FI MU

Faculty of Informatics
Masaryk University Brno

Decidable Race Condition for HMSC

by

Vojtěch Řehák
Petr Slovák
Jan Strejček
Loïc Hélouët

FI MU Report Series **FIMU-RS-2009-10**

 December 2009

Publications in the FI MU Report Series are in general accessible via WWW:

`http://www.fi.muni.cz/reports/`

Further information can be obtained by contacting:

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**

Decidable Race Condition for HMSC

Vojtěch Řehák* Petr Slovák Jan Strejček†
Faculty of Informatics, Masaryk University
Brno, Czech Republic
{rehak, xslovak2, strejcek}@fi.muni.cz

Loïc Hélouët
INRIA/IRISA
Rennes, France
loic.helouet@irisa.fr

December 27, 2009

Abstract

Races in Message Sequence Charts may lead to a bad interpretation of described behaviours, and are often considered as a design error. While there is a quadratic-time algorithm detecting races in *Basic Message Sequence Charts (BMSCs)*, the problem is undecidable for *High-level Message Sequence Charts (HMSCs)*. To improve this negative situation for HMSCs, we introduce two new notions: a new concept of race called *trace-race* and an extension of the HMSC formalism with *open coregions*, i.e. coregions that can extend over more than one BMSC. We present three arguments showing benefits of our notions over the standard notions of race and HMSC. First, every trace-race-free HMSC is also race-free. Second, every race-free HMSC can be equivalently expressed as a trace-race-free HMSC with open coregions. Last, the trace-race detection problem for HMSC with open coregions is decidable and PSPACE-complete (the problem is in P if the number of processes and gates is fixed).

*Partially supported by Czech Science Foundation (GAČR), grant No. 201/08/P459.
†Partially supported by Czech Science Foundation (GAČR), grant No. 201/08/P375.

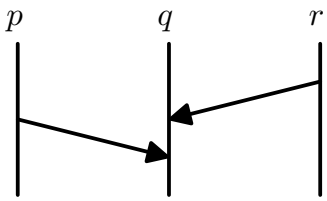

Figure 1: A BMSC containing a race

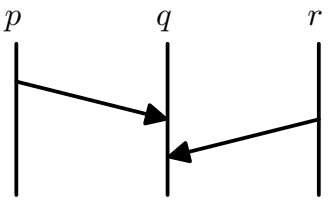

Figure 2: A similar BMSC containing a race

1 Introduction

Message Sequence Chart (MSC) [4] is a popular formalism for specification of systems composed of several asynchronous interacting components (e.g. communication protocols or multi-process systems). Its simplicity and intuitiveness come mainly from the fact that MSC describes only exchange of messages between system components, while other aspects of the system (such as content of the messages, computation steps of the components, etc.) are abstracted away. Even such an incomplete model can indicate serious errors in the designed system. This paper focuses on a common error called *race condition* or simply *race*.

MSCs are based on composition of simple chronograms called *Basic Message Sequence Charts (BMSCs)*. A BMSC consists of a finite number of *processes* and *events*. Each event is associated with some process. A message is represented by a pair of a send event on a sender process and a receive event on a receiver process. Events on each process are ordered according to their chronological succession. Besides this *visual order* $<$, there is also a *causal order* $\ll$, that is weaker than $<$. Intuitively, events e, f are in causal order $e \ll f$, if the BMSC enforces that e always precedes f. There are several definitions of causal order depending on the settings of the modelled system and semantics of the model. For example, if one process sends two messages to another process, the corresponding receive events are causally ordered if and only if the considered message transport protocol has the *FIFO property*: two messages sent from one process to another are always received in the same order. In this paper, we assume that every process has one unbounded buffer for all incoming messages and that the message transport protocol satisfies the FIFO property.

A BMSC contains a *race condition* (or simply *race*) [1] if there are two visually ordered events that are not causally ordered (i.e. they can actually occur in an arbitrary order). For example, Figure 1 depicts that the process q receives a message from r followed by a message from p. As processes and communication in BMSCs are always asynchronous,

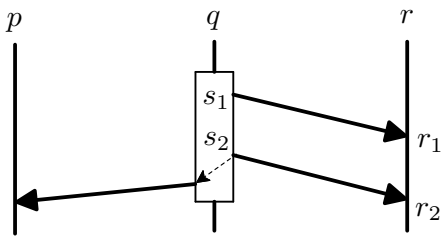

Figure 3: A BMSC containing a race between r_1, r_2

the messages can be also received in the opposite order as shown in Figure 2). In both figures, the two receive events are in race as they are ordered visually but not causally. Races in BMSC description should be considered as a design error, as they exhibit discrepancies between the intended ordering designed in a BMSC, and the ordering in a real system implemented according to this BMSC. Races in a BMSC can be detected in quadratic time [1].

While a BMSC describes only a single and finite communication scenario, its extension called *High-level Message Sequence Chart (HMSC)* [8, 1] can describe more complex interactions, with iterations and alternatives between several scenarios. An HMSC is a finite state transition system where each state is labelled by a BMSC or a reference to another HMSC. In the sequel, we will only consider HMSCs labelled by BMSCs. Each run (i.e. a path starting in the initial state and ending in a final state) in an HMSC can be understood as a single BMSC, which is a concatenation of the BMSCs labelling the states along the run. As a transitions system may contain cycles, the number of its runs may be infinite. Hence, an HMSC represents a potentially infinite set of BMSCs of unbounded size.

The definition of race was extended to HMSCs in [5]. Intuitively, an HMSC H has a race if some BMSC represented by H contains a race and H does not represent any BMSC where the two racing events are defined with the opposite visual order. Unfortunately, the problem whether a given HMSC contains a race is undecidable [5, 4].

In this paper, we propose an alternative definition of race for HMSCs called *trace-race*. Roughly speaking, an HMSC has a trace-race if some BMSC represented by H contains a race. Clearly, every trace-race-free HMSC is also race-free but not vice versa. To improve the expressive power of trace-race-free HMSCs, we extend the HMSC formalism with *open coregions*. A coregion is a standard part of the formalism that allows some events on the same process in a BMSC to be visually unordered. An open coregion is basically a coregion spread over several BMSCs. We present a transformation of an

arbitrary race-free HMSC into an equivalent trace-race-free HMSC with open coregions, where equivalence means that the two HMSCs have the same set of linearizations. Finally, we show that the problem of whether a given HMSC with open coregions contains a trace-race is decidable and PSPACE-complete. In fact, our algorithm works in polynomial time for HMSCs with fixed number of processes and gates. For definitions of gates and linearizations see Sections 2 and 3, respectively.

The rest of the paper is organized as follows. Section 2 recalls the definitions of BMSCs, HMSCs and race condition for BMSCs. The race and trace-race conditions for HMSCs are defined and compared in Section 3. Section 4 is devoted to the translation of race-free HMSCs into equivalent trace-race-free HMSCs. The decidability and time complexity of the trace-race detection problem is discussed in Section 5. Section 6 studies the space complexity is this problem. Section 7 briefly summarizes benefits of the presented notions.

2 Preliminaries

This section provides definitions of BMSCs, race condition for BMSCs, and HMSCs. We omit some features of MSCs given by the ITU standard [4], e.g. atomic actions, labelling of messages with names, timers etc. However, these restrictions are quite common, and our results can be extended to MSCs with atomic actions and message labelling using the technique of [2].

2.1 BMSCs with (open) coregions, gates, and general ordering

The basic concept of BMSCs is described in Section 1. In the visual representation of a BMSC, processes are depicted as vertical lines and messages are represented by arrows between these lines. Events located on the same process line are visually ordered from top to bottom.

A process line may contain segments called *coregions* delimiting subsets of events. Events in a coregion are a priori not in visual order, but they can be visually ordered using a relation called *general ordering*. Note that general ordering need not be a partial order. In the visual representation, coregions are represented by rectangles and general ordering is denoted by dashed arrows between pairs of ordered events (see Figure 3).

In existing MSC formalisms, coregions are limited to finite set of events located in a single BMSC. We extend the standard definition of BMSCs with *open coregions* and

gates. These features allow coregions of arbitrary size, spread over several concatenated BMSCs. Gates enable events of different BMSCs to be generally ordered within the final joined coregion. Similar ideas for connecting orders using gates or predicates was already proposed for instance in [7, 3].

A coregion can be open on top (*top-open coregion*), on bottom (*bottom-open coregion*), or open on both sides. All processes use a common gate name space $\mathcal{G}$. For each process p, we define the sets of *top gates* $p.\overline{\mathcal{G}} = \{p.\overline{g} \mid g \in \mathcal{G}\}$ and *bottom gates* $p.\underline{\mathcal{G}} = \{p.\underline{g} \mid g \in \mathcal{G}\}$ located on process p. Given a BMSC with a set of processes $\mathcal{P}$, we set $\mathcal{P}.\overline{\mathcal{G}} = \bigcup_{p\in\mathcal{P}} p.\overline{\mathcal{G}}$ and $\mathcal{P}.\underline{\mathcal{G}} = \bigcup_{p\in\mathcal{P}} p.\underline{\mathcal{G}}$ to be the sets of all top and bottom gates in this BMSC, respectively. We also extend the general ordering to range over both events and gates within an open coregion.

Definition 2.1. *Let $\mathcal{G}$ be a finite gate name space. A BMSC over $\mathcal{G}$ is a tuple*

$$M = (\mathcal{P}, E_S, E_R, P, \{<_p\}_{p\in\mathcal{P}}, \mathcal{M}, \mathcal{C}, \{\prec_C\}_{C\in\mathcal{C}})$$

where

- *$\mathcal{P}$ is a finite set of processes.*
- *E_S and E_R are disjoint finite sets of* send *and* receive events*, respectively. We set $E = E_S \cup E_R$.*
- *$P : E \to \mathcal{P}$ is a mapping that associates each event with a process.*
- *$<_p$ is a total order on all events on a process p.*
- *$\mathcal{M} \subseteq (E_S \times E_R)$ is a bijective mapping, relating every send with a unique receive. We assume that a process cannot send a message to itself, i.e. $P(e) \neq P(f)$ whenever $(e, f) \in \mathcal{M}$. For any $(e, f) \in \mathcal{M}$, we use $\mathcal{M}(e)$ to denote the receive event f, and $\mathcal{M}^{-1}(f)$ to denote the send event e.*
- *$\mathcal{C}$ is a finite set of pairwise disjoint* coregions*, where a coregion $C \in \mathcal{C}$ is a consistent nonempty subset of events and gates of a single process p, i.e.*
  - *$\emptyset \neq C \subseteq P^{-1}(p) \cup p.\overline{\mathcal{G}} \cup p.\underline{\mathcal{G}}$ for some $p \in \mathcal{P}$*
  - *if $e <_p d <_p f$ and $e, f \in C$, then $d \in C$.*

  A coregion C containing a top gate is called top-open *and it has to contain all top gates $p.\overline{\mathcal{G}}$ and satisfy that if $e <_p f$ and $f \in C$ then $e \in C$. A coregion C containing a bottom*

gate is called bottom-open *and it has to contain all bottom gates* $p.\underline{\mathcal{G}}$ *and satisfy that if* $e <_p f$ *and* $e \in C$ *then* $f \in C$. *A coregion which is both top-open and bottom-open is called just* open.

- $\prec_C$ *is an acyclic relation called* general ordering *on elements in* C *such that* $\prec_C \subseteq (p.\overline{\mathcal{G}} \cup P^{-1}(p)) \times (p.\underline{\mathcal{G}} \cup P^{-1}(p))$, *where* p *is the process containing the coregion* C.

The definition says that a top-open coregion has to contain all top gates. As coregions are pairwise disjoint, there is at most one top-open coregion. Similarly, each BMSC contains at most one bottom-open coregion. Note that we do not impose that coregions contain events. For example, an open coregion covering an inactive process can connect top and bottom gates in a BMSC.

In the visual representation, an open coregion is depicted as a rectangle without the side(s) which are open. Gates are represented by small squares on the corresponding missing side of these rectangles. As gates are always depicted in the same order, their names become redundant (and they are often omitted). Dashed arrows represent general ordering. For example of BMSCs with open coregions see Figure 4. Recall that dashed arrows represent a general ordering.

2.2 Visual order, causal order and race in BMSCs

Every BMSC induces two preorders on events: visual $<$ and causal $\ll$ ordering. The *visual order* represents the order of events directly described by the BMSC. Loosely speaking, $<$ is the reflexive and transitive closure of total orders $<_p$ of events on each process, excluding the order of events within each coregion, plus general ordering and the order generated by the FIFO property and the fact that every send event precedes the corresponding receive event. The visual order is actually defined over the union of events and gates.

Definition 2.2. *Let* $M = (\mathcal{P}, E_S, E_R, P, \{<_p\}_{p\in\mathcal{P}}, \mathcal{M}, \mathcal{C}, \{\prec_C\}_{C\in\mathcal{C}})$ *be a BMSC over* $\mathcal{G}$. *A* visual order $<$ *given by* M *is the least preorder* $< \subseteq (\mathcal{P}.\overline{\mathcal{G}} \cup E) \times (\mathcal{P}.\underline{\mathcal{G}} \cup E)$ *such that*

(i) $<$ *contains the relation*

$$\Big(\bigcup_{p\in\mathcal{P}} <_p \;\setminus\; \bigcup_{C\in\mathcal{C}} C \times C\Big) \;\cup\; \Big(\bigcup_{C\in\mathcal{C}} \prec_C\Big) \;\cup\; \mathcal{M},$$

(ii) $<$ *respects the FIFO property, i.e. for every* $e, f \in E_S$ *such that* $P(e) = P(f)$ *and* $P(\mathcal{M}(e)) = P(\mathcal{M}(f))$*, it holds that* $e < f$ *implies* $\mathcal{M}(e) < \mathcal{M}(f)$.[1]

Note that one can define a BMSC where the preorder $<$ is not a partial order (e.g. a combination of a general order and the FIFO property can induce a cycle $e < f < e$ where $e \neq f$). Such a situation is clearly a design error and can be detected using arbitrary algorithm for cycle detection. In the sequel, we always assume that $<$ is a partial order.

In contrast to the visual order, the *causal order* captures the partial order of events that has to be respected by all executions as it is enforced by the semantics of the design. Hence, the causal order represents the interpretation of a BMSC relevant to its implementation.

Definition 2.3. *Given a BMSC* $M = (\mathcal{P}, E_S, E_R, P, \{<_p\}_{p\in\mathcal{P}}, \mathcal{M}, \mathcal{C}, \{\prec_C\}_{C\in\mathcal{C}})$ *over* $\mathcal{G}$*, we define a* causal order $\ll$ *as the least partial order on* E *such that* $e \ll f$*, if*

- $(e, f) \in \mathcal{M}$*, i.e.* send and receive events of each message are ordered*, or*
- $P(e) = P(f)$ *and* $e < f$ *and* $f \in E_S$*, i.e.* any send event is delayed until all previous events took place*, or*
- $P(e) = P(f)$ *and* $\exists e', f' \in E$ *such that* $e' < f'$*,* $P(e') = P(f')$*,* $(e', e) \in \mathcal{M}$ *and* $(f', f) \in \mathcal{M}$*, i.e.* causal order respects the FIFO property*.*

Lemma 2.4. *For every BMSC it holds that* $\ll \subseteq <$*. Moreover, for each send event* $f \in E_S$ *it holds* $e < f \iff e \ll f$*.*

Proof. The relation $\ll \subseteq <$ follows directly from the definitions of the orders. Hence, it remains to show that for each send event f it holds that $e < f$ implies $e \ll f$.

Let e, b be events satisfying $e < f$ and f is a send event. Further, let $\prec$ be a relation defined in the same way as $<$ but without the application of the reflexive and transitive closure. In other words, $\prec$ is the least set such that it

(i) contains the relation

$$\Big(\bigcup_{p\in\mathcal{P}} <_p \;\setminus\; \bigcup_{C\in\mathcal{C}} C \times C\Big) \;\cup\; \big(\bigcup_{C\in\mathcal{C}} \prec_C\big) \;\cup\; \mathcal{M},$$

[1]The FIFO property is usually not included in the definition of a visual order. However, once we choose the FIFO message passing setting, violating this property should be considered as a design error and it can be easily detected. Hence, we included the property directly in our definition.

(ii) respects the FIFO property, i.e. for every $e, f \in E_S$ such that $P(e) = P(f)$ and $P(\mathcal{M}(e)) = P(\mathcal{M}(f))$, it holds that $e < f$ implies $\mathcal{M}(e) \prec \mathcal{M}(f)$.

As $e < f$, there exists a sequence of events $e_0, e_1, \ldots, e_n$ such that $e = e_0 \prec e_1 \prec \ldots \prec e_n = f$. We prove by induction that on n that $e \ll f$.

Base case $n = 0$, i.e. $e = f$: As $\ll$ is reflexive, $e \ll f$.

Inductive step $n > 0$: Let e_i be the first event in the sequence $e_0, \ldots, e_n$ such that $P(e_i) = P(f)$. Definition of $\ll$ directly implies that $e_i \ll f$. If $e_i = e_0 = e$, we are done. Otherwise $P(e_{i-1}) \neq P(e)$ and the definition of $\prec$ implies that $(e_{i-1}, e_i) \in \mathcal{M}$. Hence, $e_{i-1} \ll e_i$. Further, e_{i-1} is a send action and by induction hypothesis we get $e \ll e_{i-1}$. To sum up, $e \ll e_{i-1} \ll e_i \ll f$ and thus $e \ll f$.

□

A race is defined as a difference between visual and causal order on events.

Definition 2.5. *If a BMSC contains some events e, f satisfying $e < f$ and $e \not\ll f$, we say that the BMSC contains a* race (between events e, f). *Otherwise, the BMSC is called* race-free.

Theorem 2.6 ([1])**.** *The problem whether a given BMSC with n events contains a race is decidable in time $\mathcal{O}(n^2)$.*

Note that [1] deals with BMSCs without any coregions. However, an extension of this theorem to BMSCs with (possibly open) coregions and general ordering is straightforward.

The following lemma says that a BMSC contains a race if and only if it contains a race between two events on the same process.

Lemma 2.7. *A BMSC contains a race if and only if there are two events e, f such that $e < f$, $e \not\ll f$, and $P(e) = P(f)$.*

Proof. The implication "$\Longleftarrow$" follows immediately from the definition of race. We prove the other direction. Let us assume that a BMSC $(\mathcal{P}, E_S, E_R, P, \{<_p\}_{p\in\mathcal{P}}, \mathcal{M}, \mathcal{C}, \{\prec_C\}_{C\in\mathcal{C}})$ contains a race between events e', f', i.e. $e' < f'$ and $e' \not\ll f'$. Further, let $\prec$ be a relation defined in the same way as $<$ but without the application of the reflexive and transitive closure. In other words, $\prec$ is the least set such that it

(i) contains the relation

$$\Big(\bigcup_{p\in\mathcal{P}} <_p \;\setminus\; \bigcup_{C\in\mathcal{C}} C\times C\Big) \;\cup\; \big(\bigcup_{C\in\mathcal{C}} \prec_C\big) \;\cup\; \mathcal{M},$$

(ii) respects the FIFO property, i.e. for every $e, f \in E_S$ such that $P(e) = P(f)$ and $P(\mathcal{M}(e)) = P(\mathcal{M}(f))$, it holds that $e < f$ implies $\mathcal{M}(e) \prec \mathcal{M}(f)$.

As $e' < f'$, there exist events $e_0, e_1, \ldots, e_n$ such that $e' = e_0 \prec e_1 \prec \ldots \prec e_n = f'$. Note that $n > 0$ as $e' \not\ll f'$ and $\ll$ is reflexive. We prove by induction on n that there exist e, f satisfying $e < f$, $e \not\ll f$, and $P(e) = P(f)$.

Base case $n = 1$, i.e. $e' \prec f'$: As $e' \not\ll f'$, we get that $(e', f') \notin \mathcal{M}$ and, similarly, (e', f') could not have been added to $\prec$ as a result of the FIFO property. Definition of $\prec$ therefore implies that $(e', f') \in <_p$ for some $p \in \mathcal{P}$ or $(e', f') \in \prec_C$ for some $C \in \mathcal{C}$. In both cases, it holds that $P(e') = P(f')$.

Inductive step $n > 1$: Either $e_1 \ll f'$, which implies $e' \not\ll e_1$ and then also $P(e') = P(e_1)$ using same argument as in the base case; or $e_1 \not\ll f'$ and the statement follows directly from the induction hypothesis.

□

2.3 HMSCs

An HMSC is a finite directed graph with an *initial* state and a set of *final* states, where each state is labelled with a BMSC.

Definition 2.8. *An* HMSC *is a tuple* $(S, \to, s_0, S_F, L, \mathcal{L})$ *where*

- S *is a finite set of states,* $s_0 \in S$ *is an* initial state, $S_F \subseteq S$ *is a set of* final states,
- $\to \subseteq S \times S$ *is a transition relation,*
- $\mathcal{L}$ *is a finite set of BMSCs over a common gate name space,*
- $L(s) : S \to \mathcal{L}$ *is a mapping assigning to each state a BMSC.*

A sequence of states $\sigma = s_1 s_2 \cdots s_k$ *is a* path, *if* $(s_i, s_{i+1}) \in \to$ *for every* $1 \le i < k$. *A path is a* run *if* $s_1 = s_0$ *and* $s_k \in S_F$.

To give a semantics of HMSCs, we need to define a concatenation operation on BMSCs. Intuitively, the concatenation of BMSCs M_1 and M_2 is done by gluing the corresponding process lines together with the BMSC M_2 drawn beneath M_1. If M_1 and M_2 contain bottom-open and top-open coregions on a process p, respectively, then the two coregions are merged and each bottom gate $p.\underline{g}$ of the upper open coregion is joined with the corresponding top gate $p.\overline{g}$ of the lower open coregion. Further, whenever an event e of the upper coregion is generally ordered with a joined gate and this joined gate is generally ordered with an event f of the lower coregion, the events e, f become generally ordered in the newly created coregion. The joined gates are then removed. If M_1 contains a bottom-open coregions on a process p that is not in M_2, then the coregion remains bottom-open. However, if M_1 contains a bottom-open coregion on a process p and M_2 contains the process p without any top-open coregion on it, then the bottom side of the coregion is closed.

Definition 2.9. *Let* $M_i = (\mathcal{P}_i, E_{Si}, E_{Ri}, P_i, \{<_{ip}\}_{p\in\mathcal{P}}, \mathcal{M}_i, \mathcal{C}_i, \{\prec_{iC}\}_{C\in\mathcal{C}_i})$ *for* $i = 1, 2$ *be two BMSCs over a common gate name space* $\mathcal{G}$ *and such that the sets* $E_{S1} \cup E_{R1}$ *and* $E_{S2} \cup E_{R2}$ *are disjoint (we can always rename events so that the sets become disjoint). The concatenation of* M_1 *and* M_2 *is the BMSC* $M_1 \cdot M_2 = (\mathcal{P}_1 \cup \mathcal{P}_2, E_{S1} \cup E_{S2}, E_{R1} \cup E_{R2}, P_1 \cup P_2, \{<_p\}_{p\in\mathcal{P}}, \mathcal{M}_1 \cup \mathcal{M}_2, \mathcal{C}, \{\prec_C\}_{C\in\mathcal{C}})$ *where*

$$<_p = \begin{cases} <_{1p} \text{ if } p \in \mathcal{P}_1 \setminus \mathcal{P}_2 \\ <_{2p} \text{ if } p \in \mathcal{P}_2 \setminus \mathcal{P}_1 \\ \text{transitive closure of } <_{1p} \cup <_{2p} \cup (P_1^{-1}(p) \times P_2^{-1}(p)) \\ \text{if } p \in \mathcal{P}_1 \cap \mathcal{P}_2 \end{cases}$$

and $\mathcal{C}$ *contains all coregions* C *of the following five kinds:*

1. $C \in \mathcal{C}_1$ *and* C *is not bottom-open or* C *is on a process* $p \in \mathcal{P}_1 \setminus \mathcal{P}_2$*. We set* $\prec_C = \prec_{1C}$*.*

2. $C \in \mathcal{C}_2$ *and* C *is not top-open or* C *is on a process* $p \in \mathcal{P}_2 \setminus \mathcal{P}_1$*. We set* $\prec_C = \prec_{2C}$*.*

3. $C = C_1 \setminus p.\underline{\mathcal{G}}$ *for some* $C_1 \in \mathcal{C}_1$ *such that* $p.\underline{\mathcal{G}} \subseteq C_1$ *and* $p.\overline{\mathcal{G}} \cap C_2 = \emptyset$ *for all* $C_2 \in \mathcal{C}_2$*, i.e.* C *corresponds to a coregion of* $\mathcal{C}_1$ *that is bottom-open but there is no matching top-open coregion in* $\mathcal{C}_2$ *(note that* C *is closed on the bottom side). We set* $\prec_C = \prec_{1C_1} \cap (C \times C)$*.*

4. $C = C_2 \setminus p.\overline{\mathcal{G}}$ *for some* $C_2 \in \mathcal{C}_2$ *such that* $p.\overline{\mathcal{G}} \subseteq C_2$ *and* $p.\underline{\mathcal{G}} \cap C_1 = \emptyset$ *for all* $C_1 \in \mathcal{C}_1$*, i.e.* C *corresponds to a coregion of* $\mathcal{C}_2$ *that is top-open but there is no matching bottom-open coregion in* $\mathcal{C}_1$*. We set* $\prec_C = \prec_{2C_2} \cap (C \times C)$*.*

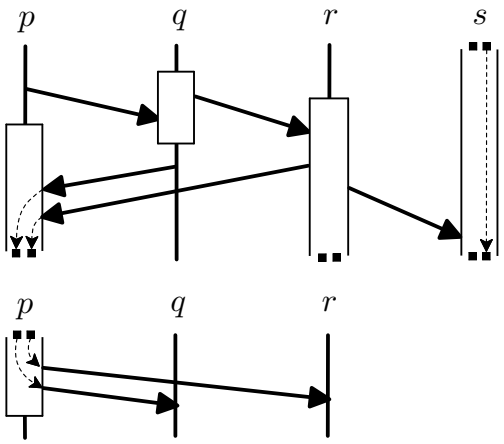

Figure 4: BMSCs M_1 (upper) and M_2

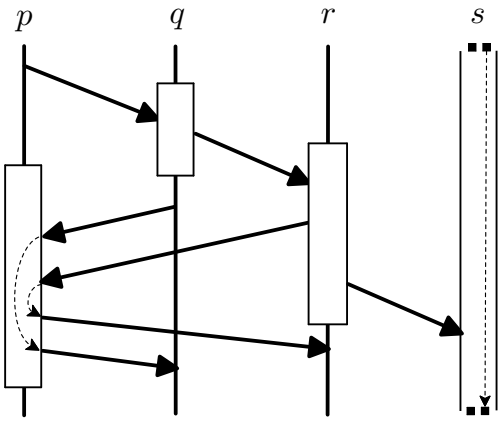

Figure 5: Concatenation $M_1 \cdot M_2$

5. $C = (C_1 \setminus p.\underline{\mathcal{G}}) \cup (C_2 \setminus p.\overline{\mathcal{G}})$ *for some* $C_1 \in \mathcal{C}_1$ *and* $C_2 \in \mathcal{C}_2$ *satisfying* $p.\underline{\mathcal{G}} \subseteq C_1$ *and* $p.\overline{\mathcal{G}} \subseteq C_2$*, i.e.* C *is a bottom-open coregion of* $\mathcal{C}_1$ *merged with the matching top-open coregion of* $\mathcal{C}_2$*. We set*

$$\prec_C = \{(e,f) \mid \Big((e,f) \in \prec_{1C_1} \text{ and } f \notin p.\underline{\mathcal{G}}\Big) \text{ or } \Big((e,f) \in \prec_{2C_2} \text{ and } e \notin p.\overline{\mathcal{G}}\Big) \\ \text{or } \Big((e, p.\underline{g}) \in \prec_{1C_1} \text{ and } (p.\overline{g}, f) \in \prec_{2C_2} \text{ for some } g \in \mathcal{G}\Big)\}.$$

Note that if visual orders of M_1 and M_2 are partial orders, then the visual order of $M_1 \cdot M_2$ is also a partial order. An example of two BMSCs and their concatenation is provided by Figures 4 and 5.

Each path $s_1 s_2 \cdots s_k$ of an HMSC represents a single BMSC given by concatenation of the BMSCs assigned to $s_1, s_2, \ldots, s_k$, i.e. a path $\sigma = s_1 s_2 \cdots s_k$ represents the BMSC $L(\sigma) = L(s_1) \cdot L(s_2) \cdot \ldots \cdot L(s_k)$. Hence, an HMSC represents a set of BMSCs corresponding

to its runs. As an HMSC may contain a cycle, the represented set of BMSCs can be infinite and there is no bound on the size (i.e. number of events) of such BMSCs.

3 Race conditions in HMSCs

First we explain the idea of race conditions for a set of BMSCs. Let us consider a system where two processes p and r send a message to a third process q, that receives them in arbitrary order. This behaviour can be specified (even without any coregion) by two BMSCs depicted in Figures 1 and 2. Even if both BMSCs contain a race, the specification given by this pair of BMSCs should be considered as race-free because both permutations of the two receive events on process q allowed by causal ordering are included in the specification.

The race condition for a set of BMSCs can formulated very simply using the following terminology. An *execution induced by a BMSC* M is a totally ordered set $(E, \subset)$, where E is the set of events of M and $\subset$ is a linear extension of the causal order $\ll$ given by M. We say that such an execution $(E, \subset)$ *corresponds to a BMSC* M' if M' has the same set of events and $\subset$ is a linear extension of the visual order $<$ of M'.

Definition 3.1. *We say that a set of BMSCs contains a* race *if there exists an execution induced by some BMSC of the set and not corresponding to any BMSC of the set.*

The race condition for HMSCs introduced in [5] follows the same principle. Unfortunately, we cannot directly say that an HMSC contains a race if it represents a set of BMSCs containing a race. The problem is that the BMSCs represented by the HMSC are constructed with the concatenation operation during which events can be renamed. Therefore, the events are replaced by *labels* keeping the information about sending and receiving processes. Further, the linearly ordered executions are replaced by words called *linearizations*.

Definition 3.2. *Let* $M = (\mathcal{P}, E_S, E_R, P, \{<_p\}_{p\in\mathcal{P}}, \mathcal{M}, \mathcal{C}, \{\prec_C\}_{C\in\mathcal{C}})$ *be a BMSC. We define an auxiliary function label* $: E \to \{p!q, p?q \mid p, q \in \mathcal{P}\}$ *such that*

$$label(e) = \begin{cases} p!q & \text{if } e \in E_S,\ p = P(e), \text{ and } q = P(\mathcal{M}(e)) \\ p?q & \text{if } e \in E_R,\ p = P(e), \text{ and } q = P(\mathcal{M}^{-1}(e)). \end{cases}$$

A linearization *of* M *w.r.t. a partial order* $\sqsubset \in \{<, \ll\}$ *is a word* $label(e_1)label(e_2)\cdots label(e_n)$ *such that* $E = \{e_1, e_2, \cdots, e_n\}$ *and* $e_i \sqsubset e_j$ *implies* $i < j$. *Moreover, we define* $Lin_\sqsubset(M)$ *to be the set of all linearizations of* M *w.r.t* $\sqsubset$.

Intuitively, $Lin_{\ll}(M)$ represents all executions induced by M, while $Lin_{<}(M)$ represents all executions corresponding to M.

The definition of linearization can be extended to HMSCs as follows.

Definition 3.3. Linearizations *of an HMSC* H *w.r.t.* $\sqsubset \in \{<, \ll\}$ *is the set*

$$Lin_{\sqsubset}(H) = \bigcup_{\sigma \text{ is a run of } H} Lin_{\sqsubset}(L(\sigma)).$$

Now we can recall the race condition for HMSCs introduced in [5].

Definition 3.4. *An HMSC* H *contains a* race *if* $Lin_{<}(H) \neq Lin_{\ll}(H)$.

This definition of race has several drawbacks. First of all, the problem whether an HMSC contains a race is undecidable even if we restrict the problem to HMSCs without coregions [5, 6]. Further, as soon as we consider HMSCs with coregions, this notion of race does not tally with the definition of race for BMSCs. For example, the BMSC drawn in Figure 3 contains a race as the messages from q to r can be sent in arbitrary order while the receive events r_1, r_2 are visually ordered. If we look at this BMSC as an HMSC with only one state, then there is no race with respect to Definition 3.4 as both events r_1, r_2 are represented in linearizations by the same label $r?q$ and therefore the information about their order is lost.

A simple definition of a trace-race follows.

Definition 3.5. *An HMSC* H *contains a* trace-race *if there is a run* σ *of* H *such that the BMSC* $L(\sigma)$ *contains a race.*

If an HMSC H contains no trace-race, then each its run σ represents a race-free BMSC $L(\sigma)$. As visual and causal orders of a race-free BMSC coincide, we get that $Lin_{<}(L(\sigma)) = Lin_{\ll}(L(\sigma))$. Hence, every trace-race-free HMSC is also race-free.

The inverse implication does not hold. For example, the race-free HMSC of Figure 6 has a trace-race (the HMSC describes the system discussed at the beginning of this section).

As the definition of trace-race does not replace events by labels, trace-race tallies with the definition of race for BMSCs, i.e. a BMSC has a race if and only if it has a trace-race when seen as a single state HMSC.

It is commonly agreed that designers should avoid races in HMSCs. We provide an intuitive explanation why we think that designers should actually avoid trace-races as well. Let H be a race-free HMSC with a trace-race. As H has a trace-race, there has

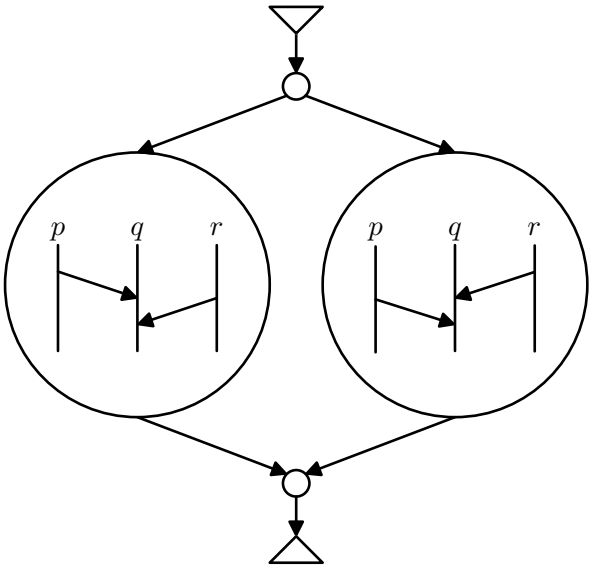

Figure 6: An race-free HMSC with a trace-race

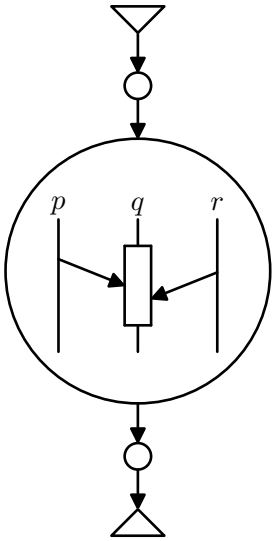

Figure 7: A trace-race-free HMSC

to be a run σ such that $L(\sigma)$ induces an execution not corresponding to $L(\sigma)$. As H is race-free, this execution corresponds to some BMSC $L(\sigma')$, where $\sigma' \neq \sigma$ is another run of H.[2] Lemma 2.4 implies that all executions corresponding to a run are also induced by the run. Hence, the execution is in fact induced by (at least) two different runs of the HMSC. This is a potential source of errors as an implementation of this kind of description tends to violate the "write things once" programming principle. Moreover, trace-race-free HMSCs are usually more compact and their use may encourage a cleaner way of the system design. For example, compare the trace-race-free system depicted on Figure 7, which models the same behaviour as the race-free HMSC of Figure 6.

4 Transformation of HMSCs into trace-race-free HMSCs

We present a transformation of an arbitrary HMSC H into a trace-race-free HMSC H'. This transformation consists only in relabelling the states of the original HMSC with BMSCs that have the same processes, events, and causal orders as the original BMSCs, but they induce different visual orders. As the structure of H' remains the same, it has the same set of runs as H. The HMSC is changed in such a way that both visual and causal orders of the BMSC corresponding to a run σ in H' are the same as the causal order of the BMSC corresponding to σ in H. Hence, $Lin_<(H') = Lin_{\ll}(H') = Lin_{\ll}(H)$ and H' is trace-race-free. Moreover, if H was race-free (i.e. $Lin_<(H) = Lin_{\ll}(H)$), then

[2] In fact, the linearization corresponding to the mentioned execution has to be in $Lin_<(L(\sigma'))$.

$Lin_<(H) = Lin_\ll(H) = Lin_<(H') = Lin_\ll(H')$ and we say that H and H' are equivalent. The transformation of the original HMSC H proceeds in two steps.

1. We modify each BMSC M in a state of H such that each process is covered with a coregion open on both sides, while BMSCs represented by the resulting HMSC remain the same as those represented by H: the same events on the same processes with the same visual and causal orders. We use general orderings and two fresh gate names pre, suc to induct the same visual (and hence also causal) orders. The definition of concatenation implies that, on a process p, all events of BMSCs preceding M in some run of H are visually ordered before all events of M (except those in a top-open coregion). This relation is preserved using the gate $p.\overline{pre}$. Similarly, all events of M (except those in a bottom-open coregion) are visually ordered before all events of BMSCs succeeding M in some run of H. This relation is preserved using the gate $p.\underline{suc}$.

   More precisely, the general ordering $\prec_{C'}$ of a coregion C' open on both sides and covering a process p is defined as the least relation satisfying the following conditions (where $\mathcal{G}' = \mathcal{G} \cup \{pre, suc\}$ and $<$ refers to the original visual order of M):

   - For every $e \in (E \cup p.\overline{\mathcal{G}}), f \in (E \cup p.\underline{\mathcal{G}})$, if $e < f$, then $(e, f) \in \prec_{C'}$.
   - For every $e \in E$, if e is not in a top-open coregion in M, then $(p.\overline{pre}, e) \in \prec_{C'}$.
   - For every $e \in E$, if e is not in a bottom-open coregion in M, then $(e, p.\underline{suc}) \in \prec_{C'}$.
   - $p.\overline{suc} \times (E \cup p.\underline{\mathcal{G}'}) \subseteq \prec_{C'}$.
   - $(E \cup p.\overline{\mathcal{G}'}) \times p.\underline{pre} \subseteq \prec_{C'}$.

2. Now we restrict the general orderings to induce visual orders equivalent to the original causal orders. Due to Definition 2.3, it is sufficient to generally order pairs (e, f) such that $e < f$ and $f \in E_S$ (where $<$ refers to the original visual order). Formally, we replace every general ordering $\prec_{C'}$ computed in the previous step with $\prec_{C'} \cap ((\mathcal{P}.\overline{\mathcal{G}'} \cup E) \times (\mathcal{P}.\underline{\mathcal{G}'} \cup E_s))$.

5 Trace-race detection problem for HMSCs

This section deals with decidability and time complexity of the *trace-race detection problem*, i.e. the problem whether a given HMSC (possibly with open coregions and gates)

contains a trace-race. We assume that each state of a given HMSC $H = (S, \to, s_0, S_F, L, \mathcal{L})$ appears on some run of H (the states violating this property can be easily detected and eliminated). Recall that H contains a trace-race if and only if there is a run σ such that the BMSC $L(\sigma)$ contains a race. There is such a run if and only if there is a path $\pi = s_0 s_1 s_2 \dots s_k$ starting in the initial state and such that

1. the BMSC $L(s_k)$ contains a race, or
2. there is an event e in a race-free BMSC $L(s_0 \dots s_{k-1})$ and an event f in a race-free BMSC $L(s_k)$ such that $L(\pi)$ contains a race between e and f.

The races of the first kind can be easily detected due to Theorem 2.6. In the rest of this section, we focus on detection of the races of the second kind. The problem cannot be directly reduced to the detection of races in BMSCs as there could be infinitely many paths starting in s_0. Our detection technique relies on a precise characterization of races appearing in concatenation $M_1 \cdot M_2$ of two race-free BMSCs M_1, M_2. For this, we need two new functions describing sets of *joined gates* of the form $p.g$. Intuitively, a joined gate $p.g$ for a concatenation $M_1 \cdot M_2$ is a reference to a gate that appears as a bottom gate $p.\underline{g}$ in M_1 and is identified with the corresponding top gate $p.\overline{g}$ of M_2 during concatenation. We denote by $p.\mathcal{G}$ and $\mathcal{P}.\mathcal{G}$ the sets all joined gates over gate name space $\mathcal{G}$ and the process p or all processes $\mathcal{P}$, respectively.

Definition 5.1. *Given a BMSC* $M = (\mathcal{P}, E_S, E_R, P, \{<_p\}_{p\in\mathcal{P}}, \mathcal{M}, \mathcal{C}, \{\prec_C\}_{C\in\mathcal{C}})$ *over gate name space* $\mathcal{G}$*, we define two functions* $\downarrow(), \uparrow() : (E_S \cup E_R) \to 2^{\mathcal{P}.\mathcal{G}}$ *as*

$$\downarrow(e) = \{p.g \in \mathcal{P}.\mathcal{G} \mid e < p.\underline{g}\} \quad \textit{and} \quad \uparrow(e) = \{p.g \in \mathcal{P}.\mathcal{G} \mid p.\overline{g} < e\}.$$

In the context of a concatenation $M_1 \cdot M_2$, $\downarrow(e)$ and $\uparrow(e)$ always refer to the values of these functions in the BMSC M_i (where $i \in \{1, 2\}$) originally containing the event e. The characterization of races in $M_1 \cdot M_2$ is formulated in Lemma 5.6. The lemma assumes that the two race-free BMSCs M_1, M_2 are in the *special form* where all events and gates on each process are covered by a coregion open on both sides. Note that every BMSC can be converted to this form by the first step of the transformation presented in the previous section. Lemma 5.6 with the characterization is preceded by four auxiliary lemmata.

Lemma 5.2. *Let* e, f *be events of a BMSC such that* f *is a receive event. It holds that*

$$e \ll f \iff e < \mathcal{M}^{-1}(f) \vee (P(e) = P(f) \wedge P(\mathcal{M}^{-1}(e)) = P(\mathcal{M}^{-1}(f)) \wedge \mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f)).$$

Proof. The implication "$\Longleftarrow$" follows immediately from the definition of $\ll$: if $e < \mathcal{M}^{-1}(f)$, then Lemma 2.4 implies $e \ll \mathcal{M}^{-1}(f)$ and thus $e \ll f$; if the second part of the disjunction holds, $e \ll f$ due to the FIFO property.

We prove the direction "$\Longrightarrow$". Let us assume that $e \ll f$. Further, let $\prec\!\!\!\ll$ be a relation defined in the same way as $\ll$ but without the application of the reflexive and transitive closure. In other words, $\prec\!\!\!\ll$ is the least set such that $e \prec\!\!\!\ll f$ iff

(i) $(e, f) \in \mathcal{M}$, i.e. *send and receive events of each message are ordered*, or

(ii) $P(e) = P(f)$ and $e < f$ and $f \in E_S$, i.e. *any send event is delayed until all previous events took place*, or

(iii) $P(e) = P(f)$ and $\exists e', f' \in E$ such that $e' < f'$, $P(e') = P(f')$, $(e', e) \in \mathcal{M}$ and $(f', f) \in \mathcal{M}$, i.e. $\prec\!\!\!\ll$ satisfies the FIFO property.

As $e \ll f$, there exist events $e_0, e_1, \ldots, e_n$ such that $e = e_0 \prec\!\!\!\ll \ldots \prec\!\!\!\ll e_{n-1} \prec\!\!\!\ll e_n = f$. We prove the Lemma by induction on n:

Base case $n = 0$, i.e. $e = f$: Then $\mathcal{M}^{-1}(e) = \mathcal{M}^{-1}(f)$ and thus the second part of the disjunction holds.

Base case $n = 1$, i.e. $e \prec\!\!\!\ll f$: As f is a receive event, we get $e \prec\!\!\!\ll f$ either due to (i) (for which the first part of the disjunction holds), or due to (iii) (for which the second part of the disjunction is satisfied).

Inductive step $n > 1$: If e_{n-1} is a send event, i.e. $e_{n-1} = \mathcal{M}^{-1}(f)$, then $e \ll e_{n-1}$ and therefore also $e < e_{n-1} = \mathcal{M}^{-1}(f)$. When e_{n-1} is a receive event, we get $\mathcal{M}^{-1}(e_{n-1}) < \mathcal{M}^{-1}(f)$ (as $e_{n-1} \prec\!\!\!\ll f$ due to (iii)) and we may use the induction hypothesis:

- either $e < \mathcal{M}^{-1}(e_{n-1})$ which then gives us also $e < \mathcal{M}^{-1}(f)$;
- or the second part of the disjunction holds for e and e_{n-1}, and therefore it holds also for e and f.

□

Lemma 5.3. *Let $M_1 \cdot M_2$ be a concatenation of two BMSCs, e be an event of M_1, and f be an event of M_2. It holds that*

$$\downarrow(e) \cap \uparrow(f) \neq \emptyset \implies e < f.$$

Proof. Let $p.g \in \downarrow(e) \cap \uparrow(f)$ for some $p \in \mathcal{P}$. Hence, Hence, $e < p'.\underline{g'}$ in M_1 and $p'.\overline{g'} < f$ in M_2. Due to definition of concatenation, it holds that $e < f$ in $M_1 \cdot M_2$. □

Lemma 5.4. *Let $M_1 \cdot M_2$ be a concatenation of two race-free BMSCs in the special form, e be an event of M_1, and f be an event of M_2. It holds that*

$$\begin{aligned} \downarrow(e) \cap \uparrow(f) = \emptyset \quad \Longrightarrow \quad & e \not< f \vee \\ & \exists \textit{ a receive event } f' \textit{ in } M_2 \textit{ such that } f' < f \wedge \\ & \textit{label}(e) = \textit{label}(f') \wedge \mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f'). \end{aligned}$$

Proof. We prove the following equivalent statement instead.

$$\begin{aligned} e < f \quad \Longrightarrow \quad & \downarrow(e) \cap \uparrow(f) \neq \emptyset \vee \\ & \exists \text{ a receive event } f' \text{ in } M_2 \text{ such that } f' < f \wedge \\ & \textit{label}(e) = \textit{label}(f') \wedge \mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f'). \end{aligned}$$

Let $\prec$ be a relation defined in the same way as $<$ but without the application of the reflexive and transitive closure. In other words, $\prec$ is the least set such that it

(i) contains the relation[3] $(\bigcup_{C \in \mathcal{C}} \prec_C) \cup \mathcal{M}$,

(ii) respects the FIFO property, i.e. for every $e', f' \in E_S$ such that $P(e') = P(f')$ and $P(\mathcal{M}(e')) = P(\mathcal{M}(f'))$, it holds that $e' < f'$ implies $\mathcal{M}(e') \prec \mathcal{M}(f')$.

As $e < f$, there exist events $e_0, e_1, \ldots, e_n$ such that $e = e_0 \prec e_1 \prec \ldots \prec e_n = f$. Note that $n > 0$ as e and f were originally in different BMSCs. We prove the statement by induction on the weighted length of the sequence of $\prec$ steps. The weight of each step induced by the item (i) is 1. The weight of each step $\mathcal{M}(e') \prec \mathcal{M}(f')$ induced by the item (ii) is equal to the least weight of a sequence $e' \prec \ldots \prec f'$ for $e' < f'$.

Base case $n = 1$, i.e. $e \prec f$: First note, that $(e, f) \notin \mathcal{M}$ as each originates from a different BMSC. Therefore, from the definition of the $\prec$ relation, $P(e) = P(f) = p$. There are two cases:

- if (e, f) has been added to $\prec$ due to the FIFO property (item (ii) of the $\prec$ definition), then $f = f'$ is a receive event such that $\textit{label}(e) = \textit{label}(f')$ and $\mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f')$ hold;
- if $(e, f) \in (\bigcup_{C \in \mathcal{C}} \prec_C)$, then e and f must have been connected through some gate joined during the concatenation, i.e. $\downarrow(e) \cap \uparrow(f) \neq \emptyset$.

[3] As the BMSCs are in the special form, it holds $\left(\bigcup_{p \in \mathcal{P}} <_p\right) \setminus \left(\bigcup_{C \in \mathcal{C}} C \times C\right) = \emptyset$.

Inductive step $n > 1$: We have $e \prec e_1 < f$.

Let e_1 be in M_1. The induction hypothesis for $e_1 < f$ implies that

- either $\downarrow(e_1) \cap \uparrow(f) \neq \emptyset$: As $e < e_1$, it holds that $\downarrow(e) \supseteq \downarrow(e_1)$ and so $\downarrow(e) \cap \uparrow(f) \neq \emptyset$;
- or $\exists$ a receive event f' in M_2 such that $f' < f \wedge \mathit{label}(e_1) = \mathit{label}(f') \wedge \mathcal{M}^{-1}(e_1) < \mathcal{M}^{-1}(f')$: As M_1 is race-free, $e < e_1$ implies that $e \ll e_1$. Hence, due to Lemma 5.2, it holds that
  - either $e < \mathcal{M}^{-1}(e_1)$: Due to induction hypothesis for $\mathcal{M}^{-1}(e_1) < \mathcal{M}^{-1}(f')$, it holds that $\downarrow(\mathcal{M}^{-1}(e_1)) \cap \uparrow(\mathcal{M}^{-1}(f')) \neq \emptyset$ (the second disjunct cannot be valid as $\mathcal{M}^{-1}(e_1)$ is a send event). From $e < \mathcal{M}^{-1}(e_1)$ and $f' < f$, it follows that $\downarrow(e) \cap \uparrow(f) \neq \emptyset$;
  - or $P(e) = P(e_1) \wedge P(\mathcal{M}^{-1}(e)) = P(\mathcal{M}^{-1}(e_1)) \wedge \mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(e_1))$: Hence, $\mathit{label}(e) = \mathit{label}(e_1) = \mathit{label}(f')$ and $\mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(e_1) < \mathcal{M}^{-1}(f')$.

Let e_1 be in M_2. The induction hypothesis for $e < e_1$ implies that

- either $\downarrow(e) \cap \uparrow(e_1) \neq \emptyset$: As $e_1 < f$, it holds that $\uparrow(e_1) \subseteq \uparrow(f)$ and so $\downarrow(e) \cap \uparrow(f) \neq \emptyset$;
- or $\exists$ a receive event f' in M_2 such that $f' < e_1 \wedge \mathit{label}(e) = \mathit{label}(f') \wedge \mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f')$: As $e_1 < f$, it holds that $f' < f$.

□

Lemma 5.5. *Let $M_1 \cdot M_2$ be a concatenation of two race-free BMSCs in the special form, e be an event of M_1, and f be an event of M_2. It holds that*

$$\begin{aligned} e < f \iff\ & \downarrow(e) \cap \uparrow(f) \neq \emptyset \vee \\ & \exists \textit{ a receive event } f' \textit{ in } M_2 \textit{ such that } f' < f \wedge \\ & \mathit{label}(e) = \mathit{label}(f') \wedge \mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f'). \end{aligned}$$

Proof. This lemma follows from Lemma 5.3, Lemma 5.4, and the fact that

$$\begin{aligned} &\exists \text{ a receive event } f' \text{ in } M_2 \text{ such that } f' < f \wedge \\ &\quad \mathit{label}(e) = \mathit{label}(f') \wedge \mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f') \implies e < f. \end{aligned}$$

□

Now we formulate and prove the characterization.

Lemma 5.6. *Let $M_1 \cdot M_2$ be a concatenation of two race-free BMSCs M_1, M_2 in the special form, e be an event of M_1, and f be an event of M_2 such that $P(e) = P(f) = p$. Then e and f are in race if and only if all the following conditions hold.*

1. f *is a receive event*
2. $\downarrow(e) \cap \uparrow(f) \cap p.\mathcal{G} \neq \emptyset$
3. $\downarrow(e) \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset$
4. $label(e) = label(f) \implies \downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset$
5. $\forall$ *receive events* f' *of* M_2 *such that* $f' < \mathcal{M}^{-1}(f)$ *:*
   $label(e) = label(f') \implies \downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f')) = \emptyset$

Proof. "$\implies$": We divide the proof into the following cases.

(1) If f is not a receive event, then $e < f \implies e \ll f$ and so, e and f are not in race.

(2) Due to Lemma 5.3, $\downarrow(e) \cap \uparrow(\mathcal{M}^{-1}(f)) \neq \emptyset$ implies $e < \mathcal{M}^{-1}(f)$. As $\mathcal{M}^{-1}(f)$ is a send event, it holds that $e \ll \mathcal{M}^{-1}(f)$. Hence, $e \ll f$ and e and f are not in race.

(3) Similarly, due to Lemma 5.3, $\downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f)) \neq \emptyset$ implies that $\mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f)$. Therefore, $label(e) = label(f) \wedge \downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f)) \neq \emptyset$ implies (due to the FIFO property) that $e \ll f$. Hence, e and f are not in race.

(4) Let f' be a receive event of M_2 such that $f' < \mathcal{M}^{-1}(f)$, $label(e) = label(f')$, and $\downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f')) \neq \emptyset$. Due to Lemma 5.3, $\downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f')) \neq \emptyset$ implies that $\mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f')$. Therefore, $label(e) = label(f') \wedge \downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f')) \neq \emptyset$ implies (due to the FIFO property) that $e \ll f'$. As $f' < \mathcal{M}^{-1}(f)$ implies $f' \ll \mathcal{M}^{-1}(f)$, it holds that $e \ll f$, and so e and f are not in race.

(5) It remains to show that if $\downarrow(e) \cap \uparrow(f) \cap p.\mathcal{G} = \emptyset$, then e and f are not in race. We discuss two cases.

- Let $\downarrow(e) \cap \uparrow(f) = \emptyset$. Then, due to Lemma 5.4, either $e \not< f$ or there is an event f' specified in Lemma 5.4. Due to the FIFO property, $e \ll f'$. As M_2 is race-free, $f' \ll f$ holds. Hence, existence of f' implies $e \ll f$ and e and f are not in race.
- Let $(\downarrow(e) \cap \uparrow(f)) \setminus p.\mathcal{G} \neq \emptyset$, say $p'.g \in \downarrow(e) \cap \uparrow(f)$ for some $p' \neq p$. Then there is a send event f' in M_2 on the process p' such that $e < f' < f$ and $p'.g \in \downarrow(e) \cap \uparrow(f')$. Therefore, $e \ll f' < f$. As M_2 is race-free, it holds that $e \ll f' \ll f$ and e and f are not in race.

"$\Longleftarrow$": Due to Lemma 5.3, it follows from $\downarrow(e) \cap \uparrow(f) \cap p.\mathcal{G} \neq \emptyset$ that $e < f$.

Due to Lemma 5.4, $\downarrow(e) \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset$ implies that $e \not< \mathcal{M}^{-1}(f)$ or there is a receive event f'' in M_2 such that $f'' < \mathcal{M}^{-1}(f) \wedge \mathit{label}(e) = \mathit{label}(f'') \wedge \mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f'')$. However, the last conjunct of our precondition ($\forall$ receive events $f' \ldots$) implies that $\downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f'')) = \emptyset$, which contradicts $\mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f'')$ (due to Lemma 5.4 for send events). Hence, it holds that $e \not< \mathcal{M}^{-1}(f)$ and Lemma 5.2 says that $e \ll f$ iff $\mathit{label}(e) = \mathit{label}(f)$ and $\mathcal{M}^{-1}(e) < \mathcal{M}^{-1}(f)$. Now the precondition $\mathit{label}(e) = \mathit{label}(f) \implies \downarrow(\mathcal{M}^{-1}(e)) \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset$ and Lemma 5.4 applied on send events $\mathcal{M}^{-1}(e)$ and $\mathcal{M}^{-1}(f)$ imply that $e \not\ll f$. Hence, e and f are in race. □

In Lemma 5.6, the precondition $P(e) = P(f)$ is not a serious restriction thanks to Lemma 2.7. The characterization says that to decide whether an event e of M_1 is in race with some event of M_2, one needs to know only $\mathit{label}(e)$, $\downarrow(e)$ and, if e is a receive action, then also $\downarrow(\mathcal{M}^{-1}(e))$. Triples $(\mathit{label}(e), \downarrow(e), \downarrow(\mathcal{M}^{-1}(e)))$ for receive events e and $(\mathit{label}(e), \downarrow(e), \emptyset)$ for send events e are called *footprints* of M_1. Note that the number of footprints for a fixed set of processes $\mathcal{P}$ and a gate name space $\mathcal{G}$ is bounded by $2 \cdot |\mathcal{P}|^2 \cdot 2^{|\mathcal{P}| \cdot |\mathcal{G}|} \cdot 2^{|\mathcal{P}| \cdot |\mathcal{G}|}$. Extending function P to labels as $P(p!q) = P(p?q) = p$, Lemma 5.6 can be reformulated as follows:

Lemma 5.7. *Let M_1 and M_2 be two race-free BMSCs in the special form. The concatenation $M_1 \cdot M_2$ contains a race if and only if there is a receive event f in M_2 and a footprint (l, F, F') of M_1 such that all the following conditions hold.*

1. $P(l) = P(f) = p$
2. $F \cap \uparrow(f) \cap p.\mathcal{G} \neq \emptyset$
3. $F \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset$
4. $l = \mathit{label}(f) \implies F' \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset$
5. $\forall$ *receive events* f' *of* M_2 *such that* $f' < \mathcal{M}^{-1}(f)$ *:*
   $l = \mathit{label}(f') \implies F' \cap \uparrow(\mathcal{M}^{-1}(f')) = \emptyset$

If the concatenation $M_1 \cdot M_2$ contains no race, we can easily compute its set of footprints. The computation uses the following auxiliary function.

Definition 5.8. *For every BMSC M over a gate name space $\mathcal{G}$ and a set of processes $\mathcal{P}$, and for every set $G \subseteq \mathcal{P}'.\mathcal{G}$ of joined gates we define $G{\downarrow_M}$ as*

$$G{\downarrow_M} = \{p'.g' \mid p.\overline{g} < p'.\underline{g'} \text{ where } < \text{ is the visual order in } M\} \cup \{p.g \in G \mid p \notin \mathcal{P}\}.$$

Lemma 5.9. *Let $\mathcal{FP}_1$ be the set of all footprints of a race-free BMSC M_1 and $\mathcal{FP}_2$ be the set of all footprints of a race-free BMSC M_2 such that $M_1 \cdot M_2$ is also race-free. Then the set of all footprints of the concatenation $M_1 \cdot M_2$ is equal to*

$$\mathcal{FP}_2 \cup \{(l, F{\downarrow_{M_2}} \cup \overline{F}, F'{\downarrow_{M_2}}) \mid (l, F, F') \in \mathcal{FP}_1 \wedge \overline{F} = \{p.g \mid \exists \text{ a receive event } f \text{ of } M_2 \text{ such that } f < p.\underline{g} \wedge \mathit{label}(f) = l \wedge F' \cap {\downarrow}(\mathcal{M}^{-1}(f)) \neq \emptyset\}\}.$$

Proof. It follows from the definition of the footprint, that any footprint $(l, F, F') \in \mathcal{FP}_2$ has to be a footprint of the concatenation as well. Each footprint (l, F, F') of M_1 is updated using the functions $F{\downarrow_{M_2}}$ and $F'{\downarrow_{M_2}}$. This update does not reflect the relations induced by the FIFO property, which are therefore added by the set $\overline{F}$.

One can readily confirm that all computed footnotes are footnotes of $M_1 \cdot M_2$. The other inclusion can be proven by induction similar to the one in the proof of Lemma 2.7. □

Now, we are ready to present our trace-race detection algorithm. Given a HMSC $H = (S, \rightarrow, s_0, S_F, L, \mathcal{L})$, we remove all states that are not on any run of H and we transform all the remaining BMSCs to the special form. Then the algorithm checks whether the BMSCs in states are race-free. Further, the algorithm computes all pairs $(s, (l, F, F'))$, where s is a state and (l, F, F') is a footprint of $L(s)$. In general, such a pair $(s, (l, F, F'))$ represents a footprint of a BMSC $L(\sigma)$ for some path $\sigma = s_0 \dots s_k$ starting in the initial state and leading to s. Each computed pair $(s, (l, F, F'))$ is then processed: for all $(s, s') \in \rightarrow$, the algorithm uses Lemma 5.7 to decide whether a BMSC M with the footprint (l, F, F') concatenated with $L(s')$ contains a race. If a race is found the algorithm halts. Otherwise, the algorithm employs Lemma 5.9 to calculate the footprint (l', F'', F''') corresponding to concatenation of (l, F, F') and $L(s')$. The pair $(s', (l', F'', F'''))$ is stored for subsequent processing unless it has been already processed or it is waiting to be processed. If all computed pairs are processed and no race is found, the HMSC is trace-race-free. The number of pairs is bounded and each pair is processed at most once, the algorithm eventually terminates. The algorithm is described formally in Figure 8.

Theorem 5.10. *Given an HMSC $H = (S, \rightarrow, s_0, S_f, L, \mathcal{L})$ (with open coregions and gates) over a gate name space $\mathcal{G}$, the problem whether H contains a trace-race is decidable in time $\mathcal{O}(|S|^2 \cdot b^3 \cdot |\mathcal{P}|^2 \cdot 2^{2 \cdot |\mathcal{P}| \cdot (|\mathcal{G}|+2)})$, where b is the size of the largest BMSC in $\mathcal{L}$. Hence, the problem is in P if the number of processes and gates is fixed.*

Input: an HMSC $H = (S, \rightarrow, s_0, S_F, L, \mathcal{L})$

Output: "race" if the HMSC contains a trace-race, "no race" otherwise

 # initialization
remove states that are not on any run in H
transform all BMSCs in states to the special form
for every s of S **do**
  if $L(s)$ contains a race **then return** "race"
  $TODO := \{(s, (l, F, F')) \mid (l, F, F')$ is a footprint of $L(s)\}$
 # processing the pairs
$DONE := \emptyset$
while $TODO \neq \emptyset$ **do**
  take $(s, (l, F, F'))$ from $TODO$
  add $(s, (l, F, F'))$ to $DONE$
  for every s' such that $(s, s') \in \rightarrow$ **do**
    # race checking
    for every $f \in E_R$ of $L(s')$ **do**
      if $P(l) = P(f) = p$ and $F \cap \uparrow(f) \cap p.(\mathcal{G} \cup \{pre, suc\}) \neq \emptyset$
        and $F \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset$
        and $(\ l = label(f) \implies F' \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset\ )$
        and $\forall$ receive events f' of $L(s')$ such that $f' < \mathcal{M}^{-1}(f)$
          it holds that $(\ l = label(f') \implies F' \cap \uparrow(\mathcal{M}^{-1}(f')) = \emptyset\ )$
      then return "race"
    # new footprint computation
    $\overline{F} := \{p.g \mid \exists$ a receive event f of $L(s')$ such that
      $f < p.\underline{g} \ \wedge\ label(f) = l \ \wedge\ F' \cap \downarrow(\mathcal{M}^{-1}(f)) \neq \emptyset\}$
    if $(s', (l, F{\downarrow_{L(s')}} \cup \overline{F}, F'{\downarrow_{L(s')}})) \notin TODO \cup DONE$ **then**
      add $(s', (l, F{\downarrow_{L(s')}} \cup \overline{F}, F'{\downarrow_{L(s')}}))$ to $TODO$
return "no race"

Figure 8: Trace-race detection algorithm

Proof. The correctness of the algorithm follows from Lemmata 2.7, 5.7 and 5.9. The complexity bound has been derived as follows:

- **initialization**: Removal of states that are not on any run in H can be done in $\mathcal{O}(|S|^2)$. Then we compute visual and causal orders for all BMSCs in the remaining states of H. This can be done in $\mathcal{O}(|S| \cdot b^2)$. The conversion of all BMSCs in states

of H to the special form (due to the first part of the transformation presented in Section 4) can be done in $\mathcal{O}(|S| \cdot b^2)$. The conversion does not modify the number of events in the BMSCs, but the gate name space is extended with names *pre*, *suc*. The for-cycle runs in $\mathcal{O}(|S| \cdot b^2)$.

- **processing the pairs**: The number of pairs is bounded by $|S| \cdot 2|\mathcal{P}|^2 \cdot 2^{|\mathcal{P}|\cdot(|\mathcal{G}|+2)} \cdot 2^{|\mathcal{P}|\cdot(|\mathcal{G}|+2)}$ and each pair is processed at most once. The race checking for a fixed pair and a fixed state s' can be done in $\mathcal{O}(b^3)$ time. The computation of a new footprint takes $\mathcal{O}(b^2)$ time. Hence, the whole while-cycle is evaluated in $\mathcal{O}(|S|^2 \cdot b^3 \cdot |\mathcal{P}|^2 \cdot 2^{2\cdot|\mathcal{P}|\cdot(|\mathcal{G}|+2)})$.

The overall complexity is $\mathcal{O}(|S|^2 \cdot b^3 \cdot |\mathcal{P}|^2 \cdot 2^{2\cdot|\mathcal{P}|\cdot(|\mathcal{G}|+2)})$. □

6 Space complexity of the trace-race detection problem

In this section, we show that the trace-race detection problem for HMSCs is PSPACE-complete.

In the algorithm of Figure 8, each tuple $(s, (l, F, F'))$ is of a polynomial size to the size of the given HMSC. Therefore, the sets *TODO* and *DONE* are of an exponential size of the given HMSC. To prove that the problem is in PSPACE, we present a nondeterministic algorithm, where the sets *TODO* and *DONE* are supplied by a nondeterministic choice of the race inducing footprint as well as the successive states on the race inducing path of the given HMSC. To prevent infinite runs of the nondeterministic algorithm, we introduce a counter i which counts the number of derived pairs $(s, (l, F, F'))$. If i is greater then the number of all possible pairs, we stop the computation. The formal description of the nondeterministic algorithm is in Figure 9.

Lemma 6.1. *The trace-race detection problem for HMSC (with open coregions and gates) is in* PSPACE.

Proof. The correctness of the algorithm follows from Lemmata 2.7, 5.7 and 5.9. The space complexity is polynomial as we need to store only the current footnote and the value of i. The size of a footnote is polynomial in the size of H. The value of i is bounded by $|S|\cdot 2|\mathcal{P}|^2\cdot 2^{2\cdot|\mathcal{P}|\cdot(|\mathcal{G}|+2)}$ and therefore it can be also stored in a polynomial space. Hence, the trace-race detection problem is in NPSPACE. As PSPACE=NPSPACE, it is in PSPACE. □

Input: an HMSC $H = (S, \rightarrow, s_0, S_F, L, \mathcal{L})$

Output: the HMSC contains a trace-race iff the algorithm can return "race"

 # initialization
remove states that are not on any run in H
transform all BMSCs in states to the special form
choose a state s of S
choose a footprint (l, F, F') of $L(s)$
$i := 0$
while $i < |S| \cdot 2|\mathcal{P}|^2 \cdot 2^{2 \cdot |\mathcal{P}| \cdot (|\mathcal{G}|+2)}$ **do**
  $i := i + 1$
  # processing the pair $(s, (l, F, F'))$
  choose s' such that $(s, s') \in \rightarrow$
  for every $f \in E_R$ of $L(s')$ **do**
    # race checking
    if $P(l) = P(f) = p$ and $F \cap \uparrow(f) \cap p.(\mathcal{G} \cup \{pre, suc\}) \neq \emptyset$
      and $F \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset$
      and $(\ l = label(f) \implies F' \cap \uparrow(\mathcal{M}^{-1}(f)) = \emptyset\ )$
      and $\forall$ receive events f' of $L(s')$ such that $f' < \mathcal{M}^{-1}(f)$
        it holds that $(\ l = label(f') \implies F' \cap \uparrow(\mathcal{M}^{-1}(f')) = \emptyset\ )$
    then return "race"
  # new footprint computation
  $\overline{F} := \{p.g \mid \exists$ a receive event f of $L(s')$ such that
    $f < p.\underline{g} \ \wedge\ label(f) = l \ \wedge\ F' \cap \downarrow(\mathcal{M}^{-1}(f)) \neq \emptyset\}$
  $(l, F, F') := (l, F{\downarrow_{L(s')}} \cup \overline{F}, F'{\downarrow_{L(s')}})$
  $s := s'$
return "don't know"

Figure 9: Nondeterministic trace-race detection algorithm

Lemma 6.2. *The problem whether a given HMSC (with open coregions and gates) contains a trace-race is* PSPACE*-hard.*

Proof. We prove this lemma by a reduction of the word problem for linear bounded automata, which is known to be PSPACE-complete,onto the trace race condition problem for HMSCs with open coregions and gates.

First, let us recall the definition of the word problem for linear bounded automata.

Instance: A word w and a linear bounded automaton A.
Question: Is the word w accepted by the automaton A?

Let A be a linear bounded automaton $(Q, q_0, \Sigma, \Gamma, \delta, F)$. Without loss of generality, let us assume that the set of states $Q = \{q_0, q_1, \cdots, q_n\}$, the input alphabet $\Sigma = \{0, 1\}$, the tape alphabet $\Gamma = \{\triangleright, 0, 1, \triangleleft\}$, the set of accepting states $F = \{q_n\}$. Let $w \in \{0,1\}^*$ be an input word and $w[i] \in \{0,1\}$ be the i-th letter of w, for $1 \leq i \leq |w|$.

In what follows, we construct an HMSC $H_{w,A} = (S, \to, s_0, \{s_f\}, L, \mathcal{L})$, where $S = \{s_0, s_{init}, s_{comp}, s_{accept}, s_f\} \cup \{s_{qbq'b'D} \mid q, q' \in Q \wedge b, b' \in \Gamma \wedge D \in \{L, R\} \wedge (q', b', D) \in \delta(q, b)\}$. The transition relation $\to$ is defined as

$$\{(s_0, s_{init}), (s_{init}, s_{comp}), (s_{comp}, s_{accept}), (s_{accept}, s_f)\} \cup$$
$$(\{s_{comp}\} \times \{s_{qbq'b'D} \mid q, q' \in Q \wedge b, b' \in \Gamma \wedge D \in \{L, R\} \wedge (q', b', D) \in \delta(q, b)\}) \cup$$
$$(\{s_{qbq'b'D} \mid q, q' \in Q \wedge b, b' \in \Gamma \wedge D \in \{L, R\} \wedge (q', b', D) \in \delta(q, b)\} \times \{s_{comp}\}).$$

The mapping L assigns the empty BMSC to the states s_0, s_{comp}, and s_f, which serves as connection points only. For every other state s, the mapping L assigns a BMSC M_s of $\mathcal{L}$ to s. In every BMSC of $\mathcal{L}$ (except of the empty BMSC), there are no events and only one process p with one (both top- and bottom-) open coregion containing all gates of $p.\overline{\mathcal{G}}$ and $p.\underline{\mathcal{G}}$, where $\mathcal{G} = \{g_0, g_1\} \cup Q \cup \{t_i, b_i \mid 0 \leq i \leq |w| + 1\}$. Therefore, defining general ordering within each BMSC of $\mathcal{L}$ is the last and the most important part of the $H_{w,A}$ construction. General ordering of top gates $p.\overline{g_0}, p.\overline{g_1}$ and bottom gates of $p.\underline{\mathcal{G}}$ will store configurations of A. Intuitively, connecting to $p.\overline{g_0}$ stands for FALSE or 0 and connecting to $p.\overline{g_1}$ stands for TRUE or 1. Gates $p.\underline{Q}$ mimic the actual control state of A. Each pair $p.\underline{t_i}$, $p.\underline{b_i}$ stores the tape value on the i-th position on the right-hand side from the reading/writing head (i is counted modulo the tape length). The gate $p.\underline{t_i}$ is connected to $p.\overline{g_1}$ if the i-th symbol is terminal, i.e. $\triangleright$ or $\triangleleft$; otherwise, it is connected to $p.\overline{g_0}$. The gates $p.\underline{b_i}$ store the value 0 or 1, or marks $\triangleright$ off $\triangleleft$.

The general ordering of the BMSC $M_{s_{init}}$ is defined as:

$$\{p.\overline{g_0}\} \times \left(\begin{array}{l} \{p.\underline{g_0}\} \cup \{p.\underline{q_i} \mid 0 < i \leq n\} \cup \\ \{p.\underline{t_i} \mid 1 \leq i \leq |w|\} \cup \{p.\underline{b_0}\} \cup \{p.\underline{b_i} \mid 1 \leq i \leq |w| \wedge w[i] = 0\} \end{array} \right)$$
$$\cup$$
$$\{p.\overline{g_1}\} \times \left(\begin{array}{l} \{p.\underline{g_1}\} \cup \{p.\underline{q_0}\} \cup \\ \{p.\underline{t_0}, p.\underline{t_{|w|+1}}\} \cup \{p.\underline{b_{|w|+1}}\} \cup \{p.\underline{b_i} \mid 1 \leq i \leq |w| \wedge w[i] = 1\} \end{array} \right).$$

The general ordering of the BMSC $M_{s_{qbq'b'D}}$ is defined as:

$$
\begin{aligned}
&\{(p.\overline{g_0}, p.\underline{g_0}), (p.\overline{g_1}, p.\underline{g_1})\} \\
&\cup \\
&\{p.\overline{q_i} \mid 1 \leq i \leq n \wedge q_i \neq q\} \times \{p.\underline{g_0}\} \\
&\cup \\
&\{p.\overline{q_i} \mid 1 \leq i \leq n \wedge q_i = q\} \times \{p.\underline{g_1}\} \\
&\cup \\
&\{(p.\overline{t_0}, p.\underline{g_i}), (p.\overline{b_0}, p.\underline{g_j}) \mid (b \in \{0,1\} \implies (i = 0 \wedge j = b)) \wedge \\
&\qquad\qquad (b = \triangleright \implies (i = 1 \wedge j = 0)) \wedge \\
&\qquad\qquad (b = \triangleleft \implies (i = 1 \wedge j = 1))\} \\
&\cup \\
&\{p.\overline{g_0}\} \times \{p.\underline{q_i} \mid 1 \leq i \leq n \wedge q_i \neq q'\} \\
&\cup \\
&\{p.\overline{g_1}\} \times \{p.\underline{q_i} \mid 1 \leq i \leq n \wedge q_i = q'\} \\
&\cup \\
&\{(p.\overline{g_i}, p.\underline{t_k}), (p.\overline{g_j}, p.\underline{b_k}) \mid (D = L \implies k = 1) \wedge \\
&\qquad\qquad (D = R \implies k = |w| + 1) \wedge \\
&\qquad\qquad (b' \in \{0,1\} \implies (i = 0 \wedge j = b)) \wedge \\
&\qquad\qquad (b' = \triangleright \implies (i = 1 \wedge j = 0)) \wedge \\
&\qquad\qquad (b' = \triangleleft \implies (i = 1 \wedge j = 1))\} \\
&\cup \\
&\{(p.\overline{t_i}, p.\underline{t_j}), (p.\overline{b_i}, p.\underline{b_j}) \mid 1 \leq i \leq |w| + 1 \wedge \\
&\qquad\qquad (D = L \implies j = i + 1 \mod (|w| + 2)) \wedge \\
&\qquad\qquad (D = R \implies j = i - 1 \mod (|w| + 2))\}
\end{aligned}
$$

The general ordering of the BMSC $M_{s_{accept}}$ is defined as:

$$
\begin{aligned}
&(\{p.\overline{g_0}\} \cup \{p.\overline{q_i} \mid 0 \leq i < n\}) \times \{p.\underline{g_0}\} \\
&\cup \\
&(\{p.\overline{g_1}\} \cup \{p.\overline{q_n}\}) \times \{p.\underline{g_1}\}.
\end{aligned}
$$

"Cheating" during the simulated computation (to continue in a different computation) will order $p.\overline{g_0}$ before $p.\underline{g_1}$, or $p.\overline{g_1}$ before $p.\underline{g_0}$. Therefore, in $L(H_{w,A})$, there is a BMSC with $p.\overline{g_0} \not< p.\underline{g_1}$ and $p.\overline{g_1} \not< p.\underline{g_0}$ if and only if A accepts w.

To show the reduction proving this lemma, we need an HMSC H such that in $L(H)$, there is a BMSC $p.\overline{g_0} \not< p.\underline{g_1}$ or $p.\overline{g_1} \not< p.\underline{g_0}$ if and only if A accepts w. In other words, we need to eliminate all BMSC where $p.\overline{g_0} < p.\underline{g_1} \wedge p.\overline{g_1} \not< p.\underline{g_0}$ or $p.\overline{g_0} \not< p.\underline{g_1} \wedge p.\overline{g_1} < p.\underline{g_0}$.

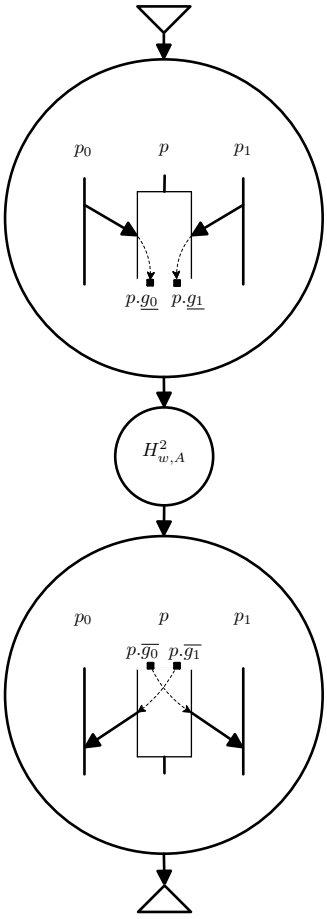

Figure 10: The HMSC H

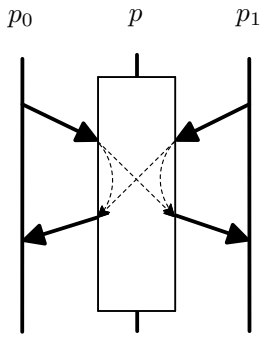

Figure 11: A (race-free) BMSC of a cheating computation

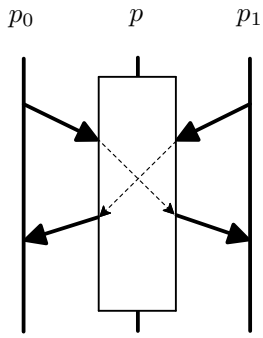

Figure 12: A BMSC (with race) of an accepting computation

We construct an HMSC $H^2_{w,A}$ from the HMSC $H_{w,A}$ adding one more copy of gate names $Q' \cup \{t'_i, b'_i \mid 0 \leq i \leq |w| + 1\}$ into $\mathcal{G}$. The new gates will be connected by general ordering in the same way as their non-primed counterparts but the roles of g_0 and g_1 will be exchanged with respect to them, i.e. $p.\overline{g_0}$ will stand for TRUE or 1, and $p.\overline{g_0}$ will stand for FALSE or 0. Therefore, in $H^2_{w,A}$, if a cheating cause $p.\overline{g_0} < p.\underline{g_1}$ via the original gates, then the primed gates cause $p.\overline{g_1} < p.\underline{g_0}$ at the same BMSC. Therefore, in $L(H^2_{w,A})$, there is a BMSC with $p.\overline{g_0} \not< p.\underline{g_1}$ and $p.\overline{g_1} \not< p.\underline{g_0}$ if A accepts w; otherwise, all BMSC of $L(H^2_{w,A})$ includes $p.\overline{g_0} < p.\underline{g_1}$ and $p.\overline{g_1} < p.\underline{g_0}$.

Now, we construct an HMSC H of Figure 10. It holds that there is a trace-race in H if and only if there is a BMSC in $L(H^2_{w,A})$ such that $p.\overline{g_0} \not< p.\underline{g_1}$ or $p.\overline{g_1} \not< p.\underline{g_0}$. Therefore, there is a trace-race in H if and only if A accepts w. □

The Lemmata 6.1 and 6.2 directly implies Theorem 6.3.

Theorem 6.3. *The trace-race detection problem is* PSPACE*-complete.*

7 Conclusions

We have introduced two new notions for HMSCs: an extension of the formalism with *open coregions* and a new race condition for HMSCs called *trace-race*. Definitions of race and trace-race directly imply that every trace-race-free HMSC is also race-free. We have shown that every race-free HMSC can be translated into an equivalent trace-race-free HMSC using open coregions, where by equivalence we mean that the two HMSCs represent sets of BMSCs with identical linearizations. Hence, trace-race-free HMSCs with open coregions are as expressive as race-free HMSCs with open coregions (and we conjecture that trace-race-free HMSCs with open coregions are in fact strictly more expressive than race-free HMSCs without open coregions). While the race detection problem is undecidable even for HMSCs without coregions [5], we have demonstrated that the trace-race detection problem is decidable (and PSPACE-complete) for HMSCs with open coregions. Therefore, HMSCs with open coregions and the trace-race notion appear as good candidates for tractable analysis of race ambiguities in scenario based designs.

The trace-race detection algorithm is implemented in *Sequence Chart Studio* (a Microsoft Visio add-on available at `http://scstudio.sourceforge.net/`). The studio currently supports HMSCs with closed coregions only (a support of open coregions is planned too).

Acknowledgment. We would like to thank Philippe Darondeau for an important hint.

References

[1] R. Alur, G.J. Holzmann, and D. Peled. An Analyzer for Message Sequence Charts. In *TACAS'96*, LNCS, pages 35–48. Springer, 1996.

[2] Philippe Darondeau, Blaise Genest, and Loïc Hélouët. Products of message sequence charts. In Roberto M. Amadio, editor, *Foundations of Software Science and Computational Structures, FOSSACS 2008*, volume 4962 of *Lecture Notes in Computer Science*, pages 458–473. Springer, 2008.

[3] Thomas Gazagnaire and Loïc Hélouët. Event correlation with boxed pomsets. In John Derrick and Jüri Vain, editors, *Formal Techniques for Networked and Distributed*

Systems - FORTE 2007, volume 4574 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2007.

[4] ITU Telecommunication Standardization Sector - Study group 17. ITU recommandation Z.120, Message Sequence Charts (MSC), 2004.

[5] A. Muscholl and D. Peled. Message Sequence Graphs and Decision Problems on Mazurkiewicz Traces. In *MFCS'99*, volume 1672 of *LNCS*, pages 81–91. Springer, 1999.

[6] A. Muscholl and D. Peled. Analyzing Message Sequence Charts. In *SAM 2000: Proceedings of the 2nd Conference on MSC and SDL*, pages 3–17, Grenoble, 2000. VERIMAG, IRISA, SDL Forum.

[7] Vaughan R. Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, 1986.

[8] Ekkart Rudolph, Peter Graubmann, and Jens Grabowski. Tutorial on Message Sequence Charts. *Computer Networks and ISDN Systems*, 28(12):1629–1641, 1996.