Appeared in: Information and Computation 204:920–956, 2006.

# A Kleene Theorem and Model Checking Algorithms for Existentially Bounded Communicating Automata

Blaise Genest[*], Dietrich Kuske[†], and Anca Muscholl[‡]

**Abstract**

The behavior of a network of communicating automata is called *existentially bounded* if communication events can be scheduled in such a way that the number of messages in transit is always bounded by a value that depends only on the machine, not the run itself. We show a Kleene theorem for existentially bounded communicating automata, namely the equivalence between communicating automata, globally-cooperative compositional message sequence graphs, and monadic second order logic. Our characterization extends results for universally bounded models, where for each and every possible scheduling of communication events, the number of messages in transit is uniformly bounded [15, 17]. As a consequence, we give solutions in the spirit of [22] for various model checking problems on networks of communicating automata that satisfy our optimistic restriction.

## 1 Introduction

Communicating finite-state machines (CFM for short), or equivalently, FIFO channel systems, are a fundamental model for concurrent systems. Unfortunately, these machines are too powerful to be amenable for automatic verification since they are Turing equivalent [8].

Several papers aimed at identifying variants of these machines or approximated behaviors thereof, that are suited for automated verification methods. For example, for lossy FIFO systems the reachability problem is known to be decidable [1, 11], albeit of non-primitive recursive complexity [28].

Another approach to obtain decidability of model checking questions is based on the representation of the set of reachable configurations, (including channel contents)

[*]IRISA, Campus de Beaulieu, Rennes, France
Work done while being affiliated with LIAFA, Université Paris 7, supported by the ACI Versydis.
[†]Institut für Informatik, Universität Leipzig, Germany
Some of the results were obtained when D. Kuske was affiliated with the Technische Universität Dresden. Supported by DAAD-PROCOPE D/0333596.
[‡]LIAFA, Université Paris 7, France
Work supported by the ACI Versydis.

by some finite automaton, see e.g. [4, 5, 6]. Often this approach requires to relax the operations on channels, which yields an over-approximation of the result.

The approach taken by our paper goes beyond regular representations of reachable configurations. We use instead partial order methods for describing the behavior of a CFM. Formally, the behaviors are described by *Message sequence charts* (MSC for short), a diagram notation described by the ITU norm Z.120 [16]. The advantage of reasoning about CFMs using MSCs is both succinctness and comprehension, since a single diagram subsumes a set of sequential runs of the CFM. Since MSCs are a partial order formalism, we reason about CFM and MSC properties using partial order logics such as monadic second order logic (MSO for short) over MSCs.

The MSC model has become popular in telecommunication through its visual representation, depicting the involved processes as vertical lines, and each message as an arrow between the source and the target processes, according to their occurrence order. The Z.120 standard has also extended the notation to *MSC-graphs*, which consist of finite transition systems, where each state is labeled by an MSC. This formalism actually corresponds to regular expressions over MSCs, and it is not a necessarily executable model. The interest in considering MSC-graphs is their practical impact in designing communication protocols. CMSC-graphs were proposed in [13] as a generalization of MSC-graphs, corresponding to regular expressions over communication events. Their introduction was motivated by the fact that MSC-graphs and CFM are incomparable, however, the language of any CFM can be described by some CMSC-graph.

An early line of work considered universally bounded MSCs, only. In terms of a CFM, this amounts to saying that every CFM run can be executed with channels of fixed size, no matter how events are scheduled. Equivalently, there exists some (uniform) bound on the number of messages in transit, at any time. Since the size of the communication channels is fixed uniformly, this constraint turns a CFM into a finite state device. Checking that a CFM is universally bounded is undecidable, and recently heuristics were proposed for solving this problem [19]. On the other hand, universal-boundedness for MSC-graphs can be enforced by a syntactic restriction [2, 25]. Over universally bounded MSCs, the rich theory of regular languages extends very well: automata (CFMs), logic (monadic second order) and MSC-expressions (regular MSC-graphs) are all equivalent [15] (see also [21, 17]). Moreover, model checking in the realm of universally bounded MSC models is decidable.

The drawback of models with universally bounded communication channels is the limited expressive power. Intuitively, universal channel bounds require message acknowledgments, which can be difficult to impose in general. For instance, basic protocols of producer-consumer type (such as e.g. the USB protocol [29]) are not universally bounded, since the communication is one-way. In this paper we relax this restriction on channels in order to capture more interesting behaviors, such as USB. The idea is to require an *existential bound* on channels. This means roughly that every MSC run must have *some* scheduling of events that respects a given channel bound (other schedules might exceed the bound). In other words, runs *can* be executed with bounded channels, provided that we schedule the events conveniently. For instance, in a producer-consumer setting, the scheduling alternates between producer and consumer actions. This requirement is perfectly legitimate in practice, since real life protocols must be executable with limited communication channels. When a channel overflow happens, then the sender stops temporarily until some message is consumed from the queue.

In a nutshell, we have two objectives in this paper: First, we look for a robust class of MSC models, i.e., one with equivalent characterizations in terms of logics, regular expressions and automata. Second, we want a class with decidable model checking problem, which of course requires some restrictions on the models we consider.

The main result of the paper is that communicating finite-state machines, monadic second order logic and globally-cooperative CMSC-graphs are equivalent over existentially bounded MSCs. Thus, we proved an extension of the corresponding result from [15] to the more complex setting of existentially-bounded MSCs. The MSO logic used here is based on the MSC partial order and the message relation, as employed also in [23]. As shown by [7], this logic is in general more powerful than the existential fragment of MSO using only the immediate process successor together with the message relation. In particular, it follows from our main result that CFMs and globally-cooperative CMSC-graphs can be complemented relative to the set of existentially-$B$-bounded MSCs for any bound $B$. We do not know how to prove this explicitly without exploiting the equivalence to MSO, which is trivially closed under negation. Another consequence of the main result is that several interesting model checking instances are decidable in this setting. We can check 1) whether all existentially-$B$-bounded MSCs accepted by a CFM satisfy an MSO formula, for any bound $B$, and 2) whether the language of a safe CMSC-graph is included in (intersects, respectively) the language of a CFM.

*Overview.* In Section 2 we define the formalisms used in the paper – message sequence charts, communicating automata, MSO, and Mazurkiewicz traces. Section 3 describes the way model checking works using representative executions. This leads to the restriction of CMSC-graphs to safe and globally-cooperative graphs, and to an intimate relation between existentially bounded sets of MSCs and Mazurkiewicz traces. In Section 4 we state the equivalence of several specification formalisms for MSCs and give some of the transformations. The proof is completed in Section 5 where we present the construction of a CFM from a regular set of representatives. Finally, Section 6 shows that model checking is possible for all formalisms considered in this paper.

*Related work.* Existential channel bounds appear in [18] and implicitly in [13] (realizable CHMSCs). Our paper generalizes several results about expressivity and model checking for MSCs with universally bounded channels [2, 25, 15, 17, 21]. Without the restriction of universally bounded channels, [22, 23] shows how to use representative executions in model checking against MSO properties and [14] does this against MSC-graph properties. Recall that we use the logic from [15, 23] that talks about the partial order of an MSC. The paper [7] shows that the existential fragment of the weaker MSO based on the immediate successor is expressively equivalent to CFMs without any restrictions.

# 2 Definitions

## 2.1 Message sequence charts

The communication framework used in our paper is based on sequential processes that exchange asynchronously messages over point-to-point, error-free FIFO channels. Let $\mathcal{P}$ be a finite set of process identities that we fix throughout this paper. Processes act by either sending a message, that is denoted by $p!q$ meaning that process $p$ sends to process $q$, or by receiving a message, that is denoted by $p?q$, meaning that process $p$ receives from process $q$. Thus we do not use different message contents in our notation. In the same line, we do not consider local events, that is, events which are neither send nor receive. This is done for convenience and the reader might convince himself/herself that proofs work (with small alterations) in the more general setting as well.

For any process $p \in \mathcal{P}$, we define a local alphabet (set of event types on $p$) $\Sigma_p = \{p!q, p?q \mid q \in \mathcal{P} \setminus \{p\}\}$ and set $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$. For the rest of the paper, whenever a pair of processes $p, q \in \mathcal{P}$ communicate, we will implicitly assume that $p \neq q$.

We introduce now the notation of *(compositional) message sequence charts*, that is usually employed for describing scenarios of communication. The *message sequence chart* notation (MSC for short) corresponds to the Z.120 standard of the ITU. Theoretical work has revealed several deficiencies of the standard notation of MSCs and MSC-graphs, which motivated the extended notation of *compositional message sequence charts, CMSC* for short. We will be mainly interested in MSCs as a complete formalism, but we will use CMSCs as a kind of technical tool.

**Definition 2.1** *A* compositional message sequence chart *(*CMSC*) is a tuple* $M = (E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ *where*

- *$E$ is a finite set of events*
- *$\lambda : E \to \Sigma$ maps each event to a type, and we set*
  - *$E_p = \{e \in E \mid \lambda(e) \in \Sigma_p\}$ the set of events of process $p$,*
  - *$S = \{e \in E \mid \exists p, q \in \mathcal{P} : \lambda(e) = p!q\}$ the set of send events, and*
  - *$R = E \setminus S$ the set of receive events*
- *$<_p$ is a total strict order on $E_p$, for any $p \in \mathcal{P}$*
- $\mathrm{msg} : S \to R$ *is an injective partial mapping satisfying*
  - *if* $\mathrm{msg}(s) = r$*, then there are $p, q \in \mathcal{P}$ distinct such that $\lambda(s) = p!q$ and $\lambda(r) = q?p$,*
  - *if $s_1 <_p s_2$, $\lambda(s_1) = \lambda(s_2) = p!q$, and* $\mathrm{msg}(s_1)$, $\mathrm{msg}(s_2)$ *are defined, then* $\mathrm{msg}(s_1) <_q \mathrm{msg}(s_2)$,

*such that the relation $\leq := (\bigcup_{p \in \mathcal{P}} <_p \cup \{(s, \mathrm{msg}(s)) \mid s \in S\})^*$ is a partial order, called* visual order*.*

*A* message sequence chart *[16] is a CMSC $(E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ such that the message mapping* $\mathrm{msg} : S \to R$ *is defined everywhere and surjective.*

*For a CMSC $M$, we will write $P(e) = p$ if $e \in E_p$, i.e., $\lambda(e) \in \Sigma_p$. Moreover, we write $e \lessdot_p f$ if $e$ is the immediate predecessor of $f$ on process $p$, i.e., $e <_p f$ and $e <_p g \leq_p f$ implies $g = f$.*

The second requirement on the function msg of the definition above ensures that it is order preserving on its domain. Intuitively, messages are received in the same order in which they are sent, i.e., we deal with FIFO-channels. For this reason, we will refer to this property of CMSCs as FIFO.

This definition of a CMSC differs from the original one in [13] in that we do not consider message contents (called "names" in [13]), as done e.g. in [15, 17]. As already noted there, one could add message contents to the formalism without sacrificing any of the results.

Figure 1 depicts an MSC. In that picture, there are two processes named $p$ and $q$. The two vertical lines denote the time axis of these two processes, i.e., the relations $<_p$ and $<_q$, respectively. For simplicity, the picture only indicates whether a given event is a sent or a receive event, since we have only two processes, it should be obvious, which process is sending to (is receiving from) which process. Arrows between events on distinct process lines indicate the mapping msg. Hence, the MSC from Figure 1 denotes the sending and receiving of three messages from process $p$ to process $q$ and of two messages from process $q$ to process $p$. Both processes first send their respective messages before they receive anything. The MSC does not specify the content of these messages.

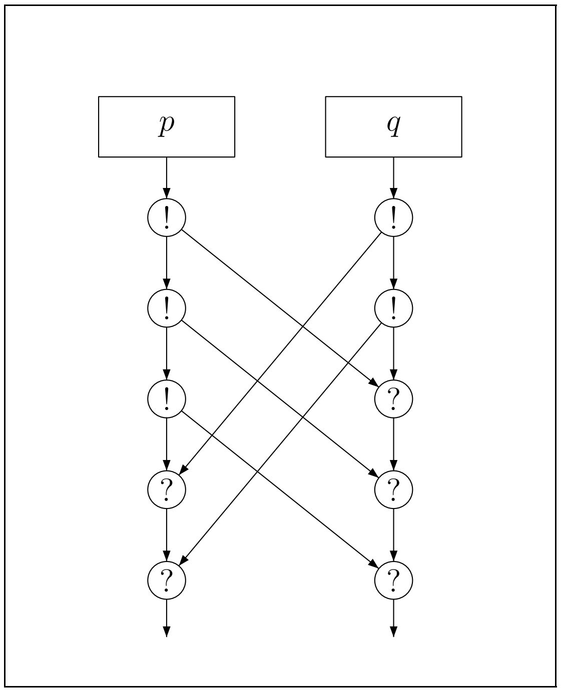

Figure 1: A message sequence chart $M$

We can view a CMSC $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ as a poset $(E, \leq, \lambda)$. For a given poset, elements from $E$ are called *events*. Any linear extension of $\leq$ is called a *linearization* of $M$. We represent it as a word $u = u_1 \cdots u_n$ over the alphabet $\Sigma$. Thus, the set $\mathrm{Lin}(M)$ of linearizations of the CMSC $M$ is a subset of $\Sigma^*$. For a set (or language) of CMSCs $\mathcal{M}$, we write $\mathrm{Lin}(\mathcal{M}) = \bigcup_{M\in\mathcal{M}} \mathrm{Lin}(M)$. Note that we can recover an *MSC* from any of its linearizations, thanks to the FIFO condition.

Let $B$ be some positive integer. A word (linearization) $w \in \Sigma^*$ is *$B$-bounded* if for any prefix $u$ of $w$ and any $p, q \in \mathcal{P}$, the number of occurrences of $p!q$ in $u$ exceeds that of occurrences of $q?p$ in $u$ by at most $B$. An MSC $M$ is *existentially $B$-bounded* (*∃-$B$-bounded* for short) if it has some $B$-bounded linearization $w \in \mathrm{Lin}(M)$. Let $\mathrm{Lin}^B(M) \subseteq \mathrm{Lin}(M)$ denote the set of $B$-bounded linearizations of $M$ – by definition, this set is non-empty iff $M$ is ∃-$B$-bounded. An MSC $M$ is *universally $B$-bounded* if $\mathrm{Lin}(M) = \mathrm{Lin}^B(M)$.

As an example, the word $(q!p)^2\,[(p!q)\,(q?p)]^3\,(p?q)^2$ is a 2-bounded linearization of the MSC $M$ from Fig. 1, i.e., $M$ is ∃-2-bounded. But the MSC $M$ is not ∃-1-bounded: suppose $w \in \mathrm{Lin}(M)$ is 1-bounded. Let $u$ be the minimal prefix of $w$ containing two occurrences of $p!q$. Since, in $w$, any occurrence of $p!q$ has to precede any occurrence of $p?q$, there is no occurrence of $p?q$ in $u$. Since $w$ is 1-bounded, $u$ has to contain an occurrence of $q?p$. Since $w$ is a linearization of $M$, the word $u$ contains two occurrences of $q!p$. Since $u$ does not contain any occurrence of $p?q$, the word $w$ is not 1-bounded, a contradiction. Hence the MSC $M$ is ∃-2-bounded, but not ∃-1-bounded.

The class of all CMSCs, resp. MSCs and ∃-$B$-bounded MSCs, will be denoted $\mathbb{CMSC}$, resp. $\mathbb{MSC}$ and $\mathbb{MSC}^B$. An algorithm for checking whether an MSC $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ is ∃-$B$-bounded is based on the following relation $\prec_B \subseteq E \times E$, see [18]:

Let $\prec_B = \mathrm{msg} \cup \bigcup_{p\in\mathcal{P}} <_p \cup \mathrm{rev}$, where rev is given as:

$$\begin{aligned} \mathrm{rev}(r) = s' \quad \text{iff} \qquad & \mathrm{msg}(s) = r,\ \lambda(s) = \lambda(s'),\ \text{ and} \\ & |\{x \in E \mid s <_p x \leq_p s', \lambda(s) = \lambda(x)\}| = B \end{aligned}$$

That is, the relation rev maps a receive $r$ with $r = \mathrm{msg}(s)$ to the send $s'$ that is the $B$-th event with $\lambda(s') = \lambda(s)$ and $s < s'$ (if such an event exists).

**Lemma 2.2** *[18] An MSC $M$ is ∃-$B$-bounded iff the relation $\prec_B = \mathrm{msg} \cup \bigcup_{p\in\mathcal{P}} <_p \cup \mathrm{rev}$ is acyclic.*

We say that $\mathcal{M} \subseteq \mathbb{MSC}$ is an ∃-$B$-bounded set of MSCs if $\mathcal{M} \subseteq \mathbb{MSC}^B$; $\mathcal{M}$ is *existentially-bounded* (or ∃-bounded) if $\mathcal{M} \subseteq \mathbb{MSC}^B$ for some $B$. Similarly, $\mathcal{M}$ is called universally $B$-bounded if every $M \in \mathcal{M}$ is universally $B$-bounded; and $\mathcal{M}$ is *universally-bounded*, if it is universally $B$-bounded for some $B$.

## 2.2 Communicating finite-state machines

The most natural formalism to describe (asynchronous) communication protocols are *communicating finite-state machines* (CFM for short) that we define in this section.

CFMs are a basic model for distributed algorithms based on asynchronous message passing.

**Definition 2.3** *[8] A* communicating finite-state machine *(*CFM*) is a tuple* $\mathcal{A} = (C, (\mathcal{A}_p)_{p\in\mathcal{P}}, F)$ *where*

- $C$ *is a finite set of* message contents *or* control messages.
- $\mathcal{A}_p = (S_p, \rightarrow_p, \iota_p)$ *is a finite labeled transition system over the alphabet* $\Sigma_p \times C$ *for any* $p \in \mathcal{P}$ *(i.e.,* $\rightarrow_p \subseteq S_p \times (\Sigma_p \times C) \times S_p$*) with initial state* $\iota_p \in S_p$.
- $F \subseteq \prod_{p\in\mathcal{P}} S_p$ *is a set of global final states.*

The first approach to define the behavior of a CFM considers these machines as sequential devices that accept linearizations of MSCs. More precisely, one defines from the CFM $\mathcal{A} = (C, (\mathcal{A}_p)_{p\in\mathcal{P}}, F)$ a $(\Sigma \times C)$-labeled, infinite transition system as follows. A configuration of $\mathcal{A}$ consists of a tuple of local states and of channel contents, i.e., it is an element $((s_p)_{p\in\mathcal{P}}, (w_{p,q})_{p,q\in\mathcal{P}})$ of $\prod_{p\in\mathcal{P}} S_p \times \prod_{p,q\in\mathcal{P}} C^*$. For two configurations, an action $a \in \Sigma_p$, and a control message $c \in C$, we have

$$((s^1_p)_{p\in\mathcal{P}}, (w^1_{p,q})_{p,q\in\mathcal{P}}) \xrightarrow{a,c} ((s^2_p)_{p\in\mathcal{P}}, (w^2_{p,q})_{p,q\in\mathcal{P}})$$

if

- $s^1_p \xrightarrow{a,c}_p s^2_p$ is a transition of the local machine $\mathcal{A}_p$ and $s^1_q = s^2_q$ for $q \neq p$.
- Send events: if $a = p!q$, then $w^2_{p,q} = w^1_{p,q}c$ (i.e., message $c$ is inserted into the channel from $p$ to $q$) and $w^1_{p',q'} = w^2_{p',q'}$ for $(p', q') \neq (p, q)$ (i.e., all other channels are unchanged)
- Receive events: if $a = p?q$, then $w^1_{q,p} = cw^2_{q,p}$ (i.e., message $c$ is deleted from the channel from $q$ to $p$) and $w^1_{q',p'} = w^2_{q',p'}$ for $(q', p') \neq (q, p)$ (i.e., all other channels are unchanged).

A *sequential run* of $\mathcal{A}$ is a sequence $d_1, (a_1, c_1), d_2, (a_2, c_2), \ldots, (a_n, c_n), d_{n+1}$ with $d_i$ configurations, $a_i \in \Sigma$ and $c_i \in C$ such that $d_i \xrightarrow{a_i,c_i} d_{i+1}$ for all suitable $i$. It is accepting if $d_1 = ((\iota_p)_{p\in\mathcal{P}}, (\varepsilon)_{p,q\in\mathcal{P}})$ and $d_{n+1} = (f, (\varepsilon)_{p,q\in\mathcal{P}})$ for some $f \in F$. Finally, $L(\mathcal{A}) \subseteq \Sigma^*$ is the set of words $a_1a_2\cdots a_n$ such that there exists an accepting sequential run $d_1, (a_1, c_1), d_2, (a_2, c_2), \ldots, (a_n, c_n), d_{n+1}$.

The alternative definition of the semantics of a CFM $\mathcal{A}$ uses MSCs for representing successful runs. This idea goes back to Mazurkiewicz traces and asynchronous automata (see [30], where it is used for obtaining the equivalence between MSO and asynchronous automata). Here, we define a partial order run of a CFM as an MSC, the events of which are labeled consistently by local states of the CFM. To this purpose, let $M = (E, \lambda, \text{msg}, (<_p)_{p\in\mathcal{P}})$ be an MSC and $\rho : E \rightarrow \bigcup_{p\in\mathcal{P}} S_p$ be a mapping labeling each event on process $p$ by some local state from $S_p$. For this mapping, we define a second mapping $\rho^- : E \rightarrow \bigcup_{p\in\mathcal{P}} S_p$ as follows. Let $e \in E_p$. If there is $e' \in E_p$ such that $e' \lessdot_p e$ then $\rho^-(e) = \rho(e')$. Otherwise (i.e., if $e$ is minimal in $(E_p, <_p)$), we set $\rho^-(e) = \iota_p$. The idea behind these notations is fairly simple: $\rho(e)$ is the state of the local machine $\mathcal{A}_p$ *after* executing event $e$, whereas $\rho^-(e)$ is the state the local machine was in *before* executing $e$. Then the mapping $\rho$ is a *run* if for any $s \in E$

with $\lambda(s) = p!q$ and $\mathrm{msg}(s) = r$, there is some control message $c \in C$ such that $\rho^-(s) \xrightarrow{p!q,c}_p \rho(s)$ and $\rho^-(r) \xrightarrow{q?p,c}_q \rho(r)$.

Now let $\rho$ be a run on the MSC $M$. If $E_p \neq \emptyset$, let $s_p = \rho(e_p)$ where $e_p$ is the maximal event in $(E_p, <_p)$. Otherwise, define $s_p = \iota_p$. The run $\rho$ is *successful* if the tuple $(s_p)_{p \in \mathcal{P}}$ belongs to the set of global final states $F$. An MSC is *accepted* by the CFM $\mathcal{A}$ if it admits a successful run. We will denote by $\mathcal{L}(\mathcal{A})$ the set of MSCs accepted by $\mathcal{A}$. Clearly, an MSC can admit several (accepting) runs of a CFM.

It is straightforward to prove the relation between the two languages of a CFM:

**Proposition 2.4** *Let $\mathcal{A}$ be a CFM. Then $L(\mathcal{A}) = \mathrm{Lin}(\mathcal{L}(\mathcal{A}))$.*

## 2.3 CMSC-graphs

An MSC stands for a single communication scenario. In order to specify the behavior of a communicating system, it is necessary to describe (finite or infinite) sets of MSCs. The simplest way is to list all possible scenarios in a library (possibly distinguishing between positive and negative scenarios). However, such libraries are huge, therefore hard to manipulate. Another simple way to do this was proposed in the Z.120 standard through high-level MSCs (denoted here as MSC-graphs). We define now the more general CMSC-graphs [13], that can be viewed formally as a kind of regular expressions over communication events.

We need first to define the composition of two CMSCs. Intuitively, to compose CMSCs $M_1$ and $M_2$, we glue the corresponding process lines together and draw the second CMSC below the first one. Since we deal with CMSCs instead of MSCs, the composition is slightly more subtle, and we need to make it formal. First we need the restriction of a CMSC $M = (E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ to a subset $F \subseteq E$ of events: It is the CMSC $M|_F = (F, \lambda^F, \mathrm{msg}^F, (<_p^F)_{p \in \mathcal{P}})$ with $\lambda^F = \lambda|_F$, $\mathrm{msg}^F(s) = r$ if $\mathrm{msg}(s) = r$ and $s, r \in F$, and $<_p^F \, = \, <_p \cap (F \times F)$.

**Definition 2.5** *Let $M_i = (E^i, \lambda^i, \mathrm{msg}^i, (<_p^i)_{p \in \mathcal{P}})$ for $i = 1, 2$ be CMSCs. The composition $M_1 \cdot M_2$ is the set of CMSCs $M = (E_1 \uplus E_2, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ such that*

- *$M|_{E_i} = M_i$ for $i = 1, 2$, and*
- *$e \in E_2$ and $e \leq e'$ imply $e' \in E_2$ for any $e, e' \in E_1 \uplus E_2$ (i.e., the process lines are glued putting $M_1$ above $M_2$ and at most messages from $M_1$ to $M_2$ are added).*

*This composition can naturally be extended to a binary operation on sets of CMSCs by $\mathcal{M}_1 \cdot \mathcal{M}_2 = \bigcup_{M_1 \in \mathcal{M}_1, M_2 \in \mathcal{M}_2} M_1 \cdot M_2$.*

As an illustration of this definition, consider the two CMSCs $M_1$ and $M_2$ depicted in Figure 2. Figure 3 shows some more CMSCs (the first one is actually an MSC). The common feature of all these six CMSCs is that $M_1$ is a downwards closed substructure, $M_2$ is an upwards closed substructure, and there are no further events. Hence all these CMSCs are compositions of $M_1$ and $M_2$ (and there are no further ones). Thus, Figure 3 depicts the set $M_1 \cdot M_2$.

Note that the set $M_1 \cdot M_2$ is non-empty for any CMSCs $M_1$ and $M_2$ (in any case, one can just put the two CMSCs one after the other and concatenate the process lines). If $e$ is an unmatched send in $M_2$, then it is unmatched in any CMSC $M$ from $M_1 \cdot M_2$ as well: if there was a matching receive $e'$, then it would have to be an event of $M_2$ by the second item in the definition. But $M_2$ is a substructure of $M$, i.e., there are no additional edges between events from $M_2$. Similarly, any unmatched receive in $M_1$ remains unmatched in $M$. Thirdly note that the concatenation of CMSC-languages is associative. Last but not least, as we saw in the illustrating example in Figure 3, the product $M_1 \cdot M_2 \cdot M_3 \cdot \ldots M_n$ can contain more than one CMSC. However, it can contain at most one MSC for the following reason: for any two distinct MSCs $N_1$ and $N_2$, there exists a process $p \in \mathcal{P}$ such that $N_1|_{E_p^1}$ and $N_2|_{E_p^2}$ differ (where $E_p^i$ is the set of events of process $p$ in the MSC $N_i$). But if $N_1$ and $N_2$ belong to the product above, then $N_i|_{E_p^i}$ is the concatenation of the words $M_j|_{E_p^j}$. Hence $N_1|_{E_p^1} = N_2|_{E_p^2}$.

Compositions $M_1 \star M_2$ of CMSCs were also defined in [13, 23]. Differently from our considerations here, they considered only the first CMSC from Figure 3 as a legitimate composition. The definition from [13, 23] differs from ours in several other aspects. First, it is only a partial operation. In particular, it is not defined if the first factor contains an unmatched receive event. In addition, the composition from [13, 23] is, even if defined, not associative. But in relevant cases, the two definitions are closely related: $(\cdots((M_1 \star M_2) \star M_3) \cdots \star M_n)$ is defined and an MSC iff $M_1 \cdot M_2 \cdot M_3 \cdots M_n$ contains an MSC, in which case these two MSCs are equal.

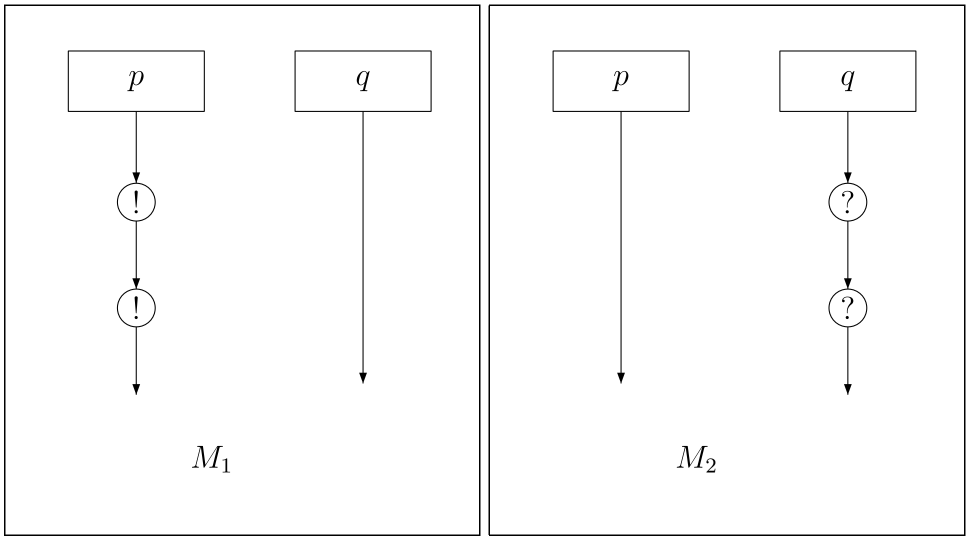

Figure 2: CMSCs $M_1$ and $M_2$

**Definition 2.6** *A* CMSC-graph *is a labeled graph* $G = (V, \rightarrow, \lambda, V^0, V^f)$ *where*

- $V$ *is the finite set of vertices.*
- $V^0, V^f \subseteq V$ *are sets of initial/final vertices respectively.*

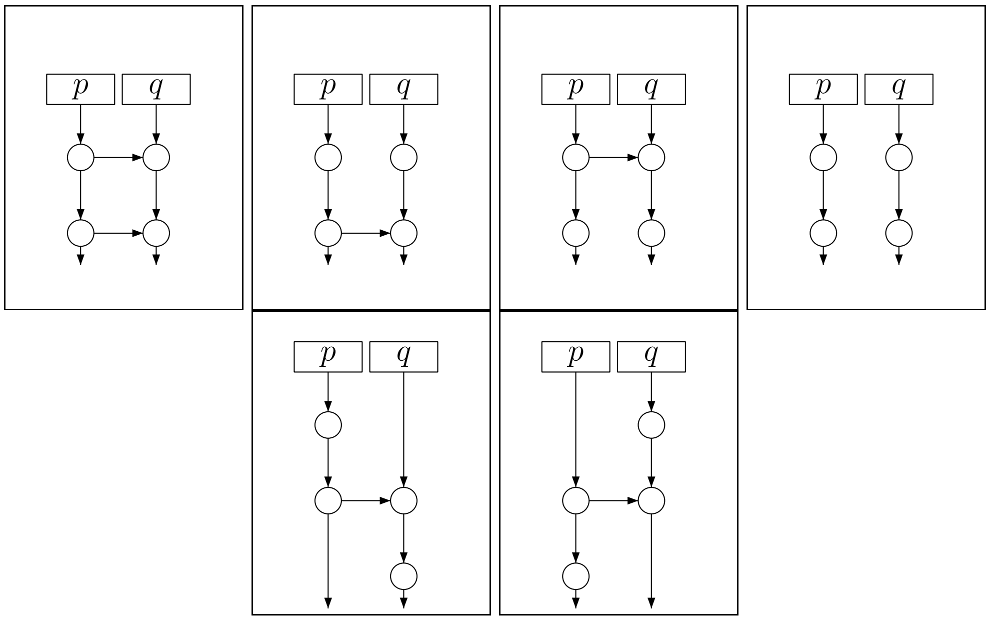

Figure 3: All compositions of $M_1$ and $M_2$

- $\rightarrow \subseteq V \times V$ *is the set of edges.*
- $\lambda : V \rightarrow \mathbb{CMSC}$ *labels a node* $v$ *with the CMSC* $\lambda(v)$.

A path in the CMSC-graph $G$ is a sequence $v_1, \ldots, v_n$ of nodes in $V$ such that $v_i \rightarrow v_{i+1}$ for all $i$. It is accepting if $v_1 \in V^0$ and $v_n \in V^f$. An MSC is accepted by $G$ if it labels some accepting path of $G$. The set of all MSCs accepted by $G$ is denoted $\mathcal{L}(G)$. A CMSC-graph is an *MSC-graph*, if all nodes are labeled by MSCs.

CMSC-graphs have been considered in [13] and [23]. Here, we deviate from the definition in [13] by introducing final states. Madhusudan and Meenakshi's version of CMSC-graphs [23] has final states like ours. On the other hand, they require that any path starting in an initial node admits some composition that is a CMSC without unmatched receive events. Thus, our definition is more general than those considered in [13, 23] (although, later, we will consider safe CMSC-graphs that coincide with the CMSC-graphs from [23]).

Figure 4 depicts a CMSC-graph. Here, we have two processes named *host* and *function*. In the leftmost node of the CMSC-graph, one finds a CMSC. This CMSC describes that *host* first sends an initialization message, and then another message to *function*. Immediately after receiving a message, *function* acknowledges. While the sending of these acknowledgments is part of the current CMSC node, their receiving by *host* is located in the next node. Thus, after executing this CMSC, the channel from *function* to *host* contains two acknowledgments, while the other channel is empty. Altogether, the CMSC-graph describes all MSCs where *function* immediately acknowledges messages it gets. The first message from *host* to *function* initializes the transfer. *Host* then starts sending the actual message. Since it does not get the acknowledgment in time, it must resend it, before getting the first acknowledgment.

Then it iterates between sending a message and receiving the acknowledgment from the message before, and at the end the channel is empty.

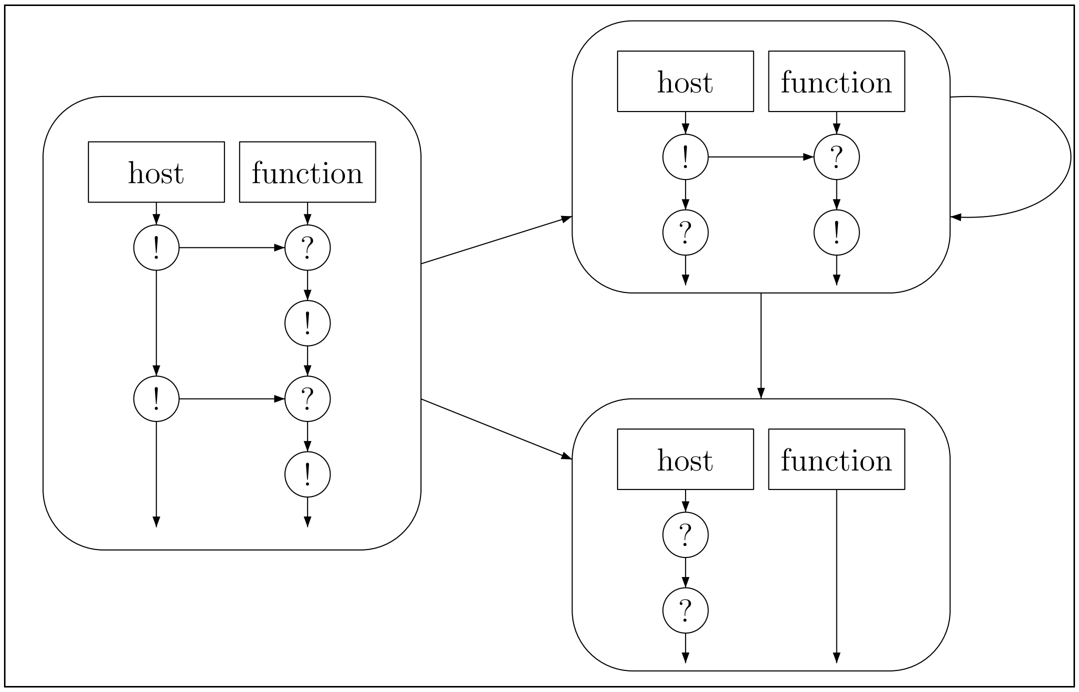

Figure 4: A CMSC-graph specifying transactions of USB 1.1.

The size $|M|$ of a CMSC $M$ is the number of its events. The size $|G|$ of a CMSC-graph $G$ is $\sum_{v \in V} |\lambda(v)|$.

## 2.4 Monadic second order logic

Logic is a classical formalism used to describe properties of various structures, like words, trees, pomsets, graphs etc. This also applies to structures like MSCs. Thus, after CFMs and CMSC-graphs, logic is another means for specifying sets of MSCs. We consider here monadic second order logic, that is the classical formalism over the structures mentioned above. The syntax is defined as follows.

**Definition 2.7** *For a set $\mathcal{R}$ of binary relations,* MSO($\mathcal{R}$)-formulas over the alphabet $\Gamma$ *are defined by the syntax*

$$\varphi ::= v_a(x) \mid R(x,y) \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists X \varphi \mid \exists x \varphi$$

*where $R \in \mathcal{R}$, $a \in \Gamma$, $x, y$ are first order variables, and $X$ is a second order variable.*

The relations in $\mathcal{R}$ used in the paper are the message relation msg, the visual order $\leq$, the process order $(<_p)_{p \in \mathcal{P}}$ and the immediate process successor $(\lessdot_p)_{p \in \mathcal{P}}$. An MSO($\mathcal{R}$)-formula over the alphabet $\Sigma$ is interpreted on an MSC $M = (E, \lambda, \text{msg}, (<_p)_{p \in \mathcal{P}})$ as expected. We have $M \models v_a(x)$ if $\lambda(x) = a$, $M \models x \leq y$ if $x \leq y$ in the visual order on $M$, and $M \models \mathrm{msg}(x,y)$ if $x \in S$ and $\mathrm{msg}(x) = y$. Moreover, $M \models (x <_p y)$ if $x <_p y$ and $M \models (x \lessdot_p y)$ if $y$ is the immediate successor of $x$ w.r.t. $<_p$.

For an MSO-formula $\varphi$ over $\Sigma$ without free variables, let $\mathcal{L}(\varphi)$ denote the set of MSCs that satisfy $\varphi$. We will also consider existential monadic second order logic (EMSO). An EMSO formula is of the form $\exists X_1 \dots X_n \varphi$ with $\varphi$ a first order formula.

An MSO($\leq$)-formula over an alphabet $\Gamma$ can be interpreted on $\Gamma$-labeled partial orders $M = (E, \leq, \lambda)$ with $\lambda : E \to \Gamma$ as usual, by letting $M \models v_a(x)$ if $\lambda(x) = a$ and $M \models x \leq y$ if $x \leq y$. Note that words over $\Gamma$ can be considered in a natural way as $\Gamma$-labeled linear orders. Using this interpretation, we write $w \models \varphi$ to denote that the word $w$ (more precisely: the associated linear order) satisfies $\varphi$. By $L(\varphi)$ we denote the set of *words* over $\Gamma$ that satisfy $\varphi$.

We discuss now some subtle differences between the logics MSO($(\lessdot_p)_{p\in\mathcal{P}}, \mathrm{msg}$) and MSO($\leq, \mathrm{msg}$) when applied to MSCs: Note that $x \lessdot_p z$ is equivalent to $x < z \wedge \forall y (x < y \leq z \to y = z) \wedge \bigvee_{p\in\mathcal{P}} \bigvee_{a,b\in\Sigma_p} v_a(x) \wedge v_b(z)$. Hence any formula $\varphi$ from MSO($(\lessdot_p)_{p\in\mathcal{P}}, \mathrm{msg}$) can be translated into an equivalent formula $\psi$ from MSO($\leq, \mathrm{msg}$). Note that $\psi$ is an existential formula whenever $\varphi$ is existential.

Conversely, the formula $x \leq y$ is equivalent to $\forall X (x \in X \wedge \forall z, z' (z \in X \wedge (\bigvee_{p\in\mathcal{P}} z \lessdot_p z' \vee \mathrm{msg}(z,z')) \to z' \in X) \to y \in X)$. Hence, it is also possible to translate any formula $\psi$ from MSO($\leq, \mathrm{msg}$) into an equivalent formula $\varphi$ from MSO($(\lessdot_p)_{p\in\mathcal{P}}, \mathrm{msg}$). But even if $\psi$ is existential, the resulting formula $\varphi$ is not existential anymore.

Thus, the full logics MSO($\leq, \mathrm{msg}$) and MSO($(\lessdot_p)_{p\in\mathcal{P}}, \mathrm{msg}$) are equally expressive, but the existential fragment of the former could be more expressive than the existential fragment of the latter (which is actually the logic considered in [7]).

## 2.5 Mazurkiewicz traces

We will establish a relationship between the expressive power of the different MSC formalisms presented until now. The main tool will be Mazurkiewicz traces [24, 9] that we introduce next.

A *trace alphabet* is a pair $(\Omega, I)$ consisting of an alphabet $\Omega$ and a symmetric and irreflexive relation $I \subseteq \Omega^2$. The relation $I$ will be referred to as the *independence relation*; its complement $D = \Omega^2 \setminus I$ is the *dependence relation*.

Let $\sim_I \subseteq \Omega^* \times \Omega^*$ be the congruence on the free monoid $\Omega^*$ generated by the equations $ab \sim_I ba$ for all $(a,b) \in I$. A *trace* is an equivalence class $[w]_I$ of this equivalence relation. Further, the *$I$-closure* of a set $L \subseteq \Omega^*$ is the set $[L]_I = \bigcup_{w\in L} [w]_I$ of all words that are $\sim_I$-equivalent to some element of $L$. If $L = [L]_I$, we say that $L$ is *closed under $I$-commutation* (or $I$-closed for short).

An alternative way to define traces is via labeled partially ordered sets. Any word $u = a_1 a_2 \cdots a_n$ with $a_i \in \Omega$ defines a labeled poset $t_u = (E, \leq_I, \lambda)$ where

- $E = \{1, \dots, n\}$,
- $\lambda(i) = a_i$, and
- $\leq_I$ is the least partial order on $E$ such that $i \leq_I j$ whenever $i < j$ and $\lambda(i) D \lambda(j)$.

It belongs to the very basics of trace theory that two words $u$ and $v$ are equivalent w.r.t. $\sim_I$ iff the two labeled partial orders $t_u$ and $t_v$ are isomorphic. This implies in particular that $[u]_I$ is the set of linearizations of the labeled poset $t_u$. At places, it will be useful to consider a trace not as an equivalence class of words, but as (an isomorphism class of) labeled partial orders $t_u$. This allows in particular to interpret MSO($\leq_I$)-formulas in a trace and thereby to define notions like $[u]_I \models \varphi$ for a trace $[u]_I$.

Let $(E, \leq, \lambda)$ be an $\Omega$-labeled partially ordered set. Then there exists a word $u \in \Omega^*$ with $t_u \cong (E, \leq, \lambda)$ iff we have for any $e, f \in E$

- $e \lessdot f$ implies $(\lambda(e), \lambda(f)) \in D$, and
- if $e$ and $f$ are incomparable, then $(\lambda(e), \lambda(f)) \in I$.

We end this section by recalling some fundamental results from Mazurkiewicz trace theory (cf. [9]). Below, we say that a finite automaton $\mathcal{A}$ is *$D$-loop-connected* if for every loop of $\mathcal{A}$, the set of letters labeling the loop induces a connected subgraph of $(\Omega, D)$. By loop we mean a (not necessarily simple) cycle. Asynchronous automata are defined at the beginning of Section 5.

**Theorem 2.8** *Let $(\Omega, I)$ be a trace alphabet and let $L \subseteq \Omega^*$ be $I$-closed.*

1. *(Ochmański's theorem [26]) $L$ is regular iff there exists some $D$-loop-connected automaton $\mathcal{A}$ with $L = [L(\mathcal{A})]_I$.*
2. *(Zielonka's theorem [31]) $L$ is regular iff it is accepted by a deterministic asynchronous automaton.*
3. *$L$ is regular iff $L = \{u \in \Omega^* \mid t_u \models \varphi\}$ for some MSO($\leq_I$) formula $\varphi$ [30, 10].*

## 3 Model checking CMSC-graphs

In this section, we single out classes of CMSC-graphs such that both questions $\mathcal{L}(G) \overset{?}{\subseteq} \mathcal{L}(G')$ and $\mathcal{L}(G) \cap \mathcal{L}(G') \overset{?}{=} \emptyset$ become decidable (cf. Prop. 3.7). Recall that in general, these problems are undecidable for unrestricted CFMs or CMSC-graphs.

### 3.1 Strategy and definitions

Let $\mathcal{M}$ be a set of MSCs. A *set of representatives* (or *representative set*) of $\mathcal{M}$ is a set $X \subseteq \Sigma^*$ such that $X \subseteq \mathrm{Lin}(\mathbb{MSC})$ and $\mathcal{M} = \{M \in \mathbb{MSC} \mid X \cap \mathrm{Lin}(M) \neq \emptyset\}$. If $X$ is a *regular* set of representatives, then there exists $B$ such that $X$ consists of $B$-bounded linearizations only [23]. So, if such a set exists, $\mathcal{M}$ is $\exists$-bounded. Conversely, for any $\exists$-$B$-bounded set $\mathcal{M}$, the set $\mathrm{Lin}^B(\mathcal{M})$ of $B$-bounded linearizations is a set of representatives (but not necessarily regular).

Representative sets based on $B$-bounded linearizations can be used e.g. to do model checking beyond regular MSC languages, as shown in [23, 14]. The basic ideas are as follows: let $B \in \mathbb{N}$ and suppose $\mathcal{M} \subseteq \mathbb{MSC}^B$ and $\mathcal{M}' \subseteq \mathbb{MSC}$ are sets of MSCs where the former is assumed to be $\exists$-$B$-bounded. Then $\mathcal{M} \cap \mathcal{M}' = \emptyset$ iff

$\mathrm{Lin}^B(\mathcal{M}) \cap \mathrm{Lin}^B(\mathcal{M}') = \emptyset$ and $\mathcal{M} \subseteq \mathcal{M}'$ iff $\mathrm{Lin}^B(\mathcal{M}) \subseteq \mathrm{Lin}^B(\mathcal{M}')$. Now suppose that

(1) $\mathcal{M}$ has a regular set of $B$-bounded representatives accepted by the automaton $\mathcal{A}$, and

(2) $\mathrm{Lin}^B(\mathcal{M}')$ is accepted by the automaton $\mathcal{A}'$.

Then we can decide both the questions "$\mathcal{M} \cap \mathcal{M}' \overset{?}{=} \emptyset$" and "$\mathcal{M} \overset{?}{\subseteq} \mathcal{M}'$" since they are equivalent to the corresponding questions for the languages of $\mathcal{A}$ and $\mathcal{A}'$.

In order to use this observation, we therefore need mechanisms for the specification of sets of MSCs $\mathcal{M}$ that allow to calculate an automaton

(1) that accepts some regular set of $B$-bounded representatives of $\mathcal{M}$, or

(2) that accepts $\mathrm{Lin}^B(\mathcal{M})$.

Let $G = (V, \rightarrow, \lambda, V^0, V^f)$ be a CMSC-graph. For any node $v \in V$, choose a linearization $\ell(v) \in \Sigma^*$ of the MSC $\lambda(v)$ and set

$$K_G = \{\ell(v_1) \cdots \ell(v_n) \mid (v_1, \ldots, v_n) \text{ is an accepting path of } G\}$$

Then $K_G$ is regular and, for any $M \in \mathcal{L}(G)$, we have $K_G \cap \mathrm{Lin}(M) \neq \emptyset$. But $K_G$ can contain words that are no linearizations of MSCs. Hence, in general, $K_G$ is not necessarily a set of representatives. For instance, consider the CMSC-graph $G$ consisting of two nodes $v_0, v_1$ with $v_0$ labeled by a CMSC consisting of a send $p!q$ and $v_1$ labeled by a CMSC consisting of a receive $q?p$. The transitions are $v_0 \rightarrow v_0, v_0 \rightarrow v_1$ and $v_1 \rightarrow v_1$. Moreover, $v_0$ is the initial state and $v_1$ the final one. We have $K_G = (p!q)^+(q?p)^* \ni p!q(q?p)^2$. Since $p!q(q?p)^2$ is no linearization of any MSC, the set $K_G$ is not the representative set of any set of MSCs. The definition below ensures that $K_G$ is a representative set:

**Definition 3.1** *A CMSC-graph $G = (V, \rightarrow, \lambda, V^0, V^f)$ is* safe *if for any accepting path $(v_0, v_1, \ldots, v_n)$ in $G$ (i.e., $v_0 \in V^0$ and $v_n \in V^f$), the set $\lambda(v_0) \cdot \lambda(v_1) \cdots \lambda(v_n)$ contains an MSC.*

Note that the MSC whose existence is required above is uniquely determined since any product of CMSCs contains at most one MSC. Safe CMSC-graphs are precisely the CMSC-graphs considered in [23] (called "CMSG" there).

**Lemma 3.2** *Let $G$ be a safe CMSC-graph. Then $K_G$ is a regular set of $|G|$-bounded representatives of $\mathcal{L}(G)$. In particular, $\mathcal{L}(G)$ is $\exists$-$B$-bounded for any $B \geq |G|$.*

*Proof.* The regularity of $K_G$ is obvious by the very definition. Next, let $M \in \mathcal{L}(G)$. Then there exists an accepting path $(v_0, v_1, \ldots, v_n)$ in $G$ with $M \in \lambda(v_0) \cdot \lambda(v_1) \cdots \lambda(v_n)$. Since $\lambda(v_i)$ is a convex substructure of $M$, the word $\ell(v_0)\ell(v_1)\ldots\ell(v_n)$ is a linearization of $M$. Hence $K_G \cap \mathrm{Lin}(M) \neq \emptyset$. Conversely let $M \in \mathbb{MSC}$ with $K_G \cap \mathrm{Lin}(M) \neq \emptyset$. Then there exists an accepting path $(v_0, v_1, \ldots, v_n)$ in $G$ with $\ell(v_0)\ell(v_1)\ldots\ell(v_n) \in \mathrm{Lin}(M)$ implying $M \in \lambda(v_0) \cdot \lambda(v_1) \cdots \lambda(v_n)$. Hence, indeed, $K_G$ is a regular set of representatives of $\mathcal{L}(G)$.

Now we show that the set $K_G$ consists of $|G|$-bounded linearizations only. So let $\pi = (v_0, v_1, \dots, v_n)$ be an accepting path in $G$ with $w = \ell(v_0)\ell(v_1)\dots\ell(v_n) \in K_G$ and let $u$ be some prefix of $w$. Let $i$ be minimal such that $u$ is a prefix of $u' = \ell(v_0)\ell(v_1)\dots\ell(v_i)$. We first "shorten" the word $u'$. Suppose $|u'| > |G|$. Since $|G|$ is the sum of the number of events of MSCs $\lambda(v)$ for $v \in V$, we obtain $i > |V|$. Hence there are $0 \leq a < b \leq i$ with $v_a = v_b$. Then the number of $p!q$-events in $\ell(v_a)\ell(v_{a+1})\dots\ell(v_{b-1})$ equals that of $q?p$-events (otherwise, the successful path $(v_0, v_1, \dots, (v_a, v_{a+1}, \dots v_{b-1})^2, v_b, \dots v_n)$ would not define any MSC). Deleting all loops in this path repeatedly, we find a path $\pi' = (v'_0, v'_1, \dots, v'_k)$ with mutually distinct nodes, $v_0 = v'_0$ and $v'_k = v_i$ such that, for any $p, q \in \mathcal{P}$ distinct, the difference of $p!q$- and $q?p$-events in $\pi$ equals that in $\pi'$. In addition, the length of $v' = \ell(v'_0)\ell(v'_1)\dots\ell(v'_k)$ is at most $|G|$ since the nodes $v'_j$ are mutually distinct. Recall that there is a suffix $u''$ of $\ell(v_i)$ such that $u' = uu''$. Hence there is a word $v$ with $v' = vu''$. Since $|v| \leq |G|$, the number of $p!q$-events and that of $q?p$-events differ by at most $|G|$. Hence the same holds for $u$. □

Note that the lemma says that safe CMSC-graphs satisfy requirement (1) above (namely, the construction of an automaton accepting a regular set of representatives). Next, we define another restriction on CMSC-graphs that allows to satisfy requirement (2).

**Definition 3.3** *The* communication graph *of a set $A \subseteq \Sigma$ is a graph whose vertices are the processes involved in $A$, and there is an (undirected) edge between vertices $p, q$ iff $A$ contains both a send $p!q$ from $p$ to $q$ and a receive $q?p$ on $q$ from $p$. The* communication graph *of a CMSC $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ is the communication graph of $A = \lambda(E)$. A path $(v_1, v_2, \dots, v_n)$ in a CMSC-graph is* connected *if any CMSC from $\ell(v_1) \cdot \ell(v_2) \cdots \ell(v_n)$ has a connected communication graph. A CMSC-graph $G$ is* loop-connected *if every loop of $G$ is connected.*

*A* globally-cooperative CMSC-graph *(or* gc-CMSC-graph *for short) is a CMSC-graph that is safe and loop-connected.*

Since any MSC-graph is clearly safe, it is globally-cooperative if and only if it is loop-connected which was the intention of the definition of globally-cooperative MSC-graphs in [14].

The CMSC-graph in Figure 4 is globally-cooperative.

Prop. 3.6 will show that, indeed, $\mathrm{Lin}^B(\mathcal{L}(G))$ is (effectively) regular for any globally-cooperative CMSC-graph for a suitable $B \leq |G|$. Before we can embark on this proof in Section 3.3, we investigate in Section 3.2 the relation between $\exists$-$B$-bounded sets of MSCs and traces. This connection will be crucial in the proof of the regularity of $\mathrm{Lin}^B(\mathcal{L}(G))$ for $G$ an gc-CMSC-graph. It will reappear later when we investigate the relation between MSO, CFMs, and gc-CMSCs.

## 3.2 Existential bounds and traces

Let $B$ be a positive integer that we fix for this section. We define a trace alphabet $(\Omega, I)$ with $\Omega_p = \Sigma_p \times \{0, \dots, B-1\}$ for $p \in \mathcal{P}$ and $\Omega = \bigcup_{p\in\mathcal{P}} \Omega_p$. The dependence relation $D \subseteq \Omega \times \Omega$ is given by $(x, i)D(y, j)$ if either $P(x) = P(y)$ or $\{(x, i), (y, j)\} =$

$\{(p!q,n),(q?p,n)\}$ for some $p,q,n$. Then $I = \Omega^2 \setminus D$ is symmetric and irreflexive, hence $(\Omega, I)$ is a trace alphabet.

We define now a mapping $\widetilde{\ }: \Sigma^* \to \Omega^*$ by numbering the events of the same type modulo $B$. Let $\widetilde{x_1 \cdots x_m} = (x_1,n_1)\dots(x_m,n_m)$, with $n_i = |\{j \le i \mid x_j = x_i\}| \bmod B$, i.e., modulo $B$, there are $n_i$ occurrences of the letter $x_i$ in the prefix $x_1x_2\dots x_i$. We also consider the projection $\pi : \Omega^* \to \Sigma^*$ given by $\pi(x,n) = x$ for $(x,n) \in \Omega$. A word $u \in \Omega^*$ is $B$-bounded if $\pi(u) \in \Sigma^*$ is $B$-bounded. We denote $\tilde{L} = \{\tilde{u} \mid u \in L\}$ for $L \subseteq \Sigma^*$.

Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ be an MSC. For $e \in E_p$, let $\lambda_I(e) = (\lambda(e), n)$ with $n = |\{f \in E \mid f \le_p e, \lambda(f) = \lambda(e)\}| \bmod B$ (i.e., $n$ is the number of events below $e$ labeled by the same element of $\Sigma$, modulo $B$). We associate with $M$ the structure[1] $\mathrm{tr}(M) = (E, \prec_B^*, \lambda_I)$. Figure 5 depicts the result when applying this operation to the MSC $M$ from Figure 1 with $B = 2$. Note that there is one additional edge from the first occurrence of $(q?p,0)$ to the second occurrence of $(p!q,0)$, this edge is an rev-edge. Since $M$ is $\exists$-2-bounded, the relation $\prec_2$ is acyclic by Lemma 2.2 and $\mathrm{tr}(M) := (E, \prec_2^*, \lambda_I)$ is an $\Omega$-labeled partial order. The reader can easily check that $\prec_1$ is not acyclic.

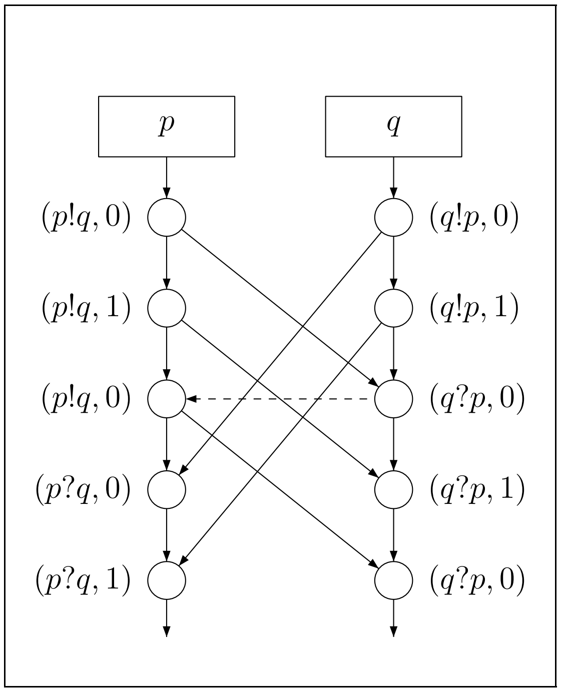

Figure 5: Trace $\mathrm{tr}(M)$ associated with the MSC $M$ of Fig. 1.

**Lemma 3.4** *Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ be an $\exists$-$B$-bounded MSC.*

1. *The labeled poset* $\mathrm{tr}(M)$ *is a trace over* $(\Omega, I)$.

[1]Recall that $\prec_B = \mathrm{msg} \cup \bigcup_{p\in\mathcal{P}} <_p \cup \mathrm{rev}$.

2. *If $u$ is a $B$-bounded linearization of $M$, then* $\mathrm{tr}(M) = t_{\tilde{u}}$ *and* $\mathrm{Lin}^B(M) = \pi([\tilde{u}]_I)$.

*Proof.* To show that $\mathrm{tr}(M)$ is a trace, let $e, f \in E$ be distinct. If $e \prec_B^* f$ and there is no event properly between these two, then $e \prec_B f$ and therefore $\lambda_I(e) D \lambda_I(f)$ by the very definition of $\prec_B$ and $D$. Now let $e$ and $f$ be incomparable w.r.t. $\prec_B^*$ and suppose $\lambda_I(e) D \lambda_I(f)$. Then $e$ and $f$ are executed by distinct processes, i.e., we get w.l.o.g. $\lambda_I(e) = (p?q, n)$ and $\lambda_I(f) = (q!p, n)$ from the definition of $D$. Let $\mathrm{msg}(f) = e'$ hence $f \prec_B e'$. Since $\lambda(e) = \lambda(e')$, these two events are related by $<_p$ and therefore by $\prec_B^*$. Since $e$ and $f$ are incomparable, we obtain $e <_p e'$. Since $\mathrm{msg}(f) = e'$, there are as many $p?q$-labeled events below $e'$ as there are $q!p$-labeled nodes below $f$. But this number equals $n$ (modulo $B$) and therefore the number of $p?q$-labeled events below $e$. By the very definition of $\prec_B$, this implies $e \prec_B f$ contradicting our assumption. Hence $\mathrm{tr}(M)$ is indeed a trace [9].

Next consider a $B$-bounded linearization $u$ of $M$. Since $u$ is $B$-bounded, it is also a linearization of $(E, \prec_B^*, \lambda)$. Hence $\tilde{u} \in \mathrm{Lin}(E, \prec_B^*, \lambda_I)$. Since $\mathrm{tr}(M)$ is a trace we obtain $\mathrm{tr}(M) = t_{\tilde{u}}$.

Now let $v$ be another $B$-bounded linearization of $M$. Then $[\tilde{v}]_I = \mathrm{Lin}(\mathrm{tr}(M)) = [\tilde{u}]_I$ implies $\tilde{v} \sim_I \tilde{u}$ and therefore $v = \pi(\tilde{v}) \in \pi([\tilde{u}]_I)$ which proves $\mathrm{Lin}^B(M) \subseteq \pi([\tilde{u}]_I)$. Conversely let $v \in \pi([\tilde{u}]_I)$. Then there exists $v' \sim_I \tilde{u}$ with $\pi(v') = v$. Hence $v' \in \mathrm{Lin}(\mathrm{tr}(M))$. Since the relation $\prec_B^*$ contains the visual order $<$ of $M$, the word $v'$ is also a linearization of $(E, \leq, \lambda_I)$. Hence $v = \pi(v') \in \mathrm{Lin}(M)$. Furthermore, $v' \in \mathrm{Lin}(E, \prec_B^*, \lambda_I)$ implies that $v$ is also a linearization of $(E, \prec_B^*, \lambda)$. □

## 3.3 Model checking

Let $G$ be a safe CMSC. Then we saw that the language $K_G$ is a regular set of representatives. In particular, $\mathcal{L}(G)$ is $\exists$-$B$-bounded for some $B \in \mathbb{N}$. In general, this does not imply that $\mathrm{Lin}^B(\mathcal{L}(G))$ is regular (the desired property (2) as explained in Section 3.1). We now exhibit the relation between $\exists$-$B$-bounded MSCs and traces to show that $\mathrm{Lin}^B(\mathcal{L}(G))$ is regular provided $G$ is a gc-CMSC-graph.

**Lemma 3.5** *Let $G$ be a gc-CMSC-graph, let $K_G$ be the regular language described in Section 3.1, and let $|G| \leq B \in \mathbb{N}$. Then there exists a $D$-loop-connected automaton $\mathcal{B}$ with $|G|B^{2|\mathcal{P}|}$ many states such that*

1. $L(\mathcal{B}) = \widetilde{K_G}$ *and*
2. *if $t_i, u_j \in \Omega^*$ are non-empty words with $t_i I u_j$ for $i < j$ such that $w = t_0 u_1 \cdots t_{k-1} u_k t_k$ labels some path in $\mathcal{B}$, then $k < k_0 = (|\mathcal{P}|^2 B + |\mathcal{P}|)|G|$.*

*Proof.* In the proof of Lemma 3.2, we saw that there is a finite automaton $\mathcal{A}$ with $|G|$ states that accepts $K_G \subseteq \Sigma^*$. From $\mathcal{A}$, we construct an automaton $\mathcal{B}$ as follows: states of $\mathcal{B}$ are tuples $(r, (n_a)_{a \in \Sigma})$ where $r$ is a state of $\mathcal{A}$ and $n_a \in \{0, \dots B-1\}$ are counters. The initial state consists of the initial state of $\mathcal{A}$ together with all counters being 0. There is a transition $(r, (n_a)_{a\in\Sigma}) \xrightarrow{(b,n)} (r', (n'_a)_{a\in\Sigma})$ of $\mathcal{A}$ iff $r \xrightarrow{b} r'$ in $\mathcal{B}$, $n = n'_b = (n_b + 1) \bmod B$, and $n_a = n'_a$ for $a \neq b$. A state $(r, (n_a)_{a\in\Sigma})$ is final in $\mathcal{B}$ if $r$ was final in $\mathcal{A}$. In $\mathcal{B}$, there is an $u$-labeled path from the initial state to $(r, (n_a)_{a\in\Sigma})$ iff $n_a = |u|_a \bmod B$, $\widetilde{\pi(u)} = u$ and there is a $\pi(u)$-labeled path in $\mathcal{A}$ from the initial state to $r$. Hence $\mathcal{B}$ accepts $\widetilde{L(\mathcal{A})} = \widetilde{K_G}$. From Lemma 3.4, we therefore get $\pi([L(\mathcal{B})]_I) = \mathrm{Lin}^B(G)$. Now let $u$ be the label of some loop in $\mathcal{B}$. Since $\mathcal{B}$ accepts only words of the form $\tilde{v}$ with $v$ a linearization of some MSC, we get

- for any $p, q \in \mathcal{P}$ and $n \in \{0, \ldots, B-1\}$, $(p!q, n)$ appears in $u$ iff $(q?p, n)$ appears in $u$ and
- for any $n, m \in \{0, \ldots, B-1\}$ and $a \in \Sigma$, $(a, n)$ appears in $u$ iff $(a, m)$ appears in $u$.

Furthermore, $\pi(u)$ labels a loop in $\mathcal{A}$, too. Since $G$ is globally-cooperative, the alphabet of $\pi(u)$ has a connected communication graph. Thus, the alphabet of $u$ is connected, i.e., $\mathcal{B}$ is $D$-loop-connected.

To show the second statement by contradiction, assume $k \geq (|\mathcal{P}|^2 B + |\mathcal{P}|)|G|$. By the construction of $\mathcal{B}$ from $\mathcal{A}$, there is an $\pi(w)$-labeled path in $\mathcal{A}$. Since $\mathcal{A}$ has $|G|$ states, we have a set $J$ of $|\mathcal{P}|^2 B + |\mathcal{P}|$ indices such that the subpath labeled by $\pi(t_i)$ starts in the same node $v$ of $\mathcal{A}$ for all $i \in J$. In particular, for any two consecutive $i, j \in J$, $\pi(t_i \cdots u_{j-1})$ labels a loop around $v$. Since $G$ is globally-cooperative, for each such loop there is either some process occurring in both $\pi(t_i \cdots t_{j-2})$ and $\pi(u_{i+1} \cdots u_{j-1})$ (shared process) or some $p!q$ in $\pi(t_i \cdots t_{j-2})$ and $q?p$ in $\pi(u_{i+1} \cdots u_{j-1})$, or vice-versa (shared channel). There are at most $|\mathcal{P}|$ such loops where $\pi(t_i \cdots t_{j-2})$ and $\pi(u_{i+1} \cdots u_{j-1})$ share a process, since $t_i I u_j$ for every $i < j$ (the same process cannot be shared in two different loops). Therefore, we have at least $|\mathcal{P}|^2 B$ loops that share a channel (second case above). Thus, there are $p, q \in \mathcal{P}$ and $n \in \{0, \ldots, B-1\}$ and at least two loops such that the corresponding $t$- and $u$-subpaths contain $(p!q, n)$ and $(q?p, n)$, resp. (or vice versa). But this means that some $t_i$ in the first loop is not independent from some $u_j$ in the second loop, contradicting $t_i I u_j$ for all $i < j$. $\square$

**Proposition 3.6** *Let $G$ be a gc-CMSC-graph and $|G| \leq B \in \mathbb{N}$. Then $\mathrm{Lin}^B(G)$ is regular and one can construct an automaton of size at most $|G|^{5|\mathcal{P}|^4 B^2 |G|}$ recognizing it.*

*Proof.* Let $\mathcal{B}$ be the automaton constructed in Lemma 3.5. By Lemma 3.5(2) and [25], there exists an automaton accepting $[L(\mathcal{B})]_I$, of size $(|\mathcal{B}|^2 2^{|\Omega|})^{k_0}$, which is at most $(|G|^2 B^{4|\mathcal{P}|} 2^{|\Omega|})^{(|\mathcal{P}|B+1)|\mathcal{P}||G|}$. One can check that the last value is asymptotically less than $|G|^{5|\mathcal{P}|^4 B^2 |G|}$. $\square$

Now, we get the first decidable model checking problem. This statement was known for MSC-graphs in the case where both $G$ and $G'$ are globally-cooperative [14].

**Proposition 3.7** *The following problems are decidable*
*input: safe CMSC-graph $G$ and gc-CMSC-graph $G'$*
*questions: Is the intersection $\mathcal{L}(G) \cap \mathcal{L}(G')$ empty? Does $\mathcal{L}(G) \subseteq \mathcal{L}(G')$ hold?*

*Proof.* Let $B = \max |G|, |G'|$. Then, by Lemma 3.2, the set $\mathcal{L}(G)$ admits a regular set $K_G \subseteq \mathrm{Lin}^B(\mathbb{MSC})$ of representatives. By Prop. 3.6, the set $\mathrm{Lin}^B(\mathcal{L}(G'))$ is regular. Since $\mathcal{L}(G)$ is $\exists$-$B$-bounded, we get $\mathcal{L}(G) \cap \mathcal{L}(G') = \emptyset$ iff $\mathrm{Lin}^B(\mathcal{L}(G)) \cap \mathrm{Lin}^B(\mathcal{L}(G')) = \emptyset$. Since $K_G$ is a set of $B$-bounded representatives of $\mathcal{L}(G)$, this is equivalent to the emptiness of $K_G \cap \mathrm{Lin}^B(\mathcal{L}(G'))$. But this last question is decidable since both sets are effectively regular.

Similarly, $\mathcal{L}(G) \subseteq \mathcal{L}(G')$ iff $\mathrm{Lin}^B(\mathcal{L}(G)) \subseteq \mathrm{Lin}^B(\mathcal{L}(G'))$ since $\mathcal{L}(G)$ is $\exists$-$B$-bounded. But $\mathrm{Lin}^B(\mathcal{L}(G)) \subseteq \mathrm{Lin}^B(\mathcal{L}(G'))$ iff $K_G \subseteq \mathrm{Lin}^B(\mathcal{L}(G'))$ since $K_G$ is a set of $B$-bounded representatives. Since these two sets are effectively regular, the inclusion problem is decidable as well. $\square$

**Remark 3.8** The complexity of the two model checking instances in Proposition 3.7 is PSPACE for the intersection and EXPSPACE for the inclusion. The reason is that the automaton recognizing $\mathrm{Lin}^B(\mathcal{L}(G'))$ is exponential in both $|G|$ and $|G'|$ (whereas the automaton for the representative set $K_G$ is polynomial).

Similar model checking problems can be formulated for CFMs and MSO-sentences. To show their decidability, we will proceed as above, i.e., compute regular sets of representatives and automata for the set of all $B$-bounded linearizations. These calculations are the core of the following two sections. In Section 6, we will come back to the model checking problem (see Cor. 6.1).

## 4 A Kleene theorem for existentially bounded MSCs

The main result is stated in the following theorem, which generalizes the results of [17, 15, 21] from universally bounded to existentially bounded sets of MSCs. We use a unified proof technique, interpreting MSCs as traces and applying known constructions for traces.

**Theorem 4.1** *Let $B \in \mathbb{N}$ and $\mathcal{M} \subseteq \mathbb{MSC}^B$ be a set of $\exists$-$B$-bounded MSCs. Then the following assertions are equivalent:*

*(1) $\mathcal{M} = \mathcal{L}(\mathcal{A})$ for some CFM $\mathcal{A}$.*

*(2) $\mathcal{M} = \mathcal{L}(\varphi)$ for some EMSO($<_p, \mathrm{msg}$) formula $\varphi$.*

*(3) $\mathcal{M} = \mathcal{L}(\varphi)$ for some MSO($\leq, \mathrm{msg}$) formula $\varphi$.*

*(4) $\mathcal{M} = \mathcal{L}(G)$ for some gc-CMSC-graph $G$.*

*(5) $\mathrm{Lin}^B(\mathcal{M})$ is a regular set of representatives for $\mathcal{M}$.*

Similar results were known before: [15] proves the equivalence of (1), (3), and (5) for universally bounded sets of MSCs. In addition, they show that in this case of universally bounded sets of MSCs, deterministic CFMs have the full expressive power (we do not know whether this is the case for existentially bounded sets of MSCs as well). Their proof uses ideas from the theory of Mazurkiewicz traces, but these ideas have to be reproved in the more complex setting of MSCs. The main focus of [17] are universally bounded sets of infinite MSCs where, again, the equivalence of (1), (3), and (5) is shown. In particular, it is shown that deterministic CFMs with Muller acceptance have the same expressive power as monadic second order logic. The proofs in [17] are based on trace theory but, differently from [15], it uses a different technique to transfer known results directly from traces to CFMs. This technique is based on the encoding presented in Section 3.2 and it allows to preserve determinism of distributed automata. In particular, this gives an alternative proof of results in [15]. Sections 3.2 and 5.1 extend Kuske's technique [17] to existentially bounded sets. For arbitrary sets of MSCs, [7] proves the equivalence of (1) and (2). They also show that the logic $\mathrm{MSO}(\leq, \mathrm{msg})$ is properly more powerful and that deterministic CFMs are properly weaker than general CFMs. The paper [23] proves in particular that a set of MSCs has a regular set of representatives iff it is the language of some safe CMSC-graph (note that both these notions are weaker than those in (4) and (5), resp.). In [21], it is shown that a finitely generated MSC language is the language of a loop-connected MSC-graph (called c-HMSC there) iff it is definable in $\mathrm{MSO}(\leq)$.

Recall that the equivalence of (1) and (2) was shown (even for arbitrary sets of MSCs) in [7]. The implication (2) to (3) is immediate. Proposition 3.6 shows the implication (4) to (5). We will show that (3) implies (5), that (5) implies (4), and finally that (5) implies (1).

The proofs use the trace alphabet $(\Omega, I)$ and in particular Theorem 2.8 at crucial points: For showing that (3) implies (5), we use the equivalence between $\mathrm{MSO}(<_I)$ and regular sets of traces [30, 10], i.e., Theorem 2.8(3). To prove that (4) and (5) are equivalent, we will use Ochmański's Theorem 2.8(1) [26]. We provide here an alternative proof to the one in [23]. Finally, to prove (5) implies (1), we will use Zielonka's Theorem 2.8(2) [31] and simulate asynchronous automata by CFMs. More precisely, we first build a CFM $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') \cap \mathbb{MSC}^B = \mathcal{M}$. Then we construct a CFM $\mathcal{A}''$ that generates precisely the set of $\exists$-$B$-bounded MSCs (this is actually the most difficult part of the proof). Since the intersection of CFM-accepted languages can be accepted by a CFM, (1) follows.

From a logical point of view, the implication (3)$\Rightarrow$(2) is of particular interest since it states the collapse of the quantifier alternation hierarchy. This collapse was known before for words and for traces [30]. A subtle point here is the use of the predecessor relation $\lessdot_p$ vs. the partial order relation $\leq$. As discussed before, any $\mathrm{MSO}(\leq, \mathrm{msg})$-formula can easily be translated into an equivalent formula from $\mathrm{MSO}((\lessdot_p)_{p\in\mathcal{P}}, \mathrm{msg})$. Thus, the nontrivial part of this implication concerns the collapse of the quantifier alternation hierarchy of the logic $\mathrm{MSO}((\lessdot_p)_{p\in\mathcal{P}}, \mathrm{msg})$ into its existential fragment. To do this translation, we proceed indirectly, as for words or traces, by showing (3)$\Rightarrow$(1)$\Rightarrow$(2): the MSO formula is transformed into a CFM whose behavior can be described by a formula of $\mathrm{EMSO}((\lessdot_p)_{p\in\mathcal{P}}, \mathrm{msg})$. A weaker consequence is that the logic $\mathrm{MSO}(\leq, \mathrm{msg})$ also collapses into its existential fragment. It is likely that this weaker consequence can be shown using the trace alphabet $(\Omega, I)$ and the corresponding statement for traces [30].

Note also that the implication (5)$\Rightarrow$(3) is shown indirectly via CFMs. An alternative proof based on the trace alphabet $(\Omega, I)$ and Theorem 2.8(3) seems possible as well.

## 4.1 From MSO to regular sets of representatives

We start proving the implication (3)⇒(5), i.e., we show that the set of $B$-bounded linearizations of $\mathcal{M}$ is regular whenever $\mathcal{M} = \mathcal{L}(\varphi) \subseteq \mathbb{MSC}^B$ for some MSO$(\leq, \mathrm{msg})$ formula $\varphi$ and some $B$. This fact was already shown by [23] for model checking CMSC-graphs against MSO$(\leq, \mathrm{msg})$, but with a different proof technique. Here, for the sake of completeness, we apply trace theory, using a result that allows to go from an MSO formula over traces to a regular set of words [10, 30].

**Proposition 4.2** *Let $\varphi$ be an MSO$(\leq, \mathrm{msg})$ formula and $B \in \mathbb{N}$ such that $\mathcal{L}(\varphi) \subseteq \mathbb{MSC}^B$. Then $\mathrm{Lin}^B(\mathcal{L}(\varphi))$ is a regular set of representatives.*

*Proof.* We recall that $\leq_I$ denotes the partial order of the trace $\mathrm{tr}(M)$ associated with the $\exists$-$B$-bounded MSC $M$.

Since the visual order $\leq$ of MSCs is the reflexive and transitive closure of $\mathrm{msg} \cup \bigcup_{p\in\mathcal{P}} <_p$, we can assume that $\varphi$ only uses the message relation msg and the process order $<_p$, $p \in \mathcal{P}$.

With the formula $\varphi$ over MSCs we associate an MSO formula $\widetilde{\varphi}$ on traces over $(\Omega, I)$ as follows. Every predicate $v_a(x)$ in $\varphi$ is replaced by $\bigvee_{0\leq n<B} v_{(a,n)}(x)$ in $\widetilde{\varphi}$. Every predicate $\mathrm{msg}(x, y)$ is replaced by

$$x \leq_I y \wedge \bigvee_{\substack{a=p!q,b=q?p\\ 0\leq n<B}} v_{(a,n)}(x) \wedge v_{(b,n)}(y) \wedge \forall z : (v_{(b,n)}(z) \wedge x \leq_I z) \rightarrow y \leq_I z \ .$$

This formula expresses that for $x$ labeled by $(p!q, n)$, the node $y$ is the smallest one labeled by $(q?p, n)$ with $x \leq_I y$. Then, for $M \in \mathbb{MSC}^B$, we have $M \models \varphi$ iff $\mathrm{tr}(M) \models \tilde{\varphi}$.

Finally, we define the formula $\widehat{\varphi}$ as the conjunction of $\widetilde{\varphi}$ with a formula expressing that the trace over $(\Omega, I)$ is associated with an $\exists$-$B$-bounded MSC. For this, it suffices to state that for each event type $a \in \Sigma$ and each node labeled by some $(a, n)$, the next node labeled by $(a, m)$ satisfies $m = n + 1 \bmod B$, and that the msg relation is a bijection between sends and receives (for the last condition we use the formula given above for $\mathrm{msg}(x, y)$).

By Thm. 2.8(3), $L = \{u \in \Gamma^* \mid t_u \models \widehat{\varphi}\}$ is a regular language. The construction of $\widehat{\varphi}$ ensures $t_u \models \widehat{\varphi}$ iff there exists $M \in \mathbb{MSC}$ with $M \models \varphi$ and $t_u = \mathrm{tr}(M)$. Hence, from Lemma 3.4, we get $L = \widetilde{\mathrm{Lin}^B}(\mathcal{L}(\varphi))$. Hence, $\mathrm{Lin}^B(\mathcal{L}(\varphi))$ is the projection of the regular language $L$ and therefore regular as well. $\mathrm{Lin}^B(\mathcal{L}(\varphi))$ is a set of representatives of $\mathcal{L}(\varphi)$ since this set is $\exists$-$B$-bounded. $\square$

## 4.2 From regular sets of representatives to CMSC-graphs

We now demonstrate the implication (5)⇒(4).

**Proposition 4.3** *Let $\mathcal{M}$ be a set of $\exists$-$B$-bounded MSCs such that* $\mathrm{Lin}^B(\mathcal{M}) \subseteq \Sigma^*$ *is regular. Then there exists a globally-cooperative CMSC-graph $G$ with $\mathcal{L}(G) = \mathcal{M}$.*

*Proof.* Since the mapping $\tilde{}$ is a rational transduction (see e.g. [3]), the set $L = \widetilde{\mathrm{Lin}^B(\mathcal{M})} \subseteq \Omega^*$ is regular as well. By Lemma 3.4, the set $L$ is $I$-closed. Thus, we can apply Ochmański's Theorem 2.8 (1) for obtaining a $D$-loop-connected automaton $\mathcal{B}$ with $[L(\mathcal{B})]_I = L$. From $\mathcal{B}$ we obtain an automaton $\mathcal{A}$ over the alphabet $\Sigma$ by replacing each label $(a, n) \in \Omega$ by $a$. Since all words in $L$ are of the form $\tilde{u}$ for some $u \in \Sigma^*$, we get $\widetilde{L(\mathcal{A})} = L(\mathcal{B}) \subseteq L$ and therefore $L(\mathcal{A}) \subseteq \mathrm{Lin}^B(\mathcal{M})$. Thus, all successful paths in $\mathcal{A}$ are labeled by $B$-bounded linearizations of MSCs from $\mathcal{M}$. Conversely, if $M \in \mathcal{M}$, then there is $u \in \mathrm{Lin}^B(M)$ with $\tilde{u} \in L(\mathcal{B})$ and therefore $u \in L(\mathcal{A})$. Thus, $L(\mathcal{A}) = \mathrm{Lin}^B(\mathcal{M})$.

Now let $\rho$ be a loop in the automaton $\mathcal{A}$ and let $A \subseteq \Sigma$ be the set of labels appearing in this loop. Then $\rho$ is also a loop in the automaton $\mathcal{B}$ with label set $A' \subseteq \Omega$. Since $\mathcal{B}$ is $D$-loop-connected, $A'$ is a connected subset of $(\Omega, D)$. Since $\mathcal{B}$ accepts only words of the form $\tilde{u}$ for some linearization $u$ of an MSC, we have that $(p!q, n) \in A'$ iff $(p?q, n) \in A'$. Hence the $D$-connectedness of $A'$ implies the connectedness of the communication graph of $\{a \mid \exists n : (a, n) \in A'\}$.

To obtain a CMSC-graph, we transform $\mathcal{A}$ in such a way that labels move from transitions to nodes. The resulting CMSC-graph $G$ is safe since $\mathcal{A}$ accepts only linearizations of MSCs, and globally-cooperative since loops in $\mathcal{A}$ are labeled by sets $A \subseteq \Sigma$ whose communication graph is connected. □

## 5 From regular representatives to CFM

In order to obtain a CFM from a regular set of representatives, we will use Zielonka's theorem, which characterizes regular trace languages by a distributed automaton model, namely by asynchronous automata[2].

**Definition 5.1** *An asynchronous automaton over the trace alphabet $(\Omega, I)$ is a tuple $\mathcal{B} = ((K_e, \delta_e, k_e^0)_{e \in \Omega}, Acc)$ such that for any $e \in \Omega$:*

- *$K_e$ is a finite set of local states,*
- *$\delta_e : \prod_{(e,f) \in D} K_f \to K_e$ is a local transition function,*
- *$k_e^0 \in K_e$ is a local initial state,*

*and $Acc \subseteq \prod_{e \in \Omega} K_e$ is a set of global accepting states.*

The idea is that an asynchronous automaton consists of local components, one for each letter $e \in \Omega$. When the $e$-component executes the action $e$, its new state results from the current states corresponding to the letters depending on $e$. Only at the very end of a run, there is a global synchronization through final states.

Next we define runs of asynchronous automata. Intuitively, a run can be seen as a labeling of the pomset by local states, that is consistent with the transition relations.

[2]The definition we give actually corresponds to deterministic asynchronous cellular automata, see [31, 9].

Let $(E, \leq, \lambda_I)$ be a trace over $(\Omega, I)$, $\theta : E \to \bigcup_{e\in\Omega} K_e$ a mapping, and $t \in E$ with $\lambda_I(t) = e$. For $f \in \Omega$ with $(e, f) \in D$, we define $\theta_f^-(t) = \theta(t_f)$ if $t_f$ is the maximal $f$-labeled event of $E$ properly below $t$. If no such event exists, $\theta_f^-(t) = k_f^0$ is the $f$-component of the initial state of $\mathcal{B}$. The mapping $\theta$ is a *run* if for any $t \in E$ with $\lambda(t) = e$, we have $\theta(t) = \delta_e((\theta_f^-(t))_{(e,f)\in D})$. Next, for $e \in \Omega$ let $k_e = \theta(t_e)$ where $t_e$ is the maximal $e$-labeled event of $E$ (if such an event exists), and $k_e = k_e^0$ otherwise. The run $\theta$ is successful provided that $(k_e)_{e\in\Omega} \in \mathrm{Acc}$ is a (global) accepting state. The set of traces $\mathcal{L}(\mathcal{B})$ accepted by $\mathcal{B}$ is the set of traces that admit a successful run. By Thm. 2.8(2), an $I$-closed set of words $L$ is regular iff there exists an asynchronous automaton $\mathcal{B}$ with $L = \mathrm{Lin}(\mathcal{L}(\mathcal{B}))$.

## 5.1 A CFM recognizing regular representatives

Since we will construct several CFMs in this and the subsequent section, we start with a general result that extracts the common part of all these constructions. Let $K$ be a finite set, $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ an MSC. Furthermore, let $\gamma : E \to K$ be a mapping and $(k_p^0)_{p\in\mathcal{P}} \in K^{\mathcal{P}}$ be a tuple of initial values. For each event $t \in E_p$, we define the values $\gamma^-(t), \gamma^m(t)$ as values associated with two events below $t$. Let $\gamma^-(t) = \gamma(s)$ if $s$ is the predecessor of $t$ on the same process, namely $p$. If no such predecessor exists, then $\gamma^-(t) = k_p^0$. Furthermore, let $\gamma^m(t) = \gamma(x)$ if either $\mathrm{msg}(x) = t$ or $\mathrm{rev}(x) = t$. If $t$ is a send of type $\lambda(t) = p!q$, but there is no $x \in E_q$ with $\mathrm{rev}(x) = t$, then let $\gamma^m(t) = k_q^0$.

Let updt be an (update) function from $\Sigma \times K^2$ to $2^K$. We say that $\gamma : E \to K$ is a *good labeling* of $M$ with respect to updt and the (initial) values $(k_p^0)_{p\in\mathcal{P}}$ if $\gamma(t) \in \mathrm{updt}(\lambda(t), \gamma^-(t), \gamma^m(t))$ for any $t \in E$.

In the following, we will need an auxiliary mapping *extract* : $\Omega_p \times K^{\Omega_p} \times \{0, \ldots, B-1\}^{\Sigma_p} \to K$ given by *extract*$(e, \mathrm{mem}, \mathrm{cnt}) = \mathrm{mem}(e)$. This mapping extracts from the local state $(e, \mathrm{mem}, \mathrm{cnt})$ of a CFM the value stored in mem for the last event of type $e$.

**Proposition 5.2** *Let* updt $: \Sigma \times K^2 \to 2^K$ *be a mapping and* $k_p^0 \in K$ *for* $p \in \mathcal{P}$. *Then there exists a CFM* $\mathcal{A}$ *with local state set* $\Omega_p \times K^{\Omega_p} \times \{0, \ldots, B-1\}^{\Sigma_p}$ *for* $p \in \mathcal{P}$ *with the following properties for any MSC* $M$*:*

*(1) if* $\rho$ *is a run of* $\mathcal{A}$ *on* $M$*, then* $\gamma = \mathit{extract} \circ \rho$ *is a good labeling of* $M$ *with respect to* updt *and* $(k_p^0)_{p\in\mathcal{P}}$.

*(2) if* $\gamma$ *is a good labeling of* $M$ *with respect to* updt *and* $(k_p^0)_{p\in\mathcal{P}}$*, then there exists a run* $\rho$ *of* $\mathcal{A}$ *with* $\gamma = \mathit{extract} \circ \rho$.

*Proof.* We construct a CFM $\mathcal{A}$ that guesses a labeling and accepts only if this labeling is good with respect to updt and $(k_p^0)_{p\in\mathcal{P}}$. The set of message contents is $K \times K$. A local state $(e, \mathrm{mem}, \mathrm{cnt}) \in S_p$ has the following meaning:

- cnt counts the number of occurrences modulo $B$ of each type in $\Sigma_p$. This allows to compute the $\Omega$-label of events.
- $e$ is the $\Omega$-symbol of the last event on process $p$,

- mem records the last $K$-value for each symbol in $\Omega_p$.

The initial state of process $p$ is $s_p^0 = (e_p, \overline{k_p^0}, \overline{0})$ for all $p$, where $\overline{x}$ is the constant function taking value $x$ for all arguments and $e_p$ is an arbitrary (but fixed) label from $\Omega_p$.

We next define the transition relations: for two states $(e, \mathrm{mem}, \mathrm{cnt})$ and $(e', \mathrm{mem}', \mathrm{cnt}')$ from $S_p$, $a \in \Sigma_p$, and $m \in K \times K$ (the message set), we have $(e, \mathrm{mem}, \mathrm{cnt}) \xrightarrow{a,m} (e', \mathrm{mem}', \mathrm{cnt}')$ if

(a) $\mathrm{cnt}'(a) = \mathrm{cnt}(a) + 1 \bmod B$, and $\mathrm{cnt}'(b) = \mathrm{cnt}(b)$ for all $b \neq a$,

(b) $e' = (a, \mathrm{cnt}'(a))$,

(c) $\mathrm{mem}'(f) = \mathrm{mem}(f)$ for all $f \neq e'$,

(d) $m = (\mathrm{val}, \mathrm{gss})$ with
   - If $a = p!q$, $\mathrm{mem}'(e') \in \mathrm{updt}(a, \mathrm{mem}(e), \mathrm{gss})$ and $\mathrm{val} = \mathrm{mem}'(e')$.
   - If $a = p?q$, $\mathrm{mem}'(e') \in \mathrm{updt}(a, \mathrm{mem}(e), \mathrm{val})$ and $\mathrm{gss} = \mathrm{mem}(e')$.

To understand this definition informally, consider an action $a \in \Sigma_p$ and let $(e, \mathrm{mem}, \mathrm{cnt})$ be the $p$-local state before process $p$ executes $a$. Then $e = (a', n) \in \Omega_p$ where $a'$ is the last action on process $p$ before $a$ and $n$ counts the number of occurrences of $a'$ before that last $p$-event (modulo $B$). Hence, $\mathrm{mem}(e)$ is the value of the guessed labeling at the last $p$-event.

If $a = p!q$ is a send action, the current event is the target of a rev-edge. Process $p$ guesses the value gss of the guessed labeling at the source of this edge. Furthermore, it guesses the value val of the guessed labeling at the current event. Since the guessed labeling shall be good, val has to belong to $\mathrm{updt}(a, \mathrm{mem}(e), \mathrm{gss})$. Then the pair $(\mathrm{val}, \mathrm{gss})$ is sent to process $q$.

If $a = p?q$ is a receive action, let $(v, \mathrm{gss})$ be the message received. Note that $v$ is the value of the guessed labeling at the source of the current msg-edge. Process $p$ chooses a value of the guessed labeling $\mathrm{val} \in K$. To ensure that this guessed labeling is good, the only restriction is $\mathrm{val} \in \mathrm{updt}(a, \mathrm{mem}(e), v)$. Since $e' = (a, \mathrm{cnt}(a) + 1 \bmod B)$, $\mathrm{mem}(e')$ is the value of the guessed labeling at the $B$th receive $p?q$ before the current one. Because of the intended meaning of gss (see above), process $p$ has to check whether $\mathrm{gss} = \mathrm{mem}(e')$. If this is not the case, the machine deadlocks.

If the machine does not deadlock, the new $p$-local state is obtained by updating the counting-information in cnt and recalling the value of the guessed labeling val in the memory cell $\mathrm{mem}(e)$.

Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ be an MSC and let $\gamma : E \to K$ be a good labeling with respect to updt and $(k_p^0)_{p \in \mathcal{P}}$. We define a run $\rho$ of $\mathcal{A}$ as follows: for $t \in E_p$, let $\rho(t) = (e, \mathrm{mem}, \mathrm{cnt})$ with

- $\mathrm{cnt}(a) = |\{s \in E \mid \lambda(s) = a,\ s \leq_p t\}| \bmod B$ for $a \in \Sigma_p$.
- For $(a, n) \in \Omega_p$, $\mathrm{mem}((a, n)) = \gamma(s)$ if $s \in E_p$ is maximal with $s \leq_p t$ and $\lambda_I(s) = (a, n)$. If no such $s$ exists, $\mathrm{mem}((a, n)) = k_p^0$.
- $e = \lambda_I(t)$.

Then it is not hard to check that $\rho$ is a run of $\mathcal{A}$ on $M$. This shows the second statement.

Now let $\rho$ be a run of $\mathcal{A}$ on the MSC $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$. Let $s, r \in E$ be two nodes with $\mathrm{msg}(s) = r$ and $\lambda(s) = p!q$. We write $\rho(s) = (e'_s, \mathrm{mem}'_s, \mathrm{cnt}'_s)$, $\rho^-(s) = (e_s, \mathrm{mem}_s, \mathrm{cnt}_s)$, $\rho(r) = (e'_r, \mathrm{mem}'_r, \mathrm{cnt}'_r)$, and $\rho^-(r) = (e_r, \mathrm{mem}_r, \mathrm{cnt}_r)$. Recall that $\rho^-(t)$ is the local $p$-state just before executing event $t \in E_p$. Furthermore, let $t \in E$ with $\mathrm{rev}(t) = s$ and $\rho(t) = (e_t, \mathrm{mem}_t, \mathrm{cnt}_t)$. If no such $t$ exists, define $(e_t, \mathrm{mem}_t, \mathrm{cnt}_t) = (e_q, \overline{k^0_q}, \overline{0})$. Since $\rho$ is a run, there are $\mathrm{val}, \mathrm{gss} \in K$ such that

(1) $(e_s, \mathrm{mem}_s, \mathrm{cnt}_s) \overset{\lambda(s),(\mathrm{val},\mathrm{gss})}{\longrightarrow} (e'_s, \mathrm{mem}'_s, \mathrm{cnt}'_s)$ and

(2) $(e_r, \mathrm{mem}_r, \mathrm{cnt}_r) \overset{\lambda(r),(\mathrm{val},\mathrm{gss})}{\longrightarrow} (e'_r, \mathrm{mem}'_r, \mathrm{cnt}'_r)$.

Then we have $\mathrm{gss} = \mathrm{mem}_r(e'_r)$ from (2). The definition of rev implies $\mathrm{mem}_r(e'_r) = \mathrm{mem}_t(e_t) = \gamma^m(s)$. Hence $\gamma(s) = \mathrm{mem}'_s(e'_s) \in \mathrm{updt}(\lambda(s), \mathrm{mem}_s(e_s), \mathrm{gss}) = \mathrm{updt}(\lambda(s), \gamma^-(s), \gamma^m(s))$ by (1) ensuring that $\gamma$ is good at node $s$.

Furthermore, $\mathrm{val} = \mathrm{mem}'_s(e'_s) = \gamma(s) = \gamma^m(r)$ by (1). By (2), we get $\gamma(r) = \mathrm{mem}'_r(e'_r) \in \mathrm{updt}(\lambda(r), \mathrm{mem}_r(e_r), \mathrm{val}) = \mathrm{updt}(\lambda(r), \gamma^-(r), \gamma^m(r))$. Thus, $\gamma$ is also good at $r$. Since any node of $E$ is either a send or a receive, we showed that $\gamma$ is good w.r.t. updt and $(k^0_p)_{p\in\mathcal{P}}$. □

Now let $\mathcal{M} \subseteq \mathbb{MSC}^B$ be an existentially bounded set of MSCs such that $\mathrm{Lin}^B(\mathcal{M})$ is regular. We will construct a CFM $\mathcal{A}$ that accepts an $\exists$-$B$-bounded MSC iff it belongs to $\mathcal{M}$ (the behavior on MSCs that are not $\exists$-$B$-bounded is of no concern here and will be dealt with in the subsequent section).

The general line of argument is as follows. First, it is shown that the set of traces $\mathrm{tr}(\mathcal{M} \cap \mathbb{MSC}^B)$ can be accepted by an asynchronous automaton $\mathcal{B}$. From this asynchronous automaton, we will construct a function updt and a tuple of initial values such that good labelings on $M \in \mathbb{MSC}^B$ correspond to runs of the asynchronous automaton $\mathcal{B}$ on $\mathrm{tr}(M)$. Thus, from the previous proposition, we will get a CFM $\mathcal{A}$ whose runs on $M$ correspond to runs of the asynchronous automaton on $\mathrm{tr}(M)$.

**Proposition 5.3** *Let $B \in \mathbb{N}$ and $\mathcal{M}$ a set of $\exists$-$B$-bounded MSCs with $\mathrm{Lin}^B(\mathcal{M})$ regular. Then there exists a CFM $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) \cap \mathbb{MSC}^B = \mathcal{M}$.*

*Proof.* Since the mapping ˜ is a rational transduction, the set $L = \widetilde{\mathrm{Lin}^B(\mathcal{M})} \subseteq \Omega^*$ is regular as well. By Lemma 3.4, the set $L$ is $I$-closed. Hence there exists an asynchronous automaton $\mathcal{B} = ((K_e, \delta_e, k^0_e)_{e\in\Omega}, \mathrm{Acc})$ accepting $[L]_I$ by Theorem 2.8(2).

We now associate with $\mathcal{B}$ a mapping updt that mimics the local transition functions of the asynchronous automaton. The set of values is $K = \bigcup_{p\in\mathcal{P}} K_p$ with $K_p = \prod_{f\in\Omega_p} K_f$. Thus, each $K_p$ describes the local states of events on process $p$. Let $a = p\theta q$ for $\theta \in \{!, ?\}$, $k_p, k'_p \in K_p$ and $k_q \in K_q$. Then $\mathrm{updt}(a, k_p, k_q) \subseteq K_p$. We define $k'_p \in \mathrm{updt}(a, k_p, k_q)$ iff there exists $0 \le n < B$ such that

1. $k'_p[(a, n)] = \delta_e((k_p[f])_{f\in\Omega_p}, k_q[(a', n)])$ (where $a'$ is the event matching $a = p\theta q$), and
2. $k'_p[f] = k_p[f]$ for $f \in \Omega_p \setminus \{(a, n)\}$.

Furthermore, $\mathrm{updt}(a, k_1, k_2) = \emptyset$ if $a = p\theta q$, but $k_1 \notin K_p$ or $k_2 \notin K_q$.

Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}}) \in \mathbb{MSC}^B$ and $\gamma : E \to K$ a mapping. Let furthermore $p \in \mathcal{P}$ and $e \in \Omega_p$ and define $k_e = \gamma(t)[e]$ if $t$ is the maximal element of $E_p$. If $E_p$ is empty, then set $k_e = k_p^0[e]$. We say that $\gamma$ is accepting if the tuple $(k_e)_{e\in\Omega}$ belongs to Acc, i.e., is accepting in the asynchronous automaton $\mathcal{B}$. We show that $\mathrm{tr}(M)$ is accepted by $\mathcal{B}$ iff there exists a good labeling $\gamma$ that is accepting.

First, let $\gamma$ be an accepting good labeling. For $t \in E$, let $\theta(t) = \gamma(t)[\lambda_I(t)]$. Since updt mimics the transitions of $\mathcal{B}$, the mapping $\theta$ is a run of the asynchronous automaton $\mathcal{B}$ on $tr(M)$. Using (2) in the definition of the function updt, we obtain that the final global state of this run $\theta$ is precisely $(k_e)_{e\in\Omega}$ as defined above. Hence $\theta$ is accepting, i.e., $\mathrm{tr}(M)$ is accepted by $\mathcal{B}$.

Conversely, assume $\theta$ is an accepting run of $\mathcal{B}$ on $\mathrm{tr}(M)$. We define a labeling $\gamma$ from $\theta$. For an event $t$ on $E_p$ and $f \in \Omega_p$, let $\gamma(t)[f] = \theta(u)$, where $u$ is the last event of $tr(M)$ before $t$ that has type $f$.

Since updt mimics the transitions of $\mathcal{B}$, the mapping $\gamma$ is a good labeling of $M$ w.r.t. updt and the initial states of $\mathcal{B}$. We can define easily a run $\rho$ of the CFM on $M$ such that for any event $t$ with $\rho(t) = (e, \mathrm{mem}, \mathrm{cnt})$, we have $\gamma(t) = \mathrm{mem}(e)$. Moreover, $(\gamma(t_p)[f])_{f\in\Omega_p} = (\theta(t_f))_{f\in\Omega_p}$, where $t_f$ is the last event of type $f$ in $tr(M)$, that is, $\rho$ is an accepting run of the CFM.

To conclude, let $\mathcal{A}$ be the CFM from Prop. 5.2. Its accepting states can be changed such that a run is accepting iff the associated good labeling is accepting. Hence, $\mathcal{A}$ accepts $M \in \mathbb{MSC}^B$ iff $\mathrm{tr}(M)$ is accepted by $\mathcal{B}$ iff $M \in \mathcal{M}$. Thus, $\mathcal{L}(\mathcal{A}) \cap \mathbb{MSC}^B = \mathcal{M}$. □

Thus, we managed to construct a CFM that checks membership in $\mathcal{M} \subseteq \mathbb{MSC}^B$ provided that the input MSC is $\exists$-$B$-bounded. The following two sections explain how to build a CFM $\mathcal{A}'$ which accepts precisely $\mathbb{MSC}^B$. Taking the direct product of these two machines will show that $\mathcal{M}$ can be accepted by a CFM whenever $\mathrm{Lin}^B(\mathcal{M})$ is a regular set of representatives.

## 5.2 Characterizing $\exists$-$B$-bounded MSCs

Lemma 2.2 provides a characterization for $\exists$-$B$-bounded MSCs via the relation $\prec_B = \mathrm{msg} \cup \bigcup_{p\in\mathcal{P}} <_p \cup \mathrm{rev}$. Although this characterization was very useful to establish the relation between $\exists$-$B$-bounded MSCs and traces, it is global and it does not provide directly a way to check $\exists$-$B$-boundedness using finite and distributed memory. This section refines the characterization such that a CFM can test it.

First, we replace the order relations $<_p$ on processes by type functions $\delta_a$ ($a \in \Sigma$). For an MSC $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ and a type $a \in \Sigma$ we define the partial function $\delta_a : E \to E$ by $\delta_a(e) = f$ iff for some $p \in \mathcal{P}$ we have $e <_p f$, $\lambda(f) = a$ and for every $g$ with $e <_p g <_p f$, $\lambda(g) \neq a$. That is, $\delta_a(e)$ is the first event after $e$ on the same process of type $a \in \Sigma$. Let $\mathcal{R} = \mathrm{msg} \cup \bigcup_{a\in\Sigma} \delta_a \cup \mathrm{rev}$. Since $(\bigcup_{a\in\Sigma_p} \delta_a)^* =<_p$, the MSC $M$ is $\exists$-$B$-bounded iff the relation $\mathcal{R}$ is acyclic. Note that $<_p$ has unbounded outdegree whereas $\delta_a$ has unbounded indegree but outdegree at most one.

Besides the partial functions $\delta_a$, we also use a partial function $\delta_\sharp = \mathrm{msg} \cup \mathrm{rev}$ (note that this is well-defined since the domains of these two functions are disjoint).

By induction, we define $\delta_{\sigma\sigma'} = \delta_{\sigma'} \circ \delta_\sigma$ for $\sigma, \sigma' \in (\Sigma \cup \{\sharp\})^*$. For the empty word, we set $\delta_\varepsilon(e) = e$ for all $e$.

**Lemma 5.4** *Let $\tau \in (\Sigma \cup \{\sharp\})^*$. Let furthermore $M$ be an MSC with $e, f \in E$ such that $\lambda(e) = \lambda(f)$.*

*(1) Assume that $\delta_\tau(e)$ and $\delta_\tau(f)$ are defined. Then they have the same type, i.e., $\lambda(\delta_\tau(e)) = \lambda(\delta_\tau(f))$.*

*(2) Let $f \leq e$ be such that $\delta_\tau(e)$ is defined. Then $\delta_\tau(f)$ is defined too, and $\delta_\tau(f) \leq \delta_\tau(e)$.*

*Proof.* We first prove (1) by induction on the length of $\tau$. The base case $\tau = \varepsilon$ is trivial. Now let $\tau = a\sigma$ with $a \in \Sigma \cup \{\sharp\}$.

If $a = \sharp$ and $\lambda(e) = p!q$ is a send event, then $\delta_a(e) = \mathrm{msg}(e)$ implying $\lambda(\delta_a(e)) = q?p$. Since $\lambda(f) = p!q$, we similarly get $\lambda(\delta_a(f)) = q?p$.

Now suppose $a = \sharp$ and $\lambda(e) = p?q$ is a receive event. Then $\delta_a(e) = \mathrm{rev}(e)$ implying, as above, $\lambda(\delta_a(e)) = \lambda(\delta_a(f))$.

If $a \in \Sigma$, then $\lambda(\delta_a(e)) = a = \lambda(\delta_a(f))$.

Now we can apply the induction hypothesis to $\sigma$, since $\delta_\sigma$ is defined on $\delta_a(e)$ and on $\delta_a(f)$ and these two nodes carry the same label. This finishes the proof of (1).

The second statement is shown similarly by induction on the length of $\tau$ where, again, the base case $\tau = \varepsilon$ is trivial. As before, let $\tau = a\sigma$ with $a \in \Sigma \cup \{\sharp\}$.

First suppose $a = \sharp$ and $\lambda(e) = p!q = \lambda(f)$. Since $M$ is an MSC, there exists $r \in E$ with $\mathrm{msg}(e) = r$. Hence $\delta_\sharp(f) \leq r$ is defined.

Now suppose $a = \sharp$ and $\lambda(e) = p?q = \lambda(f)$. Since $s = \delta_\sharp(e) = \mathrm{rev}(e)$, the node $s$ is the send event associated with the receive event number $B$ after $e$. Since $f \leq e$ is also a receive event of the same type, there exist $B - 1$ receive events after $f$, and because of FIFO, $\delta_\sharp(f) = \mathrm{rev}(f) \leq \mathrm{rev}(e) = \delta_\sharp(e)$.

If $a \in \Sigma$, since $\delta_a(e)$ is defined, there is an event of type $a$ after $e$. Since $f \leq e$, there is also an event of type $a$ after $f$, and $\delta_a(f) \leq \delta_a(e)$.

We can apply the induction hypothesis to $\tau'$, since $\delta_\sigma$ is defined on $\delta_a(e)$, $\lambda(\delta_\sigma(f)) = \lambda(\delta_\sigma(e))$ and $\delta_a(f) \leq \delta_a(e)$. □

Now let $e \in E$ be an event of type $\lambda(e) = x_0$ and $\tau \in (\Sigma \cup \{\sharp\})^*$. We say that $e$ defines a *$(x_0, \tau)$-cycle of length $|\tau|$* if the event $\delta_\tau(e)$ exists and satisfies $\delta_\tau(e) \leq_p e$ for some $p \in \mathcal{P}$. That is, the start and endpoint of the cycle are on the same process $p$, with the endpoint preceding the starting point. Clearly, this implies a cycle for the relation $\mathcal{R}$.

The next lemma shows that it suffices to test $(x_0, \tau)$-cycles of bounded length for knowing whether an MSC $M$ is $\exists$-$B$-bounded or not.

**Lemma 5.5** *Let $M$ be an MSC. Then $M$ is not $\exists$-$B$-bounded iff there exists an event $e$ defining a $(x_0, \tau)$-cycle, for some $x_0 \in \Sigma$ and $\tau \in \sharp(\Sigma\,\sharp)^*$ with $|\tau| \leq 2|\mathcal{P}|$.*

*Proof.* If there exists such an $(x_0, \tau)$-cycle, then the relation $\mathcal{R}$ is not acyclic implying that $M$ is not $\exists$-$B$-bounded.

Conversely, suppose that $M$ is not $\exists$-$B$-bounded, hence the relation $\mathcal{R}$ contains some cycle. We choose $\sigma = a_1 a_2 \cdots a_n \in (\Sigma \cup \{\sharp\})^*$ of minimal length such that there exists $x_0 \in \Sigma$ and an event $e_1$ defining a $(x_0, \sigma)$-cycle. Set $e_i = \delta_{a_1 a_2 \cdots a_{i-1}}(e_1)$ and suppose $n > 2|\mathcal{P}|$. Then there exists some process $p \in \mathcal{P}$ and $i < j < k$ with $e_i, e_j, e_k \in E_p$. If $e_k \leq_p e_i$, then $\delta_{a_{i+1} a_{i+2} \cdots a_k}(e_i) = e_k \leq_p e_i$, i.e., $e_i$ defines a $(\lambda(e_i), a_{i+1} \cdots a_k)$-cycle properly shorter than $\sigma$, a contradiction. Hence $e_i <_p e_k$. Consider the sequence $\tau = a_1 a_2 \cdots a_i\, \lambda(e_k) a_{k+1} a_{k+1} \cdots a_n$ of length $i+1+(n-k) < n$ since $i+1 < k$. Then $\delta_{\lambda(e_k)}(e_i) \leq_p e_k$ implies $\delta_\tau(e_1) = \delta_{\lambda(e_k) a_{j+1} \cdots a_n}(e_i) \leq \delta_{a_{j+1} \cdots a_n} \leq e_1$ by Lemma 5.4, again giving rise to a shorter cycle. Thus, we showed $n \leq 2|\mathcal{P}|$.

It remains to show that $\tau \in \sharp(\Sigma\,\sharp)^*$. Assume first that $a_i, a_{i+1} \in \Sigma$. Since $\delta_\sigma(e_1)$ is defined, in particular $\delta_{a_i a_{i+1}}(e_i)$ is defined. Hence $e_i$, $e_{i+1}$, and $e_{i+2}$ belong to the same process which allows to derive a contradiction as above. Second, let $a_i = a_{i+1} = \sharp$. Then $e_i \leq_p e_{i+2}$ allows to argue as above, i.e., to replace $\sigma$ by $\tau = a_1 a_2 \cdots a_i\, \lambda(e_{i+2}) a_{i+3} \cdots a_n$, again contradicting the minimal length of $\sigma$. Thus, letters from $\Sigma$ alternate with $\sharp$ in $\sigma$.

If $a_1 \in \Sigma_p$, then $\delta_\sigma(e_1) \leq_p e_1 <_p \delta_{a_1}(e_1)$. Hence $\delta_{a_1}(e_1)$ defines a $(a_1, a_2 a_3 \cdots a_n)$-cycle. This allows to cancel all types from the beginning of $\sigma$ thereby leaving a word from $\sharp(\Sigma\,\sharp)^*$. Similarly, we can cancel all letters from $\Sigma$ from the end which finally shows $\sigma \in \sharp(\Sigma\,\sharp)^*$ as required. $\square$

Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ be an MSC, $e \in E$, $x_0 \in \Sigma$, and $\tau \in (\Sigma \cup \{\sharp\})^*$ be a word. We call event $e$ a *$(x_0, \tau)$-marker*, if it satisfies the following conditions:

- $\lambda(e) = x_0$,
- $\lambda_\tau(e)$ is defined, and
- for every event $f \in E$ with $\lambda(f) = x_0$, $e <_p f$, and such that $\delta_\tau(f)$ is defined, we have $\delta_\tau(e) \neq \delta_\tau(f)$.

In particular, $(x_0, \tau)$-markers are mapped to mutually distinct nodes $\delta_\tau(e)$. Moreover, if for some node $e \in E$, $\delta_\tau(e)$ is defined, then $e$ is below some $(\lambda(e), \tau)$-marker $f$ with $\delta_\tau(e) = \delta_\tau(f)$. Let $\mathrm{Mark}(x_0, \tau) \subseteq E$ denote the set of $(x_0, \tau)$-markers in $M$. The image of the mapping $\delta_\tau$ (restricted to $x_0$-labeled nodes) is denoted $\mathrm{CoMark}(x_0, \tau)$. Nodes from $\mathrm{CoMark}(x_0, \tau)$ are called *comarkers*. Note that $\delta_\tau$ is an order-preserving bijection from $\mathrm{Mark}(x_0, \tau)$ onto $\mathrm{CoMark}(x_0, \tau)$. In particular, the number of markers equals the number of comarkers. The next lemma characterizes MSCs that are *not* existentially $B$-bounded through the number of markers $e$ that are still in transit, in the sense that $\delta_\tau(e)$ did not yet happen on process $p$.

We write in the following $\mathrm{past}(e)$ for the set $\{f \in E \mid f \leq e\}$ of events below $e$.

**Proposition 5.6** *Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ be an MSC. Then the following are equivalent*

*(1) $M$ is not $\exists$-$B$-bounded*

*(2) There exist a sequence $\tau \in \sharp(\Sigma\,\sharp)^*$ with $|\tau| \leq 2|\mathcal{P}|$, $p \in \mathcal{P}$, $x_0 \in \Sigma_p$, and a marker $e \in \mathrm{Mark}(x_0, \tau)$ such that*

*(2a) $\lambda(e) = x_0$ and $\delta_\tau(e) \in E_p$, and*

*(2b) The difference* $|past(e) \cap \mathrm{Mark}(x_0, \tau)| - |past(e) \cap \mathrm{CoMark}(x_0, \tau)|$ *is either* $\leq 0$ *or* $> 2|\mathcal{P}|B$.

*Proof.* First suppose (1). Then, by Lemma 5.5, there exist $x_0 \in \Sigma$, $\tau \in (\Sigma \cup \{\sharp\})^*$ with $|\tau| \leq 2|\mathcal{P}|$, and $e \in E$ such that $\delta_\tau(e) \leq_p e$ for some $p \in \mathcal{P}$. Hence, in particular (2a) holds. Since, by Lemma 5.4, the set $\mathrm{Mark}(x_0, \tau)$ is mapped order-preservingly onto $\mathrm{CoMark}(x_0, \tau)$, we get $|\mathrm{past}(e) \cap \mathrm{Mark}(x_0, \tau)| - |\mathrm{past}(e) \cap \mathrm{CoMark}(x_0, \tau)| \leq 0$ proving (2b).

Conversely, suppose (2) holds, i.e., there are $\tau$, $p$, $x_0$, and $e$ satisfying (2a) and (2b). If, in (2b), we have $|\mathrm{past}(e) \cap \mathrm{Mark}(x_0, \tau)| - |\mathrm{past}(e) \cap \mathrm{CoMark}(x_0, \tau)| \leq 0$, then $\delta_\tau(e) \leq_p e$. Hence $e$ is an $(x_0, \tau)$-cycle implying (1) by Lemma 5.5. So suppose $|\mathrm{past}(e) \cap \mathrm{Mark}(x_0, \tau)| - |\mathrm{past}(e) \cap \mathrm{CoMark}(x_0, \tau)| > 2|\mathcal{P}|B$. Then there exist $(x_0, \tau)$-markers $e_0 < e_1 < e_2 \cdots < e_{2|\mathcal{P}|B} \leq e$ with $\delta_\tau(e_i) \not\leq e$. By (2a) and Lemma 5.4, this implies $e < \delta_\tau(e_0) < \delta_\tau(e_1) < \cdots < \delta_\tau(e_{2|\mathcal{P}|B})$.

Let $\tau = a_1 a_2 \cdots a_k$ and define $\tau_i = a_1 \cdots a_i$ for $0 \leq i \leq k$. We show that there exist sequences $\sigma_i \in (\Sigma \cup \{\sharp\})^*$ such that $\delta_{\sigma_i}(\delta_{\tau_i}(e_0)) = \delta_{\tau_{i-1}}(e_{(i+1)B})$ for all $1 \leq i \leq k$.

Since the events $\delta_\tau(e_j)$ form a properly increasing sequence, we get $\delta_{\tau_{i-1}}(e_{iB}) <_q \delta_{\tau_{i-1}}(e_{iB+1}) <_q \cdots <_q \delta_{\tau_{i-1}}(e_{iB+B})$ for some process $q \in \mathcal{P}$ and all these events are of the same type $c$ (i.e., carry the same label w.r.t. $\lambda$) by Lemma 5.4. Now let $a = a_i$. By case distinction, we choose $\sigma_i \in (\Sigma \cup \{\sharp\})^*$ such that $\delta_{\sigma_i}(\delta_{\tau_i}(e_{iB})) = \delta_{\tau_{i-1}}(e_{iB+B})$:

- If $a = \sharp$, then set $b = \sharp$. Again, we have to distinguish two cases depending on whether the type $c$ is a send or a receive event.
  - First, let $c$ be a send event. Hence, properly above $\delta_{\tau_{i-1}}(e_{iB})$ and below $\delta_{\tau_{i-1}}(e_{iB+B})$, there are at least $B$ send events of the same type $b$. Thus, $cp(\delta_{\tau_{i-1}}(e_{iB})) = \delta_b(\delta_{\tau_{i-1}}(e_{iB})) \leq \delta_{\tau_{i-1}}(e_{iB+B})$. To obtain $\sigma_i$, extend $b$ by the appropriate number of $\lambda(\delta_{\tau_{i-1}}(e_{iB+B}))$.
  - Now suppose $c$ is a receive event. Hence, properly above $\delta_{\tau_{i-1}}(e_{iB})$ and below $\delta_{\tau_{i-1}}(e_{iB+B})$, there are at least $B$ receive events of the same type. Thus, $\mathrm{rev}(\delta_{\tau_{i-1}}(e_{iB})) = \delta_b(\delta_{\tau_{i-1}}(e_{iB})) \leq \delta_{\tau_{i-1}}(e_{iB+B})$. To obtain $\sigma_i$, extend $b$ by the appropriate number of $\lambda(\delta_{\tau_{i-1}}(e_{iB+B}))$.
- Now suppose $a \in \Sigma_p$. Since $\delta_{\tau_{i-1}}(e_{iB}) <_p \delta_{\tau_i}(e_{iB}) <_p \delta_{\tau_i}(e_{iB+B})$, we get $\delta_{\tau_i}(e_{iB}) \leq_p \delta_{\tau_{i-1}}(e_{iB+B})$. This time, $\sigma_i$ consists of the appropriate number of $\lambda(\delta_{\tau_{i-1}}(e_{iB+B}))$.

Thus, we have

$$\begin{array}{ll} \delta_{\sigma_1}\delta_{\sigma_2}\cdots\delta_{\sigma_{k-1}}\delta_{\sigma_k}\delta_{\tau_k}(e_0) & = \\ \delta_{\sigma_1}\delta_{\sigma_2}\cdots\delta_{\sigma_{k-1}}\delta_{\tau_{k-1}}(e_B) & = \\ \delta_{\sigma_1}\delta_{\sigma_2}\cdots\delta_{\tau_{k-2}}(e_{2B}) & = \\ \vdots & \\ \delta_{\tau_{k-k}}(e_{kB}) & = \quad e_{kB} <_p \delta_{\tau_k}(e_0) \ . \end{array}$$

Hence, $\delta_{\tau_k}(e_0)$ is a $(\delta_{\tau_k}(e_0), \sigma)$-cycle with $\sigma = \sigma_k \sigma_{k-1} \ldots \sigma_1$ implying that the relation $\mathcal{R}$ is not acyclic. Thus, we proved that $M$ is not $\exists$-$B$-bounded. $\square$

## 5.3 A CFM recognizing $\mathbb{MSC}^B$

In this section we show how to construct a CFM checking the non-existence of cycles in $(E, \mathcal{R})$ using the characterization by bounded cycles provided in Section 5.2. We will exclude these $(x_0, \tau)$-cycles one by one. So let us fix some type $x_0 \in \Sigma_p$ and a sequence $\tau \in \sharp(\Sigma\sharp)^*$. Consider some MSC $M$. If there is some node $e \in E$ of type $\lambda(e) = x_0$ and such that $\delta_\tau(e)$ exists, but is not in $E_p$, then by Lemma 5.4(1), no MSC will ever admit a $(x_0, \tau)$-cycle. Thus, it suffices to consider a pair $(x_0, \tau)$ where $\delta_\tau(e) \in E_p$ whenever defined and $\lambda(e) = x_0 \in \Sigma_p$. We will construct a CFM that accepts an MSC $M$ iff $M$ does not contain any $(x_0, \tau)$-cycle.

Let $\tau = a_1 a_2 \cdots a_k$ and $\tau_i = a_1 \cdots a_i$ for all $i \leq k$. A *weak* $(x_0, \tau)$*-marker* is a node $s$ such that for some $i \leq k$, node $s$ is an $(x_0, \tau_i)$-marker and either $i = k$ or $\delta_{\tau_{i+1}}(s)$ is undefined. Note that in particular, any marker is a weak marker (set $i = k$).

For two events $s$ and $t$ and a number $0 \leq i \leq k$, we write $R(s, t, i)$ if one of the following holds

$R_1(s,t,i)$: $\delta_{\tau_i}(s) = t$ or

$R_2(s,t,i)$: $i < k$, $a_{i+1} \in \Sigma_q$, $\delta_{\tau_i}(s) <_q t$ and, if $\delta_{\tau_{i+1}}(s)$ is defined, then $t <_q \delta_{\tau_{i+1}}(s)$.

Intuitively, we have $R(s, t, i)$ if and only if either $\delta_{\tau_i}(s) = t$, or else the event $\delta_{\tau_i}(s)$ has been already located on process $q$ before event $t$, and the next event $\delta_{\tau_{i+1}}(s)$ is expected after $t$ on $q$.

We let $K = \{0, \heartsuit, \dagger\}^{\{0,1,\ldots,k\}}$ be a set of functions. For an MSC $M$, a mapping $\gamma : E \to K$ is a *valid marking* if the following hold for any $t \in E$ and $0 \leq i < k$:

(V1) $\gamma(t)(i) \in \{\heartsuit, \dagger\}$ iff there is $s \in E$ satisfying $\lambda(s) = x_0$ and $R(s, t, i)$.

(V2) $\gamma(t)(i) = \heartsuit$ iff there is $s \in E$ satisfying $\lambda(s) = x_0$, $\gamma(s)(0) = \heartsuit$ and $R(s, t, i)$.

(V3) If $\gamma(t)(0) = \heartsuit$, then $t$ is a weak $(x_0, \tau)$-marker.

(V4) If $t$ is a $(x_0, \tau)$-marker, then $\gamma(t)(0) = \heartsuit$.

Valid mappings will encode a search for all markers. In order to do this, the nodes $\delta_{\tau_i}(s)$ for nodes $s$ with $\lambda(s) = x_0$ have to be found. Thus, the intuitive meaning of $\gamma(t)(i) \in \{\heartsuit, \dagger\}$ is that the node $\delta_{\tau_i}(s)$ has been found and we are searching for the node $\delta_{\tau_{i+1}}(s)$. Moreover, $\gamma(t)(i) = \heartsuit$ means that search started in a node $s$ which was a (weak) marker.

In several steps, we will construct an update-function whose good labelings are precisely the valid markings. This, in conjunction with Prop. 5.6, will then be used to check for the non-existence of $(x_0, \tau)$-cycles.

**Lemma 5.7** *Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ be an MSC and $\gamma : E \to K$ a mapping. Then $\gamma$ satisfies (V1) for all $t \in E$ and $0 \leq i \leq k$ iff for any $t \in E$ and $0 \leq i < k$, the following hold*

*(L1)* $\gamma(t)(0) \in \{\heartsuit, \dagger\}$ *iff* $\lambda(t) = x_0$

*(L2)* $\gamma(t)(i+1) \in \{\heartsuit, \dagger\}$ *iff*

- $a_{i+1} \in \Sigma$, $\lambda(t) = a_{i+1}$ *and* $\gamma^-(t)(i) \in \{\heartsuit, \dagger\}$, *or*
- $a_{i+1} = \sharp$ *and* $\gamma^m(t)(i) \in \{\heartsuit, \dagger\}$, *or*

- $\lambda(t) \neq a_{i+2} \in \Sigma$ *and* $\gamma^-(t)(i+1) \in \{\heartsuit, \dagger\}$.

*Proof.* Informally, the first two cases in condition (L2) correspond to $R_1(s,t,i+1)$, whereas the third case corresponds to $R_2(s,t,i+1)$.

The lemma is shown by induction on $i$.

Base case $i=0$. Since $a_1 = \sharp$, $R(s,t,0)$ holds iff $\delta_{\tau_0}(s) = t$ and $\lambda(s) = x_0$. Since $\delta_{\tau_0}(s) = s$, we showed the equivalence of (L1) and (V1) for $i=0$.

Next suppose that the lemma has been shown for all $t \in E$ and for some $i$ with $0 \leq i < k$. We have to prove the equivalence of (V1) for $i+1$ and (L2).

So suppose (V1) holds for $i+1$ and let $\gamma(t)(i+1) \neq 0$. Then, by (V1), there exists $s \in E$ with $\lambda(s) = x_0$ and $R(s,t,i+1)$. The first two alternatives of (L2) arise from $R_1(s,t,i+1)$, i.e., $\delta_{\tau_{i+1}}(s) = t$:

- First suppose $a_{i+1} \in \Sigma_p$. Then $i+1 < k$ and $a_{i+2} = \sharp$. Since $\delta_{\tau_{i+1}}(s) = t$, we get $\lambda(t) = a_{i+1}$. Furthermore, there exists $e \in E$ with $\delta_{\tau_i}(s) \leq_p e \lessdot_p \delta_{\tau_{i+1}}(s) = t$. Hence, by (V1), $\gamma(e)(i) \neq 0$ and therefore $\gamma^-(t)(i) \neq 0$.
- Next suppose $a_{i+1} = \sharp$. Setting $e = \delta_{\tau_i}(s)$ yields $\delta_\sharp(e) = t$ and therefore $\gamma^m(t)(i) = \gamma(e)(i) \neq 0$ by (V1).

Now suppose $R_2(s,t,i+1)$ with $a_{i+2} \in \Sigma_q$. Then $\delta_{\tau_{i+1}}(s) <_q t$. Hence there exists $e \in E$ with with $e \lessdot_p t$. Then $R(s,e,i+1)$ holds implying $\gamma(e)(i+1) \neq 0$ from (V1) and therefore $\gamma^-(t)(i+1) \neq 0$. This finishes the proof that (L2) follows from (V1).

It remains to infer (V1) for $i+1$ from (L2). To this aim, let (by contradiction) $t \in E$ be minimal (wrt. the visual order $\leq$) such that $\gamma(t)(i+1) \neq 0$, but there is no $s \in E$ with $\lambda(s) = x_0$ and $R(s,t,i+1)$.

- If $\lambda(t) = a_{i+1} \in \Sigma$ and $\gamma^-(t)(i) \neq 0$, let $e \in E$ with $e \lessdot_q t$. Then $\gamma(e)(i) \neq 0$ which, by the induction hypothesis, implies the existence of $s \in E$ with $\lambda(s) = x_0$ and $R(s,e,i)$. Since $t$ is the successor of $e$ on process $q$, we get $t \leq_q \delta_{\tau_{i+1}}(s)$. Now $\lambda(t) = a_{i+1}$ implies $t = \delta_{\tau_{i+1}}(s)$. Hence $R_1(s,t,i+1)$ contradicts our assumption.
- If $a_{i+1} = \sharp$ and $\gamma^m(t)(i) \neq 0$, then let $e \in E$ with $\delta_\sharp(e) = t$. Then we get $\gamma(e)(i) \neq 0$. Thus, by the induction hypothesis and $a_{i+1} = \sharp$, there exists $s \in E$ with $\lambda(s) = x_0$ and $R_1(s,e,i)$, i.e., $\delta_{\tau_i}(s) = e$. Now $\delta_\sharp(e) = t$ implies $\delta_{\tau_{i+1}}(s) = t$, again contradicting our assumption.
- Finally consider the case $\lambda(t) \neq a_{i+2} \in \Sigma$ and $\gamma^-(t)(i+1) \neq 0$. Let $e \in E$ with $e \lessdot_p t$. Then $\gamma(e)(i+1) = \gamma^-(t)(i+1) \neq 0$. Since we chose $t$ minimal, there exists $s \in E$ with $\lambda(s) = x_0$ and $R(s,e,i+1)$. Since $\lambda(t) \neq a_{i+2}$, we obtain $R(s,t,i+1)$ which again contradicts our assumption.

This finishes the indirect proof. Thus, (V1) for $i+1$ follows from (L2). ☐

In a similar way, the following lemma can be shown:

**Lemma 5.8** *Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ be an MSC and $\gamma : E \to K$ a mapping satisfying (V1) (i.e., (L1) and (L2)). Then $\gamma$ satisfies (V2) iff, for any $t \in E$ and $0 \leq i < k$, we have*

*(L3)* $\gamma(t)(i+1) = \heartsuit$ *iff*

- $a_{i+1} \in \Sigma$, $\lambda(t) = a_{i+1}$ *and* $\gamma^-(t)(i) = \heartsuit$, *or*
- $a_{i+1} = \sharp$ *and* $\gamma^m(t)(i) = \heartsuit$, *or*
- $\lambda(t) \neq a_{i+2} \in \Sigma$ *and* $\gamma^-(t)(i+1) = \heartsuit$.

Informally, condition (L4) in the next lemma states that if $s$ is a (weak) marker, then for every $i$, whenever node $\delta_{\tau_i}(s)$ has already been found and $\delta_{\tau_{i+1}}(s)$ is searched later on the same process, no other event $\delta_{\tau_i}(s')$ can occur on this process *before* $\delta_{\tau_{i+1}}(s)$ is eventually found.

**Lemma 5.9** *Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p \in \mathcal{P}})$ be an MSC and $\gamma : E \to K$ a mapping satisfying (V1) and (V2) (i.e., (L1), (L2) and (L3)). Then $\gamma$ satisfies (V3) iff, for any $t \in E$ and $1 \leq i < k$, we have*

*(L4) $\gamma^-(t)(i) = \heartsuit$ and $\lambda(t) \neq a_{i+1} \in \Sigma$ imply $\gamma^m(t)(i-1) = 0$*

*Proof.* First assume (V3) holds. Let $t \in E$ and $1 \leq i < k$ with $\gamma^-(t)(i) = \heartsuit$ and $\lambda(t) \neq a_{i+1} \in \Sigma_p$. Since every other letter in $\tau$ is $\sharp$, this implies $a_i = \sharp$. Furthermore $t \in E_p$. Since $\gamma^-(t)(i) = \heartsuit$, we have $\gamma(e)(i) = \heartsuit$ for the node $e \in E_p$ with $e \lessdot_p t$. Hence, by (V2), there exists $s \in E$ with $\lambda(s) = x_0$, $\gamma(s)(0) = \heartsuit$ and $R(s, t, i)$. Since $\lambda(t) \neq a_{i+1}$, this ensures in particular that $\delta_{\tau_{i+1}}(s) = \delta_{a_{i+1}}(t)$ or none of them is defined. Furthermore, by (V3), $s$ is a weak marker. Suppose, towards a contradiction, $\gamma^m(t)(i-1) \neq 0$. Then, for node $f \in E$ with $\delta_\sharp(f) = t$ we have $\gamma(f)(i-1) \neq 0$. Hence, by (V1), there is $s' \in E$ with $\lambda(s') = x_0$ and $R(s', f, i-1)$. Since $a_i = \sharp$, $R_2(s', f, i-1)$ is impossible, i.e., $\delta_{\tau_{i-1}}(s') = f$ implying $\delta_{\tau_i}(s') = t$. Thus, by what we showed above, $\delta_{\tau_{i+1}}(s') = \delta_{a_{i+1}}(t) = \delta_{\tau_{i+1}}(s)$. Since $s$ is a weak marker, this implies $s' \leq s$. Hence, Lemma 5.4(2) implies $t = \delta_{\tau_i}(s') \leq \delta_{\tau_i}(s) \leq e \lessdot_p t$, a contradiction. Thus, we inferred (L4) from (V3).

Conversely, assume (L4). Towards a contradiction, let $s \in E$ with $\gamma(s)(0) = \heartsuit$, but $s$ is not a weak marker. By (V2), we obtain $\lambda(s) = x_0$. Since $s$ is no weak marker, there exist a weak marker $s' \in E$ and $0 < i \leq k$ such that $s < s'$, $\lambda(s') = x_0$, and $\delta_{\tau_i}(s) = \delta_{\tau_i}(s')$. Let $0 \leq j < i$ be maximal with $\delta_{\tau_j}(s) \neq \delta_{\tau_j}(s')$ and therefore $\delta_{\tau_j}(s) < \delta_{\tau_j}(s')$ by Lemma 5.4(2). Note that, since $j < i$ and $\delta_{\tau_i}(s')$ is defined, the events $\delta_{\tau_{j+1}}(s)$ and $\delta_{\tau_{j+1}}(s')$ are defined. First suppose $a_{j+1} = \sharp$. Then, by the definition of $\delta_\sharp$, we get $\delta_{\tau_{j+1}}(s) < \delta_{\tau_{j+1}}(s')$ contradicting the maximality of $j$. Thus, $a_{j+1} \in \Sigma_p$ for some process $p$. Since every odd letter in $\tau$ equals $\sharp$, we get $a_j = \sharp$ and $j \geq 1$. Since $\delta_{\tau_{j+1}}(s) = \delta_{\tau_{j+1}}(s')$ is defined, the nodes $\delta_{\tau_j}(s)$ and $\delta_{\tau_j}(s')$ belong to $E_p$ implying $\delta_{\tau_j}(s) <_p \delta_{\tau_j}(s') =: t$. Then there exists $e \in E_p$ with $e \lessdot_p t$ implying $\delta_{\tau_j}(s) \leq_p e \lessdot_p \delta_{\tau_j}(s') <_p \delta_{\tau_{j+1}}(s)$. Hence, by (V2), we get $\gamma(e)(j) = \heartsuit$ implying $\gamma^-(t)(j) \neq 0$. Furthermore, $t = \delta_{\tau_j}(s')$ lies properly between $\delta_{\tau_j}(s)$ and $\delta_{\tau_{j+1}}(s)$. Hence $\lambda(t) \neq a_{j+1}$. Let $f = \delta_{\tau_{j-1}}(s')$ implying $\delta_\sharp(f) = t$ and, by (V1), $\gamma(f)(j-1) = \gamma^m(t)(j-1) \neq 0$. Hence, by (L4) we have $\dagger = \gamma^-(t)(j) = \gamma(e)(j)$. But this contradicts $\gamma(e)(j) = \heartsuit$ as shown before. $\square$

The condition (L5) in the next lemma states that $\delta_\tau(s)$ is labeled by $\heartsuit$ only when $s$ is a marker.

**Lemma 5.10** *Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ be an MSC and $\gamma : E \to K$ a mapping satisfying (V1), (V2), and (V3) (i.e., (L1), (L2), (L3) and (L4)). Then $\gamma$ satisfies (V4) iff, for any $t \in E$, we have*

*(L5)* $\gamma(t)(k) \neq \dagger$

*Proof.* Suppose (V4) holds and let $t \in E$ with $\gamma(t)(k) \neq 0$. Then, by (V1), there is $s \in E$ with $\lambda(s) = x_0$ and $R(s, t, k)$ implying $\delta_\tau(s) = t$. Choose $s$ maximal subject to these restrictions. Then $s$ is a $(x_0, \tau)$-marker. Hence, by (V4), $\gamma(s)(0) = \heartsuit$ implying, by (V2), $\gamma(t)(k) = \heartsuit \neq \dagger$.

Conversely, let $s$ be a marker. Then $t := \delta_\tau(s)$ is defined. Hence, by (V1), $\gamma(t)(k) \neq 0$ implying, by (L5), $\gamma(t)(k) = \heartsuit$. Thus, by (V2), there exists $s' \in E$ with $\lambda(s') = x_0$, $\gamma(s')(0) = \heartsuit$, and $\delta_\tau(s') = t$. Since $s$ is a marker, we get $s' \leq s$. By (V3), $s'$ is a weak marker. Since $\delta_\tau(s')$ is defined, it is actually a marker implying $s \leq s'$. Thus, we showed $s = s'$ and therefore $\gamma(s)(0) = \heartsuit$. □

For the next proposition, recall that an update function $\mathrm{updt} : \Sigma \times K^2 \to 2^K$ takes as arguments the $\Sigma$-label of the node, the value of the immediate predecessor on the same process, and the value of the predecessor w.r.t. the msg or rev arc, resp.

**Proposition 5.11** *There is a function* $\mathrm{updt} : \Sigma \times K^2 \to 2^K$ *such that, for any MSC $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ and any mapping $\gamma : E \to K$, the following are equivalent*

1. *$\gamma$ is a good mapping w.r.t.* updt *and* $\overrightarrow{0}$
2. *$\gamma$ is a valid mapping.*

*Proof.* For $a \in \Sigma$ and $f, g, h \in K$, let $f \in \mathrm{updt}(a, g, h)$ iff the following hold for any $0 \leq i < k$:

(U1) $f(0) \neq 0$ iff $a = x_0$.

(U2) $f(i + 1) \neq 0$ iff

- $a = a_{i+1} \in \Sigma$ and $g(i) \neq 0$, or
- $a_{i+1} = \sharp$ and $h(i) \neq 0$, or
- $a \neq a_{i+2} \in \Sigma$ and $g(t)(i + 1) \neq 0$.

(U3) $f(i + 1) = \heartsuit$ iff

- $a = a_{i+1} \in \Sigma$ and $g(i) = \heartsuit$, or
- $a_{i+1} = \sharp$ and $h(i) = \heartsuit$, or
- $a \neq a_{i+2} \in \Sigma$ and $g(t)(i + 1) = \heartsuit$.

(U4) $g(i) = \heartsuit$ and $a \neq a_{i+1} \in \Sigma$ imply $h(i - 1) = 0$.

(U5) $f(k) \neq \dagger$.

Since (U1-5) are just reformulations of (L1-5), the mapping $\gamma$ is good iff it satisfies (L1-5) and therefore (by the previous lemmas) (V1-4). □

**Lemma 5.12** *Let $M = (E, \lambda, \mathrm{msg}, (<_p)_{p\in\mathcal{P}})$ be an MSC. Then the following are equivalent*

*(1) For any* $e \in \mathrm{Mark}(x_0, \tau)$*, we have*

$$|past(e) \cap \mathrm{Mark}(x_0, \tau)| - |past(e) \cap \mathrm{CoMark}(x_0, \tau)| \in \{1, 2, \dots, kB\}$$

*(2) There exists a good labeling* $\gamma$ *w.r.t. the function* updt *from the previous proposition and* $\overrightarrow{0}$ *and an event* $s$ *with* $\lambda(s) = x_0$ *such that*

$$\begin{aligned}|\{e \in E \mid e \leq s, \gamma(e)(0) = \heartsuit, \lambda(e) = x_0\}| - |\{e \in E \mid e \leq s,\ \gamma(e)(k) = \heartsuit\}|\\ = |\{e \in E \mid e > s,\ \gamma(e)(k) = \heartsuit\}|\end{aligned}$$

*and*

$$\begin{aligned}|\{e \in E \mid e \leq t, \gamma(e)(0) = \heartsuit, \lambda(e) = x_0\}| - |\{e \in E \mid e \leq t,\ \gamma(e)(k) = \heartsuit\}|\\ \in \{1, \dots, 2|\mathcal{P}|B\}\end{aligned}$$

*for any* $t \leq s$ *with* $\lambda(t) = x_0$*.*

*Proof.* First suppose (1). Let $\gamma$ be the unique mapping satisfying (V1), (V2) such that $\gamma(t)(0) = \heartsuit$ iff $t$ is a marker. Then $\gamma$ is a good labeling w.r.t. updt and $\overrightarrow{0}$. Furthermore,

$$\begin{aligned}\mathrm{Mark}(x_0, \tau) &= \{s \in E \mid \gamma(s)(0) = \heartsuit\}\\ \mathrm{CoMark}(x_0, \tau) &= \{s \in E \mid \gamma(s)(k) = \heartsuit\}\end{aligned}$$

With $s$ the maximal marker in $M$, the first condition in (2) follows by elementary arithmetic. Let $t \leq s$ with $\lambda(t) = x_0$. By (1), the difference of the number of markers below $t$ and that of comarkers below $t$ belongs to $\{1, \dots, 2|\mathcal{P}|B\}$. This ensures the second condition in (2).

Conversely, let $\gamma$ be a good labeling and $s \in E$ with $\lambda(s) = x_0$ such that the conditions in (2) are satisfied. Let $\mathrm{wmark}(x_0, \tau)$ denote the set of weak $(x_0, \tau)$-markers in $M$. Then $\mathrm{Mark}(x_0, \tau) \subseteq \mathrm{wmark}(x_0, \tau)$ since any marker is a weak marker. Furthermore, by Lemma 5.4, the set of markers $\mathrm{Mark}(x_0, \tau)$ is downwards closed in the set of weak markers $\mathrm{wmark}(x_0, \tau)$.

Set $X = \{s \in E \mid \gamma(s)(0) = \heartsuit\}$ and $Y = \{t \in E \mid \gamma(t)(k) = \heartsuit\}$. Then, since $\gamma$ satisfies (V1-4), $Y = \mathrm{CoMark}(x_0, \tau)$ and $\mathrm{Mark}(x_0, \tau) \subseteq X \subseteq \mathrm{wmark}(x_0, \tau)$. Hence $\mathrm{Mark}(x_0, \tau)$ is downwards closed in $X$ and

$$\begin{aligned}\{e \in E \mid e \leq s,\ \gamma(e)(k) = \heartsuit\} &= \{e \in \mathrm{CoMark}(x_0, \tau) \mid e \leq s\} \text{ and}\\ \{e \in E \mid e \geq s,\ \gamma(e)(k) = \heartsuit\} &= \{e \in \mathrm{CoMark}(x_0, \tau) \mid e \geq s\}\end{aligned}$$

Since the number of markers in $M$ equals that of comarkers, the first condition in (2) implies

$$\{e \in E \mid e \leq s,\ \gamma(e)(0) = \heartsuit, \lambda(e) = x_0\} = \mathrm{Mark}(x_0, \tau)$$

That is, a node $s$ is a marker iff $\gamma(e)(0) = \heartsuit$.

Now let $t$ be some $(x_0, \tau)$-marker in $M$. Then $t \leq s$ and $\lambda(t) = x_0$. By the above remark we get

$$\{e \in E \mid e \leq t, \gamma(e)(0) = \heartsuit, \lambda(e) = x_0\} = \{e \in \mathrm{Mark}(x_0, \tau) \mid e \leq t\} \text{ and}$$
$$\{e \in E \mid e \leq t, \gamma(e)(k) = \heartsuit\} = \{e \in \mathrm{CoMark}(x_0, \tau) \mid e \leq t\}.$$

By the second condition in (2), the difference of the sizes of these sets belongs to the set $\{1, 2, \ldots, kB\}$. Thus, $M$ satisfies (1). □

**Proposition 5.13** *There exists a CFM $\mathcal{A}$ that accepts an MSC $M$ iff $M$ satisfies condition (1) in Lemma 5.12.*

*Proof.* Let updt be the function from Proposition 5.11 and let $\mathcal{A}'$ be the CFM from Prop. 5.2. Since this CFM computes all good labelings, it can check condition (2) from Lemma 5.12. Now the result follows from Lemma 5.12. □

**Proposition 5.14** *Let $B > 0$ be a natural number. Then there exists a CFM $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathbb{MSC}^B$.*

*Proof.* By the previous proposition, for any $x_0 \in \Sigma$ and $\tau \in \sharp(\Sigma\,\sharp)^*$, the set of MSCs satisfying condition (1) in Lemma 5.12 can be accepted by some CFM. Since $\mathbb{MSC}^B$ is the intersection of finitely many such sets (Prop. 5.6), the result follows. □

**Theorem 5.15** *Let $\mathcal{M}$ be a set of MSCs with $Lin^B(\mathcal{M})$ a regular set of representatives of $\mathcal{M}$. Then there exists a CFM $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{M}$.*

*Proof.* Follows from Propositions 5.14 and 5.3. □

# 6 Further results

## 6.1 Some more model checking

In Section 3, we explained how to do model checking in the realm of CMSC-graphs. Since Theorem 4.1 is effective, we get the following results:

**Corollary 6.1** *The following problems are decidable*

*(1) input: safe CMSC-graph $G$ and an MSO$(\leq, \mathrm{msg})$-sentence $\varphi$
question: Does $\mathcal{L}(G) \subseteq \mathcal{L}(\varphi)$ hold?*

*(2) input: CFM $\mathcal{A}$, $B \in \mathbb{N}$ and an MSO$(\leq, \mathrm{msg})$-sentence $\varphi$
question: Does $\mathcal{L}(\mathcal{A}) \cap \mathbb{MSC}^B \subseteq \mathcal{L}(\varphi)$ hold?*

*(3) input: safe CMSC-graph $G$ and CFM $\mathcal{A}$
question: Does $\mathcal{L}(G) \cap \mathcal{L}(\mathcal{A}) = \emptyset$ hold?*

*(4) input: safe CMSC-graph $G$ and CFM $\mathcal{A}$
question: Does $\mathcal{L}(G) \subseteq \mathcal{L}(\mathcal{A})$ hold?*

*Proof.* (1) Since $G$ is safe, the set $\mathcal{L}(G)$ is, by Lemma 3.2, $\exists$-$B$-bounded with $B = |G|$. By Prop. 5.14, one can construct a CFM $\mathcal{A}_B$ with $\mathcal{L}(\mathcal{A}_B) = \mathbb{MSC}^B$. Theorem 4.1(1)$\Rightarrow$(3) yields a sentence $\varphi_B$ from MSO($\leq$, msg) with $\mathcal{L}(\varphi_B) = \mathcal{L}(\mathcal{A}_B) = \mathbb{MSC}^B$. Let $\psi = \varphi \wedge \varphi_B$. Since $\mathcal{L}(\psi)$ is $\exists$-$B$-bounded, there exists a gc-CMSC-graph $G'$ with $\mathcal{L}(G') = \mathcal{L}(\psi)$ by Theorem 4.1(3)$\Rightarrow$(4). Note that $\mathcal{L}(G) \subseteq \mathcal{L}(\varphi)$ iff $\mathcal{L}(G) \subseteq \mathcal{L}(\varphi) \cap \mathbb{MSC}^B = \mathcal{L}(G')$. But the inclusion $\mathcal{L}(G) \subseteq \mathcal{L}(G')$ is decidable by Prop. 3.7.

(2) Prop. 5.14 gives a CFM $\mathcal{A}_B$ with $\mathcal{L}(\mathcal{A}_B) = \mathbb{MSC}^B$. Now a direct-product construction allows to build a CFM $\mathcal{A}'$ with $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}_B) = \mathcal{L}(\mathcal{A}) \cap \mathbb{MSC}^B$. This allows to apply Theorem 4.1(1)$\Rightarrow$(4) which yields a gc-CMSC-graph $G$ with $\mathcal{L}(G) = \mathcal{L}(\mathcal{A}) \cap \mathbb{MSC}^B$. Note that $G$ is in particular safe. As in (1), we obtain another gc-CMSC-graph $G'$ with $\mathcal{L}(G') = \mathcal{L}(\varphi) \cap \mathbb{MSC}^B$. Since $\mathcal{L}(\mathcal{A}) \cap \mathbb{MSC}^B \subseteq \mathcal{L}(\varphi)$ iff $\mathcal{L}(\mathcal{A}) \cap \mathbb{MSC}^B \subseteq \mathcal{L}(\varphi) \cap \mathbb{MSC}^B$ (i.e., iff $\mathcal{L}(G) \subseteq \mathcal{L}(G')$), the decidability follows from Prop. 3.7.

(3) Let $B = |G|$. By Lemma 3.2, the set $K_G$ is regular set of $B$-bounded representatives of $\mathcal{L}(G)$. It is easily seen that it can be accepted by a finite automaton $\mathcal{B}_G$. Recall that, in Sect. 2.2, we presented an infinite transition system whose accepting paths are labeled by $\mathrm{Lin}(\mathcal{L}(\mathcal{A}))$ by Prop. 2.4. If, in that transition system, we restrict to states $((s_p)_{p\in\mathcal{P}}, (w_{p,q})_{p,q\in\mathcal{P}})$ with $|w_{p,q}| \leq B$ for all $p, q \in \mathcal{P}$ (i.e., we restrict the channels to capacity $B$), we end up with a finite automaton $\mathcal{B}$ accepting $\mathrm{Lin}(\mathcal{L}(\mathcal{A}) \cap \mathbb{MSC}^B)$. Thus, we get $\mathcal{L}(G) \cap \mathcal{L}(\mathcal{A}) = \emptyset$ iff $L(\mathcal{B}_G) \cap L(\mathcal{B}) = \emptyset$ which is decidable.

(4) The same reasoning yields $\mathcal{L}(G) \subseteq \mathcal{L}(\mathcal{A})$ iff $L(\mathcal{B}_G) \subseteq L(\mathcal{B})$ which is decidable. □

Among the model checking instances covered by the proposition above, the only case that was already known is (1) [23]. This problem has non-elementary complexity and the same holds for (2) [20]. We determine the complexity of the model checking procedures for (3) and (4): The finite automaton $\mathcal{B}_G$ has at most $|G|$ many states while the number of states of $\mathcal{B}$ is $\prod_{p\in\mathcal{P}} |S_p| \cdot \prod_{p,q\in\mathcal{P}} C^B$ (where $S_p$ is the set of $p$-local states and $C$ the set of message contents of $\mathcal{A}$), hence it is exponential in $\mathcal{P}$ and $B = |G|$. Thus, the problems (3) and (4) are in PSPACE and EXPSPACE, resp.

## 6.2 Monadic second order logic

It is rather obvious that any formula $\varphi$ from (E)MSO($(<_p)_{p\in\mathcal{P}}$, msg) can be rewritten into a formula $\psi$ from (E)MSO($\leq$, msg) with $\mathcal{L}(\varphi) = \mathcal{L}(\psi)$ since $x <_p y$ is equivalent to $x \leq y \wedge x \neq y \wedge \bigvee_{p\in\mathcal{P}} \bigvee_{a,b\in\Sigma_p} (v_a(x) \wedge v_b(y))$. Conversely, $x \leq y$ iff there are mutually distinct processes $p_1, p_2, \ldots, p_n$ and $u_i, v_i \in E_{p_i}$ with $x = v_1$, $\mathrm{msg}(v_i) = u_{i+1} <_p v_{i+1}$ for $1 \leq i < n$ and $u_n = y$. Since $n \leq |\mathcal{P}|$, this can be expressed as a first-order formula over $((<_p)_{p\in\mathcal{P}}, \mathrm{msg})$, i.e., the logics (E)MSO($(<_p)_{p\in\mathcal{P}}$, msg) and (E)MSO($\leq$, msg) are equally expressive.

From [7, Cor. 5.7], we know that the logics MSO($\leq$) and EMSO($\leq$, msg) are incomparable.

Now let $B \in \mathbb{N}$ and $\varphi$ be some sentence from MSO($\leq$, msg) such that any MSC from $\mathcal{L}(\varphi)$ is universally-$B$-bounded. Then, by [7] and [15], there exists a sentence $\psi$

in MSO($\leq$) with $\mathcal{L}(\psi) = \mathcal{L}(\varphi)$; the converse is immediate.

The same holds for existentially bounded sets of MSCs:

**Proposition 6.2** *Let $\varphi$ be an (E)MSO($(\leq_p)_{p\in\mathcal{P}}, \mathrm{msg}$) formula and $B \in \mathbb{N}$ with $\mathcal{L}(\varphi) \subseteq \mathbb{MSC}^B$. Then there exists an (E)MSO($\leq$) formula $\psi$ with $\mathcal{L}(\varphi) = \mathcal{L}(\psi)$.*

*Proof.* Let $X_0, \dots, X_{B-1}$ be set variables that are not used in $\varphi$. Then we write down a formula $\varphi_0$ expressing that $X_n$ contains precisely those events $e$ with $\lambda_I(e) = (a, n)$ for some $a \in \Sigma$ (see Section 3.2). From now on, we assume that $\varphi_0$ holds. Then it makes sense to write $v_{a,n}(x)$ for $v_a(x) \wedge x \in X_n$ since this formula holds of an event $e$ iff $\lambda_I(e) = (a, n)$.

We also use the abbreviation $x <_m y$ for $\bigvee_{p\neq q, n<B} v_{p!q,n}(x) \wedge v_{q?p,n}(y) \wedge x \leq y \wedge \forall z((x \leq z \wedge v_{q?p,n}(z)) \to y \leq z)$. Then $x <_m y$ expresses that there are processes $p, q \in \mathcal{P}$ and a number $0 \leq n < B$ such that $\lambda_I(x) = (p!q, n)$ and $y$ is the first event above $x$ with $\lambda_I(y) = (q?p, n)$. Note that for any send-event $e$ there is a receive event $f$ with $e <_m f$ since the number of sends equals that of receives and since receives have to follow sends. Thus, $<_m$ is a function from the set of send events $S$ into the set of receive events $R$.

Since $x_1 <_m y_1$, $x_2 <_m y_2$, $\widetilde{\lambda}(x_1) = \widetilde{\lambda}(x_2)$ and $x_1 \leq x_2$ imply $y_1 \leq y_2$, the function $<_m$ is even order-preserving. Thus, for $\exists$-$B$-bounded MSCs, the functions $<_m$ and msg coincide.

Let $\varphi_1$ be obtained from $\varphi$ by replacing all occurrences of $(x, y) \in \mathrm{msg}$ by $x <_m y$. Then an $\exists$-$B$-bounded MSC $M$ satisfies $\exists X_0, \dots, X_{B-1}(\varphi_0 \wedge \varphi_1)$ iff $M$ satisfies $\varphi$. However, if $M$ is not $\exists$-$B$-bounded this equivalence might not hold.

We use a second formula $\varphi_2$ which is $\forall x, x', y, (x <_m y \wedge x' <_m y \to x = x')$ stating that the function $<_m$ is injective. Since in an MSC, the number of sends $p!q$ equals that of receives $q?p$ for any $p, q \in \mathcal{P}$, the formula $\varphi_2$ states that $<_m$ is actually an order-preserving bijection from $S$ onto $R$. Hence, $<_m$ corresponds to the message relation (on $\exists$-$B$-bounded MSCs).

It remains to express that the MSC is $\exists$-$B$-bounded. For this we can apply Theorem 4.1 to the CFM constructed in Proposition 5.14, thus obtaining an equivalent EMSO($(\lessdot_p)_{p\in\mathcal{P}}, \mathrm{msg}$) formula. Replacing in this formula the message relation msg by $<_m$ yields a formula $\varphi_3$. Now let $M$ be an MSC and let $X_0, \dots, X_{B-1}$ be sets of events satisfying $\varphi_0$. Then $M$ satisfies $\varphi_2 \wedge \varphi_3$ iff $M$ is $\exists$-$B$-bounded and msg $=<_m$. Hence, with $\psi = \exists X_0, \dots, X_{B-1}(\varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3)$, we obtain $\mathcal{L}(\varphi) = \mathcal{L}(\psi)$. $\square$

## 7 Conclusion

We showed the equivalence of several formalisms for the specification of existentially bounded sets of MSCs, namely communicating finite machines, monadic second order logic, globally-cooperative compositional message sequence graphs, and regular sets of representatives. Corresponding results were known for universally bounded sets of MSCs. The new results were obtained by an adaptation of Kuske's technique that allows to transfer results on Mazurkiewicz traces to the realm of MSCs. Our construction of a CFM that accepts all existentially $B$-bounded MSCs is based on a new characterization of this class of MSCs. Since the different formalisms can be transformed effectively into each other, we obtain the decidability of several model checking problems involving CFMs, message sequence graphs, and monadic second order logic.

The main questions left open in this paper concern the implementation of MSC specifications. In practice, there are two important issues for CFMs: deadlock-freeness and determinism. There are two sources of non-determinism in the CFMs we constructed:

1. The simulation of an asynchronous automaton by a CFM requires guessing of information which is not yet available due to the lack of the rev-edges that the asynchronous automaton can access.
2. The CFM that accepts all existentially $B$-bounded MSCs guesses the markers for detecting cycles.

In the setting of universally bounded sets of MSCs, these issues did not come up. The first could be circumvented since the rev-edges were not necessary to obtain a trace from a universally $B$-bounded MSC. To check whether a MSC is universally $B$-bounded, a deterministic CFM suffices that locally counts the number of messages in transit.

It seems unlikely that our expressivity results hold for deterministic CFMs. A related question is that of freedom of deadlock. There are clearly CFMs with universally $B$-bounded behavior that cannot be transformed into an equivalent deadlock-free CFM. So the problem arises to characterize the existentially $B$-bounded behaviors of deadlock-free CFMs.

## References

[1] P. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[2] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *CONCUR'99*, LNCS 1664, pp. 114-129, 1999.

[3] J. Berstel. Transductions and context-free languages. Teubner Studienbücher, Stuttgart, 1979.

[4] A. Bouajjani and P. Habermehl. Symbolic Reachability Analysis of FIFO-Channel Systems with Nonregular Sets of Configurations. *Theoretical Computer Science*, 221(1-2):211–250, 1999.

[5] B. Boigelot, P. Godefroid, B. Willems and P. Wolper. The Power of QDDs. In *SAS'97*, LNCS 1302, pp. 172-186, 1997.

[6] B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. *Formal Methods in System Design*, 14(3):237-255 (1999).

[7] B. Bollig and M. Leucker. Message-Passing Automata are expressively equivalent to EMSO Logic. *Theoretical Computer Science*, to appear, 2005. (An extended

abstract appeared under the same title in *CONCUR'04*, LNCS 3170, pp. 146-160, 2004.)

[8] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323-342, 1983.

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

[10] W. Ebinger and A. Muscholl. Logical definability on infinite traces. *Theoretical Computer Science*, 154:67–84, 1996. (An extended abstract appeared under the same title in *ICALP'93*, LNCS 700, pp. 335–346, 1993.)

[11] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1,2):63–92, 2001.

[12] B. Genest, M. Minea, A. Muscholl, and D. Peled. Specifying and verifying partial order properties using template MSCs. In *FoSSaCS'04*, LNCS 2987, pp. 195-210, 2004.

[13] E. Gunter, A. Muscholl, and D. Peled. Compositional Message Sequence Charts. *International Journal on Software Tools for Technology Transfer (STTT)* 5(1): 78-89 (2003). (An extended abstract appeared in *TACAS'01*, LNCS 2031, pp. 496–511, 2001.)

[14] B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state High-level MSCs: Model checking and realizability. In *ICALP'02*, LNCS 2380, pp.657-668, 2002. Journal version in *Journal of Computer and System Sciences*, in press, December 2005.

[15] J. G. Henriksen, M. Mukund, K. Narayan Kumar, M. Sohoni and P. Thiagarajan. A Theory of Regular MSC Languages. *Information and Computation*, 202(1):1–38, 2005.

[16] ITU-TS recommendation Z.120, Message Sequence Charts, Geneva, 1999.

[17] D. Kuske. Regular sets of infinite message sequence charts. *Information and Computation*, 187:80–109, 2003.

[18] M. Lohrey and A. Muscholl. Bounded MSC communication. *Information and Computation*, 189:135–263, 2004.

[19] S. Leue, R. Mayr, and W. Wei. A scalable incomplete test for the boundedness of UML RT models. In *TACAS'04*, LNCS 2988, pp. 327–341, 2004.

[20] A.R. Meyer. Weak monadic second order theory of one successor is not elementary recursive. In: *Proc. Logic Colloquium*, Lecture Notes in Mathematics vol. 453, Springer 1975, pp. 132-154.

[21] R. Morin. Recognizable Sets of Message Sequence Charts. In *STACS'02*, LNCS 2285, pp. 523-534, 2002.

[22] P. Madhusudan. Reasoning about Sequential and Branching Behaviours of Message Sequence Graphs. In *ICALP'01*, LNCS 2076, pp. 809-820, 2001.

[23] P. Madhusudan and B. Meenakshi. Beyond Message Sequence Graphs In *FSTTCS'01*, LNCS 2245, pp. 256-267, 2001.

[24] A. Mazurkiewicz. Concurrent program schemes and their interpretation. Technical report, DAIMI Report PB-78, Aarhus University, 1977.

[25] A. Muscholl and D. Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In *MFCS'99*, LNCS 1672, pp. 81-91, 1999.

[26] E. Ochmański. Regular behaviour of concurrent systems In *Bulletin of the EATCS* 27, pp.56-67, 1985.

[27] D. Peled. Specification and Verification of Message Sequence Charts. In *FORTE/PSTV'00*, pp. 139-154, 2000.

[28] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letter*, 83(5):251–261, 2002.

[29] USB 1.1 specification, available at http://www.usb.org/developers/docs/usbspec.zip

[30] W. Thomas. On logical definability of trace languages. In V. Diekert, editor, *Proceedings of a workshop of the ESPRIT BRA No 3166: Algebraic and Syntactic Methods in Computer Science (ASMICS) 1989*, Report TUM-I9002, Technical University of Munich, pages 172–182, 1990.

[31] W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. – Informatique Théorique et Applications*, 21:99-135, 1987.