Detecting Races in Ensembles of Message Sequence Charts

Edith Elkind[1], Blaise Genest [2], and Doron Peled[3]

[1] Department of Computer Science, University of Liverpool
Liverpool L69 3BX, United Kingdom
[2] CNRS & IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
[3] Department of Computer Science, Bar Ilan University, Ramat Gan 52900, Israel

Abstract. The analysis of message sequence charts (MSCs) is highly important in preventing common problems in communication protocols. Detecting race conditions, i.e., possible discrepancies in event order, was studied for a single MSC and for MSC graphs (a graph where each node consists of a single MSC, also called HMSC). For the former case, this problem can be solved in quadratic time, while for the latter case it was shown to be undecidable. However, the prevailing real-life situation is that a collection of MSCs, called here an *ensemble*, describing the different possible scenarios of the system behavior, is provided, rather than a single MSC or an MSC graph. For an ensemble of MSCs, a potential race condition in one of its MSCs can be compensated by another MSC in which the events occur in a different order. We provide a polynomial algorithm for detecting races in an ensemble. On the other hand, we show that in order to prevent races, the size of an ensemble may have to grow exponentially with the number of messages. Also, we initiate the formal study of the standard MSC *coregion* construct, which is used to relax the order among events of a process. We show that by using this construct, we can provide more compact race-free ensembles; however, detecting races becomes NP-complete.

1 Introduction

Software verification is an inherently difficult task. It is well known that it is undecidable for general domains. Moreover, even for finite domains many problems in this area are computationally intractable. In particular, this is often the case for problems that deal with concurrent processes. Another difficulty in applying verification methods for software is the technology transfer, i.e., providing the users (which are programmers and software engineers) with an easy-to-use and intuitive notation. It is thus beneficial to be able to analyze a notation that is already in use by software developers.

Message sequence charts (MSCs) is a formalism that is widely used in software engineering community and is formally described in [8]. This standard specification formalism consists of a textual notation, and a corresponding graphical notation. One MSC represents the relative order between message send and receive events (and sometimes also local events). A collection of charts represents alternative executions, which can also be organized into a graph, where each node is an MSC. The latter construction is called an MSC graph, or a High-level MSC (HMSC).

In recent years, several algorithms for analyzing MSCs and HMSCs have been suggested [4, 6, 10–13]. Perhaps the first problem to be analyzed was that of detecting race conditions [2]. This problem arises because in the MSC notation, the events of each process must be totally ordered. On the other hand, because of limited control on speed of message propagation, messages do not always arrive in the order specified by the MSC. In other words, there are two partial orders associated with each MSC: the *visual order*, which corresponds to the graphical description of the MSC, and the *causal order*, i.e., the order that is under the control of the programmer (e.g., sending a message after receiving another one, but on the other hand, not the order between two messages received from different processes). A race condition is defined as a discrepancy between the two orders.

A polynomial-time transitive closure-based algorithm for race detection was proposed in [2]. This algorithm is used in various tools, in particular in Bell Labs' uBET tool [7]. In [15], the authors generalize this problem to HMSCs and show that in this setting, it becomes undecidable. This is also the case for many other problems for HMSCs, such as HMSC intersection and LTL model checking of HMSCs [3, 14]. Intuitively, these undecidability results follow from the fact that the HMSC notation describes a system with no bound on the number of messages in transit. This complication can be avoided either by fixing a bound on the process queues, or by imposing various structural restrictions. The latter approach is taken by [3, 14]; their proofs proceed by bounding the size of the queues as a function of the HMSC.

Alur et al. [1] analyze several decision problems for a set of MSCs, rather than for a single MSC or an HMSC. In particular, they consider the problem of deciding whether a combination of behaviors for different processes specified by different MSCs is covered in the MSC collection. This is a very natural setting, as in many cases, the actual objects that software engineers have to deal with are collection of scenarios (described as MSCs), rather than a single MSC or an HMSC (one reason for this is that the semantics of HMSCs is not quite clear due to different ways of defining the concatenation of MSCs [15, 5]).

In this paper, we consider the problem of detecting race conditions in an ensemble of MSCs, i.e., a collection of MSCs over the same set of events. In this setting, even if in one of the MSCs the messages arrive in an order different from the one specified, another MSC in the collection may capture this alternative order. Thus, avoiding race conditions corresponds to *covering* alternative orders of events for one MSC by other MSCs in the collection. More precisely, race conditions can be defined in terms of the sets of linearizations (completions to total orders) of both orders associated with each MSC. Namely, for each MSC the visual order and the causal order typically produce different sets of linearizations (with the latter set including the former set). We say that an ensemble of MSCs contains a race if there is a linearization of the causal order of some MSC that is not a linearization of the visual order of any MSC in the ensemble.

We describe an efficient algorithm for finding race conditions, which extends the one of [2] for a single MSC. The running time of our algorithm is cubic in the representation size of the problem, i.e, the number of MSCs in the collection and the size of each MSC. However, in many cases one would need a large number of MSCs to avoid races: we describe some natural scenarios such that in any race-free collection that represents them the number of MSCs in the collection is exponential in the number of messages. Sometimes this problem can be alleviated by using *coregions*, which can be used to bundle together events in a single MSC. To the best of our knowledge, this paper provides the first formal study of the complexity and succinctness of this notational primitive, though it was already defined in [9] (see also [5, 16] for coregions in LSCs and in TMSCs). Intuitively, it removes restrictions on the relative order of events that appear in it. We provide examples in which using coregions results in an exponentially smaller race-free collection. However, the more compact representation comes at a cost, as the problem of detecting races becomes NP-complete, even for the most restrictive definition of a coregion.

2 Preliminaries

In this section, we formally define message sequence charts (MSCs) as well as two partial orders that are associated with them. Intuitively, a message sequence chart can be graphically represented as a collection of process lines, where messages are depicted as arcs that link the sending process with the receiving process. This representation implies an ordering over all events that belong to the same process, as well as between send and receive events that correspond to the same message.

Definition 1. *A message sequence chart (MSC) $M = (E, <^M, \mathcal{P}, \ell, S, R, r)$ is given by a set of events E, a partial order $<^M$ on E, a set $\mathcal{P}$ of processes, and a mapping $\ell : E \mapsto \mathcal{P}$ that associates each event in E with a process. For each process P, the set $\ell^{-1}(P)$ is totally ordered by $<_P = <^M |_P$. The event set is partitioned as $E = S \cup R$, where S and R are the sets of send and receive events, respectively. Furthermore, $r : S \mapsto R$ is a bijective mapping that relates each send with a unique receive. We assume that a process cannot send a message to itself, i.e., for any $e \in S$ we have $\ell(e) \neq \ell(r(e))$.*

Set $e <_c f$ for every e, f such that $r(e) = f$. It is required that $<^M$ is equal to the transitive closure of $<_c \cup \bigcup_{P \in \mathcal{P}} <_P$. The relation $<^M$ is called the visual order *of the message sequence chart M.*

In practical systems, there is often no way to ensure that two messages from different processes arrive in the same order. This means that the visual order may provide more ordering over events that is achievable in practice. This issue can be tackled by introducing a weaker partial order, which only orders two events if they necessarily happen in that order in any execution. There are three classes of such events. First, it is clear that each send occurs before the corresponding receive. Second, a process can condition sending a message on sending or receiving some other messages. This means that each send event always happens after all send and receive events of the same process that precede it in the visual order. Finally, we assume that processes communicate through a fifo channel, which guarantees that any two messages sent by one process to another always arrive in the correct order. This set of requirements can be formalized as follows.

Definition 2. *Given a message sequence chart $M = (E, <, \mathcal{P}, \ell, S, R, r)$, its* causal order *$\prec^M$ is a transitive closure of the precedence relation $\prec_0^M$, where for two events $e, f \in E$ we have $e \prec_0^M f$ if one of the following conditions holds:*

- *e and f are a matching send-receive pair, i.e., $r(e) = f$;*

– *$\ell(e) = \ell(f) = P$, $e <_P f$, and $f \in S$;*
– *$\ell(e) = \ell(f) = P$, $e, f \in R$, $r^{-1}(e) = e'$, $r^{-1}(f) = f'$, $\ell(e') = \ell(f') = P'$, and $e' <_{P'} f'$.*

Let $\mathcal{L}(X)$ be the set of all linearizations (i.e., completions to total order) of a partial order X. Clearly, for any message sequence chart M, $e \prec^M f$ implies $e <^M f$; however, the converse does not hold. In other words, we have $\mathcal{L}(<^M) \subseteq \mathcal{L}(\prec^M)$. To simplify notation, we write $\mathcal{L}_<(M)$ instead of $\mathcal{L}(<^M)$ and $\mathcal{L}_\prec(M)$ instead of $\mathcal{L}(\prec^M)$.

Definition 3. *We say that a message sequence chart M contains a* race condition *if there are two events $e, f \in E$ such that $e <^M f$, but e, f are unordered by $\prec^M$.*

Intuitively, a race condition means that the causal order allows more executions than the visual order, i.e., by restricting our attention to scenarios prescribed by the visual order, we may miss some (unexpected) executions. This situation is illustrated in Figure 1: in each MSC, the order of the two receive events of the second process is specified by the visual order, but not by the causal order. Hence, it is desirable to have an algorithm that detects races.

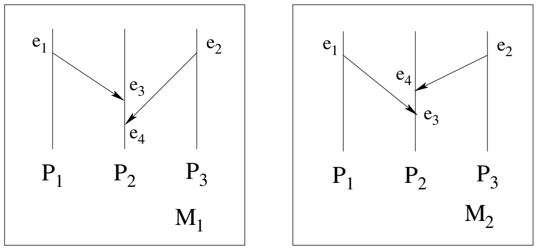

Fig. 1. Each of M_1, M_2 admits a race. In M_1, we have $e_3 <^{M_1} e_4$, but e_3 and e_4 are unordered by $\prec^{M_1}$. In M_2, we have $e_4 <^{M_2} e_3$, but e_3 and e_4 are unordered by $\prec^{M_2}$. However, taken together, M_1 and M_2 cover all possible executions.

An equivalent definition of a race is to say that the set of all linearizations of $<^M$ is strictly contained in the set of all linearizations of $\prec^M$. As we have $\mathcal{L}_<(M) \subseteq \mathcal{L}_\prec(M)$ for any M, it follows that M contains a race if and only if $\mathcal{L}_\prec(M) \neq \mathcal{L}_<(M)$. The advantage of this definition is that it is easier to generalize it to collections of MSCs, defined below.

3 Race detection in multiple MSCs

We start by introducing the notion of an *ensemble* of message sequence charts. Intuitively, it is a collection of several message sequence charts describing acceptable behaviors of the system. Consequently, the message sequence charts in an ensemble describe different partial orders (both visual and causal) on the same set of events.

Definition 4. *An ensemble of MSCs is a set $\mathbb{M} = \{M_1, \dots, M_m\}$ such that*

– *$E_1 = \cdots = E_m = E$;*
– *$\mathcal{P}_1 = \cdots = \mathcal{P}_m = \mathcal{P}$;*
– *for any $e \in E$, it holds that $\ell_1(e) = \cdots = \ell_m(e) = \ell(e)$;*
– *$S_1 = \cdots = S_m = S$, $R_1 = \cdots = R_m = R$;*
– *for any $e \in S$, it holds that $r_1(e) = \cdots = r_m(e) = r(e)$.*

Remark 1. Note that in general the admissible executions may not share the same set (and type) of events, i.e., the MSCs given in the input do not necessarily form an ensemble. However, in this case one can easily decompose the input into a collection of ensembles. Therefore, checking ensembles rather than arbitrary collections of MSCs does not lead to a loss of generality.

For an ensemble $\mathbb{M} = \{M_1, \dots, M_m\}$ of message sequence charts, we define $\mathcal{L}_<(\mathbb{M}) = \cup_{i=1,\dots,m}\mathcal{L}_<(M_i)$, $\mathcal{L}_\prec(\mathbb{M}) = \cup_{i=1,\dots,m}\mathcal{L}_\prec(M_i)$. Similarly to the case of a single message chart, we have $\mathcal{L}_<(\mathbb{M}) \subseteq \mathcal{L}_\prec(\mathbb{M})$. We say that there is a race if $\mathcal{L}_<(\mathbb{M}) \neq \mathcal{L}_\prec(\mathbb{M})$. It may be the case that each MSC in $\mathbb{M}$ is not race-free, but $\mathbb{M}$ is: for example, the two MSCs of Figure 1 form a race-free ensemble.

In the remainder of this section, we describe an algorithm that detect races in time polynomial in the total size of the message sequence charts in $\mathbb{M}$. Our approach is based on the following idea. Consider a graph G whose vertices are all permutations of events in E, and there is an edge between two vertices if the respective permutations can be obtained from each other by reversing the order of two adjacent events. Clearly, if $|E| = N$, then G has $N!$ vertices. Our algorithm does not construct this graph explicitly; however, it will be useful in proving correctness of our algorithm. It easy to see that each of the sets $\mathcal{L}_\prec(M_i)$, $\mathcal{L}_<(M_i)$, $i = 1, \dots, m$, forms a connected subgraph of this graph. Note also that $\mathcal{L}_<(M_i) \subseteq \mathcal{L}_\prec(M_i)$ for all $i = 1, \dots, m$. Moreover, if $M_i \neq M_j$, then the sets $\mathcal{L}_<(M_i)$ and $\mathcal{L}_<(M_j)$ are disjoint. To see this, note that the visual orders $<^{M_i}$ and $<^{M_j}$ differ if and only if they have different projections on some process line P_k, i.e., some events e_x and e_y with $\ell(e_x) = \ell(e_y) = P_k$ are ordered differently by $<^{M_i}$ and $<^{M_j}$. This means that for any $L_i \in \mathcal{L}_<(M_i)$ and $L_j \in \mathcal{L}_<(M_j)$ the events e_x and e_y will be ordered differently in L_i and L_j as well, i.e., $\mathcal{L}_<(M_i) \cap \mathcal{L}_<(M_j) = \emptyset$.

Proposition 1. *The ensemble $\mathbb{M}$ contains a race if and only if for some $i, j \in \{1, \dots, m\}$ and a permutation L_1 the following conditions hold: (1) $L_1 \in \mathcal{L}_\prec(M_i)$; (2) L_1 is adjacent in G to some permutation $L_2 \in \mathcal{L}_<(M_j)$; (3) $L_1 \notin \mathcal{L}_<(\mathbb{M})$.*

Proof. Clearly, if $\mathbb{M}$ does not contain a race, no such i, j and L_1 can exist, as any permutation in $\mathcal{L}_\prec(M_i)$ will be contained in $\mathcal{L}_<(\mathbb{M})$.

For the opposite direction, suppose that $\mathbb{M}$ contains a race, that is, for some $i \in \{1, \dots, m\}$ there is a permutation L such that $L \in \mathcal{L}_\prec(M_i)$, $L \notin \mathcal{L}_<(\mathbb{M})$. Consider the subgraph of G induced by $\mathcal{L}_<(\mathbb{M})$, and let $\hat{\mathcal{L}}$ be the maximal connected component of this subgraph that contains $\mathcal{L}_<(M_i)$. Clearly, we have $L \notin \hat{\mathcal{L}}$. On the other hand, as $\mathcal{L}_<(M_i) \subseteq \mathcal{L}_\prec(M_i)$, there is another permutation $L' \in \mathcal{L}_\prec(M_i)$ such that $L' \in \mathcal{L}_<(M_i) \subseteq \hat{\mathcal{L}}$. Since the set $\mathcal{L}_\prec(M_i)$ is connected, there is a path in G that stays within $\mathcal{L}_\prec(M_i)$ and leads from L to L'. The last vertex on this path is in $\hat{\mathcal{L}}$, while the first one is not. Therefore, there exist two adjacent vertices (i.e., permutations) L_1 and L_2 on this path such that (i) both L_1 and L_2 are in $\mathcal{L}_\prec(M_i)$ and (ii) L_2 is in $\hat{\mathcal{L}}$, while L_1

is not. Moreover, we have $L_2 \in \mathcal{L}_<(M_j) \subseteq \hat{\mathcal{L}}$ for some $j \in \{1, \dots, m\}$. It remains to show that $L_1 \notin \mathcal{L}_<(M_k)$ for any $k = 1, \dots, m$. For any k such that $\mathcal{L}_<(M_k) \subseteq \hat{\mathcal{L}}$, this holds since $L_1 \notin \hat{\mathcal{L}}$. Now, suppose $L_1 \in \mathcal{L}_<(M_k)$ for some k such that $\mathcal{L}_<(M_k) \not\subseteq \hat{\mathcal{L}}$. Then $\mathcal{L}_<(M_k)$ contains an element (i.e., L_1) that is adjacent to $\hat{\mathcal{L}}$. This means that the set $\hat{\mathcal{L}} \cup \mathcal{L}_<(M_k)$ is connected. However, $\hat{\mathcal{L}}$ was defined as the largest connected component of $\mathcal{L}_<(\mathbb{M})$ that contains $\mathcal{L}_<(M_i)$, a contradiction. ∎

Hence, to check for races, it is sufficient to verify if the condition of Proposition 1 holds. The straightforward approach of checking this condition for each linearization requires superpolynomial time. However, it turns out that we can partition candidate linearizations in a polynomial number of classes that correspond to certain partial orders, and check all linearizations in the same class simultaneously. Namely, consider the following algorithm `DetectRace(`$\mathbb{M}$`)`.

```
DetectRace($\mathbb{M}$);
forall $M = M_1, \dots, M_m$ do
        forall $P = P_1, \dots, P_n$ do
                $K := |\ell^{-1}(P)|$;
                for $j = 1, \dots, K-1$ do
                     $<'_P =$ Swap($M, P, j$);
                     $<'^M = <_c \cup \bigcup_{P' \neq P} <_{P'} \cup <'_P$;
                     if PO($<'^M$) and Disjoint($<'^M$) then
                        forall $N = M_1, \dots, M_m$ do
                                if PO($<'^M \cup \prec^N$) return true;
return false;
```

The function `Swap`(M, P, j) returns the total order obtained from the order of the events that belong to process P in message sequence chart M by switching the order of the jth and the $(j+1)$st event. The function `PO`(X) checks if its input relation X is a partial order, i.e., contains no cycles. The function `Disjoint`$(<'^M)$ checks that the set of linearizations of $<'^M$ is disjoint from $\mathcal{L}_<(\mathbb{M})$; its implementation is given below. All other functions are straightforward to implement.

```
Disjoint($<'^M$);
forall $N = M_1, \dots, M_m$ do
        forall $P = P_1, \dots, P_n$ do
                if $<'^M_P \neq <^N_P$ break;
        return false;
return true;
```

In words, for each message sequence chart $M \in \{M_1, \dots, M_m\}$, we consider the visual orders of all MSCs that can be obtained from M by switching the order of two consecutive events of some process. For each MSC obtained in this way, we check if it is valid, i.e., contains no cycles, using the function `PO`. If this is indeed the case, we check whether the linearizations of this MSC are not contained in $\mathcal{L}_<(\mathbb{M})$ (function `Disjoint`), and whether the union of the visual order of this MSC with the causal order of some other MSC in $\mathbb{M}$ is a partial order. If both of these conditions hold, our algorithm returns `true`, which means that $\mathbb{M}$ contains a race.

The correctness of our algorithm is proved via a sequence of lemmas.

Lemma 1. *For any $i, j = 1, \dots, m$ and any $L \in \mathcal{L}_{\prec}(M_i)$, if L is adjacent to some $L' \in \mathcal{L}_{<}(M_j)$ in G, then L can be obtained as a linearization of one of the partial orders $<'^{M_j}$ constructed by* `DetectRace`.

Proof. Consider a linearization $L \in \mathcal{L}_{\prec}(M_i) \setminus \mathcal{L}_{<}(M_j)$ that is adjacent to some $L' \in \mathcal{L}_{<}(M_j)$. Let e_x and e_y be the two events that are ordered differently in L and L'; assume without loss of generality that e_x precedes e_y in L'. If e_x and e_y are not ordered by $<^{M_j}$, then changing their order will result in another linearization of $<^{M_j}$. Hence, we assume that $e_x <^{M_j} e_y$.

By construction of the graph G, e_x and e_y have to be adjacent events in L'. Recall that in $<^{M_j}$ the event e_x has at most two immediate successors: $e_t = r(e_x)$ if $e_x \in S$, and the event e_z that immediately follows e_x on the same process line. Hence, $e_y \in \{e_z, e_t\}$. If $e_y = e_t$, i.e., e_x and e_y are a matching send–receive pair, then $e_x \prec^{M_i} e_y$; as $L \in \mathcal{L}_{\prec}(M_i)$, e_y cannot precede e_x in L. Hence, e_x and e_y are consecutive events of some process. Consequently, $L \in \mathcal{L}(<'^M)$ for $M = M_j$, $P = \ell(e_x) = \ell(e_y)$ and j equal to the position of e_x in $<_P$. ∎

Lemma 2. *For any $j = 1, \dots, m$ and any partial order $<'^{M_j}$ constructed by the algorithm* `DetectRace` *such that* `PO(`$<'^{M_j}$`) = true`*, either the set of all linearizations of $<'^{M_j}$ is disjoint from $\mathcal{L}(\mathbb{M})$ or $<'^{M_j} = <^{M_i}$ for some $i = 1, \dots, m$.*

Proof. The partial order $<'^{M_j}$ is obtained from $<^{M_j}$ by changing the order of two consecutive events of the same process. Hence, $<'^{M_j}$ also provides a total ordering on the events of each process. If some other $M_i \in \mathbb{M}$ has the same projections on all process lines, then by definition we have $<'^{M_j} = <^{M_i}$. Otherwise, for each $M_i \in \mathbb{M}$ there is a pair of events ordered differently by $<'^{M_j}$ and $<^{M_i}$. In this case, the sets $\mathcal{L}(<'^{M_j})$ and $\mathcal{L}_{<}(M_i)$ are disjoint. ∎

Lemma 3. *For any two partial orders X and Y on the same set of events, we have $\mathcal{L}(X) \cap \mathcal{L}(Y) \neq \emptyset$ if and only if $X \cup Y$ is a partial order.*

Proof. If $X \cup Y$ is a partial order, we have $\mathcal{L}(X \cup Y) \neq \emptyset$. As $\mathcal{L}(X \cup Y) \subseteq \mathcal{L}(X) \cap \mathcal{L}(Y)$, we have $\mathcal{L}(X) \cap \mathcal{L}(Y) \neq \emptyset$.

Conversely, if $X \cup Y$ is not acyclic, assume for the sake of contradiction that $\mathcal{L}(X) \cap \mathcal{L}(Y) \neq \emptyset$ and let L be a linearization in $\mathcal{L}(X) \cap \mathcal{L}(Y)$. Consider some cycle $\mathcal{C} = \{(e_1, e_2), \dots, (e_{t-1}, e_t), (e_t, e_1)\}$ in $X \cup Y$; there are two events e and f such that $(e, f) \in \mathcal{C}$, but f precedes e in L. Clearly, either $(e, f) \in X$, in which case $L \notin \mathcal{L}(X)$, or $(e, f) \in Y$, in which case $L \notin \mathcal{L}(Y)$. Hence, $L \notin \mathcal{L}(X) \cap \mathcal{L}(Y)$, a contradiction. ∎

Theorem 1. *The algorithm* `DetectRace`$(\mathbb{M})$ *returns* `true` *if and only if $\mathbb{M}$ admits a race. Moreover,* `DetectRace`$(\mathbb{M})$ *runs in time $O(m^2|E|^3)$, where $|E|$ is the number of events in any MSC in $\mathbb{M}$ and m is the number of message sequence charts in $\mathbb{M}$.*

Proof. By Proposition 1, detecting a race is equivalent to finding a permutation that satisfies conditions (1)–(3) in the statement of Proposition 1. By Lemma 1, for each $j = 1, \dots, m$, the algorithm `DetectRace`$(\mathbb{M})$ considers all linearizations that lie in the 1-neighborhood of $\mathcal{L}_{<}(M^j)$. By Lemma 2, the function `Disjoint` correctly decides if any of these linearizations is not covered by $\mathcal{L}(\mathbb{M})$. Finally, by Lemma 3, the last loop correctly determines if any of them can be a linearization of some causal order $\prec^{M_i}$, $i = 1, \ldots, m$. Hence, our algorithm never fails to detect a race. Conversely, `DetectRace`$(\mathbb{M})$ returns `true` only if it finds a linearization that lies in some $\prec^{M_i}$, $i = 1, \ldots, m$, but is not contained in $\mathcal{L}_<(\mathbb{M})$.

It remains to analyze the running time of our algorithm. For each triple M, P, j, we call `PO` once to check if $<'^M$ is acyclic, call `Disjoint` to compare $<'^M$ with all MSCs in $\mathbb{M}$, and then for each MSC in $\mathbb{M}$ compute a union of two partial orders and use `PO` again to check if it is acyclic. The function `PO` is based on computing the transitive closure. As each event has at most two immediate successors, the transitive closure computation can be done in time $O(|E|^2)$ [2]. Comparing the visual orders of two MSCs can be done in time $|E|$. Hence, we use $O(m|E|^2)$ operations for each triple M, P, j. As our algorithm only attempts to permute events that are adjacent on some process line for some MSC, we only consider $m|E|$ such triples. Hence, the running time of `DetectRace` is $O(m^2|E|^3)$, i.e., cubic in the size of the input. ∎

4 Number of MSCs needed for closedness

In the previous section, we give an algorithm that checks whether an ensemble of message sequence charts contains a race. Our algorithm is polynomial in the representation size of the ensemble, that is, the number of events and the number of MSCs in the ensemble. In this section, we investigate the relationship between these two parameters.

Two processes. Consider a message sequence chart M_2 given in Figure 2. It consists of two processes P_1 and P_2 and $n = 4k$ events $e_1, \ldots, e_n$, and describes the scenario when each process sends k messages to the other one, independently of the information it receives. Clearly, the causal order of M_2 induces a total order on all send events of each process. Because of the fifo assumption it also induces a total order on all receive events of the same process. However, the sends and receives of each process can be interleaved in an arbitrary way.

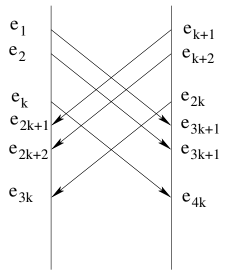

Fig. 2. An MSC with 2 processes whose causal order corresponds to $2^{\Omega(n)}$ visual orders

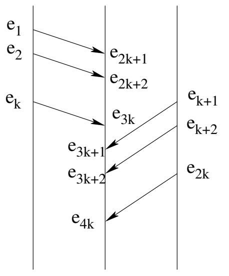

Fig. 3. An MSC with 3 processes whose causal order corresponds to $2^{\Omega(n)}$ visual orders

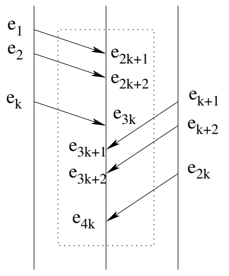

Fig. 4. A race-free MSC with a coregion of type 2

Theorem 2. *For any ensemble $\mathbb{M}$ that contains M_2, if $\mathbb{M}$ is race-free, then $\mathbb{M}$ contains $2^{\Omega(n)}$ message sequence charts.*

Proof. Recall that for any linearization L of $\prec^{M_2}$, its projection onto P_1 is a total order on the events $e_1, \dots, e_k, e_{2k+1}, \dots, e_{3k}$. Consider a collection of indices $I = \{i_1, \dots, i_k\}$, where $1 \leq i_1 < \dots < i_k \leq 2k$. We claim that for any such I, there exists a linearization L_I of $\prec^{M_2}$ such that its projection onto P_1 has the jth receive event of P_1, i.e., e_{2k+j}, in the i_jth position. Indeed, we can construct L_I by putting all send events of P_2 first, followed by all events of P_1, where the jth receive event of P_1 is in the position $k + i_j$, followed by all receive events of P_2. Formally, it can be described as a permutation $L_I = (e'_1, \dots, e'_{4k})$ of the events in E such that for $i = 1, \dots, k$ we have $e'_i = e_{k+i}$, $e'_{3k+i} = e_{3k+i}$, and for $i, \dots, 2k$ we have $e'_{k+i} = e_{2k+j}$ if and only if $i = i_j \in I$, and the remaining events in $\{e'_{k+1}, \dots, e'_{3k}\}$ are events from $\{e_1, \dots, e_k\}$, ordered so that e_i precedes e_j whenever $1 \leq i < j \leq k$.

Now, consider any ensemble $\mathbb{M}$ that contains M_2. The visual order of any MSC in $\mathbb{M}$ provides a total order on the events of P_1. Hence, for $\mathbb{M}$ to be race-free, for each set I of the form described above, it has to contain an MSC whose visual order of the events in P_1 is the same as the one given by L_I: otherwise, L_I is not contained in $\mathcal{L}_<(\mathbb{M})$. There are $\binom{2k}{k} = \Theta(2^{2k}/\sqrt{k})$ ways to choose the set I. Hence, to be race-free, $\mathbb{M}$ has to contain at least $\Theta(2^{2k}/\sqrt{k}) = 2^{\Omega(n)}$ message sequence charts. ∎

The proof of Theorem 2 depends on a subtle property of our definition of causal order. Namely, for two events $x \in R$, $y \in S$, $\ell(x) = \ell(y) = P$, we assume that if x precedes y in the visual order, then the same is true for the causal order. However, if x *follows* y in the visual order, we do not require that they are ordered in the causal order (they may still be ordered, of course, because of their causal relationships with other events). An alternative definition of causal order requires that two events are ordered in the causal order whenever they belong to the same process and at least one of them is a send event. In other words, two $x, y \in E$ such that $\ell(x) = \ell(y)$ may be unordered by the causal order only if $x, y \in R$. Of course, we still require the other properties of the causal order, i.e., ordering a send and the corresponding receive, and fifo. It is easy to see that under this definition of causal order, the argument of Theorem 2 no longer applies. Moreover, it turns out that in this case, any message sequence chart with at most two processes is race-free.

Proposition 2. *For any message sequence chart M that contains exactly two processes P_1 and P_2, we have $<^M \equiv \prec'^M$, where $\prec'$ is the variation of causal order defined above.*

Proof. Consider two events e_i and e_j of P_1 (the argument for P_2 is identical). If one of them is a send event, then, by definition, they are ordered in $\prec'^M$. Now, suppose that both of them are receive events and e_i precedes e_j in the visual order $<^M$. Then they both correspond to messages sent by P_2, i.e., $r^{-1}(e_i) = e_{i'}$, $r^{-1}(e_j) = e_{j'}$, and $\ell(e_{i'}) = \ell(e_{j'}) = P_2$. The visual order has to obey the fifo property, i.e., we have $e_{i'} <_{P_2} e_{j'}$. As both $e_{i'}$ and $e_{j'}$ are send events, they are also ordered in the causal order $\prec'^M$. Finally, by applying the fifo property to $\prec'^M$, we conclude that $r(e_{i'}) = e_i$ and $e(e_{j'}) = e_j$ are ordered in $\prec'^M$.

We have shown that $\prec'^M$ imposes an order on any two events that belong to the same process. Also, we have $e \prec'^M r(e)$ for an $e \in S$. It follows that $\prec'^M$ and $<^M$ order exactly the same events, i.e., $<^M \equiv \prec'^M$. ∎

Three processes. We will now show that with three processes we may need an exponential number of message sequence charts to avoid race conditions, even for the modified definition of causal order $\prec'$.

Theorem 3. *For any ensemble $\mathbb{M}$ that contains the MSC M_3 given in Figure 3, if $\mathbb{M}$ is race-free, then $\mathbb{M}$ contains $2^{\Omega(n)}$ message sequence charts.*

The proof of this theorem is similar to that of Theorem 2 and is omitted. It relies on the number of possible orderings of the events of the second process. As all of them are receive events, it does not depend on which version of the causal order we use.

5 MSCs with coregions

One can represent admissible orderings of events more compactly using *coregions*. A coregion is a notational primitive that covers two or more events. To the best of our knowledge, this paper is the first attempt to provide a formal analysis of the succintness of this notation. Intuitively, by putting events in a coregion we say that they can happen in any order. There are several ways to formalize this intuition, depending on what classes of events are allowed to appear in the same coregion.

Coregions that do not affect the causal order. The most restrictive approach is to only allow events not ordered by the causal order within a coregion.

Definition 5. *Given a MSC M on a set of processes $\mathcal{P} = \{P_1, \ldots, P_n\}$, a* coregion of type 1 *for M is a set of events $C = \{e_1, \ldots, e_k\}$ such that $e_1, \ldots, e_k$ are consecutive events of some process $P_i \in \mathcal{P}$ and no two events in C are ordered by $\prec^M$. The causal order $\prec^{(M,C)}$ of the pair (M, C) is the same as the causal order of M. To describe the visual order $<^{(M,C)}$ of the pair (M, C), we define the relation $<'_{P_i} = <_{P_i} \setminus \{(e_x, e_y) \mid e_x, e_y \in C\}$ and let $<^{(M,C)}$ be the partial order induced by the relation $<_c \cup \bigcup_{j \neq i} <_{P_j} \cup <'_{P_i}$.*

This definition can be extended to a message sequence chart with several coregions $C_1, \ldots, C_t$ in an obvious way. Namely, each coregion is required to consist of consecutive events of some process, and the visual order of the resulting message sequence chart is obtained from the original one by deleting all pairs (e_x, e_y) such that both e_x and e_y appear in the same coregion. We do not require that all coregions of a particular MSC are disjoint. We will sometimes abuse notation and use M to denote an MSC with one or more coregions. Also, we may refer to an MSC with coregions simply as an MSC. The exact meaning will always be clear from the context.

It is easy to see that this construction can be used to decrease the number of MSCs needed to avoid a race by exponential factor. An obvious example is provided by an MSC with $n + 1$ processes $P_0, P_1, \ldots, P_n$, in which each of $P_1, \ldots, P_n$ sends a single message to P_0. In the absence of coregions, the visual order of P_0 imposes a total order on all n receive events, while the causal order allows for any ordering of them. Hence, we need $n!$ message sequence charts to avoid races, one for each possible permutation of the receive events. On the other hand, if we are allowed to use coregions, we can simply put all receive events inside a coregion, thus covering all linearizations admitted by the causal order. The savings are not limited to MSCs with unbounded number of processes:

one can construct an example in which using coregions leads to an exponentially more compact race-free ensemble for three processes.

Unfortunately, the more compact representation has a computational cost. Namely, in Section 6 we show that for ensembles of MSCs with coregions detecting races becomes NP-hard. This result holds even if in all MSCs in the ensemble each process has at most two coregions, all coregions are of type 1, and coregions of any process do not overlap.

Coregions that may affect the causal order. In some settings, the requirement that all events in a coregion are not ordered by the causal order can be too restrictive. To increase the expressive power, we can use coregions to also express indifference about the causal order of certain events. For example, we may want to say that given two messages m_1 and m_2 from P_1 to P_2 and P_1 to P_3, respectively, it does not matter in which order they are sent. To capture this meaning, we eliminate the restriction that all events in a coregion must be independent with respect to $\prec$.

Definition 6. *Given a MSC M on a set of processes $\mathcal{P} = \{P_1, \ldots, P_n\}$, a* coregion of type 2 *for M is a set of events $C = \{e_1, \ldots, e_k\}$ such that $e_1, \ldots, e_k$ are consecutive events of some $P_i \in \mathcal{P}$.*

However, it turns out that for this definition of coregion, we cannot describe the set of all linearizations implied by an MSC by a single partial order.

Example 1. Consider an MSC that corresponds to one process sending two messages to another one. Formally, we set $M = (E, <^M, \mathcal{P}, \ell, S, R, r)$, where $S = \{s_1, s_2\}$, $R = \{r_1, r_2\}$, $E = S \cup R$, $\mathcal{P} = \{P_1, P_2\}$, $\ell(s_1) = \ell(s_2) = P_1$, $\ell(r_1) = \ell(r_2) = P_2$. Suppose that we would like to use coregions to express that the messages can be sent in any order. A natural way to do this is to set $<_{P_1} = (s_1, s_2)$, $<_{P_2} = (r_1, r_2)$, $C_1 = \{s_1, s_2\}$, $C_2 = \{r_1, r_2\}$. Because of the fifo rule, the set $\mathcal{L}$ of all linearizations that correspond to the causal order of this MSC consists of 4 elements, namely $L_1 = (s_1, s_2, r_1, r_2)$, $L_2 = (s_1, r_1, s_2, r_2)$, $L_3 = (s_2, s_1, r_2, r_1)$, and $L_4 = (s_2, r_2, s_1, r_1)$. We will now show that this set of linearizations cannot correspond to a single partial order. For the sake of contradiction, suppose that there is a partial order X with this set of linearizations. Clearly, if for some $e, f \in E$, e precedes f in some $L_i \in \mathcal{L}$, then $(f, e) \notin X$. This allows us to derive $(s_i, s_j) \notin X$, $(r_i, r_j) \notin X$, $(r_i, s_j) \notin X$ for any $i, j = 1, 2$, and also $(s_2, r_1) \notin X$, $(s_1, r_2) \notin X$. Hence, X can contain at most two elements, namely, (s_1, r_1) and (s_2, r_2). But then (s_1, s_2, r_2, r_1) would be a linearization of X, which is a contradiction.

Note also that, in contrast to the case of MSCs with coregions of type 1, the set L is not connected.

Consequently, the definition of the set of all linearizations of an MSC with coregions of type 2 is more complicated than that for a simple MSC.

Definition 7. *Given a message sequence chart M and a collection $C_1, \ldots, C_t$ of coregions of type 2 for M, let $\mathbb{M}$ be the ensemble of all valid message sequence charts that can be obtained from M by permuting the events in each of the coregions arbitrarily. Let $\mathcal{L}_{\prec}(M, C_1, \ldots, C_t)$ denote the set of all linearizations of the causal order of $(M, C_1, \ldots, C_t)$, and let $\mathcal{L}_{<}(M, C_1, \ldots, C_t)$ denote the set of all lineariza-*

tions of the visual order of $(M, C_1, \ldots, C_t)$. *We define* $\mathcal{L}_{\prec}(M, C_1, \ldots, C_t) = \mathcal{L}_{\prec}(\mathbb{M})$*,* $\mathcal{L}_{<}(M, C_1, \ldots, C_t) = \mathcal{L}_{<}(\mathbb{M})$.

One can verify that the informal argument in Example 1 is consistent with Definition 7, i.e., the set $\mathcal{L} = \{L_1, L_2, L_3, L_4\}$ is exactly the set of all linearizations that correspond to $\mathcal{L}_{\prec}(M, C_1, C_2)$. Moreover, for an MSC that only contains coregions of type 1, the two definitions result in the same set of linearizations, both for causal and for visual order. We will say that an MSC M with coregions $C_1, \ldots, C_t$ *captures* an ordinary MSC M' if $M' \in \mathbb{M}$, where $\mathbb{M}$ is the ensemble of message sequence charts constructed from M as in Definition 7.

By generalizing Example 1 to the case when P_1 sends n messages to P_2, we can see that, compared to MSCs with coregions of type 1, using coregions of type 2 may result in exponentially smaller race-free ensembles. Indeed, one can represent this scenario by a single race-free MSC and two coregions of type 2: one for all send events, and one for all receive events. On the other hand, an ensemble of MSCs with coregions of type 1 needs $n!$ MSCs to be race-free. Another example is given by MSC in Figure 4, which is obtained from the MSC in Theorem 3 by putting all events of P_2 in a coregion. This MSC is race-free; however, some of the events inside the coregion are ordered because of the fifo rule, so we cannot use a coregion of type 1. Nevertheless, the worst-case complexity of race detection is the same in both cases: in the next section, we show that race detection is NP-complete for ensembles of MSCs with coregions of both types.

6 Hardness results

Formally, an instance of the problem of race detection is given by an ensemble $\mathbb{M}$ whose elements are MSCs with coregions, i.e., each element of $\mathbb{M}$ is a list of the form $(M, C_1, \ldots, C_t)$, where M is a message sequence chart, and each C_i, $i = 1, \ldots, t$, is a coregion for M. We say that $\mathbb{M}$ is a "yes"-instance if and only if it admits a race.

Theorem 4. *The problem of race detection in ensembles of MSCs with coregions is NP-complete, even if each MSC in the ensemble only has coregions of type* 1*, each process has at most two coregions, and no two coregions can overlap.*

Proof. It is easy to see that this problem is in NP for coregions of both types: given an ensemble $\mathbb{M}$, a candidate linearization L and an MSC $M_i \in \mathbb{M}$, we can check that $L \in \mathcal{L}_{\prec}(M_i)$ and $L \notin \mathcal{L}_{<}(M_j)$ for all $M_j \in \mathbb{M}$.

For the opposite direction, the proof is by reduction from HAMILTONIAN PATH problem. Recall that an instance of HAMILTONIAN PATH is given by a graph $G = (V, E)$, $|V| = n$. It is considered to be a "yes"-instance if and only if it contains a simple path of length $n-1$, i.e., there is an ordering $v_{i_1}, \ldots, v_{i_n}$ of the elements of V such that $(v_{i_j}, v_{i_{j+1}}) \in E$ for all $j = 1, \ldots, n-1$.

Given an instance $G = (V, E)$ of HAMILTONIAN PATH, we construct an ensemble $\mathbb{M}$ of MSCs that corresponds to it. Consider an MSC M_0 that contains $2n+1$ processes, which are partitioned into two sets $\mathcal{P}_1 = \{P_0, \ldots, P_n\}$ and $\mathcal{P}_2 = \{P_v \mid v \in V\}$. Intuitively, M_0 describes the scenario where each of the processes in $\mathcal{P}_1$ sends a single message to each process in $\mathcal{P}_2$. Formally, we set $S = \{s_i^v\}_{i=0,\ldots,n, v \in V}$, $R = \{r_i^v\}_{i=0,\ldots,n, v \in V}$, $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$. Also, for all $i = 0, \ldots, n, v \in V$ we define $r(s_i^v) = r_i^v$, $\ell(s_i^v) = P_i$, $\ell(r_i^v) = P_v$. Fix an ordering $<^V$ on the vertices of V. In M_0, all events of each process are ordered lexicographically, i.e., for any $P_i \in \mathcal{P}_1$ we have $s_i^u <_{P_i} s_i^v$ if and only if $u <^V v$ and for any $P_v \in \mathcal{P}_2$ we have $r_i^v <_{P_v} r_j^v$ if and only if $i < j$. We will now construct an ensemble $\mathbb{M}$ that contains M_0. By definition, each $M \in \mathbb{M}$ has the same set of processes, the same sets of send and receive events, and the same mappings r and ℓ as M_0. We will also require that all $M \in \mathbb{M}$ order events in S in exactly the same way as M_0. Hence, to fully specify each $M \in \mathbb{M}$, we will only have to describe the order of the receive events for each process. Note that the causal order of M_0 imposes no restrictions on the relative order of the receives of any process. Hence, for $\mathbb{M}$ to be race-free, the linearizations of the visual orders of MSCs in $\mathbb{M}$ must cover all possible permutations of the receives.

To simplify notation, when describing the ordering of events on P_v, we will write i instead of r_i^v for $i = 1, \ldots, n$ and $\#$ instead of r_0^v. Consider a MSC M_{HP} that satisfies the following three conditions:

(1) for any $P_v \in \mathcal{P}_2$, there exists some $i \in \{1, \ldots, n\}$ such that the ordering of the receives on P_v is $(1, \ldots, i, \#, i+1, \ldots, n)$. This value of i is denoted by $i(v)$;
(2) for all $v \neq w \in V$, $i(v) \neq i(w)$;
(3) for any $v \in V$ such that $i(v) \neq n$, there exists an edge $(v, w) \in E$ such that $i(w) = i(v) + 1$.

Intuitively, M_{HP} describes a Hamiltonian path ρ in G: a vertex v is the ith vertex on ρ if $i(v) = i$. As $|V| = n$, conditions (1) and (2) imply that for any $i \leq n$, there exists a unique vertex v_i such that $i(v_i) = i$, and condition (3) means that for every $i < n$, (v_i, v_{i+1}) is an edge. Hence, $\rho = v_1 \cdots v_n$ is a Hamiltonian path. Therefore, if M_{HP} exists, then G has a Hamiltonian path; clearly, the converse is also true. We will now construct a polynomial ensemble $\mathbb{M}'$ of MSCs with coregions that captures all MSCs that violate at least one of the conditions (1), (2), or (3). Set $\mathbb{M} = \mathbb{M}' \cup \{M_0\}$; a race condition in $\mathbb{M}$ is equivalent to the existence of an MSC M_{HP} satisfying (1), (2) and (3), and hence to the existence of a Hamiltonian path in G.

The ensemble $\mathbb{M}'$ consists of three classes of MSCs with coregions: *bad order* MSCs (ones that capture MSCs that violate condition (1)), *no path* MSCs (ones that capture MSCs that violate condition (2)), and *bad path* MSCs (ones that capture MSCs that violate condition (3)).

Consider a message sequence chart that violates condition (1) for some P_v. If the first event of P_v is $\#$, then this MSC can be captured by a message sequence chart $M_{v,\#}$. In this MSC P_v starts with $\#$, followed by a coregion containing all other events of P_v in arbitrary order. For each P_w, $w \neq v$, the events of P_w are ordered arbitrarily, and there is a coregion that covers all of them. If P_v does not start with $\#$, let k be the first position in which the visual order of P_v deviates from the form prescribed by condition (1). As all events j, $j < k$, appear in their prescribed positions, the event in the position k must be l, $l > k$. We consider two cases, namely, $k \leq i(v)$ and $k > i(v)$.

All MSCs that violate condition (1) for P_v with $k \leq i(v)$ and event l in the kth position are captured by a message sequence chart $M^-_{(v,k,l)}$ defined as follows. For all P_w, $w \neq v$, the events of P_w are ordered arbitrarily, and there is a coregion that covers all of them. Moreover, the ordering of the first k events of P_v is $(1, \ldots, k-1, l)$, followed by all other events (including $\#$) in arbitrary order, and there is a coregion that consists of all events that appear after l.

Similarly, all MSCs that violate condition (1) for P_v with $k > i(v)$ and event l in the kth position are captured by a message sequence chart $M^{+}_{(v,k,l)}$ defined as follows. For all P_w, $w \neq v$, the events of P_w are ordered arbitrarily, and there is a coregion that covers all of them. The ordering of the first $k+1$ events of P_v is $(1, \ldots, k-1, \#, l)$, followed by all other events in arbitrary order. Also, there are two coregions for P_v: one that consists of all events that appear before l (including $\#$), and another one that consists of all events that appear after l. Observe that $M^{+}_{(v,k,l)}$ may also capture some MSCs where the first violation of condition (1) happens before k; nevertheless, all MSCs covered by $M^{+}_{(v,k,l)}$ violate condition (1) for P_v in position k.

Now, consider an MSC that satisfies condition (1), but violates condition (2). This happens if there are two vertices u and w such that $\#$ appears in the same position k in P_u and P_v. Hence, all such MSCs can be captured by n^3 *no path* MSCs $(M_{(u,v,k)})_{u \neq v \in V, k \leq n}$, defined as follows. In any $M_{(u,v,k)}$, for all P_w, $w \neq u, v$, the events of P_w are ordered arbitrarily, and there is a coregion that covers all of them. Furthermore, the events of P_u and P_v are ordered as $(1, \ldots, k, \#, k+1, \ldots, n)$.

Finally, we need to cover all MSCs that satisfy conditions (1) and (2), but violate condition (3). This happens if there is a pair of vertices $u, v \in V$ such that $(u, v) \notin E$, $i(u) = k$, $i(v) = k+1$. All such MSCs can be captured by at most n^3 *bad path* MSCs $(N_{(u,v,k)})_{(u,v) \notin E, k<n}$, defined as follows. In any $N_{(u,v,k)}$, for all P_w, $w \neq u, v$, the events of P_w are ordered arbitrarily, and there is a coregion that covers all of them. Furthermore, the events of P_u are ordered as $(1, \ldots, k, \#, k+1, \cdots, n)$, and the events of P_v are ordered as $(1, \ldots, k, k+1, \#, k+2, \cdots, n)$. Set

$$\mathbb{M}' = \bigcup_{v \in V} M_{v,\#} \cup \bigcup_{\substack{v \in V \\ k \neq l \leq n}} M^{-}_{(v,k,l)} \cup \bigcup_{\substack{v \in V \\ k \neq l \leq n}} M^{+}_{(v,k,l)} \cup \bigcup_{\substack{u \neq v \in V \\ k \leq n}} M_{(u,v,k)} \cup \bigcup_{\substack{(u,v) \notin E \\ k < n}} N_{(u,v,k)}.$$

Recall that $\mathbb{M} = \mathbb{M}' \cup \{M_0\}$, and observe that M_0 violates condition (1). The causal order of any $M \in \mathbb{M}$ is $\prec^{M_0}$, i.e., it puts no restrictions on the relative ordering of different receive events. For any MSCs M whose visual order violates (1), (2), or (3), there is an MSC M' in $\mathbb{M}$ (with or without coregions) such that $\mathcal{L}_{<}(M) \subseteq \mathcal{L}_{<}(M')$. On the other hand, any $M \in \mathbb{M}$ violates at least one of the conditions (1), (2), and (3).

Now, suppose that G contains a Hamiltonian path. Then there exists an MSC M_{HP} described above, which satisfies all three conditions. The set $\mathcal{L}_{<}(M_{HP})$ is not covered by $\mathcal{L}_{<}(\mathbb{M})$, i.e., there is a race. Conversely, suppose that G contains no Hamiltonian path, and let L be an arbitrary linearization of $\prec^{M_0}$. Consider the MSC M_L that is obtained by projecting L onto processes in $\mathcal{P}$. This MSC violates one of the conditions (1), (2), or (3), so we have $\mathcal{L}_{<}(M_L) \subseteq \mathcal{L}_{<}(\mathbb{M})$. As $L \in \mathcal{L}_{<}(M_L)$, the result follows. ∎

7 Conclusions

The MSC notation is important in describing scenarios of protocols. Its analysis allows one to detect common design errors. One of the most basic problems in MSCs is that of race conditions; the occurrence of events in an order that is different from the order of their appearance in the MSC. Race conditions are defined for a single MSC as the discrepancies between the visual order between events as they appear in the MSC, and the causal order, which takes into account only the order that is under the control of the system (e.g., excluding the order between receives from different processes). Equivalently, this can be defined as the discrepancy between the corresponding sets of linearizations. This relationship between partial orders and linearizations allows us to extend the problem beyond checking a single MSC. The classical algorithm for MSCs was described in [2] and was implemented in the uBET system. For a graph of MSCs (HMSC), this problem was shown to be undecidable [15].

In this paper we studied MSC ensembles, i.e., collections of MSCs for the same set of messages. In this case, a race in a single MSC of the ensemble may be compensated by another MSC with a different order of events. We describe a polynomial algorithm for race detection, which extends the algorithm of [2]. On the other hand, the existence of a polynomial algorithm can be attributed to the fact that a race-free ensemble of MSCs may need to have an exponential (in the number of events) number of MSCs.

We also studied the *coregion* construct, a part of the standard which has not been formally treated before. It allows encapsulating events (sends, receives) of a process within a box, denoting the lack of any particular order between the events in the box. We showed that by using this construct, one may achieve an exponential reduction in the size of race-free ensembles; however, race detection becomes NP-complete.

References

1. R. Alur, K. Etessami, and M. Yannakakis, Inference of message sequence charts. IEEE Transactions on Software Engineering, July 2003, Volume 29, 623–633
2. R. Alur, G. Holzmann, and D. Peled, An analyzer for message sequence charts. In *Software Concepts and Tools*, 17(2):70–77, 1996
3. R. Alur and M. Yannakakis, Model checking of message sequence charts. In *Proc. of CONCUR'99*, LNCS 1664, Springer, 114–129, 1999
4. H. Ben-Abdulla and S. Leue, Symbolic detection of process divergence and non-local choice in message sequence charts. In *Proc. of TACAS'97*, LNCS 1217, Springer, 259–274, 1997
5. W. Damm, D. Harel, LSCs: breathing life into message sequence charts. Formal Methods in System Design, Volume 19, July 2001, 45–80
6. L. Hélouët and C. Jard, Conditions for synthesis of communicating automata from HMSCs. In *5th International Workshop on Formal Methods for Industrial Critical Systems*, 2000
7. G. Holzmann, D. Peled, M. Redberg, Design tools for requirements engineering. Bell Labs Technical Journal, volume 2, 86–95, 1997
8. ITU-T Recommendation Z.120, Message Sequence Chart (MSC), Geneva, 1996
9. I. Krueger, Distributed system design with message sequence charts. Ph.D. Thesis, TU Munchen, 2000.
10. D. Kuske, Regular sets of infinite message sequence charts. *Information and Computation*, (187):80–109, 2003
11. M. Lohrey. Safe realizability of high-level message sequence charts. In *Proc. of CONCUR'02*, LNCS 2421, Springer-Verlag, 177–192, 2002
12. P. Madhusudan, Reasoning about sequential and branching behaviours of message sequence graphs. In *Proc. of ICALP'01*, LNCS 2076, Springer-Verlag, 809–820, 2001
13. M. Mukund, K. Narayan Kumar, and M. Sohoni, Synthesizing distributed finite-state systems from MSCs. In *Proc. of CONCUR'00*, LNCS 1877, Springer-Verlag, 521–535, 2000
14. A. Muscholl and D. Peled, Message sequence graphs and decision problems on Mazurkiewicz traces. In *Proc. of MFCS'99*, LNCS 1672, Springer-Verlag, 81–91, 1999
15. A. Muscholl, D. Peled, and Z. Su, Deciding properties of message sequence charts. In *Proc. of FoSSaCS'98*, LNCS 1378, Springer-Verlag, 226–242, 1998
16. B. Sengupta and R. Cleaveland, Triggered message sequence charts. *TSE* , IEEE, 2006.