Realizability of high-level message sequence charts: closing the gaps *

Markus Lohrey

Institut für Informatik, Universität Stuttgart,
Breitwiesenstr. 20-22, 70565 Stuttgart, Germany
lohrey@informatik.uni-stuttgart.de

Abstract. We study the notion of safe realizability for high-level message sequence charts (HMSCs) [2]. We show that safe realizability is EXPSPACE-complete for bounded HMSCs but undecidable for the class of all HMSCs. This solves two open problems from [2]. Moreover we prove that safe realizability is also EXPSPACE-complete for the larger class of globally-cooperative HMSCs.

1 Introduction

Message sequence charts (MSCs) are a popular visual formalism for specifying communication scenarios of asynchronous processes, where most of the details (variables, timing constraints, etc) are abstracted away. They are part of the ITU standard [16]. *High-level message sequence charts* (HMSCs) extend MSCs by allowing iteration and non-deterministic choices. In this way infinite sets of MSCs can be described.

HMSCs are a suitable formalism for the purpose of specification. On the other hand, HMSCs allow to describe communication patterns, like for instance non-local choices [5], which are quite pathological from a practical point of view. Thus HMSCs should not be considered as a model for implementations. This rises the question of realizability (or implementability): Given an HMSC (the specification), is it possible to implement it as a communicating protocol (the implementation), which shows the same behaviour as the original HMSC?

Concerning the formal definition of realizability, we follow Alur et al [1, 2], which define two notions of realizability: *weak realizability* and *safe realizability*. Both are based on the model of *communicating finite state machines* (CFMs) with FIFO queues for describing the implementation. CFMs appeared as one of the earliest abstract models for concurrent systems [6], and are used for instance in the specification language SDL [15]. An accepting run of a CFM generates in a canonical way an MSC. Thus, in [2] an HMSC H is called weakly realizable, if there exists a CFM $\mathcal{A}$ such that the set of all MSCs generated by the accepting runs of $\mathcal{A}$ is precisely the set of MSCs defined by H. In practice, such an implementation may be considered as being too weak. A very desirable further

* This work was done while the author was on leave at IRISA, Campus de Beaulieu, 35042 Rennes, France and supported by the INRIA cooperative research action FISC.

property of the implementation $\mathcal{A}$ is *deadlock-freeness*: every partial run of $\mathcal{A}$ can be completed to a run that terminates in a final state of $\mathcal{A}$. Thus, in [2] an HMSC H is called safely realizable, if there exists a *deadlock-free* CFM $\mathcal{A}$ such that the set of all MSCs generated by the accepting runs of $\mathcal{A}$ is precisely the set of MSCs defined by H.

In [2] it is shown that weak realizability is already undecidable for *bounded HMSCs*, a class of HMSCs which was introduced in [4, 21] because of its nice model-checking properties. As shown in [19], FIFO communication (i.e., message overtaking is not allowed) is the reason for this negative result: for non-FIFO communication weak realizability is decidable for bounded HMSCs. Concerning safe realizability, Alur et al prove in [2] an EXPSPACE upper bound as well as a PSPACE lower bound for safe realizability of bounded HSMCs, but the exact complexity remained open. In Section 3.1, we will prove that safe-realizability is in fact EXPSPACE-complete for bounded HMSCs. Using the same proof technique we will also show that safe realizability is undecidable for the class of all HMSCs, which solves the second open problem from [2]. Furthermore, in Section 3.2, we will extend our EXPSPACE-completeness result from bounded to *globally-cooperative HMSCs* [9, 19], which share many of the nice algorithmic properties of bounded HMSCs. Finally, in Section 4 we argue that all our results remain valid for non-FIFO communication.

Let us remark that the notion of realizability used in this paper is a quite strict one in the sense that it allows neither the introduction of new messages nor the addition of further content to already existing messages. More liberal realizations that allow the latter were studied in [9]. Other approaches to the realization problem can be also found in [7, 11].

A preliminary version of this paper appeared in [18].

2 Preliminaries

For complexity results we will use standard classes like PSPACE (polynomial space) and EXPSPACE (exponential space), see [22] for definitions.

Let Σ be an alphabet of symbols and $\Gamma \subseteq \Sigma$. We denote with $\pi_\Gamma : \Sigma^* \to \Gamma^*$ the projection morphism onto the subalphabet Γ. The empty word is denoted by ε. The length of the word $w \in \Sigma^*$ is $|w|$. For $k \in \mathbb{N}$ let $w[1, k]$ be the prefix of w of length $\min\{k, |w|\}$. For $u, v \in \Sigma^*$ we write $u \sqsubseteq v$, if u is a prefix of v.

A *pomset* is a labeled partial order $\mathcal{P} = (A, \lambda, \prec)$, i.e., $(A, \prec)$ is a partial order and $\lambda : A \to \Sigma$ is a labeling function. For $B \subseteq A$ we define the restricted pomset $\mathcal{P}\restriction_B = (B, \lambda\restriction_B, \prec\restriction_B)$. A word $\lambda(a_1)\lambda(a_2)\cdots\lambda(a_n) \in \Sigma^*$ is a *linearization* of $\mathcal{P}$ if $A = \{a_1, a_2, \ldots, a_n\}$, $a_i \neq a_j$ for $i \neq j$, and $a_i \prec a_j$ implies $i < j$ for all i, j. With $\mathrm{lin}(\mathcal{P}) \subseteq \Sigma^*$ we denote the set of all linearizations of $\mathcal{P}$.

For this paper, we use some basic notions from trace theory, see [8] for more details. An *independence relation* on the alphabet Σ is a symmetric and irreflexive relation $I \subseteq \Sigma \times \Sigma$. The complementary relation $(\Sigma \times \Sigma) \backslash I$ is also called a *dependence relation*. On Σ^* we define the equivalence relation $\equiv_I$ as the transitive reflexive closure of the symmetric relation $\{(uabv, ubav) \mid u, v \in \Sigma^*, \ (a, b) \in I\}$. For a subset $L \subseteq \Sigma^*$ we define the *I-closure* of L by

$$[L]_I = \{v \in \Sigma^* \mid \exists u \in L : u \equiv_I v\} \subseteq \Sigma^*.$$

Let $\mathcal{A}$ be a finite automaton over the alphabet Σ and assume that $\rightarrow \subseteq Q \times \Sigma \times Q$ is the transition relation of $\mathcal{A}$. Then $\mathcal{A}$ is called *loop-connected with respect to I*, if for every loop $q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_n \xrightarrow{a_n} q_1$ of $\mathcal{A}$, the set $\{a_1, \ldots, a_n\} \subseteq \Sigma$ induces a connected subgraph of $(\Sigma, \ (\Sigma \times \Sigma) \backslash I)$. For a loop connected automaton $\mathcal{A}$, one can construct an automaton $\mathcal{A}'$ of size bounded exponentially in the size of $\mathcal{A}$ such that $L(\mathcal{A}') = [L(\mathcal{A})]_I$ [21]. In general, this exponential blow-up cannot be avoided, see [21] for an example.

2.1 Message sequence charts

For the rest of this paper let P be a finite set of *processes* ($|P| \geq 2$) and $\mathbb{C}$ be a finite set of *message contents*. With $\mathrm{Ch} = \{(p, q) \in P \times P \mid p \neq q\}$ we denote the set of all *channels*. The set of *types of process* $p \in P$ is

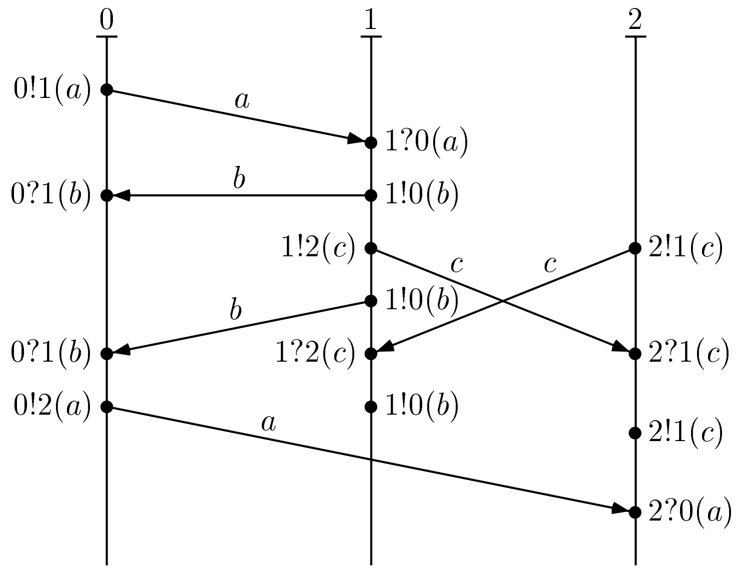
$$\Sigma_p = \{p!q(c), p?q(c) \mid q \in P \backslash \{p\}, c \in \mathbb{C}\}$$

and the set of all *types* is $\Sigma = \bigcup_{p \in P} \Sigma_p$. With $p!q(c)$ we denote the type of an event that sends from process p a message with content c to process q, whereas $p?q(c)$ denotes the type of an event that receives on process p a message with content c from process q. A *partial message sequence chart (pMSC)* over P and $\mathbb{C}$ is a tuple $M = (E, t, m, \prec)$, where:

- E is a finite set of *events*.
- $t : E \to \Sigma$ labels each event with its type. The set of events *located on process* $p \in P$ is $E_p = t^{-1}(\Sigma_p)$. Let $E_! = \{e \in E \mid \exists p, q \in P, c \in \mathbb{C} : t(e) = p!q(c)\}$ be the set of *send events* and $E_? = E \backslash E_!$ be the set of *receive events*.
- $m : D \to E_?$ is a bijection between a subset $D \subseteq E_!$ of the send events and the receive events such that $m(s) = r$ and $t(s) = p!q(c)$ implies $t(r) = q?p(c)$. In this case we also say that (s, r) is a *message* in M from process p to q with content c. If $s \in E_! \backslash D$ with $t(s) = p!q(c)$ then s is called an *unmatched send event* in M from p to q with content c.
- $\prec$ is a partial order on E, called the *visual order of M*, such that for every $p \in P$, the restriction of $\prec$ to E_p is a total order, and $\prec$ is equal to the transitive closure of

$$\{(e_1, e_2) \mid e_1 \prec e_2, \ \exists p \in P : e_1, e_2 \in E_p\} \ \cup \ \{(s, m(s)) \mid s \in D\}.$$

Partial message sequence charts are called left-closed compositional message sequence charts in [9]. Often pMSCs are further restricted to satisfy the *FIFO condition*, which means that for all $s_1, s_2 \in E_!$, if $s_1 \prec s_2$, $t(s_1) = p!q(c)$, $t(s_2) = p!q(d)$, and $s_2 \in D$, then also $s_1 \in D$ and $m(s_1) \prec m(s_2)$, i.e., message overtaking on any channel is disallowed. *For the main part of this paper we always assume the FIFO restriction without mentioning it explicitly*, only in Section 4 we briefly discuss the non-FIFO case. The pMSC definition may also include local actions, however this is not important in the present setting. We use the usual graphical representation of pMSCs, where time flows top-down, processes are drawn as vertical lines, and arrows represent messages. The following diagram shows a pMSC with two unmatched send events.



Let $M = (E, t, m, \prec)$ be a pMSC, where $m : D \to E_?$ for $D \subseteq E_!$. We also write $E(M) = E$. We identify M with the pomset $(E, t, \prec)$, and we identify pMSCs if they are isomorphic as pomsets. In particular, for $F \subseteq E$ we can define the restricted pomset $M\restriction_F$, which in general is not a pMSC. If $D = E_!$, i.e., if there are no unmatched send events, then M is called a *message sequence chart* (MSC) over P and $\mathbb{C}$. With $\mathrm{p}\mathbb{MSC}_{P,\mathbb{C}}$ (resp. $\mathbb{MSC}_{P,\mathbb{C}}$) we denote the set of all pMSCs (resp. MSCs) over P and $\mathbb{C}$. In the sequel, we will omit the subscripts P and $\mathbb{C}$, if they are clear from the context. Let $|M| = |E|$ denote the *size of* M. Let $P(M) = \{p \in P \mid E_p \neq \emptyset\}$ be the set of all processes that are active in M. More generally, for $F \subseteq E$ let $P(M\restriction_F) = \{p \in P \mid E_p \cap F \neq \emptyset\}$ be the set of all processes that participate in $M\restriction_F$. The *communication graph* $G(M)$ of M is defined as the directed graph $G(M) = (P(M), \mapsto)$, where $p \mapsto q$ if and only if there exists in M a message from p to q (with arbitrary content). Note that $G(M)$ does not contain isolated points. This is different from [4], where the set of nodes of $G(M)$ consists of all processes. For $p \in P$ let $\pi_p(M) = \pi_{\Sigma_p}(w)$, where $w \in \mathrm{lin}(M)$ is an arbitrary linearization of M (note that $\pi_{\Sigma_p}(w_1) = \pi_{\Sigma_p}(w_2)$ for all $w_1, w_2 \in \mathrm{lin}(M)$).

Let $M_i = (E_i, t_i, m_i, \prec_i)$, $i \in \{1, 2\}$, be two pMSCs over P and $\mathbb{C}$ such that $E_1 \cap E_2 = \emptyset$ and for all $(p, q) \in \mathrm{Ch}$, if there is an unmatched send event from p

to q in M_1, then there is no message from p to q in M_2 (there may be unmatched sends from p to q in M_2). Then the *concatenation* of M_1 and M_2 is the pMSC $M_1 \cdot M_2 = (E_1 \cup E_2,\, t_1 \cup t_2,\, m_1 \cup m_2,\, \prec)$, where $\prec$ is the transitive closure of

$$\prec_1 \cup \prec_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid \exists p \in P : e_1 \text{ and } e_2 \text{ are located on process } p\}.$$

For the case that $M_1, M_2 \in \mathbb{MSC}$ this corresponds to the usual definition of MSC-concatenation. Note that concatenation is only partially defined on $\mathrm{p}\mathbb{MSC}$ but totally defined on $\mathbb{MSC}$. In case $M_1 \in \mathbb{MSC}$, the concatenation $M_1 \cdot M_2$ is always defined.

Let $F \subseteq E(M)$ be an arbitrary set of events of the pMSC M. As already remarked, the pomset $N = M{\restriction_F}$ is in general not a pMSC. On the other hand, if F is *downward-closed*, i.e., $e \prec f \in F$ implies $e \in F$, then $N = M{\restriction_F}$ is again a pMSC over P and $\mathbb{C}$. We write $N \leq M$ in this case, this defines a partial order $(\mathrm{p}\mathbb{MSC}, \leq)$ on the set of pMSCs. The pomset $M{\restriction_{E\setminus F}}$ will be denoted by $M \setminus N$. In general, $M \setminus N$ is not a pMSC. On the other hand, if a send event $s \in F$ is unmatched in M whenever it is unmatched in N (i.e., no message arrows are crossing from F to its complement $E \setminus F$, this happens in particular if N is an MSC), then $M \setminus N \in \mathrm{p}\mathbb{MSC}$ and moreover $M = N \cdot (M \setminus N)$.

We say that an MSC $M \in \mathbb{MSC}$ is *atomic* if M cannot be written as $M = M_1 \cdot M_2$ for MSCs $M_1, M_2 \in \mathbb{MSC} \setminus \{\emptyset\}$, where $\emptyset$ stands for the MSC with an empty set of events. With $\mathbb{A}_{P,\mathbb{C}}$ (briefly $\mathbb{A}$) we denote the set of atomic MSCs over P and $\mathbb{C}$. Already for $|P| = 2$, the set $\mathbb{A}$ is easily seen to be infinite, see e.g. [10, Sec. 3] for an example. On $\mathbb{A}$ we define an independence relation $\mathcal{I}$ by $(A, B) \in \mathcal{I}$ if $P(A) \cap P(B) = \emptyset$. Obviously, every $M \in \mathbb{MSC}$ can be written as $M = A_1 \cdot A_2 \cdots A_m$, where $A_i \in \mathbb{A}$. Furthermore, this factorization is unique up to $\mathcal{I}$-commutations, a fact that will be crucial in Section 3.2, see [12, 19]:

Lemma 2.1 (cf [12, 19]). *If $A_1, \ldots, A_m, B_1, \ldots, B_n \in \mathbb{A}$ are atoms such that the MSCs $A_1 \cdot A_2 \cdots A_m$ and $B_1 \cdot B_2 \cdots B_n$ are equal then the words $u = A_1 A_2 \cdots A_n$ and $v = B_1 B_2 \cdots B_m$ over $\mathbb{A}$ satisfy $u \equiv_{\mathcal{I}} v$.*

The *supremum* (resp. *infimum*) of two pMSCs $M_1, M_2 \in \mathrm{p}\mathbb{MSC}$ in the partial order $(\mathrm{p}\mathbb{MSC}, \leq)$ is denoted by $\sup(M_1, M_2)$ (resp. $\inf(M_1, M_2)$). In general, $\sup(M_1, M_2)$ does not exist (whereas $\inf(M_1, M_2)$ always exists):

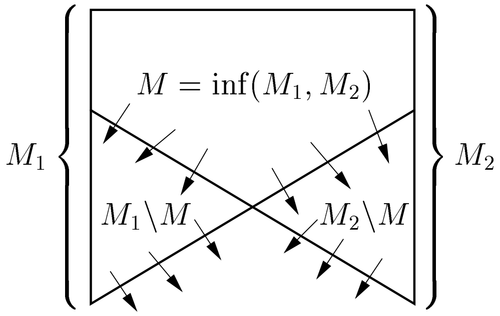
Lemma 2.2. *Let $M_1, M_2 \in \mathrm{p}\mathbb{MSC}$. Then $\sup(M_1, M_2)$ exists if and only if for all $p \in P$, either $\pi_p(M_1) \sqsubseteq \pi_p(M_2)$ or $\pi_p(M_2) \sqsubseteq \pi_p(M_1)$. Moreover, if $\sup(M_1, M_2)$ exists and $M = \inf(M_1, M_2)$ then the following holds:*

(1) $M \neq \emptyset$ *if and only if* $P(M_1) \cap P(M_2) \neq \emptyset$
(2) $P(M_1 \setminus M) \cap P(M_2 \setminus M) = \emptyset$
(3) $\sup(M_1, M_2) \setminus M_1 = M_2 \setminus M$
(4) *If $M_1 \in \mathbb{MSC}$ and there is an unmatched send event e of type $p!q(c)$ in M then $q \notin P(M_2 \setminus M)$.*

(5) If $M_1 \in \mathbb{MSC}$ then $M_2 \backslash M$ is a pMSC and $M_2 = M \cdot (M_2 \backslash M)$.
(6) If $M_1, M_2 \in \mathbb{MSC}$ then also $M \in \mathbb{MSC}$.
(7) If $M_1, M_2 \in \mathbb{A}$ and $M \neq \emptyset$ then $M_1 = M_2$.

Proof. If $\sup(M_1, M_2)$ exists then there exists $N \in \mathrm{p}\mathbb{MSC}$ such that $M_1 \leq N$ and $M_2 \leq N$. Thus $\pi_p(M_1) \sqsubseteq \pi_p(N)$ and $\pi_p(M_2) \sqsubseteq \pi_p(N)$. Hence either $\pi_p(M_1) \sqsubseteq \pi_p(M_2)$ or $\pi_p(M_2) \sqsubseteq \pi_p(M_1)$. On the other hand, if for all $p \in P$, either $\pi_p(M_1) \sqsubseteq \pi_p(M_2)$ or $\pi_p(M_2) \sqsubseteq \pi_p(M_1)$, then we can define words $u_p, v_p \in \Sigma_p^*$ $(p \in P)$ as follows: (i) if $\pi_p(M_1) \sqsubseteq \pi_p(M_2)$ then $u_p = \pi_p(M_1)$ and $v_p = \pi_p(M_2)$, and (ii) if $\pi_p(M_2) \sqsubseteq \pi_p(M_1)$ then $u_p = \pi_p(M_2)$ and $v_p = \pi_p(M_1)$. It it not difficult to see that there exist unique pMSCs M and N such that $\pi_p(M) = u_p$ and $\pi_p(N) = v_p$ for all $p \in P$. Then $M_1 \leq N$, $M_2 \leq N$, and $\sup(M_1, M_2)$ exists, in fact $N = \sup(M_1, M_2)$. Thus we have shown the first statement from the lemma. Moreover, $M = \inf(M_1, M_2)$, and (1), (2), and (3) follow immediately. For (4), assume that $M_1 \in \mathbb{MSC}$ and let s be an unmatched send event in M of type $p!q(c)$. Since $M_1 \in \mathbb{MSC}$, s has a corresponding receive event in M_1, which must be contained in $M_1 \backslash M$. Thus $q \in P(M_1 \backslash M)$. Since $P(M_1 \backslash M) \cap P(M_2 \backslash M) = \emptyset$ by (2), it follows $q \notin P(M_2 \backslash M)$, which shows (4). (5) follows easily from (4). For (6) note that if $M_1, M_2 \in \mathbb{MSC}$, then by (4), M cannot have any unmatched send events, hence $M \in \mathbb{MSC}$. Finally (5) and (6) imply (7). □

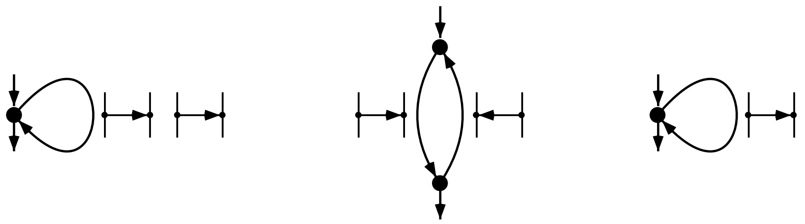
The following picture visualizes the general situation. Arrows that are leaving some region correspond to unmatched sends, and the whole region corresponds to the supremum.



The ITU standard Z.120 defines *high-level message sequence charts (HMSCs)* as finite transition systems with nodes labeled by MSCs. Here we prefer to label edges by MSCs, which does not change the expressive power of HMSCs. Thus, an HMSC H over P and $\mathbb{C}$ is a tuple $H = (V, \rightarrow, v_0, F)$, where V is a finite set of nodes, $\rightarrow \subseteq V \times \mathbb{MSC}_{P,\mathbb{C}} \times V$ is a finite set of labeled edges, $v_0 \in V$ is the initial node, and $F \subseteq V$ is the set of final nodes. Instead of $(u, M, v) \in \rightarrow$, we write $u \xrightarrow{M}_H v$. The MSC-language $\mathrm{msc}(H)$ defined by H is the set of all MSCs $M_1 \cdot M_2 \cdots M_n$, where $v_0 \xrightarrow{M_1}_H v_1 \xrightarrow{M_2}_H \cdots \xrightarrow{M_n}_H v_n \in F$ for some $v_1, \ldots, v_n \in V$. We impose the restriction that $\rightarrow \subseteq V \times \mathbb{A}_{P,\mathbb{C}} \times V$. This assumption does not change the expressiveness of HMSCs and can be easily established by adding further nodes to V. Let $\mathbb{A}_H = \{A \in \mathbb{A} \mid \exists u, v \in V : u \xrightarrow{A}_H v\}$. We may view H also as a finite automaton over the alphabet $\mathbb{A}_H$ of atoms, which accepts the set $L(H) \subseteq \mathbb{A}_H^*$ of *words over* $\mathbb{A}_H$. We will denote this automaton by H as well. An HMSC H is called *bounded* [4, 21] if for every cycle

$$v_1 \xrightarrow{A_1}_H v_2 \xrightarrow{A_2}_H \cdots \xrightarrow{A_{n-1}}_H v_n \xrightarrow{A_n}_H v_1,$$

the communication graph $G(A_1 \cdot A_2 \cdots A_n)$ is strongly connected, i.e., for all $p, q \in P(G)$ we have $p \overset{*}{\mapsto} q \overset{*}{\mapsto} p$. In [4] it is shown that for a bounded HMSC H the language $\mathrm{lin}(\mathrm{msc}(H)) \subseteq \Sigma^*$ of all linearizations of MSCs generated by H is regular, which makes several model-checking problems decidable for bounded HMSCs. On the other hand, bounded HMSCs are a quite restricted class, since they only allow the specification of behaviours where the size of communication buffers stays within some fixed bound. Thus, only finite state systems can be specified. Fortunately, many model checking problems stay decidable for a larger class of (infinite state) HMSCs: In [9], an HMSC H is called *globally-cooperative* if H, viewed as a finite automaton over the alphabet $\mathbb{A}_H$, is loop-connected with respect to the independence relation $\mathcal{I} \subseteq \mathbb{A} \times \mathbb{A}$. Globally-cooperative HMSCs were independently introduced in [19] as c-HMSCs. It is easy to see that every bounded HMSC is globally-cooperative. Finally, H is called *$\mathcal{I}$-closed* if H, viewed as a finite automaton over $\mathbb{A}_H$, satisfies $L(H) = [L(H)]_\mathcal{I}$. Thus, by [21], for a globally-cooperative HMSC H there exists an $\mathcal{I}$-closed HMSC H' of size bounded exponentially in the size of H such that $L(H') = [L(H)]_\mathcal{I}$ and thus also $\mathrm{msc}(H) = \mathrm{msc}(H')$. The diagram below shows three simple HMSCs. The first one is not globally-cooperative (and hence not bounded). The second HMSC is bounded (and hence globally-cooperative). Finally, the third HMSC is globally-cooperative but not bounded.



2.2 Communicating finite state machines

In this section we briefly introduce *communicating finite state machines* (CFMs) The tight relationship between CFMs and the theory of MSCs is well-known, see e.g. [13, 14, 17, 20].

The set of *buffer configurations* is the set $(\mathbb{C}^*)^{\mathrm{Ch}}$ of all functions from the set of channels Ch to the set $\mathbb{C}^*$ of all words over the alphabet $\mathbb{C}$ of message contents. The buffer configuration $\mathcal{B} \in (\mathbb{C}^*)^{\mathrm{Ch}}$ such that $\mathcal{B}(p, q) = \varepsilon$ for all $(p, q) \in \mathrm{Ch}$ is denoted by $\mathcal{B}_\emptyset$. Recall from the previous section that Σ_p is the set of all types of process p. A CFM over P and $\mathbb{C}$ is a tuple $\mathcal{A} = (\mathcal{A}_p)_{p \in P}$ of finite nondeterministic automata. Each $\mathcal{A}_p$ is a tuple $\mathcal{A}_p = (S_p, \Sigma_p, \delta_p, s_{0,p}, F_p)$, where S_p is the finite set of states of $\mathcal{A}_p$, $\delta_p \subseteq S_p \times \Sigma_p \times S_p$ is the transition relation of $\mathcal{A}_p$, $s_{0,p} \in S_p$ is the initial state of $\mathcal{A}_p$, and $F_p \subseteq S_p$ is the set of final states of $\mathcal{A}_p$. We say that $\mathcal{A}$ is *deterministic* if every $\mathcal{A}_p$ is deterministic, and we say that $\mathcal{A}$ is *reduced* if every $\mathcal{A}_p$ is reduced, i.e., every state of S_p is reachable from the initial state $s_{0,p}$ and from every state of S_p a final state from F_p can be reached.

The infinite set $\mathbf{S}$ of *global states of* $\mathcal{A}$ and the set $\mathbf{F}$ of *final global states of* $\mathcal{A}$ are defined by

$$\mathbf{S} = \prod_{p\in P} S_p \times (\mathbb{C}^*)^{\mathrm{Ch}} \quad \text{and} \quad \mathbf{F} = \prod_{p\in P} F_p \times \{\mathcal{B}_\emptyset\}.$$

The *initial global state of* $\mathcal{A}$ is $(\mathbf{s}_0, \mathcal{B}_\emptyset)$, where $\mathbf{s}_0 = (s_{0,p})_{p\in P}$. The *global transition relation* $\delta \subseteq \mathbf{S}\times\Sigma\times\mathbf{S}$ of $\mathcal{A}$ is defined as follows: Let $(\mathbf{s}, \mathcal{B}) \in \mathbf{S}$, where $\mathbf{s} = (s_p)_{p\in P}$, and $i, j \in P$, $c \in \mathbb{C}$. Then,

- $(s_i,\ i!j(c),\ t) \in \delta_i$ implies

  $$((\mathbf{s}, \mathcal{B}),\ i!j(c),\ (\mathbf{t}, \mathcal{C})) \in \delta,$$

  where $\mathbf{t} = (t_p)_{p\in P}$, $t_p = s_p$ for $p \neq i$, $t_i = t$, $\mathcal{C}(p,q) = \mathcal{B}(p,q)$ for $(p,q) \neq (i,j)$, and $\mathcal{C}(i,j) = c\,\mathcal{B}(i,j)$, and
- $(s_i,\ i?j(c),\ t) \in \delta_i$ and $\mathcal{B}(j,i) = wc$ for some $w \in \mathbb{C}^*$ implies

  $$((\mathbf{s}, \mathcal{B}),\ i?j(c),\ (\mathbf{t}, \mathcal{C})) \in \delta,$$

  where $\mathbf{t} = (t_p)_{p\in P}$, $t_p = s_p$ for $p \neq i$, $t_i = t$, $\mathcal{C}(q,p) = \mathcal{B}(q,p)$ for $(q,p) \neq (j,i)$, and $\mathcal{C}(j,i) = w$.

We extend the relation $\delta \subseteq \mathbf{S}\times\Sigma\times\mathbf{S}$ in the usual way to a relation $\delta \subseteq \mathbf{S}\times\Sigma^*\times\mathbf{S}$. Instead of $((\mathbf{s}, \mathcal{B}), w, (\mathbf{t}, \mathcal{C})) \in \delta$, $w \in \Sigma^*$, we write $(\mathbf{s}, \mathcal{B}) \xrightarrow{w}_{\mathcal{A}} (\mathbf{t}, \mathcal{C})$. We write $(\mathbf{s}, \mathcal{B}) \xrightarrow{*}_{\mathcal{A}} (\mathbf{t}, \mathcal{C})$ if $(\mathbf{s}, \mathcal{B}) \xrightarrow{w}_{\mathcal{A}} (\mathbf{t}, \mathcal{C})$ for some $w \in \Sigma^*$. We write $(\mathbf{s}, \mathcal{B}) \xrightarrow{w}_{\mathcal{A}}$ if $(\mathbf{s}, \mathcal{B}) \xrightarrow{w}_{\mathcal{A}} (\mathbf{t}, \mathcal{C})$ for some $(\mathbf{t}, \mathcal{C})$. Let

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists (\mathbf{t}, \mathcal{B}_\emptyset) \in \mathbf{F} : (\mathbf{s}_0, \mathcal{B}_\emptyset) \xrightarrow{w}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}_\emptyset)\}.$$

It is easy to see that for every run $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{w}_{\mathcal{A}} (\mathbf{t}, \mathcal{B})$, $w \in \Sigma^*$, that starts with empty buffers, there exists a unique pMSC $\mathrm{pmsc}(w)$ with $w \in \mathrm{lin}(\mathrm{pmsc}(w))$. Furthermore, if also $\mathcal{B} = \mathcal{B}_\emptyset$ then $\mathrm{pmsc}(w) \in \mathbb{MSC}$ and we write $\mathrm{msc}(w)$ instead of $\mathrm{msc}(w)$. Thus we can define $\mathrm{msc}(\mathcal{A}) = \{\mathrm{msc}(w) \mid w \in L(\mathcal{A})\}$. Finally, we say that $\mathcal{A}$ is *deadlock-free* if for all $(\mathbf{s}, \mathcal{B})$ such that $(\mathbf{s}_0, \mathcal{B}_\emptyset) \xrightarrow{*}_{\mathcal{A}} (\mathbf{s}, \mathcal{B})$ we have $(\mathbf{s}, \mathcal{B}) \xrightarrow{*}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}_\emptyset)$ for some $(\mathbf{t}, \mathcal{B}_\emptyset) \in \mathbf{F}$.

If $w_1, w_2 \in \mathrm{lin}(N)$ for $N \in \mathrm{p}\mathbb{MSC}$ then $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{w_1}_{\mathcal{A}} (\mathbf{t}, \mathcal{B})$ if and only if $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{w_2}_{\mathcal{A}} (\mathbf{t}, \mathcal{B})$. Thus, we may write $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{N}_{\mathcal{A}} (\mathbf{t}, \mathcal{B})$ in this case. If moreover $M \leq N \in \mathrm{p}\mathbb{MSC}$ then there is a global state $(\mathbf{u}, \mathcal{C})$ of $\mathcal{A}$ such that for all $v \in \mathrm{lin}(M)$ and $w \in \mathrm{lin}(N \backslash M)$ we have $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{v}_{\mathcal{A}} (\mathbf{u}, \mathcal{C}) \xrightarrow{w}_{\mathcal{A}} (\mathbf{t}, \mathcal{B})$. Thus, we may write $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}} (\mathbf{u}, \mathcal{C}) \xrightarrow{N\backslash M}_{\mathcal{A}} (\mathbf{t}, \mathcal{B})$.

Lemma 2.3. *Let $\mathcal{A}$ be a deterministic CFM. Let $M, M_1, M_2 \in \mathrm{p}\mathbb{MSC}$ such that $\sup(M_1, M_2)$ exists and $M = \inf(M_1, M_2)$. If*

$$(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M_1}_{\mathcal{A}} (\mathbf{s}_1, \mathcal{B}_1) \quad \textit{and} \quad (\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M_2}_{\mathcal{A}} (\mathbf{s}_2, \mathcal{B}_2)$$

then there exists a global state $(\mathbf{t}, \mathcal{B})$ of $\mathcal{A}$ such that

$$(\mathbf{s}_1, \mathcal{B}_1) \xrightarrow{M_2 \setminus M}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}) \quad \textit{and} \quad (\mathbf{s}_2, \mathcal{B}_2) \xrightarrow{M_1 \setminus M}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}).$$

Proof. Note that the case $P(M_1) \cap P(M_2) = \emptyset$, i.e., $M = \emptyset$ is obvious. For the general case note that there exist global states $(\mathbf{t}_1, \mathcal{C}_1)$ and $(\mathbf{t}_2, \mathcal{C}_2)$ such that

$$(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}} (\mathbf{t}_1, \mathcal{C}_1) \xrightarrow{M_1 \setminus M}_{\mathcal{A}} (\mathbf{s}_1, \mathcal{B}_1) \quad \text{and} \quad (\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}} (\mathbf{t}_2, \mathcal{C}_2) \xrightarrow{M_2 \setminus M}_{\mathcal{A}} (\mathbf{s}_2, \mathcal{B}_2).$$

Since $\mathcal{A}$ is deterministic, we have $(\mathbf{t}_1, \mathcal{C}_1) = (\mathbf{t}_2, \mathcal{C}_2)$. By Lemma 2.2(2), we have $P(M_1 \setminus M) \cap P(M_2 \setminus M) = \emptyset$. Then

$$(\mathbf{s}_1, \mathcal{B}_1) \xrightarrow{M_2 \setminus M}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}) \quad \text{and} \quad (\mathbf{s}_2, \mathcal{B}_2) \xrightarrow{M_1 \setminus M}_{\mathcal{A}} (\mathbf{t}, \mathcal{B})$$

for some $(\mathbf{t}, \mathcal{B})$ follows immediately. □

3 Weak and safe realizability

Let $L \subseteq \mathbb{MSC}_{P,\mathbb{C}}$ be a set of MSCs. Following [1], we say that L is *weakly realizable* if there exists a CFM $\mathcal{A}$ over P and $\mathbb{C}$ such that $\mathrm{msc}(\mathcal{A}) = L$. We say that L is *safely realizable* if there exists a deadlock-free CFM $\mathcal{A}$ over P and $\mathbb{C}$ such that $\mathrm{msc}(\mathcal{A}) = L$. [1] An HMSC H is called *weakly realizable* (*safely realizable*) if $\mathrm{msc}(H)$ is weakly realizable (safely realizable).

In [3], weak and safe realizability was also characterized by the following two conditions for sets of MSCs. Let $L \subseteq \mathbb{MSC}$.

- *Closure condition* CC_w (called CC2 in [1]). If $M \in \mathbb{MSC}$ is such that for all $p \in P$ there exists $N \in L$ with $\pi_p(M) = \pi_p(N)$ then $M \in L$.
- *Closure condition* CC_s (called CC3 in [1]). If $M \in \mathrm{p}\mathbb{MSC}$ is such that for all $p \in P$ there exists $N \in L$ with $\pi_p(M) \sqsubseteq \pi_p(N)$ then $M \leq N$ for some $N \in L$.

Then the following holds.

Lemma 3.1 (cf [3]). *Let $L \subseteq \mathbb{MSC}$.*

- *L is weakly realizable if and only if L satisfies closure condition CC_w.*
- *L is safely realizable if and only if L satisfies closure condition CC_w and closure condition CC_s.*

[1] These definitions allow local automata $\mathcal{A}_p$ with infinite state sets, but this case will never occur in this paper, since we restrict to sets of MSCs generated by HMSCs.

For the above lemma it is important that every $M \in \mathbb{pMSC}$ can be uniquely reconstructed from its projections $\pi_p(M)$, $p \in P$, which is obvious due to the FIFO-restriction.

The original definition of weak (safe) realizability suggests that the main difficulty for checking weak (safe) realizability of an HMSC is that of finding a CFM that witness weak (safe) realizability. The following lemma shows that this is in fact not the case.

Lemma 3.2. *Let L be a set of MSCs.*

- *If $\mathcal{A} = (\mathcal{A}_p)_{p \in P}$ is a CFM such that $\pi_p(L) = L(\mathcal{A}_p)$ for every $p \in P$ then L is weakly realizable if and only if* $\mathrm{msc}(\mathcal{A}) = L$.
- *If $\mathcal{A} = (\mathcal{A}_p)_{p \in P}$ is a deterministic and reduced CFM such that $\pi_p(L) = L(\mathcal{A}_p)$ for every $p \in P$ then L is safely realizable if and only if $\mathcal{A}$ is deadlock-free and* $\mathrm{msc}(\mathcal{A}) = L$.

Proof. Note that one direction in each of the two statements is trivial. For the other direction, first assume that $\mathcal{A} = (\mathcal{A}_p)_{p \in P}$ is a CFM such that $\pi_p(L) = L(\mathcal{A}_p)$ for every $p \in P$ but $\mathrm{msc}(\mathcal{A}) \neq L$. Since clearly $L \subseteq \mathrm{msc}(\mathcal{A})$, there exists $M \in \mathrm{msc}(\mathcal{A}) \setminus L$. Thus, $\pi_p(M) \in \pi_p(L)$ for all $p \in P$. Lemma 3.1 implies that L is not weakly realizable.

For the second statement assume that $\mathcal{A} = (\mathcal{A}_p)_{p \in P}$ is a deterministic and reduced CFM such that $\pi_p(L) = L(\mathcal{A}_p)$ for every $p \in P$. If $\mathrm{msc}(\mathcal{A}) \neq L$ then by the previous paragraph, L is not weakly realizable and hence not safely realizable. If $\mathcal{A}$ is not deadlock-free then there exists a pMSC M and a global state $(\mathbf{s}, \mathcal{B})$ such that $(\mathbf{s}_0, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}} (\mathbf{s}, \mathcal{B})$ but there is no global final state that is reachable from $(\mathbf{s}, \mathcal{B})$. Since every local automaton $\mathcal{A}_p$ is reduced, there exist words $w_p \in \Sigma_p^*$ such that $\pi_p(M)w_p \in L(\mathcal{A}_p) = \pi_p(L)$ for every $p \in P$. Thus, for every $p \in P$ there exists $N \in L$ with $\pi_p(M) \sqsubseteq \pi_p(N)$. We claim that there does not exist $N \in L$ with $M \leq N$ (with Lemma 3.1 this shows that L is not safely realizable). In order to deduce a contradiction, assume that $M \leq N$ for some $N \in L$. Since $L \subseteq \mathrm{msc}(\mathcal{A})$, it follows that $(\mathbf{s}_0, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}} (\mathbf{s}', \mathcal{B}') \xrightarrow{N \setminus M}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}_\emptyset)$ for a global final state $(\mathbf{t}, \mathcal{B}_\emptyset)$. Since $\mathcal{A}$ is deterministic, we obtain $(\mathbf{s}', \mathcal{B}') = (\mathbf{s}, \mathcal{B})$, which contradicts the assumption that no global final state is reachable from $(\mathbf{s}, \mathcal{B})$. □

Note that for a given HMSC H it is easy to construct a CFM with the properties from Lemma 3.2.

As already mentioned, the notions of weak and safe realizability were introduced in [1], where it was shown that for finite sets of MSCs, safe realizability can be tested in polynomial time, whereas weak realizability is coNP-complete, see also [3]. In [2], realizability was studied for HMSCs. It was shown that weak realizability is already undecidable for bounded HMSCs if FIFO communication is assumed. Under non-FIFO communication, weak realizability is decidable for bounded HMSCs [19]. Safe realizability for bounded HMSCs was shown to be in EXPSPACE, but PSPACE-hard in [2]. In Section 3.1, we will close this gap by proving that safe realizability for bounded HMSCs is EXPSPACE-complete. The proof technique used for this result will be also used in order to prove that safe realizability is undecidable for the class of all HMSCs. Moreover, in Section 3.2 we will show that safe realizability remains EXPSPACE-complete for globally-cooperative HMSCs.

3.1 Lower bound proofs

Theorem 3.3. *The following problem is EXPSPACE-complete:*

INPUT: Set P of processes, set $\mathbb{C}$ of message contents, and a bounded HMSC H over P and $\mathbb{C}$

QUESTION: Is H safely realizable?

Furthermore this problem is already EXPSPACE-complete for some fixed P and $\mathbb{C}$ (i.e., they do not belong to the input).

Proof. Membership in EXPSPACE is shown in [2] (for variable P and $\mathbb{C}$), or follows from Theorem 3.7. For the lower bound we combine ideas from [2] and [21, 23]. Let $\mathcal{M}$ be a fixed Turing-machine with an EXPSPACE-complete acceptance problem (such a machine exists, take any machine, which accepts an EXPSPACE-complete language). W.l.o.g. $\mathcal{M}$ works on an input of length n in space $2^n - 1$. Let Q be the set of states of $\mathcal{M}$ and let Δ be the tape alphabet. Furthermore, let q_0 be the initial state of $\mathcal{M}$ and q_f be the final state of $\mathcal{M}$. Let $\Box \in \Delta$ be the blank symbol. The machine $\mathcal{M}$ accepts if it reaches the final state q_f. Let us fix an input $w \in \Delta^*$ for $\mathcal{M}$ with $|w| = n$ for the further discussion. Configurations of $\mathcal{M}$ are represented as a word from $\Delta^* Q \Delta^*$ of length 2^n. A sequence $(u_1, \ldots, u_m)$ of words $u_i \in \Delta^* Q \Delta^*$ is called an *accepting computation* of $\mathcal{M}$ if $u_1 = q_0 w \Box^{2^n - n - 1}$, $|u_i| = 2^n$ $(1 \leq i \leq m)$, u_{i+1} is a successor configuration of u_i with respect to $\mathcal{M}$ $(1 \leq i < m)$, and $u_m \in \Delta^* q_f \Delta^*$.

For a number $0 \leq i < 2^n$ let $\langle i \rangle \in \{0,1\}^n$ denote the binary representation of i of length n, where moreover the least significant bit is the left-most bit. For $w = a_0 \cdots a_{2^n-1}$, $a_i \in Q \cup \Delta$, let $\beta(w) = \langle 0 \rangle a_0 \cdots \langle 2^n - 1 \rangle a_{2^n-1}$. Let $\Gamma = Q \cup \Delta \cup \{0,1\}$ and define the set $\mathbb{C}$ of message contents by $\mathbb{C} = \Gamma \cup \{\$, \ell, r\}$. [2] We will deal with the fixed set of processes $P = \{1, \ldots, 5\}$. For a symbol $a \in \Gamma$ we define the MSC $a^{(2,1)}$ (resp. $a^{(4,5)}$) over P and $\mathbb{C}$ as the unique MSC with the only linearization $2!1(a)\, 1?2(a)\, 1!2\, 2?1$ (resp. $4!5(a)\, 5?4(a)\, 5!4\, 4?5$); thus, the symbol a is send from 2 to 1 (resp. 4 to 5) and immediately confirmed. For $C = b_1 \cdots b_m \in \Gamma^*$ define the MSCs $C^{(2,1)} = b_1^{(2,1)} \cdots b_m^{(2,1)}$ and $C^{(4,5)} = b_1^{(4,5)} \cdots b_m^{(4,5)}$. For words $C_1, D_1, \ldots, C_m, D_m \in \Gamma^*$ $(m \geq 1)$ we define the MSC $M(C_1, D_1, \ldots, C_m, D_m)$ over P and $\mathbb{C}$ as shown in Figure 1, where the case $m = 3$ is shown. Finally define the following two sets of MSCs:

[2] In the following, we will also use messages without any content, the corresponding types are written as $p!q$ and $p?q$, respectively. Formally, one can introduce an additional message content nil for these messages.

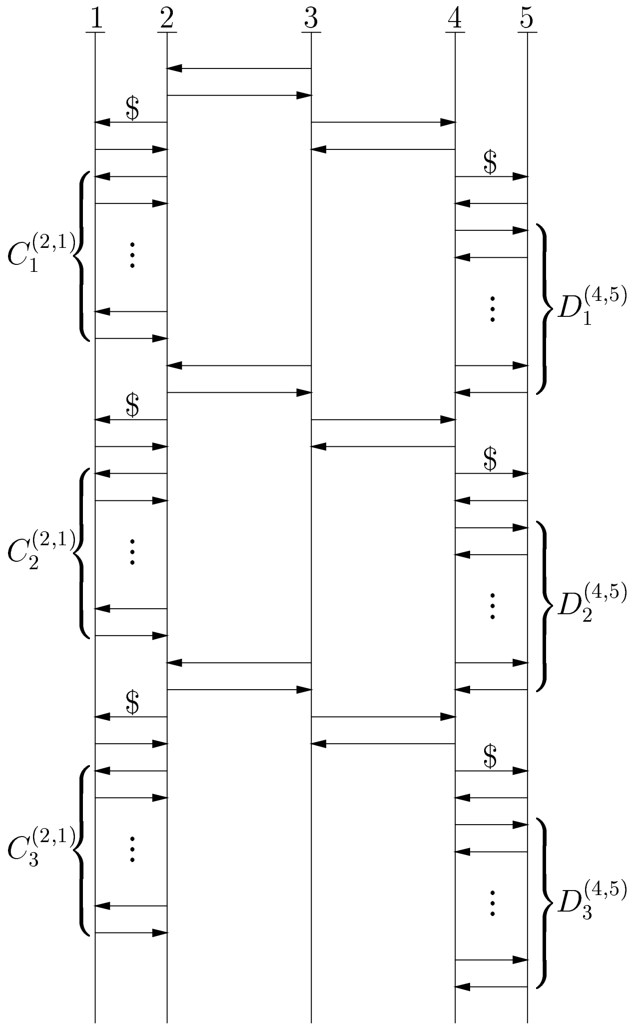


Fig. 1. $M(C_1, D_1, C_2, D_2, C_3, D_3)$

$$
\begin{aligned}
L_\ell &= \{M(C_1, D_1, \ldots, C_m, D_m) \mid m \geq 1,\ C_1, D_1, \ldots, C_m, D_m \in \Gamma^*\} \\
L_r &= L_\ell \backslash \{M(\beta(u_1), \beta(u_1), \ldots, \beta(u_m), \beta(u_m)) \mid (u_1, \ldots, u_m) \text{ is an} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{accepting computation of } \mathcal{M}\}
\end{aligned}
$$

Claim 1. There exist bounded HMSCs H_ℓ and H_r that can be constructed in time polynomial in $n = |w|$ and such that $\text{msc}(H_\ell) = L_\ell$ and $\text{msc}(H_r) = L_r$.

For L_ℓ this is clear, since all messages are immediately confirmed by messages back to the sending process. For L_r we can reuse the construction from the proof of [21, Prop. 7]. For completeness, a brief exposition follows. The set L_r contains all MSCs in L_ℓ that do *not* represent accepting computations of $\mathcal{M}$ starting on input w. Thus, $L_r = \bigcup_{i=1}^{6} L_{r,i}$, where $M(C_1, D_1, \ldots, C_m, D_m) \in L_\ell$ belongs to

– $L_{r,1}$ if some C_k or D_k is not contained in $(\{0,1\}^n \Delta)^* \{0,1\}^n Q (\{0,1\}^n \Delta)^*$.

- $L_{r,2}$ if some C_k or D_k is not contained in $0^n(Q \cup \Delta)\Gamma^* \cap \Gamma^* 1^n \Delta$.
- $L_{r,3}$ if some C_k or D_k contains a factor $\langle i \rangle a \langle j \rangle b$ with $a, b \in Q \cup \Delta$, but $j \neq i+1$.
- $L_{r,4}$ if C_1 does not belong to $\{0,1\}^* q_0 \{0,1\}^* a_1 \cdots \{0,1\}^* a_n (\{0,1\}^* \square)^*$, where $a_1 \cdots a_n = w$, or q_f does not occur in C_m.
- $L_{r,5}$ if for some k and i, C_k contains a factor $\langle i \rangle a$ and D_k contains a factor $\langle i \rangle b$, where $a, b \in Q \cup \Delta$ but $a \neq b$ (i.e., $C_k \neq D_k$).
- $L_{r,6}$ if for some k and i, D_k contains a factor $\langle i \rangle a_1 s b_1 t c_1$, C_{k+1} contains a factor $\langle i \rangle a_2 u b_2 v c_2$, where $s, t, u, v \in \{0,1\}^*$, $a_j, b_j, c_j \in Q \cup \Delta$, but there do not exist w_1, w_2 such that $w_1 a_1 b_1 c_1 w_2 \vdash_{\mathcal{M}} w_1 a_2 b_2 c_2 w_2$. Note that this is local condition on the tuple $(a_1, b_1, c_1, a_2, b_2, c_2)$.

The conditions describing $L_{r,1}$, $L_{r,2}$, $L_{r,3}$, and $L_{r,4}$ can be enforced by finite automata, which can be transformed into bounded HMSCs that operate only on the processes 1 and 2 (resp. 4 and 5). The set $L_{r,3}$ can be written as a union $\bigcup_{i=0}^{n-1} A_i \cup B_i$ where $M(C_1, D_1, \ldots, C_m, D_m)$ belongs to:

- A_i if some C_k or D_k contains a factor in $1^i \alpha \{0,1\}^{n-i-1} a \{0,1\}^i \alpha \{0,1\}^{n-i-1} b$ with $a, b \in Q \cup \Delta$ and $\alpha \in \{0,1\}$.
- B_i if some C_k or D_k contains a factor in $v \alpha \{0,1\}^{n-i-1} a \{0,1\}^i \beta \{0,1\}^{n-i-1} b$ with $a, b \in Q \cup \Delta$, $v \in \{0,1\}^i \setminus \{1^i\}$, $\alpha, \beta \in \{0,1\}$, and $\alpha \neq \beta$.

In order to generate $L_{r,5}$ and $L_{r,6}$, it is crucial that for every i, the events belonging to $C_i^{(2,1)}$ (resp. $D_i^{(4,5)}$) are causally independent from those in $D_i^{(4,5)}$ (resp. $C_{i+1}^{(2,1)}$). Thanks to the counter, we do not need concurrent iteration (i.e., loops labeled by MSCs with a non-connected communication graph). For $L_{r,5}$ for instance, we simply guess independently two positions in C_k and D_k, respectively, where $\langle i \rangle a$ and $\langle j \rangle b$, respectively, starts and verify whether $i = j$ and $a \neq b$ holds. Since the binary codings of i and j are of polynomial length, the test whether $i = j$ can be done without looping in the HMSC. Finally, note that all constructions can be done in time bounded polynomially in n. This concludes the outline of the proof of Claim 1.

Claim 2. L_ℓ is safely realizable.

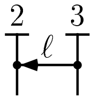
By Lemma 3.1 it suffices to verify condition CC_w and CC_s for L_ℓ. We will only check CC_w, condition CC_s can be verified analogously. Thus assume that M is an MSC such that for each $p \in \{1, \ldots, 5\}$ there exists $N \in L_\ell$ with $\pi_p(M) = \pi_p(N)$.

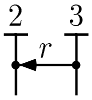
Thus $\pi_3(M) = (3!2\ 3?2\ 3!4\ 3?4)^k$ for some $k \geq 1$. Since M is an MSC, we have

$$\begin{aligned}
\pi_2(M) =& (2?3\ 2!3\ 2!1(\$)\ 2?1\ 2!1(a_{1,1})\ 2?1 \cdots 2!1(a_{1,i_1})\ 2?1) \cdots \\
& (2?3\ 2!3\ 2!1(\$)\ 2?1\ 2!1(a_{k,1})\ 2?1 \cdots 2!1(a_{k,i_k})\ 2?1) \\
\pi_4(M) =& (4?3\ 4!3\ 4!5(\$)\ 4?5\ 4!5(b_{1,1})\ 4?5 \cdots 4!5(b_{1,j_1})\ 4?5) \cdots \\
& (4?3\ 4!3\ 4!5(\$)\ 4?5\ 4!5(b_{k,1})\ 4?5 \cdots 4!5(b_{k,j_k})\ 4?5) \\
\pi_1(M) =& (1?2(\$)\ 1!2\ 1?2(a_{1,1})\ 1!2 \cdots 1?2(a_{1,i_1})\ 1!2) \cdots \\
& (1?2(\$)\ 1!2\ 1?2(a_{k,1})\ 1!2 \cdots 1?2(a_{k,i_k})\ 1!2) \\
\pi_5(M) =& (5?4(\$)\ 5!4\ 5?4(b_{1,1})\ 5!4 \cdots 5?4(b_{1,j_1})\ 5!4) \cdots \\
& (5?4(\$)\ 5!4\ 5?4(b_{k,1})\ 5!4 \cdots 5?4(b_{k,j_k})\ 5!4)
\end{aligned}$$

for some $i_1, j_1, \ldots, i_k, j_k \geq 0$. Thus $M \in L_\ell$. This proves Claim 2.

Now define the MSCs M_ℓ and M_r by

$M_\ell =$ 

$M_r =$ 

From the bounded HMSCs H_ℓ and H_r in Claim 1 it is straight-forward to construct a bounded HMSC H such that $\mathrm{msc}(H) = (M_\ell \cdot L_\ell) \cup (M_r \cdot L_r)$, where concatenation is lifted to sets of MSCs in the obvious way.

Claim 3. If $\mathcal{M}$ does not accept w then H is safely realizable: Note that if $\mathcal{M}$ does not accept w, then $L_\ell = L_r$ and $\mathrm{msc}(H) = \{M_\ell, M_r\} \cdot L_\ell$. Since L_ℓ is safely realizable by Claim 2, also $\mathrm{msc}(H)$ is safely realizable.

Claim 4. If $\mathcal{M}$ accepts w then H is not weakly realizable (and hence not safely realizable): Let $(u_1, \ldots, u_m)$ be an accepting computation of $\mathcal{M}$. Let

$$M = M(\beta(u_1), \beta(u_1), \beta(u_2), \beta(u_2), \ldots, \beta(u_m), \beta(u_m)).$$

Since $M \notin L_r$, we have $M_r \cdot M \notin \mathrm{msc}(H)$. On the other hand for all $p \in \{1, \ldots, 5\}$ there exists $N \in \mathrm{msc}(H)$ such that $\pi_p(M_r \cdot M) = \pi_p(N)$, for instance for $p \in \{1, 2, 3\}$ take $N = M_r \cdot M(\beta(u_1), C, \beta(u_2), \beta(u_2), \ldots, \beta(u_m), \beta(u_m))$ for some $C \neq \beta(u_1)$. Thus, $\mathrm{msc}(H)$ is not weakly realizable. This proves Claim 4.

Thus, by Claim 3 and Claim 4, our fixed machine $\mathcal{M}$ accepts the input w if and only if H is not safely realizable. Since the acceptance problem of $\mathcal{M}$ is EXPSPACE-complete (and EXPSPACE is by Savitch's Theorem closed under complement [22]), the theorem follows. □

Theorem 3.4. *There exist fixed sets P and $\mathbb{C}$ of processes and message contents, respectively, such that the following problem is undecidable:*
INPUT: An HMSC H over P and $\mathbb{C}$
QUESTION: Is H safely realizable?

Proof. Basically we redo the construction from the proof of Theorem 3.3. But instead of a Turing-machine with an EXPSPACE-complete acceptance problem, we use a machine $\mathcal{M}$ with an undecidable acceptance problem. Counters, as used in the proof of Theorem 3.3, are not necessary this time (and in fact cannot be used, since configurations may become arbitrarily long). Thus we redefine $\Gamma = Q \cup \Delta$ and

$$L_r = L_\ell \backslash \{M(u_1, u_1, \ldots, u_m, u_m) \mid \begin{array}{l} u_i \in \Delta^* Q \Delta^*, \ (1 \leq i \leq m) \\ u_i \vdash_{\mathcal{M}} u_{i+1} \ (1 \leq i < m) \\ u_1 = q_0 w, \ u_m \in \Delta^* q_f \Delta^* \} \end{array}$$

where w is a given input for $\mathcal{M}$. The set L_r can be generated by an (unbounded) HMSC using loops labeled with the non-connected MSCs $a^{(2,1)} \cdot a^{(4,5)}$ for $a \in \Gamma$. The rest of the construction is completely analogous to the proof of Theorem 3.3. We obtain an HMSC H such that the following holds:

– If $\mathcal{M}$ does not accept w then H is safely realizable.
– If $\mathcal{M}$ accepts w then H is not weakly realizable. □

3.2 Upper bounds for globally-cooperative HMSCs

In [19] it was shown that weak realizability is decidable for globally-cooperative HMSCs (called c-HMSCs in [19]) if non-FIFO communication is supposed. Moreover, it was argued that the methods used in the proof of this result can be also used in order to prove that safe realizability is decidable for globally-cooperative HMSCs, both for FIFO and non-FIFO communication. In this section, we prove that safe realizability is in fact EXPSPACE-complete for globally-cooperative HMSCs. Since EXPSPACE-hardness follows from Theorem 3.3, it remains to prove membership in EXPSPACE. It should be noted that the technique from [2] for proving that safe realizability is in EXPSPACE for bounded HMSCs cannot be applied to globally-cooperative HMSCs: The proof in [2] is based on the fact that the set of all linearizations of MSCs from $\mathrm{msc}(H)$ is a regular set in case H is bounded. But for globally-cooperative HMSCs this is no longer the case, see e.g. the example at the end of Section 2.1.

For the further discussion let us fix an arbitrary HMSC $H = (V, \rightarrow, v_0, F)$ over P and $\mathbb{C}$. For the main part of this section, we do not assume that H is globally-cooperative. Recall that $\mathbb{A}_H = \{A \in \mathbb{A} \mid \exists u, v \in V : u \xrightarrow{A}_H v\}$. With $\langle \mathbb{A}_H \rangle$ we denote the set of all MSCs of the form $A_1 \cdot A_2 \cdots A_n$ with $A_i \in \mathbb{A}_H$ (possibly $n = 0$, i.e., $\emptyset \in \langle \mathbb{A}_H \rangle$).

For every $p \in P$ we can easily construct in polynomial time from H a nondeterministic finite state automaton $\mathcal{A}'_p$ with $L(\mathcal{A}'_p) = \pi_p(\mathrm{msc}(H))$. Let Q_p be the set of states of $\mathcal{A}'_p$. Thus, the size of Q_p is bounded polynomially in the size of H. Using the powerset construction, we can build a deterministic and reduced automaton $\mathcal{A}_p = (S_p, \Sigma_p, \delta_p, s_{0,p}, F_p)$ such that $S_p \subseteq 2^{Q_p}$ and

$L(\mathcal{A}_p) = L(\mathcal{A}'_p) = \pi_p(\mathrm{msc}(H))$. We call the CFM $\mathcal{A} = (\mathcal{A}_p)_{p\in P}$ the *canonical implementation* of H. By Lemma 3.2, H is safely realizable if and only if $\mathcal{A}$ is deadlock-free and $\mathrm{msc}(\mathcal{A}) = \mathrm{msc}(H)$. Our main tool for checking the latter two conditions will be a finite state automaton $\mathcal{A}_\emptyset$, whose definition is inspired by [19]: $\mathcal{A}_\emptyset = (\mathbf{S}_\emptyset, \mathbb{A}_H, \delta_\emptyset, \mathbf{s}_0, \mathbf{F}_\emptyset)$ is a finite state automaton over the alphabet of atoms $\mathbb{A}_H$, where $\mathbf{s}_0 = (s_{0,p})_{p\in P}$ is the initial state, $\mathbf{S}_\emptyset \subseteq \prod_{p\in P} S_p$ is the set of all tuples $\mathbf{s}$ such that there exists $K \in \langle \mathbb{A}_H \rangle$ with $(\mathbf{s}_0, \mathcal{B}_\emptyset) \xrightarrow{K}_{\mathcal{A}} (\mathbf{s}, \mathcal{B}_\emptyset)$, $\mathbf{F}_\emptyset = \mathbf{S}_\emptyset \cap \prod_{p\in P} F_p$, and the transition relation $\delta_\emptyset$ is defined as follows: If $\mathbf{s}, \mathbf{t} \in \mathbf{S}_\emptyset$ and $A \in \mathbb{A}_H$ then $(\mathbf{s}, A, \mathbf{t}) \in \delta_\emptyset$ if and only if $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{A}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}_\emptyset)$. Notations like $\mathbf{s} \xrightarrow{A}_{\mathcal{A}_\emptyset} \mathbf{t}$ are defined as for CFMs in Section 2.2. Note that $\mathcal{A}_\emptyset$ is $\mathcal{I}$-closed, i.e., if $u \in L(\mathcal{A}_\emptyset)$ and $u \equiv_{\mathcal{I}} v$ for words $u, v \in \mathbb{A}_H^*$ then also $v \in L(\mathcal{A}_\emptyset)$, in fact, $\mathcal{A}_\emptyset$ is an asynchronous automaton in the sense of [24]. Thus, by Lemma 2.1, for $K \in \langle \mathbb{A}_H \rangle$ and $\mathbf{s}, \mathbf{t} \in \mathbf{S}_\emptyset$ we can write $\mathbf{s} \xrightarrow{K}_{\mathcal{A}_\emptyset} \mathbf{t}$ with the obvious meaning. We write $\mathbf{s} \xrightarrow{K}_{\mathcal{A}_\emptyset}$ if $\mathbf{s} \xrightarrow{K}_{\mathcal{A}_\emptyset} \mathbf{t}$ for some $\mathbf{t}$.

Note that the number of states of $\mathcal{A}_\emptyset$ is bounded by $\prod_{p\in P} S_p \leq 2^{\sum_{p\in P}|Q_p|}$, which is exponential in the size of the HMSC H. Four our purpose this size bound will be too large. But note that in order to write down a state of $\mathcal{A}_\emptyset$ we only need polynomial space.

The main part of this section is devoted to the proof of the following result:

Theorem 3.5. *The following problem is in PSPACE:*

INPUT: Set P of processes, set $\mathbb{C}$ of message contents, and an arbitrary HMSC H over P and $\mathbb{C}$

QUESTION: Does the canonical implementation $\mathcal{A}$ of H satisfy the following two properties: (i) $\mathcal{A}$ is deadlock-free and (ii) $\mathrm{msc}(\mathcal{A}) \subseteq \langle \mathbb{A}_H \rangle$ *?*

Before we go into the details of the proof of Theorem 3.5 let us first deduce a few consequences.

Theorem 3.6. *The following problem is PSPACE-complete:*

INPUT: Set P of processes, set $\mathbb{C}$ of message contents, and an $\mathcal{I}$-closed HMSC H over P and $\mathbb{C}$

QUESTION: Is H safely realizable?

Furthermore this problem is already PSPACE-complete for some fixed P and $\mathbb{C}$.

Proof. For PSPACE-hardness we can use the construction from the proof of [2, Thm. 3]. In fact, the HMSC H, constructed there, satisfies the property that $u \xrightarrow{A}_H v \xrightarrow{B}_H w$ implies $P(A) \cap P(B) \neq \emptyset$, thus H is $\mathcal{I}$-closed. Moreover, P and $\mathbb{C}$ are fixed in the construction. Hence, it remains to show membership in PSPACE. We first verify whether the canonical implementation $\mathcal{A}$ of H is both deadlock-free and satisfies $\mathrm{msc}(\mathcal{A}) \subseteq \langle \mathbb{A}_H \rangle$. If this is not the case then we can reject. By Theorem 3.5 this test can be done in polynomial space. Thus, let us assume that $\mathcal{A}$ is deadlock-free and $\mathrm{msc}(\mathcal{A}) \subseteq \langle \mathbb{A}_H \rangle$. It remains to show that $\mathrm{msc}(\mathcal{A}) = \mathrm{msc}(H)$, where the inclusion $\mathrm{msc}(H) \subseteq \mathrm{msc}(\mathcal{A})$ is trivial. Thus, we have to check whether $\mathrm{msc}(\mathcal{A}) \subseteq \mathrm{msc}(H)$. Since we already know that $\mathrm{msc}(\mathcal{A}) \subseteq \langle \mathbb{A}_H \rangle$, this is equivalent to $\mathrm{msc}(\mathcal{A}) \cap \langle \mathbb{A}_H \rangle \subseteq \mathrm{msc}(H)$. The following argument follows [19]. First note that for all $A_1, \ldots, A_m \in \mathbb{A}_H$, we have $A_1 \cdot A_2 \cdots A_m \in \mathrm{msc}(\mathcal{A})$ if and only if the *word* $A_1 A_2 \cdots A_m \in \mathbb{A}_H^*$ belongs to $L(\mathcal{A}_\emptyset)$. Hence, we have $\mathrm{msc}(\mathcal{A}) \cap \langle \mathbb{A}_H \rangle \subseteq \mathrm{msc}(H)$ if and only if $L(\mathcal{A}_\emptyset) \subseteq [L(H)]_{\mathcal{I}}$ (where H is viewed as a finite automaton over the alphabet $\mathbb{A}_H$) if and only if $L(\mathcal{A}_\emptyset) \subseteq L(H)$ (H is $\mathcal{I}$-closed) if and only if $L(\mathcal{A}_\emptyset) \cap (\mathbb{A}_H^* \backslash L(H)) = \emptyset$. This can be checked in polynomial space, by guessing a word in the intersection and storing only the current state of $\mathcal{A}_\emptyset$ (which is possible in polynomial space) and the current state of the automaton for $\mathbb{A}_H^* \backslash L(H)$ resulting from the subset construction. The latter is a subset of the set of nodes of H, hence it only needs polynomial space. $\square$

Theorem 3.7. *The following problem is EXPSPACE-complete:*

INPUT: Set P of processes, set $\mathbb{C}$ of message contents, and a globally-cooperative HMSC H over P and $\mathbb{C}$

QUESTION: Is H safely realizable?

Furthermore this problem is already EXPSPACE-complete for some fixed P and $\mathbb{C}$.

Proof. The lower bound follows from Theorem 3.3. For the upper bound we can argue as follows: For a globally-cooperative HMSC H we can by [21] construct an $\mathcal{I}$-closed HMSC H' of size bounded exponentially in the size of H such that $\mathrm{msc}(H) = \mathrm{msc}(H')$. By Theorem 3.6 we can check in space bounded polynomially in the size of H' (and thus space bounded exponentially in the size of H) whether H' and hence H is safely realizable. $\square$

The rest of this section is devoted to a proof of Theorem 3.5. Recall that we want to check whether $\mathcal{A}$ is deadlock-free and $\mathrm{msc}(\mathcal{A}) \subseteq \langle \mathbb{A}_H \rangle$. A first simplification is achieved by the following lemma.

Lemma 3.8. *The following two statements are equivalent:*

(a) $\mathcal{A}$ is deadlock-free and $\mathrm{msc}(\mathcal{A}) \subseteq \langle \mathbb{A}_H \rangle$.

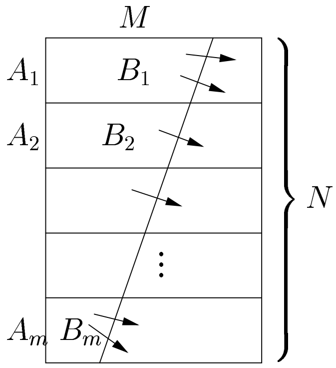
(b) $\mathcal{A}_\emptyset$ is deadlock-free and for all $\mathbf{s} \in \mathbf{S}_\emptyset$ and all $M \in \mathrm{p}\mathbb{MSC} \backslash \{\emptyset\}$ such that $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}}$ it holds

$$\exists K \in \langle \mathbb{A}_H \rangle \; \exists A \in \mathbb{A}_H \left\{ \begin{array}{l} \mathbf{s} \xrightarrow{K \cdot A}_{\mathcal{A}_\emptyset}, \; P(K) \cap P(M) = \emptyset, \\ \sup(A, M) \textit{ exists and, } \inf(A, M) \neq \emptyset \end{array} \right\}. \qquad (1)$$

Proof. First assume that (a) holds but $\mathcal{A}_\emptyset$ has a deadlock. Thus there exists a run $\mathbf{s}_0 \xrightarrow{M}_{\mathcal{A}_\emptyset} \mathbf{s}$ such that no final state of $\mathcal{A}_\emptyset$ can be reached from $\mathbf{s}$. Thus $(\mathbf{s}_0, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}} (\mathbf{s}, \mathcal{B}_\emptyset)$. Note that $M \in \langle \mathbb{A}_H \rangle$. Since by assumption $\mathcal{A}$ is deadlock-free, there exists $N \in \mathbb{MSC}$ and a final state $(\mathbf{t}, \mathcal{B}_\emptyset)$ of $\mathcal{A}$ with $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{N}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}_\emptyset)$.

Hence $M \cdot N \in \mathrm{msc}(\mathcal{A})$ and thus, by assumption, $M \cdot N \in \langle \mathbb{A}_H \rangle$, i.e., $N \in \langle \mathbb{A}_H \rangle$. It follows $\mathbf{s} \xrightarrow{N}_{\mathcal{A}_\emptyset} \mathbf{t} \in \mathbf{F}_\emptyset$, which is a contradiction.

Now assume (a) and let $\mathbf{s} \in \mathbf{S}_\emptyset$, $M \in \mathrm{p}\mathbb{MSC} \backslash \{\emptyset\}$ such that $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}}$. Since $\mathbf{s} \in \mathbf{S}_\emptyset$, the state $(\mathbf{s}, \mathcal{B}_\emptyset)$ is reachable in $\mathcal{A}$ from its initial state. Since $\mathcal{A}$ is deadlock-free, there exists $N \in \mathbb{MSC}$ such that $M \leq N$ and $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{N}_{\mathcal{A}} (\mathbf{t}, \mathcal{B}_\emptyset)$ for some final state $(\mathbf{t}, \mathcal{B}_\emptyset)$ of $\mathcal{A}$. Since $\mathrm{msc}(\mathcal{A}) \subseteq \langle \mathbb{A}_H \rangle$ we have $M \leq N = A_1 \cdot A_2 \cdots A_m$ for $A_1, \ldots, A_m \in \mathbb{A}_H$. Define $B_i = A_i{\restriction}_{E(M)}$. The following diagram visualizes the situation.



Since M is downward-closed in N, B_i must be downward-closed in A_i, i.e., $B_i \leq A_i$. Moreover, $P(A_i \backslash B_i) \cap P(B_j) = \emptyset$ for $i < j$: If e would be an event of $A_i \backslash B_i$ on process p and f would be an event of B_j on process p, then either $e \prec f$ (which is not possible, since e belongs to $N \backslash M$ and f belongs to M) or $f \prec e$ (which is not possible, since e belongs to A_i, f belongs to A_j, and $i < j$). Thus, if there is an unmatched send from p to q in B_i, then, since the corresponding receive belongs to $A_i \setminus B_i$, there cannot exist a message from p to q in some B_j with $j > i$. It follows that the concatenation $B_1 \cdot B_2 \cdots B_m$ is well defined and in fact $M = B_1 \cdot B_2 \cdots B_m$. Let $k \geq 1$ be minimal such that $B_k \neq \emptyset$, thus $B_1, \ldots, B_{k-1} = \emptyset$ and $M = B_k \cdots B_m$. Since $M \neq \emptyset$, such a k must exist. Since $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{N}_{\mathcal{A}}$, we have $\mathbf{s} \xrightarrow{A_1 \cdots A_{k-1} \cdot A_k}_{\mathcal{A}_\emptyset}$. Moreover, $P(A_1 \cdots A_{k-1}) \cap P(M) = P((A_1 \setminus B_1) \cdots (A_{k-1} \setminus B_{k-1})) \cap P(B_k \cdots B_m) = \emptyset$. Since both $A_k \leq A_k \cdots A_m$ and $M = B_k \cdots B_m \leq A_k \cdots A_m$, $\sup(A_k, M)$ exists. Finally, $B_k \neq \emptyset$ satisfies $B_k \leq A_k$ and $B_k \leq B_k \cdots B_m = M$. Thus $\inf(A_k, M) \neq \emptyset$ and (1) holds with $K = A_1 \cdots A_{k-1}$ and $A = A_k$. This concludes the proof of $(a) \Rightarrow (b)$.

It remains to prove $(b) \Rightarrow (a)$. We will show that $\neg(a)$ implies $\neg(b)$. Let us first assume that $\mathcal{A}$ is not deadlock-free, but $\mathcal{A}_\emptyset$ is deadlock-free. We have to show that (1) is false for some $\mathbf{s} \in \mathbf{S}_\emptyset$ and $M \neq \emptyset$ with $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}}$. Choose a pair $(\mathbf{s}, M) \in \mathbf{S}_\emptyset \times \mathrm{p}\mathbb{MSC}$ such that $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}} (\mathbf{t}, \mathcal{B})$, where $(\mathbf{t}, \mathcal{B})$ is a deadlock-state of $\mathcal{A}$, i.e., no final state of $\mathcal{A}$ can be reached from $(\mathbf{t}, \mathcal{B})$, and moreover $|M|$ is minimal among all pairs with this property. By assumption $\mathbf{s}$ and M exist. Since $\mathcal{A}_\emptyset$ is assumed to be deadlock-free, we must have $M \neq \emptyset$. We show that (1) does not hold for $\mathbf{s}$ and M. Assume the contrary, thus there are $K \in \langle \mathbb{A}_H \rangle$ and

$A \in \mathbb{A}_H$ such that $\mathbf{s} \xrightarrow{K \cdot A}_{\mathcal{A}_\emptyset} \mathbf{s}' \in \mathbf{S}_\emptyset$, $P(K) \cap P(M) = \emptyset$, $\sup(A, M)$ exists, and $B = \inf(A, M) \neq \emptyset$. First, since $A \in \mathbb{A}_H$ is an MSC, Lemma 2.2(5) implies that $M \backslash B$ is a pMSC. Moreover, by Lemma 2.3, $\mathcal{A}$ has the following runs.

$$\begin{tikzcd}
 & & (\mathbf{s}', \mathcal{B}_\emptyset) \arrow[dr, "M\backslash B"] & \\
 & \bullet \arrow[ur, "A"] \arrow[dr, "M"] & & (\mathbf{t}', \mathcal{B}') \\
(\mathbf{s}, \mathcal{B}_\emptyset) \arrow[ur, "K"] \arrow[dr, "M"'] & & \bullet \arrow[ur, "A\backslash B"] & \\
 & (\mathbf{t}, \mathcal{B}) \arrow[ur, "K"'] & &
\end{tikzcd}$$

Since $(\mathbf{t}, \mathcal{B})$ is a deadlock-state of $\mathcal{A}$, also $(\mathbf{t}', \mathcal{B}')$ is a deadlock-state of $\mathcal{A}$. Furthermore, since $B \neq \emptyset$ we have $|M \backslash B| < |M|$, a contradiction to the minimality of M.

Finally let us assume that $\mathrm{msc}(\mathcal{A}) \not\subseteq \langle \mathbb{A}_H \rangle$. Take $N \in \mathrm{msc}(\mathcal{A}) \backslash \langle \mathbb{A}_H \rangle$. Let $N = B_1 \cdot B_2 \cdots B_m$ be the decomposition of N into atoms. Since $N \notin \langle \mathbb{A}_H \rangle$, there exists j such that $B_1, \ldots, B_{j-1} \in \mathbb{A}_H$ but $B_j \notin \mathbb{A}_H$. Since $B_1 \cdot B_2 \cdots B_m \in \mathrm{msc}(\mathcal{A})$ we find $\mathbf{s} \in \mathbf{S}_\emptyset$ with $\mathbf{s}_0 \xrightarrow{B_1 \cdots B_{j-1}}_{\mathcal{A}_\emptyset} \mathbf{s}$ and $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{B_j}_{\mathcal{A}}$. We show that (1) is not satisfied for $\mathbf{s}$ and $M = B_j$. Assume the contrary. Thus there exists $A \in \mathbb{A}_H$ such that (among other properties) $\sup(A, B_j)$ exists and $\inf(A, B_j) \neq \emptyset$. Since A and B_j are atoms, Lemma 2.2(6) implies that $B_j = A \in \mathbb{A}_H$, a contradiction. This proves the lemma. $\square$

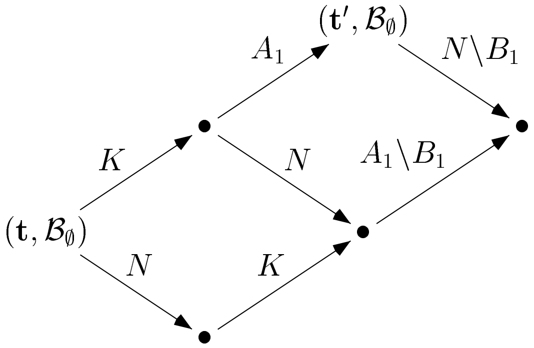
Recall that we want to check property (a) from Lemma 3.8 in PSPACE. Since PSPACE is closed under complement [22], it suffices to check $\neg(a)$ in PSPACE. Instead of $\neg(a)$, we will verify property $\neg(b)$ from Lemma 3.8 in PSPACE. Whether $\mathcal{A}_\emptyset$ has a deadlock can be easily verified in PSPACE, since states of $\mathcal{A}_\emptyset$ can be stored in polynomial space. Basically, the second alternative from $\neg(b)$ will be verified by guessing $\mathbf{s} \in \mathbf{S}_\emptyset$ and $M \in \mathrm{p}\mathbb{MSC} \backslash \{\emptyset\}$ such that $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}}$ but (1) from Lemma 3.8 is not satisfied for $\mathbf{s}$ and M. Here, another problem arises. Whereas a state $\mathbf{s} \in \mathbf{S}_\emptyset$ can be easily guessed in PSPACE, there is a priori no size bound for the pMSC M. Thus, our next goal is to bound the size of a witness M for $\neg(b)$ in Lemma 3.8 (later, we will see that we do not have to give a bound on the size of the MSC K in (1) from Lemma 3.8).

For the further consideration, let us fix some witnesses $\mathbf{s} \in \mathbf{S}_\emptyset$ and $M \in \mathrm{p}\mathbb{MSC} \backslash \{\emptyset\}$ for $\neg(b)$ from Lemma 3.8, i.e., $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}}$ but (1) from Lemma 3.8 is not satisfied for $\mathbf{s}$ and M. Furthermore, let us assume that $\mathbf{s}$ and M are chosen with this property such that $|M|$ is minimal. We will show that we can bound the size of M. For this, the following lemma will be useful.

Lemma 3.9. *Let $\mathbf{t} \in \mathbf{S}_\emptyset$ and $N \in$ pMSC such that $(\mathbf{t}, \mathcal{B}_\emptyset) \xrightarrow{N}_{\mathcal{A}}$ and $|N| < |M|$. Then there exist atoms $A_1, \ldots, A_m \in \mathbb{A}_H$ and non-empty prefixes $B_i \leq A_i$, $1 \leq i \leq m$, such that the following holds:*

- *For all send types $p!q(c) \in \Sigma$, if there is an unmatched send event of type $p!q(c)$ in B_i, then $q \notin P(B_{i+1} \cdots B_m)$.*
- *$N = B_1 \cdot B_2 \cdots B_m$ (by the first point, concatenation of the B_i is defined)*

Proof. We will prove the lemma by induction on $|N|$. The case $N = \emptyset$ is clear. Thus let us assume that $N \neq \emptyset$. Since $|N| < |M|$, the minimality of M implies that N satisfies (1) from Lemma 3.8. Thus let us take $K \in \langle \mathbb{A}_H \rangle$ and $A_1 \in \mathbb{A}_H$ such that $\mathbf{t} \xrightarrow{K \cdot A_1}_{\mathcal{A}_\emptyset} \mathbf{t}' \in \mathbf{S}_\emptyset$, $P(K) \cap P(N) = \emptyset$, $\sup(A_1, N)$ exists, and $B_1 = \inf(A_1, N) \neq \emptyset$. Since A_1 is an MSC, Lemma 2.2(4) implies that if an unmatched send event of type $p!q(c)$ exists in B_1 then $q \notin P(N \backslash B_1)$. Moreover, Lemma 2.2(5) implies that $N \backslash B_1$ is a pMSC and $N = B_1 \cdot (N \backslash B_1)$. By Lemma 2.3, $\mathcal{A}$ has the following runs:



Finally, since $B_1 \neq \emptyset$, we have $|N \backslash B_1| < |N|$. Thus we can apply the induction hypothesis to $N \backslash B_1$, which implies the statement of the lemma. □

Next fix an arbitrary maximal event e in our fixed MSC $M \neq \emptyset$, and let $N = M\restriction_{E(M) \backslash \{e\}} \in$ pMSC, i.e., we remove e from M. Since $|N| < |M|$ and $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{N}_{\mathcal{A}}$, Lemma 3.9 applies to N. Thus, we get the following two properties (C1) and (C2) for N:

(C1) $M\restriction_{E(M) \backslash \{e\}} = N = B_1 \cdot B_2 \cdots B_m$ for prefixes $B_i \leq A_i$ of atoms $A_i \in \mathbb{A}_H$.
(C2) For all send types $p!q(c) \in \Sigma$, if there is an unmatched send event of type $p!q(c)$ in B_i then $q \notin P(B_{i+1} \cdots B_m)$.

In order to bound the size of M, it suffices to give a bound on the number m. For this, consider the run

$$(\mathbf{s}, \mathcal{B}_\emptyset) = (\mathbf{s}_1, \mathcal{B}_1) \xrightarrow{B_1}_{\mathcal{A}} (\mathbf{s}_2, \mathcal{B}_2) \xrightarrow{B_2}_{\mathcal{A}} \cdots \xrightarrow{B_m}_{\mathcal{A}} (\mathbf{s}_{m+1}, \mathcal{B}_{m+1}) \qquad (2)$$

and assume that $\mathbf{s}_k = \mathbf{s}_\ell$ (but possibly $\mathcal{B}_k \neq \mathcal{B}_\ell$) for some $k < \ell$. Due to (C2), the CFM $\mathcal{A}$ can process, starting from $(\mathbf{s}_k, \mathcal{B}_k)$, also the suffix $B_\ell \cdots B_m$, i.e.,

$(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{B_1 \cdots B_{k-1} \cdot B_\ell \cdots B_m}_{\mathcal{A}} (\mathbf{s}_{m+1}, \mathcal{C})$ for some buffer configuration $\mathcal{C}$ (in general $\mathcal{C} \neq \mathcal{B}_{m+1}$). We will use this observation for a kind of pumping argument. Define $n_p = \max\{|\pi_p(A)| \mid A \in \mathbb{A}_H\}$ for $p \in P$, i.e., n_p is the maximal number of events on process p that occur in some atom from $\mathbb{A}_H$. The following lemma gives us implicitly a bound on the size of N and hence M.

Lemma 3.10. *It holds* $m < (|P| + \sum_{p \in P} n_p + 2) \cdot (1 + \prod_{p \in P} |S_p|)$.

Proof. Let $\widehat{E} \subseteq E(N)$ contain for each $p \in P$ the first n_p many events that occur in N on process p; if $|\pi_p(N)| < n_p$ then all events that occur in N on process p belong to $\widehat{E}$. Note that $|\widehat{E}| \leq \sum_{p \in P} n_p$. Hence it suffices to prove that $m < (|P|+|\widehat{E}|+2) \cdot (1+\prod_{p \in P} |S_p|)$. Assume that $m \geq (|P|+|\widehat{E}|+2) \cdot (1+\prod_{p \in P} |S_p|)$. We will deduce a contradiction to the minimality of M. In the following we have to distinguish two cases, depending on whether the maximal event e of M is a send or a receive event. The case that it is a send event is simpler, so we will only consider the case that it is a receive event, let $q?p(c)$ be the type of e. Let $s \in E(N)$ be the corresponding send event in N. Thus the type of s is $p!q(c)$, and s is the earliest unmatched send event from process p to q in N (if another unmatched send event from p to q would precede s in N then M would not satisfy the FIFO restriction).

Now we mark in the sequence $B_1, B_2, \ldots, B_m$ all positions i, such that either $P(B_1 \cdots B_{i-1}) \subsetneq P(B_1 \cdots B_i)$ or B_i contains an event from $\{s\} \cup \widehat{E}$. Thus $|P| + |\widehat{E}|+1$ many positions become marked. These markings define $|P|+|\widehat{E}|+2$ many (possibly empty) intervals in the sequence $B_1, B_2, \ldots, B_m$ that do not contain any markings. Since $m \geq (|P| + |\widehat{E}| + 2) \cdot (1 + \prod_{p \in P} |S_p|)$, at least one of these intervals has length at least $\prod_{p \in P} |S_p|$. Hence we find $k, \ell \in \{1, \ldots, m\}$ such that $k < \ell$, $\mathbf{s}_k = \mathbf{s}_\ell$ in the run (2), and the subsequence $B_k, \ldots, B_{\ell-1}$ does not contain a marking. Define $N' = B_1 \cdots B_{k-1} \cdot B_\ell \cdots B_m$, due to (C2) concatenation is defined here. Of course we have $|N'| < |N|$, and by the choice of the markings the following holds:

- The send event s still belongs to N'. Moreover, s is also the earliest unmatched send event from p to q in N'. Thus we can define a pMSC M' by adding to N' a new maximal receive event that matches the send event s.
- $P(N) = P(N')$ and thus also $P(M) = P(M')$.
- For all $p \in P$, $\pi_p(N)[1, n_p] = \pi_p(N')[1, n_p]$ and thus also $\pi_p(M)[1, n_p] = \pi_p(M')[1, n_p]$.

By the remark before Lemma 3.10, we have $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{N'}_{\mathcal{A}} (\mathbf{s}_{m+1}, \mathcal{C})$ for some buffer configuration $\mathcal{C}$. Since s is the earliest unmatched send in N' from p to q and $\mathcal{A}$ can execute the receive type $q?p(c)$ in state $\mathbf{s}_{m+1}$, also $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M'}_{\mathcal{A}}$.

We will show that also $M' \neq \emptyset$ does not satisfy (1) from Lemma 3.8, which is a contradiction to the minimality of M. For this let us take arbitrary $K \in$

$\langle \mathbb{A}_H \rangle$, $A \in \mathbb{A}_H$ such that $\mathbf{s} \xrightarrow{K \cdot A}_{\mathcal{A}_\emptyset}$, $P(K) \cap P(M') = \emptyset$, and $\sup(A, M')$ exists. We have to show that $\inf(A, M') = \emptyset$, i.e., $P(A) \cap P(M') = \emptyset$. First, note that because of $P(M) = P(M')$ we have $P(K) \cap P(M) = \emptyset$. Next, since $\sup(A, M')$ exists, $\pi_p(M)[1, n_p] = \pi_p(M')[1, n_p]$ for all $p \in P$, and $|\pi_p(A)| \leq n_p$ for all $p \in P$, Lemma 2.2 implies that also $\sup(A, M)$ exists. Thus, by the choice of M, we have $\inf(A, M) = \emptyset$, i.e., $P(A) \cap P(M) = \emptyset$, which finally implies $P(A) \cap P(M') = \emptyset$. $\square$

Thus, additionally to (C1) and (C2) we can state the following condition (C3):

(C3) The number m in (C1) satisfies $m < (|P| + \sum_{p \in P} n_p + 2) \cdot (1 + \prod_{p \in P} |S_p|)$.

Now we have all the means in order to prove Theorem 3.5.

Proof of Theorem 3.5. In order to simplify the presentation, we will give a polynomial space algorithm for the complementary problem (recall that PSPACE is closed under complement [22]). By Lemma 3.8 it suffices to check whether (b) from Lemma 3.8 does not hold. First, we check whether the finite automaton $\mathcal{A}_\emptyset$ is deadlock-free. Since states of $\mathcal{A}_\emptyset$ can be stored in polynomial space, this can be done in space bounded polynomially in the size of H without explicitly constructing $\mathcal{A}_\emptyset$. If $\mathcal{A}_\emptyset$ is not deadlock-free, we accept. Otherwise, we have to check whether a situation of the form $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}}$ with $\mathbf{s} \in \mathbf{S}_\emptyset$ and $M \neq \emptyset$ exists such that moreover (1) from Lemma 3.8 becomes false. A first approach would be to guess such a situation. But note that the size bound for M that results from (C3) is exponential in the size of H, since $\prod_{p \in P} |S_p|$ is exponential in the size of H. Thus, this idea would lead to an exponential space algorithm. But note that all we have to remember from M in order to check whether $\mathbf{s}$ and M do not satisfy (1) from Lemma 3.8, is the set of processes $P(M)$ and the tuple of prefixes $(\pi_p(M)[1, n_p])_{p \in P}$ of the projections onto the processes (whether $\sup(A, M)$ exists for some $A \in \mathbb{A}_H$ depends by Lemma 2.2 only on the prefixes $\pi_p(M)[1, n_p]$), which can be stored in polynomial space. Hence, we will guess M in an incremental way, and thereby accumulate the data $P(M)$ and $(\pi_p(M)[1, n_p])_{p \in P}$. This is achieved by the algorithm in Figure 2.

Note that all variables only need polynomial space, in particular, the binary coding of the guessed number m needs only polynomial space. Note also that in (†) in Figure 2, we only have to check whether B can be executed, starting from $\mathbf{t}$ and the *empty buffer configuration* $\mathcal{B}_\emptyset$: All unmatched sends that occurred in the past are no longer relevant due to condition (C2), which is assured by the test $P(B) \subseteq P'$.

At the end of the procedure in Figure 2, in case we have not rejected, we have guessed $\mathbf{s} \in \mathbf{S}_\emptyset$, $P_M \subseteq P$, and a tuple $(w_p)_{p \in P} \in \prod_{p \in P} \Sigma_p^*$. Furthermore, these data are guessed such that there exists a pMSC M that satisfies (C1), (C2), (C3), $(\mathbf{s}, \mathcal{B}_\emptyset) \xrightarrow{M}_{\mathcal{A}}$, $P(M) = P_M$, and $\pi_p(M)[1, n_p] \sqsubseteq w_p \sqsubseteq \pi_p(M)$. Furthermore all M satisfying these properties can be potentially guessed. It remains to check

guess $\mathbf{s} \in \mathbf{S}_\emptyset$; (corresponds to $\mathbf{s}$ from the previous discussion)
guess $m < (|P| + \sum_{p\in P} n_p + 2) \cdot (1 + \prod_{p\in P} |S_p|)$; (corresponds to m in (C3))
guess $a \in \{p!q(\cdot),\, p?q(\cdot) \mid (p,q) \in \mathrm{Ch}\}$; (corresponds to the type of the maximal event e of M, where $\cdot$ is a place holder for the message content)
$P' := P$; (contains all processes that may be active in N according to (C2) in the future)
$P_M := \emptyset$; (accumulates the set $P(M)$)
$\mathbf{t} := \mathbf{s}$; (will pass through the sequence $\mathbf{s}_1, \dots, \mathbf{s}_{m+1}$ in the run (2))
$w_p := \varepsilon$ for all $p \in P$; (accumulates the prefix $\pi_p(M)[1, n_p] \in \Sigma_p^*$)
if a is of the form $q?p(\cdot)$ for $p, q \in P$ **then**
  s-occurred := false; (indicates that the send event that corresponds to the maximal event e of M did not yet appear)
  for $i := 1$ to m **do**
    guess $B \leq A \in \mathbb{A}_H$ such that $P(B) \subseteq P'$ and $(\mathbf{t}, \mathcal{B}_\emptyset) \xrightarrow{B}_{\mathcal{A}}$; (†)
    let $(\mathbf{u}, \mathcal{B})$ be such that $(\mathbf{t}, \mathcal{B}_\emptyset) \xrightarrow{B}_{\mathcal{A}} (\mathbf{u}, \mathcal{B})$;
    $P_M := P_M \cup P(B)$; $\mathbf{t} := \mathbf{u}$;
    for all $p \in P$ **do if** $|w_p| < n_p$ **then** $w_p = w_p\, \pi_p(B)$ **endfor**
    for all unmatched send events s of B **do**
      let the type of event s be $k!\ell(d)$;
      $P' := P' \backslash \{\ell\}$;
      if $k = p,\ \ell = q,$ s-occurred = false, and
      s is the earliest unmatched send from p to q in B **then**
        $P_M := P_M \cup \{q\}$; $w_q := w_q\; q?p(d)$; s-occurred := true
    endfor
  endfor
  if s-occurred = false **then** reject **endif**
elseif a is of the form $p!q(\cdot)$ for $p, q \in P$ **then**
  analogous (but simpler) to the previous case, and hence omitted

Fig. 2.

whether $\mathbf{s}$ and the implicitly guessed M do not satisfy (1) from Lemma 3.8. By Lemma 2.2 this is equivalent to the following property:

$$\forall K \in \langle \mathbb{A}_H \rangle\ \forall A \in \mathbb{A}_H : \left\{ \begin{array}{l} \mathbf{s} \xrightarrow{K\cdot A}_{\mathcal{A}_\emptyset} \ \wedge \\ P(K) \cap P_M = \emptyset \wedge \\ P(A) \cap P_M \neq \emptyset \end{array} \right\} \ \Rightarrow\ \exists p \in P \left\{ \begin{array}{l} w_p \not\sqsubseteq \pi_p(A) \wedge \\ \pi_p(A) \not\sqsubseteq w_p \end{array} \right\}.$$

It remains to eliminate the unbounded quantifier $\forall K \in \langle \mathbb{A}_H \rangle$. For this we define the restricted finite automaton $\mathcal{A}'_\emptyset$ by removing from $\mathcal{A}_\emptyset$ all transitions of the form $\mathbf{t_1} \xrightarrow{A} \mathbf{t_2}$ with $P(A) \cap P_M \neq \emptyset$. Then the property above is equivalent to

$$\forall \mathbf{t} \in \mathbf{S}_\emptyset \; \forall A \in \mathbb{A}_H : \left\{ \begin{array}{l} \mathbf{s} \xrightarrow{*}_{\mathcal{A}'_\emptyset} \mathbf{t} \xrightarrow{A}_{\mathcal{A}_\emptyset} \; \wedge \\ P(A) \cap P_M \neq \emptyset \end{array} \right\} \; \Rightarrow \; \exists p \in P \left\{ \begin{array}{l} w_p \not\sqsubseteq \pi_p(A) \wedge \\ \pi_p(A) \not\sqsubseteq w_p \end{array} \right\}.$$

This property can be easily checked in PSPACE (without explicitly constructing the automata $\mathcal{A}'_\emptyset$ and $\mathcal{A}_\emptyset$, which have exponential size). If it holds we accept, otherwise we reject. □

4 Non-FIFO communication

For all results in Section 3 we have restricted to FIFO communication. In this section we briefly discuss the non-FIFO case. Note that the obvious fact that under FIFO communication, every MSC M can be reconstructed from its projections $\pi_p(M)$, $p \in P$, is false for non-FIFO communication (take two messages with identical contents, which are received in M_1 in the order in which they were sent, whereas in M_2 they are received in reverse order). On the other hand if we forbid at least overtaking of messages with identical message contents, this fact still holds, see also [19]. Formally, we require that for all $s_1, s_2 \in E_!$, if $s_1 \prec s_2$, $t(s_1) = p!q(c) = t(s_2)$, and $s_2 \in D$, then also $s_1 \in D$ and $m(s_1) \prec m(s_2)$. Let us assume this for the further discussion. Then for every tuple $(w_p)_{p\in P} \in \prod_{p\in P} \Sigma_p^*$ there exists at most one pMSC M with $\pi_p(M) = w_p$.

For the non-FIFO case, the concatenation of two pMSCs M_1 and M_2 is defined if whenever there is an unmatched send event from p to q with content c in M_1, then there is no message from p to q with content c in M_2. With these modifications, Lemma 2.1 (see [19]) and Lemma 2.2 remain valid for non-FIFO communication.

Also our CFM model has to be slightly altered for the non-FIFO case. The set $\mathbb{C}^{\mathrm{Ch}}$ of buffer configurations has to be replaced by $\mathbb{N}^{\mathrm{Ch}\times\mathbb{C}}$. For a given buffer configuration $\mathcal{B} \in \mathbb{N}^{\mathrm{Ch}\times\mathbb{C}}$, the value $\mathcal{B}((p,q),c)$, where $(p,q) \in \mathrm{Ch}$ and $c \in \mathbb{C}$, represents the number of messages with content c in the channel from p to q, see also [19]. Transitions in this CFM model are defined analogously to the FIFO case in Section 2.2. Then also Lemma 2.3, Lemma 3.1, and Lemma 3.2 remain true.

In order to transfer upper bounds for realizability from FIFO to non-FIFO communication, we can make use of a simple polynomial time reduction, which eliminates message contents. Let H be an HMSC over P and $\mathbb{C}$ with respect to non-FIFO communication. Thus only overtaking of messages with identical content is forbidden. For every two processes $p, q \in P$ and every message content $c \in \mathbb{C}$ we introduce a new process (p,c,q). Moreover, a message from process p to q with content c is replaced by a message from p to (p,c,q) (without any content), which is immediately followed by a message from (p,c,q) to q (without any content). The resulting HMSC H' works without message contents, formally it is defined over a singleton message content alphabet, and it does not contain overtaking messages. It is easy to see that H is weakly (safely) realizable with respect to non-FIFO communication if and only if H' is weakly (safely) realizable with respect to non-FIFO communication. But note that for a singleton message content alphabet, the FIFO restriction is in fact needless. Thus, H' is weakly (safely) realizable with respect to non-FIFO communication if and only if it is weakly (safely) realizable with respect to FIFO communication. Of course, this construction transforms an $\mathcal{I}$-closed (bounded, globally-cooperative) HMSC into an $\mathcal{I}$-closed (bounded, globally-cooperative) HMSC, and it yields a fixed set of processes if we start with a fixed set of processes and message contents. Hence, all upper bounds can be transfered from FIFO to non-FIFO communication.

Concerning our lower bound proofs in Section 3.1, note that in the constructions there, every message is immediately confirmed, which implies that the absence of the FIFO restriction has no effect (the same holds for the PSPACE-hardness proof in [2]). Altogether we obtain the following results:

Theorem 4.1. *The following holds for non-FIFO communication:*

- *The following problem is PSPACE-complete:*
  INPUT: Set P of processes, set $\mathbb{C}$ of message contents, and an $\mathcal{I}$-closed HMSC H over P and $\mathbb{C}$
  QUESTION: Is H safely realizable?
- *The following problem is EXPSPACE-complete:*
  INPUT: Set P of processes, set $\mathbb{C}$ of message contents, and a globally-cooperative (resp. bounded) HMSC H over P and $\mathbb{C}$
  QUESTION: Is H safely realizable?
- *The following problem is undecidable:*
  INPUT: Set P of processes, set $\mathbb{C}$ of message contents, and an HMSC H over P and $\mathbb{C}$
  QUESTION: Is H safely realizable?

Moreover all these results hold already for some fixed P and $\mathbb{C}$.

Note also that the HMSC H in the proof of Theorem 3.4 (resp. Theorem 3.3) is either safely realizable (if $\mathcal{M}$ does not accept w) or not even weakly realizable (if $\mathcal{M}$ accepts w). Hence we obtain

Theorem 4.2. *There exist fixed P and $\mathbb{C}$ such that the following holds for non-FIFO communication:*

- *The following problem is undecidable:*
  INPUT: An HMSC H over P and $\mathbb{C}$
  QUESTION: Is H weakly realizable?

– *The following problem is EXPSPACE-hard:*
INPUT: A bounded HMSC H over P and $\mathbb{C}$
QUESTION: Is H weakly realizable?

For the latter problem, no primitive recursive upper bound is presently known, since the decidability proof in [19] uses a reduction to the reachability problem for Petri nets.

Finally, for $\mathcal{I}$-closed HMSCs, it is easy to modify the PSPACE-hardness proof from [2], in order to show PSPACE-hardness of weak realizability for $\mathcal{I}$-closed HMSCs under non-FIFO communication.

5 Summary

The following table summarize all existing as well as our new results on realizability.

	finite	$\mathcal{I}$-closed	bounded	globally-cooperative	general
safe realizability (FIFO or non-FIFO)	PTIME [1]	PSPACE-complete	EXPSPACE-complete [1] and this paper	EXPSPACE-complete	undecidable
weak realizability (FIFO)	coNP-complete [1]	undecidable [2]	undecidable [2]	undecidable [2]	undecidable [2]
weak realizability (non-FIFO)	coNP-complete [1]	decidable [19], PSPACE-hard	decidable [19], EXPSPACE-hard	decidable [19], EXPSPACE-hard	undecidable

Only in the case of non-FIFO communication the precise complexity of weak realizability for globally-cooperative (resp. $\mathcal{I}$-closed, bounded) HMSCs remains open.

Acknowledgments. I am grateful to Anca Muscholl for many fruitful discussions on the topic of this paper. Thanks also to the anonymous referees for pointing out some inaccuracies in a previous version of this paper.

References

1. R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. In *Proceedings of the 22nd International Conference on on Software Engineering (ICSE 2000), Limerick (Ireland)*, pages 304–313. ACM Press, 2000.
2. R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP 2001), Crete (Greece)*, number 2076 in Lecture Notes in Computer Science, pages 797–808. Springer, 2001.

3. R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *IEEE Transactions on Software Engineering*, 29(7):623–633, 2003.
4. R. Alur and M. Yannakakis. Model checking of message sequence charts. In J. C. M. Baeten and S. Mauw, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR 99), Eindhoven (The Netherlands)*, number 1664 in Lecture Notes in Computer Science, pages 114–129. Springer, 1999.
5. H. Ben-Abdallah and S. Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In E. Brinksma, editor, *Proceedings of the 3rd International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS '97), Enschede (The Netherlands)*, number 1217 in Lecture Notes in Computer Science, pages 259–274, 1997.
6. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the Association for Computing Machinery*, 30(2):323–342, 1983.
7. B. Caillaud, P. Darondeau, L. Hélouët, and G. Lesventes. HMSCs as partial specifications . . . with Petri nets as completion. In F. Cassez, C. Jard, B. Rozoy, and M. D. Ryan, editors, *4th Summer School on Modelling and Verification of Parallel Processes (MOVEP 2000), Nantes (France)*, number 2067 in Lecture Notes in Computer Science, pages 125–152, 2000.
8. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.
9. B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs: Model-checking and realizability. In P. Widmayer, F. T. Ruiz, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP 2002), Malaga (Spain)*, number 2380 in Lecture Notes in Computer Science, pages 657–668. Springer, 2002.
10. E. Gunter, A. Muscholl, and D. Peled. Compositional message sequence charts. In T. Margaria and W. Yi, editors, *7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, (TACAS 2001), Genova (Italy)*, volume 2031 of *Lecture Notes in Computer Science*, pages 496–511. Springer, 2001.
11. L. Hélouët and C. Jard. Conditions for synthesis of communicating automata from HMSCs. In *5th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2000), Berlin (Germany)*, 2000.
12. L. Hélouët and P. Le Maigat. Decomposition of message sequence charts. In *2nd Workshop on SDL and MSC (SAM 2000), Grenoble (France)*, pages 46–60, 2000.
13. J. G. Henriksen, M. Mukund, K. N. Kumar, and P. Thiagarajan. On message sequence graphs and finitely generated regular MSC languages. In M. Nielsen and B. Rovan, editors, *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP 2000), Geneva (Switzerland)*, number 1853 in Lecture Notes in Computer Science, pages 675–686. Springer, 2000.
14. J. G. Henriksen, M. Mukund, K. N. Kumar, and P. Thiagarajan. Regular collections of message sequence charts. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science (MFCS'2000), Bratislava (Slovakia)*, number 1893 in Lecture Notes in Computer Science, pages 675–686. Springer, 2000.
15. ITU. Recommendation Z.100. Specification and Description Language (SDL). 1994.
16. ITU. Recommendation Z.120. Message Sequence Charts. 1996.
17. D. Kuske. A further step towards a theory of regular msc languages. In H. Alt and A. Ferreira, editors, *Proceedings of the19th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2002), Juan les Pins (France)*, number 2285 in Lecture Notes in Computer Science, pages 489–500. Springer, 2002.
18. M. Lohrey. Safe realizability of high-level message sequence charts. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR 2002), Brno (Czech Republic)*, number 2421 in Lecture Notes in Computer Science, pages 177–192. Springer, 2002.
19. R. Morin. Recognizable sets of message sequence charts. In H. Alt and A. Ferreira, editors, *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2002), Juan les Pins (France)*, number 2285 in Lecture Notes in Computer Science, pages 523–534. Springer, 2002.
20. M. Mukund, K. N. Kumar, and M. A. Sohoni. Synthesizing distributed finite-state systems from MSCs. In C. Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR 2000), University Park, PA (USA)*, number 1877 in Lecture Notes in Computer Science, pages 521–535. Springer, 2000.

21. A. Muscholl and D. Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In M. Kutylowski, L. Pacholski, and T. Wierzbicki, editors, *Proceedings of the 24th International Symposium on Mathematical Foundations of Computer Science (MFCS'99), Szklarska Poreba (Poland)*, number 1672 in Lecture Notes in Computer Science, pages 81–91. Springer, 1999.
22. C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.
23. I. Walukiewicz. Difficult configurations – on the complexity of LTrL. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP 98), Aalborg (Denmark)*, number 1443 in Lecture Notes in Computer Science, pages 140–151. Springer, 1998.
24. W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. — Informatique Théorique et Applications*, 27:99–135, 1985.