus SD \wedge$
$\forall (n, p) \in \{1, ...j\} \times \{j+1, ...k\} \setminus \{(1, k), (j, j+1)\}\ (x_n, x_p) \notin SD \cup SD^{-1} \wedge$
$\forall n \in \{1, ...k\}\ \forall p \in \{n+2, ...k\}\ \ (x_p, x_n) \notin SD$

So there are only two such basic examples for semi-dependence alphabets (all other cases can be reduced to one of them by contraction of nodes in the graph):

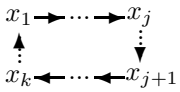


In the first one any chord would destroy the situation $[cabd\rangle \parallel [bdca\rangle = [abdc\rangle \cup [dcab\rangle \neq [u\rangle$ for any u. In the second one the situation $[dcab\rangle \parallel [bdca\rangle = [abdc\rangle \cup [cdba\rangle \neq [u\rangle$ for any u works independently from the existence of the arc from a to d. Now we come to the proof of Theorem 4:

Proof. $ii \Rightarrow i$: Take the shortest cycle of this kind and choose
$u_i = x_{n+1} \dots x_k x_1 x_2 \dots x_n$ such that $(x_{n+1}, x_n) \notin SD_i$ and

$$(x_n, x_{n+1}) \notin SD_i \vee \exists l \leq m, \neq i, (x_n, x_{n+1}) \in (SD_l \cup SD_l^{-1}) \cap [u_l\rangle_l.$$

So the synchronization has a cycle with the two soft arcs (x_k, x_1) and (x_j, x_{j+1}) and no cycle of hard arcs.
$i \Rightarrow ii$: Choose v, v', u_i (length-)minimal with $\forall i \Pi^A_{A_i}(v) = u_i, \forall i \Pi^A_{A_i}(v') = u_i$ and $\neg\exists w$ with $\forall i\ \Pi^A_{A_i}(w) = u_i, v, v' \in [w\rangle$. Construct G as the union of all (graphs of) $[u_i\rangle_i$. Construct G' from G by replacing every $b \dashrightarrow a$ by $a \longrightarrow b$ if $a = x_1 \longrightarrow x_2 \longrightarrow \dots x_j = b$ in G. Because of v, v' the graph G' contains no cycle of hard arcs. So G' must have a cycle with at least two soft arcs



because otherwise G' would be the graph for $[w\rangle$ which must contain every hard arc of G anyway because of $\forall i\ \Pi^A_{A_i}(w) = u_i$ and cycles with one soft arc are deleted in the construction of G'. The shortest cycle of this kind has no chord.
Now we can find a cycle with at least two soft arcs consisting only of original arcs from G and having only hard new chords in the direction of the cycle, which do not separate the two soft arcs with the following construction: We replace a new arc in cycle by the path of original hard arcs, which caused its existence. This may lead to a forbidden chord, but then we can find a shorting of the cycle with at least two soft arcs and no forbidden chord, where either the number of

soft arcs or the number of new hard arcs is reduced, so the construction must terminate.
Because of the minimality of the u_i there are no other vertices in G' except those in the cycle and no element of A appears twice. Furthermore the cycle can not contain soft chords and therefore there can be no arcs in the opposite direction. Now every arc in the cycle must come from one of the $[u_i\rangle_i$'s. But for every $[u_i\rangle_i$ the cycle must be open at a certain arc from x_n to x_{n+1} and it could be that $(x_n, x_{n+1}) \in SD_i$, then an arc at this place must be delivered from some other $[u_l\rangle_l$. But this $[u_l\rangle_l$ must be open at another arc from x_p to x_{p+1}. This is, what the last part of the formula in *ii* says. □

Theorem 6. *The assertions of Theorem 4 as well as the assertions of Corollary 5 are* **co-NP***-complete to decide for semi-commutation systems.*

It is easy to see that the problem is in **co-NP**. The proof of hardness in [Rei94] uses a similar graph construction as in [DOR94] with the only difference, that we have to replace the two directed arcs by two pairs of directed arc to two additional nodes.
The complexity class $\mathbf{\Sigma}_2^P$ contains those languages which are accepted by an NP-machine with access to an NP-oracle. (For more background see e. g. [BDG88].) In analogy to another result in [DOR94] the following holds:

Theorem 7. *The following problem is $\mathbf{\Sigma}_2^P$-complete: Given two dependence alphabets (A, D) and (A, D') such that $D' \subseteq D$.*
Does there exist a semi-commutation system SC such that

$$D = \{(a,b) \in A \times A \mid ab \Longrightarrow ba \notin SC \cap SC^{-1}\} \text{ and}$$
$$D' = \{(a,b) \in A \times A \mid ab \Longrightarrow ba \notin SC \cup SC^{-1}\} \text{ and}$$
$$\forall u, v \in A^* \ \exists w \in A^* \text{ with } [u\rangle \parallel [v\rangle = [w\rangle \text{ or } = \emptyset \ ?$$

This means can unstable synchronization be avoided by choosing the direction of asymmetric dependencies?

The same result holds for $2m$ dependence alphabets. Because the direction of the most arcs in the cycle is of no influence, we can not use the same graph construction as in [DOR94] for the proof; instead a different construction in [Rei94], which makes use of the direction of chords, works in a similar way to reduce the truth of quantified boolean formulae (see [Sto77], [Wra77]) to the problem.

Remark. Given $2m$ dependence alphabets (A_i, D_i) and (A_i, D_i') such that $D_i' \subseteq D_i$ for all $0 < i \leq m$. The direction of asymmetric arcs can be chosen for m semi-dependence alphabets in such a way that semi-traces having the same letters in the intersection are always synchronizable, that means the direction of asymmetric arcs can be changed in a way such that there is no directed cycle consisting of arcs from at least two of the dependencies iff there is no such cycle having at most one asymmetric arc. The property is hereby in **NLOGSPACE**.

4 The inclusion of semi-traces

For a valid representing evaluation of a concurrent system it is a basic question, whether another evaluation is also valid. If the system is described by partial commutation, it is the following trace-equivalence problem: Given a symmetric semi-commutation system C and two words w, w'; is it true that $w \overset{*}{\underset{C}{\Longrightarrow}} w'$?

The trace-equivalence problem is proved to be in $\mathbf{TC}^0$ in [AG91] by describing in first order logic enlarged by majority quantifiers, whether for each dependent pair (a, b) for every position in w there is a corresponding position in w' with the same number of preceding a's and b's. By a result of [BIS90] this follows to be in $\mathbf{TC}^0$ (but not in AC^0). The same is applied to asymmetric semi-commutation systems in [Rei94]:

Theorem 8. *The following semi-trace-inclusion problem is in* $\mathbf{TC}^0$*: Given an asymmetric semi-commutation system SC by the semi-dependence alphabet* (A, SD) *and two words* w, w'*; is it true that* $w \overset{*}{\underset{SC}{\Longrightarrow}} w'$ *respectively* $[w'\rangle \subseteq [w\rangle$*?*

5 Local checking of synchronizability of semi-traces

In [DV89] an easy testable ($\mathbf{TC}^0$), necessary condition for the synchronizability of traces was described. The *local checking property* is a criterion for the sufficiency of this condition. It is **co-NP**-complete [DOR94]. The necessary condition for the synchronizability of two semi-traces $[u_1\rangle_1, [u_2\rangle_2$ is the following:

$$\Pi^{A_1}_{A_1\cap A_2}(u_1) \overset{*}{\underset{SC'}{\Longrightarrow}} \Pi^{A_2}_{A_1\cap A_2}(u_2) \text{ for } SC' \text{ defined by } SD' = SD_1 \cap SD_2^{-1}$$

According to Theorem 8 this condition can be tested in $\mathbf{TC}^0$.

Lemma 9. *The above condition is sufficient, iff the* composition [RW91] *of the semi-commutation systems* SC_1 *and* SC_2^{-1} *is a semi-commutation system*[4]*.*

Proof. If the composition of the semi-commutation systems SC_1 and SC_2^{-1} is a semi-commutation system, the composition SC' is determined by $SD' = SD_1 \cap SD_2^{-1}$ over $A_1 \dot\cup \{a_1, \dots a_n\} = A_2 \dot\cup \{b_1, \dots b_m\}$. If $\Pi^{A_1}_{A_1\cap A_2}(u_1) \overset{*}{\underset{SC'}{\Longrightarrow}} \Pi^{A_2}_{A_1\cap A_2}(u_2)$ then $u_1 a_1^{|u_2|_{a_1}} \dots a_n^{|u_2|_{a_n}} \overset{*}{\underset{SC'}{\Longrightarrow}} u_2 b_1^{|u_1|_{b_1}} \dots b_m^{|u_1|_{b_m}}$, since
$(A_1 \times \{a_1, \dots a_n\}) \cap SD_1 = \emptyset$ and $(A_2 \times \{b_1, \dots b_m\}) \cap SD_2 = \emptyset$. Thus we have

$$u_1 a_1^{|u_2|_{a_1}}, \dots a_n^{|u_2|_{a_n}} \overset{*}{\underset{SC_1}{\Longrightarrow}} s \overset{*}{\underset{SC_2^{-1}}{\Longrightarrow}} u_2 b_1^{|u_1|_{b_1}}, \dots b_m^{|u_1|_{b_m}}$$

and s is in the synchronization of $[u_1\rangle_1$ and $[u_2\rangle_2$.
If the condition is sufficient, then for every $w_1 \overset{*}{\underset{SC'}{\Longrightarrow}} w_2$ there is a synchronization $s \in [\Pi_{A_1}(w_1)\rangle_1 \parallel [\Pi_{A_2}(w_2)\rangle_2$ and the derivation $w_1 \overset{*}{\underset{SC'}{\Longrightarrow}} w_2$ is composed of $w_1 \overset{*}{\underset{SC_1}{\Longrightarrow}} s \overset{*}{\underset{SC_2^{-1}}{\Longrightarrow}} w_2$. □

[4] this means that $f_{SC_1} \circ f_{SC_2^{-1}} = f_{SC'}$ for a semi-commutation system SC'.

Theorem 10. *The local checking property for synchronizability of two semi-traces is* **co-NP***-complete.*

Proof. A criterion describing, whether the composition of two semi-commutation systems is a semi-commutation system, is given in [RW91]. It is easy to see that this criterion is in **co-NP** and since the criterion for confluence, which was shown to be **co-NP**-complete in [DOR94], is a special case for this, the problem is **co-NP**-complete, too. □

Remark. If we use a pairwise application of the condition for the local checking for m semi-commutation systems, we get the same result with an immediate generalization of [RW91]. We conjecture that this is also the case for conditions regarding projections to sub-alphabets of constant size.

Remark. It is easy to see that for regular string languages R_1, R_2 the synchronization is also regular but even in the special case for symmetric dependencies it is undecidable whether $f_{SC}(R_1) \parallel f_{SC}(R_2) \neq \emptyset$ according to [AH89]; for more information see [Rei94].

Acknowledgment: *I thank Volker Diekert, Klaus-Jörn Lange and an anonymous referee for helpful remarks.*

References

[AG91] C. Àlvarez and J. Gabarró. The parallel complexity of two problems on concurrency. *Information Processing Letters*, 38:61–70, 1991.

[AH89] IJ. J. Aalbersberg and H. J. Hoogeboom. Characterizations of the decidability of some problems for regular trace languages. *Mathematical Systems Theory*, 22:1–19, 1989.

[BDG88] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity* I. Number 11 in EATCS Monographs on Theoretical Computer Science. Springer, Berlin-Heidelberg-New York, 1988.

[BIS90] D.M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. of Comp. and Syst. Sciences*, 41:274–306, 1990.

[CGL+92] M. Clerbout, D. Gonzalez, M. Latteux, E. Ochmanski, Y. Roos, and P.A. Wacrenier. Recognizable morphisms on semi commutations. Tech. Rep. LIFL I.T.-238, Université des Sciences et Technologies de Lille (France), 1992.

[CL87] M. Clerbout and M. Latteux. Semi-Commutations. *Information and Computation*, 73:59–74, 1987.

[Cle84] M. Clerbout. *Commutations Partielles et Familles de Langages.* Thèse, Université des Sciences et Technologies de Lille (France), 1984.

[Die94] V. Diekert. A partial trace semantics for Petri nets. *Theoretical Computer Science*, 134:87–105, 1994. Special issue of ICWLC 92, Kyoto (Japan).

[DOR94] V. Diekert, E. Ochmański, and K. Reinhardt. On confluent semi-commutation systems – decidability and complexity results. *Information and Computation*, 110:164–182, 1994. A preliminary version was presented at ICALP'91, Lecture Notes in Computer Science 510 (1991).

[DR95] V. Diekert and G. Rozenberg, editors. *The Book of Traces.* World Scientific, Singapore, 1995.

[DV89] V. Diekert and W. Vogler. On the synchronization of traces. *Mathematical Systems Theory*, 22:161–175, 1989. A preliminary extended abstract appeared at MFCS 88, Lecture Notes in Computer Science 324 (1988) 271-279.

[HK89] D. V. Hung and E. Knuth. Semi-commutations and Petri nets. *Theoretical Computer Science*, 64:67–81, 1989.

[Imm88] N. Immerman. Nondeterministic space is closed under complement. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[Joh90] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Algorithms and Complexity*, volume A of *Handbook of Theoretical Computer Science*, chapter 2, pages 67–161. Elsevier, 1990.

[Maz87] A. Mazurkiewicz. Trace theory. In W. Brauer et al., editors, *Petri Nets, Applications and Relationship to other Models of Concurrency*, number 255 in Lecture Notes in Computer Science, pages 279–324, Berlin-Heidelberg-New York, 1987. Springer.

[Och90] E. Ochmański. Semi-Commutation and Petri Nets. In V. Diekert, editor, *Proceedings of the ASMICS workshop Free Partially Commutative Monoids, Kochel am See 1989*, Report TUM-I9002, Technical University of Munich, pages 151–166, 1990.

[Och92] E. Ochmański. Modelling concurrency with semi-commutations. In I. M. Havel and V. Koubek, editors, *Proceedings of the 17th Symposium on Mathematical Foundations of Computer Science (MFCS'92), Prague, (Czechoslovakia), 1992*, number 629 in Lecture Notes in Computer Science, pages 412–420, Berlin-Heidelberg-New York, 1992. Springer.

[OW93] E. Ochmański and P. A. Wacrenier. On regular compatibility of semi-commutations. In Andrzej Lingas, Rolf Karlsson, and Svante Carlsson, editors, *Proceedings of the 20th International Colloquium on Automata, Languages and Programming (ICALP'93), Lund (Sweden) 1993*, number 700 in Lecture Notes in Computer Science, pages 445–456, Berlin-Heidelberg-New York, 1993. Springer.

[Par90] Ian Parberry. A primer on the complexity theory of neural networks. In R.B. Banerji, editor, *Formal Techniques in Artificial Intelligence*, Amsterdam, 1990. North-Holland.

[Rei94] K. Reinhardt. *Prioritätszählerautomaten und die Synchronisation von Halbspursprachen*. Dissertation, Institut für Informatik, Universität Stuttgart, 1994.

[Ruz81] W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1981.

[RW91] Y. Roos and P. A. Wacrenier. Composition of two semi commutations. In A. Tarlecki, editor, *Proceedings of the 16th Symposium on Mathematical Foundations of Computer Science (MFCS'91), Kazimierz Dolny (Poland) 1991*, number 520 in Lecture Notes in Computer Science, pages 406–414, Berlin-Heidelberg-New York, 1991. Springer.

[Sto77] L. J. Stockmeyer. The polynomial time hierarchy. *Theoret. Comput. Sci.*, 3:1–22, 1977.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Wra77] C. Wrathall. Complete sets and the polynomial hierarchie. *Theoret. Comput. Sci.*, 3:23–33, 1977.

This article was processed using the LaTeX macro package with the LLNCS document class.