Bounded MSC Communication

Markus Lohrey [a,1], Anca Muscholl [b]

[a] *Universität Stuttgart, FMI,*
Universitätsstr. 38, 70569 Stuttgart, Germany

[b] *LIAFA, Université Paris VII*
2, place Jussieu, case 7014
75251 Paris cedex 05, France

Abstract

Message sequence charts (MSCs) and high-level message sequence charts (HMSCs) are popular formalisms for the specification of communication protocols between asynchronous processes. An important concept in this context is the size of the communication buffers used between processes. Since real systems impose limitations on the capacity (or speed) of communication links, we ask whether a given HMSC can be implemented with respect to a given buffer size imposed by the environment. We introduce four different measures for buffer sizes and investigate for each of these measures the complexity of deciding whether a given MSC (or HMSC, or nested MSC) satisfies a given bound on the buffer size. The complexity of these problems varies between the classes P, NP, and coNP.

Key words: Message sequence charts, channel boundedness, computational commplexity.

1 Introduction

Message sequence charts (MSC) as well as high-level message sequence charts (HMSC) are popular visual formalisms for the specification of communication protocols between asynchronous processes that communicate over reliable, point-to-point channels. The big advantage of this kind of specification, compared say to logics or automata, is that it emphasizes the concurrent behavior

Email addresses: `lohrey@informatik.uni-stuttgart.de` (Markus Lohrey), `anca@liafa.jussieu.fr` (Anca Muscholl).

[1] Research supported by the INRIA cooperative research action FISC.

of processes in form of diagrams, forgetting about implementation-specific details as variables, timing constraints, etc. This makes MSCs interesting for describing *scenarios*, in form of typical positive or negative examples of behaviors.

When MSC specifications get to be implemented, the real system architecture usually imposes limitations on the capacity (or speed) of communication links. This leads to the problem of checking whether an MSC specification will meet some given size constraints on the communication channels. A typical example is checking *process divergence* [3], where divergence means that one process can send a message an unbounded number of times ahead of the receiving process. In this paper we refine the analysis of divergence, by considering concrete bounds on the size of channels.

We consider four different measures for the channel requirements of (H)MSCs. These measures result from two orthogonal dimensions: In the first dimension we distinguish whether *all* linearizations of an MSC M satisfy a certain buffer bound ($\forall$-boundedness), respectively whether *at least one* linearization of M respects the bound ($\exists$-boundedness). In the second dimension we distinguish between measuring the buffer size over all channels (*global* boundedness), respectively on each channel separately, taking the maximum over all channels (*local* boundedness).

The local version of boundedness is important for implementing an MSC specification in a distributed process environment, which imposes size restrictions on specific channels. Global boundedness arises naturally when one simulates MSC executions on a single-processor environment, for instance for test purposes.

Universal channel boundedness is a safety requirement, expressing that any interleaving of the MSC execution is possible within the constraints imposed by the environment. Universal channel boundedness is met for instance by regular (or bounded) HMSCs [2,15,6,7]. This subclass has been proposed in order to have a decidable model-checking problem. It is characterized by having a regular set of MSC-linearizations, which means in particular that channels are universally bounded. However, checking that an HMSC can be implemented by a finite-state machine is undecidable, in general [6].

Existential channel boundedness can be interesting for simulation purposes, since there it suffices to consider at least one interleaving for each MSC execution. Furthermore, an existential channel bound on an MSC M can suggest that the specification given by M can be enhanced by timers that ensure the bound universally, by slowing down certain processes. HMSCs are by definition existentially channel bounded, but not universally bounded in general. It is interesting to note that an existential bound on the channels provides the decidability of model-checking partial-order MSO properties against (compositional) HMSCs [12].

The results obtained in this paper can be seen as a more specific analysis of channel boundedness. Indeed, we show that one is able to compute the minimal channel bounds for which an (H)MSC is existentially (universally, resp.) bounded. For each of our buffer measures we investigate the complexity of deciding whether a given MSC (resp. HMSC, nested MSC [5]) satisfies a given bound on the channel size. The complexity of these problems varies between the classes P, NP, and coNP, see Table 1 in Section 7 for a summary of our results. A short version of this paper appeared in [11].

2 Preliminaries

For complexity results we will use standard classes like non-deterministic logarithmic space (NL), polynomial time (P), non-deterministic polynomial time (NP) and coNP (complements of NP-problems), see [17] for definitions.

A *linearization* of a partially ordered set $(A, \prec)$ is a total order on A that extends the partial order $\prec$. The transitive (transitive and reflexive) closure of a binary relation E is the least transitive relation E^+ (transitive and reflexive relation E^*) containing E. A *transitive reduction of* E is a minimal relation $F \subseteq E$ such that $E^+ = F^+$. For any alphabets $A \subseteq B$ and any word $w \in B^*$ we define $|w|_A$ as the number of symbols from A in w.

A directed graph (V, E) is *weakly connected*, or just *connected*, if the undirected graph $(V, E \cup E^{-1})$ is connected. A *connected component* of (V, E) is a maximal connected subgraph of (V, E). We say that (V, E) is *strongly connected* if for all $v, w \in V$ we have $(v, w), (w, v) \in E^*$. Finally, (V, E) is *locally strongly connected* if every connected component of (V, E) is strongly connected.

2.1 Message sequence charts

Following the ITU norm Z.120 a *message sequence chart (MSC) M* is a tuple $(\mathcal{E}, P, \lambda, t, m, \prec)$, where:

- $\mathcal{E}$ is a finite set of *events*.
- P is a finite set of *processes*.
- $\lambda : \mathcal{E} \to P$ associates with each event e a process $\lambda(e)$ on which e is located.
- $t : \mathcal{E} \to \{S, R\}$ associates with each event a type. Events in $t^{-1}(S)$ are called *send events*, events in $t^{-1}(R)$ are called *receive events*.

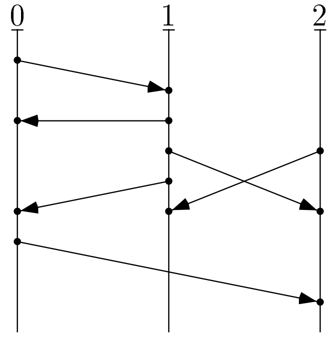

Fig. 1. An MSC.

- $m : t^{-1}(S) \to t^{-1}(R)$ is a bijection. A pair $(s, m(s))$ with s a send event is also called a *message* from process $p = \lambda(s)$ to process $q = \lambda(m(s))$. The *channel type* of s (resp. $m(s)$) is defined as $S(p,q)$ (resp. $R(p,q)$). The channel type of $e \in \mathcal{E}$ is denoted $\mathrm{ct}(e)$.
- $\prec$ is a partial order on $\mathcal{E}$, also called the *visual order of* M. We require that the set of events $\lambda^{-1}(p)$ is totally ordered by $\prec$, for every $p \in P$, and that $\prec$ equals the transitive closure of the *acyclic* relation

$$\bigcup_{p \in P} \prec\restriction_{\lambda^{-1}(p)} \;\cup\; \{(s, m(s)) \mid t(s) = S\}\,.$$

Note that $\prec$ is uniquely determined by the function m and the total orders $\prec\restriction_{\lambda^{-1}(p)}$ $(p \in P)$. In computational problems, we may therefore represent M by the tuple $(\mathcal{E}, P, \lambda, t, m, (\prec\restriction_{\lambda^{-1}(p)})_{p \in P})$ Since each component in this tuple can be represented by a data structure of size $O(|\mathcal{E}|)$, its is appropriate to define the size $|M|$ of the MSC M by $|\mathcal{E}|$. We use the usual graphical representation of MSCs, where time flows top-down and processes are drawn as vertical lines. Figure 1 shows an MSC over three processes $0, 1, 2$.

Often MSCs are further restricted to satisfy the *FIFO-condition*, which means that whenever there are two send events s_1 and s_2 with $\mathrm{ct}(s_1) = \mathrm{ct}(s_2)$ and $s_1 \prec s_2$ then also $m(s_1) \prec m(s_2)$. Thus message overtaking on any channel is disallowed, where a *channel* is a pair (p,q) of distinct processes. For instance, the above MSC satisfies the FIFO-restriction. The MSC definition may also include message contents or local actions, however this is not important in the present setting. The complexity results in this paper mostly hold independently of the FIFO-restriction (respectively whether message contents are allowed or not). This is due to the fact that all lower bound proofs in this paper hold under the FIFO-restriction, whereas all upper bound proofs (excepting those for nested MSCs in Section 5) hold without the FIFO-restriction.

Let $M = (\mathcal{E}, P, \lambda, t, m, \prec)$ be an MSC. A *linearization* of M is a linearization of the visual order $(\mathcal{E}, \prec)$. Let L be a linearization of M and let $b \in \mathbb{N}$. We say that L is *globally-bounded* by b if $|K|_{t^{-1}(S)} - |K|_{t^{-1}(R)} \leq b$ for every prefix K of L. We say that L is *locally-bounded* by b if for all channels (p,q) and every prefix K of L it holds $|K|_{\mathrm{ct}^{-1}(S(p,q))} - |K|_{\mathrm{ct}^{-1}(R(p,q))} \leq b$. We say that M is $\exists^b_{\mathrm{glob}}$-bounded (resp. $\exists^b_{\mathrm{loc}}$-bounded) if there exists a linearization of M which is globally-bounded (resp. locally-bounded) by b. We say that M is $\forall^b_{\mathrm{glob}}$-bounded (resp. $\forall^b_{\mathrm{loc}}$-bounded) if every linearization of M is globally-bounded (resp. locally-bounded) by b. Of course for $Q \in \{\exists, \forall\}$, if M is Q^b_{glob}-bounded, then M is also Q^b_{loc}-bounded. Vice versa, if M is Q^b_{loc}-bounded, then M is Q^c_{glob}-bounded, where $c = |P| \cdot (|P| - 1) \cdot b$. The MSC in Figure 1 is for instance $\exists^2_{\mathrm{glob}}$-bounded, $\forall^4_{\mathrm{glob}}$-bounded, $\exists^1_{\mathrm{loc}}$-bounded, and $\forall^2_{\mathrm{loc}}$-bounded, and moreover, these bounds are optimal.

For $Q \in \{\exists, \forall\}$ and $Y \in \{\mathrm{loc}, \mathrm{glob}\}$, we define Q_Y-MSC-BOUNDED as the following decision problem.

INPUT: MSC M and positive integer b.

QUESTION: Is M Q^b_Y–bounded?

Instead of speaking about prefixes of linearizations of MSCs, it is sometimes more convenient to consider configurations of MSCs. A *configuration* C of M is a downward-closed subset $C \subseteq \mathcal{E}$ of events, i.e., if $e \prec f \in C$ then also $e \in C$. A prefix of a linearization of M defines in the obvious way a unique configuration of M. Vice versa, for every configuration C of M there exists at least one prefix K of a linearization of M such that K defines C.

Let C be a configuration of the MSC M. The number of messages (s, r) in M with $s \in C$ and $r \notin C$ is denoted by $\mathrm{gus}(C, M)$ (**g**lobally **u**nmatched **s**ends). The maximum over all channels (p, q) of the number of messages (s, r) in M with $\mathrm{ct}(s) = S(p, q)$, $\mathrm{ct}(r) = R(p, q)$, $s \in C$, and $r \notin C$ is denoted by $\mathrm{lus}(C, M)$ (**l**ocally **u**nmatched **s**ends). By definition, M is $\forall^b_{\mathrm{glob}}$-bounded (resp. $\forall^b_{\mathrm{loc}}$-bounded) if and only if $\mathrm{gus}(C, M) \leq b$ (resp. $\mathrm{lus}(C, M) \leq b$) for every configuration C of M.

In the remainder of this section we define *high-level message sequence charts*. For this we need first to define the (sequential) product of two MSCs. Let $M_i = (\mathcal{E}_i, P, \lambda_i, t_i, m_i, \prec_i)$, $i = 1, 2$, be two MSCs over the same set P of processes, where furthermore $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$. Then the concatenation of M_1 and M_2 is the MSC $M_1 M_2 = (\mathcal{E}_1 \cup \mathcal{E}_2,\ P,\ \lambda_1 \cup \lambda_2,\ t_1 \cup t_2,\ m_1 \cup m_2,\ \prec)$, where

$$\prec\ =\ (\prec_1 \cup \prec_2 \cup \{(e_1, e_2) \mid e_1 \in \mathcal{E}_1, e_2 \in \mathcal{E}_2,\ \lambda_1(e_1) = \lambda_2(e_2)\})^+.$$

Intuitively, $M_1 M_2$ results from appending M_2 at M_1 by gluing them together at corresponding process lines.

The standard ITU definition Z.120 defines a *high-level message sequence chart (HMSC)* as a finite transition system with nodes labeled by finite MSCs. Formally, let an HMSC H be given as $H = (V, \rightarrow, P, \mu, v)$, where $(V, \rightarrow)$ is

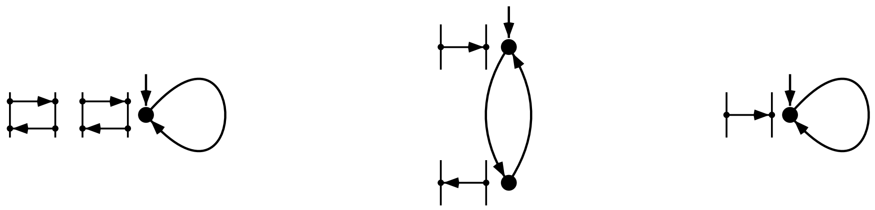

Fig. 2. Three HMSCs

a finite transition system with initial node v, P is the set of processes and μ maps every node $u \in V$ to a finite MSC $\mu(u)$ over the set of processes P. The MSC-language $\mathrm{msc}(H)$ defined by H is the set of all MSCs $\mu(u_1)\mu(u_2)\cdots$, where $u_1 = v$ and $u_1 \to u_2 \to \cdots$ is a (finite or infinite) maximal path in $(V, \to)$ (formally, for infinite paths we have to define the limit of the sequence $\mu(u_1 \cdots u_k)_{k \geq 1}$). We impose the restriction that every node u is accessible from the initial node v. Figure 2 shows three simple examples for HMSCs, where the initial node v is marked with an incoming arrow.

Let $Q \in \{\exists, \forall\}$, $Y \in \{\mathrm{loc}, \mathrm{glob}\}$, and $b \in \mathbb{N}$. We say that an HMSC H is Q^b_Y-bounded, if all $M \in \mathrm{msc}(H)$ are Q^b_Y-bounded. Finally, we define the decision problem Q_Y-HMSC-BOUNDED as follows:

INPUT: HMSC H and positive integer b.

QUESTION: Is H Q^b_Y-bounded?

The first HMSC in Figure 2 is Q^1_{loc}-bounded and Q^2_{glob}-bounded for both $Q = \exists$ and $Q = \forall$. The second HMSC is Q^1_Y-bounded for all $Y \in \{\mathrm{loc}, \mathrm{glob}\}$ and $Q \in \{\exists, \forall\}$. Finally, the third HMSC is $\exists^1_Y$-bounded for both $Y = \mathrm{loc}$ and $Y = \mathrm{glob}$ but it is neither $\forall^b_{\mathrm{loc}}$-bounded nor $\forall^b_{\mathrm{glob}}$-bounded for any $b \in \mathbb{N}$.

2.2 Pebble games

As we will see in Section 3 there is a tight connection between the existential-global boundedness problem and pebble games on directed graphs. In this section we recall the definition of pebble games and related results.

Let $G = (V, E)$ be a finite *directed acyclic graph* (dag) with node set V and edge set $E \subseteq V \times V$. A *game-configuration* is a subset of V (which corresponds to the set of nodes that are currently pebbled). For two game-configurations $C_1, C_2 \subseteq V$ and a node $v \in V$ we write $C_2 = C_1 \dot\cup \{v\}$ whenever $C_2 = C_1 \cup \{v\}$ and $v \notin C_1$, i.e., C_2 is the disjoint union of C_1 and $\{v\}$. A *move* in G is a pair (C_1, C_2) of game-configurations such that one of the following three cases holds:

(1) There exists $w \in C_1$ with $C_2 = C_1 \backslash \{w\}$ (remove the pebble from node

w).

(2) There exists a node $v \in C_2$ such that $C_2 = C_1 \dot{\cup} \{v\}$ and for all $u \in V$ with $(u, v) \in E$ it holds $u \in C_1$ (put a pebble on node v provided that each direct predecessor has a pebble).

(3) There exist nodes $w \in C_1$, $v \in C_2$ with $(w, v) \in E$, $C_2 = (C_1 \backslash \{w\}) \dot{\cup} \{v\}$, and for all $u \in V$ with $(u, v) \in E$ it holds $u \in C_1$ (move the pebble from node w to node v, provided that all direct predecessors of v have a pebble). This last rule is referred to as the *move rule*.

More precisely we say that (C_1, C_2) is an *i-move*, $i \in \{1, 2, 3\}$, if case (i) above holds. Let $b \in \mathbb{N}$. We say that the dag G can be *b-pebbled* if there exists a sequence $C_1, C_2, \ldots, C_n \subseteq V$ of game-configurations such that the following holds:

(a) $C_1 = C_n = \emptyset$ and $|C_i| \leq b$ for $1 \leq i \leq n$,

(b) For every node $v \in V$ there exists *exactly one* $i \in \{1, \ldots, n-1\}$ such that $v \notin C_i$ and $v \in C_{i+1}$. That is, each node is pebbled exactly once.

(c) (C_i, C_{i+1}) is a move for $1 \leq i < n$.

If instead of (c) we require that (C_i, C_{i+1}) is a 1-move or a 2-move for all $1 \leq i < n$, then we say that the dag G can be *b-pebbled without the move rule*. It is important to note that by condition (b) every node of G has to be pebbled exactly once, i.e., re-pebbling is not allowed. This fact is crucial for the NP-upper bound in the following result due to R. Sethi.

Theorem 2.1 ([18]). *The following problem is NP-complete:*

INPUT: Finite dag G with only one node of out-degree 0, positive integer b.

QUESTION: Can G be b-pebbled?

For the further consideration we need the variant of the theorem above where the move rule is not allowed.

Corollary 2.2. *The following problem is NP-complete:*

INPUT: Finite dag G, positive integer b.

QUESTION: Can G be b-pebbled without the move rule?

Proof. Membership in NP is obvious. We prove NP-hardness by showing that a finite dag G with exactly one node of out-degree 0 can be b-pebbled if and only if G can be $(b+1)$-pebbled without the move rule.

First, any dag G that can be b-pebbled with the move rule, can be $(b+1)$-pebbled without the move rule. Now assume that G can be $(b+1)$-pebbled without the move rule and let $\emptyset = C_1, C_2, \ldots, C_n = \emptyset$ be a play such that $|C_i| \le b+1$ for $1 \le i \le n$, every C_i results from C_{i-1} by a 1-move or a 2-move, and finally every node is pebbled exactly once. We show that G can be b-pebbled by successively eliminating all C_i with $|C_i| = b+1$. The following arguments are similar to those in the proof of Theorem A in [21]. Let $|C_i| = b+1$. Then we must have $2 \le i \le n-1$. Moreover, C_i must be obtained from the preceding game configuration by pebbling some node v, and the successor configuration of C_i is obtained by removing a pebble from some node w. Formally, we have $C_i = C_{i-1} \dot\cup \{v\}$, and $C_{i+1} = C_i \setminus \{w\}$ for some nodes $v, w \in C_i$ such that $u \in C_{i-1}$ for all $u \in V$ with $(u,v) \in E$. We can distinguish the following three cases:

Case 1. If $(w,v) \in E$ then we replace the play $C_1, C_2, \ldots, C_n$ by

$$C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n.$$

Note that (C_{i-1}, C_{i+1}) is a move according to the move rule.

Case 2. If $(w,v) \notin E$ and $w \neq v$ then we can remove the pebble from w before pebbling v, i.e., we replace the play by $C_1, \ldots, C_{i-1}, C_{i-1} \setminus \{w\}, C_{i+1}, \ldots, C_n$.

Case 3. If $w = v$ then v must be the unique node with out-degree 0, because otherwise we would have to pebble all direct successors of v before removing the pebble from v (recall that re-pebbling is not allowed). Thus v is the last node that is pebbled. So we can choose any $u \in V$ with $(u,v) \in E$ and replace the move (C_{i-1}, C_i) by the 3-move $(C_{i-1}, C_i \setminus \{u\})$. $\square$

Remark 2.3. *Note that if a dag G contains more than one node of out-degree 0, then it is no longer the case that G can be b-pebbled if and only if G can be $(b+1)$-pebbled without the move rule. A counterexample for this would be the dag with node set $\{a,b,c,d\}$ and edges $(a,c), (a,d), (b,c)$, and (b,d). It can be 3-pebbled without the move rule but it cannot be 2-pebbled by allowing the move rule.*

For a finite dag $G = (V,E)$ let $\overline{V} = \{\overline{v} \mid v \in V\}$ be a disjoint copy of the node set V and let $\tilde{E} = \{(u,v), (v,\overline{u}) \mid (u,v) \in E\} \cup \{(u,\overline{u}) \mid u \in V\}$. Note that the graph $\tilde{G} = (V \cup \overline{V}, \tilde{E})$ is again a finite dag.

Lemma 2.4. *A finite dag $G = (V,E)$ can be b-pebbled without the move rule if and only if there exists a linearization ℓ of the dag $\tilde{G}$ such that for every prefix k of ℓ we have $|k|_V - |k|_{\overline{V}} \le b$.*

Proof. The idea is to associate with a linearization ℓ of $\tilde{G}$ a pebbling strategy and vice versa as follows: The occurrence of $v \in V$ in ℓ corresponds to the instant where we put a pebble on v, whereas $\overline{v} \in \overline{V}$ corresponds to the instant where we remove the pebble from v. Then edges $(v, \overline{u}) \in \tilde{E}$ with $(u, v) \in E$ express the fact that before we remove a pebble from u, all direct successors of u have to be pebbled. By this observation the lemma follows immediately. $\square$

3 Bounded communication in finite MSCs

In this section we will determine the complexity of the four decision problems Q_Y-MSC-BOUNDED for $Q \in \{\exists, \forall\}$ and $Y \in \{\text{loc}, \text{glob}\}$.

For local boundedness we can argue by considering some additional ordering on events. Let us fix a bound b and an MSC $M = (\mathcal{E}, P, \lambda, t, m, \prec)$. With M and b we associate a binary relation $\rightsquigarrow_b$ on $\mathcal{E}$ as follows. We let $r \rightsquigarrow_b s$ whenever for some channel (p, q) and some $i \geq 1$ we have that s is the $(i+b)$-th send of channel type $S(p, q)$, whereas r is the i-th receive of channel type $R(p, q)$. In case we deal with a fixed buffer size b, we also write $r \rightsquigarrow s$ instead of $r \rightsquigarrow_b s$.

Lemma 3.1. *Let M be an MSC and $b \in \mathbb{N}$. Let $\prec$ be the visual order of M and $\rightsquigarrow_b$ be the relation associated with M and b. Then the following holds:*

(1) M is $\exists^b_{\text{loc}}$-bounded if and only if the relation $\prec \cup \rightsquigarrow_b$ is acyclic.
(2) M is $\forall^b_{\text{loc}}$-bounded if and only if $\rightsquigarrow_b$ is contained in $\prec$.

Proof. M is $\exists^b_{\text{loc}}$-bounded if and only if there exists a linearization L of M such that for every i and every channel (p, q) the i-th receive of channel type $R(p, q)$ precedes the $(i + b)$-th send of channel type $S(p, q)$. But this is equivalent to saying that there exists a linearization of M such that for all r, s with $r \rightsquigarrow_b s$, r precedes s in the linearization.

Similarly, M is $\forall^b_{\text{loc}}$-bounded if and only if for every i and every channel (p, q) the i-th receive r of channel type $R(p, q)$ precedes the $(i + b)$-th send s of channel type $S(p, q)$ in every linearization of M. But this is equivalent to $r \prec s$ for all r, s with $r \rightsquigarrow_b s$. $\square$

Since it can be checked in linear time whether a directed graph is acyclic and the size of the relation $\rightsquigarrow_b$ is in $O(|M|)$, we obtain the following result from Lemma 3.1(1):

Proposition 3.2. $\exists_{\text{loc}}$-MSC-BOUNDED *can be solved in linear time.*

Moreover, since by [1] the visual order $\prec$ of an MSC $M = (\mathcal{E}, P, \lambda, t, m, \prec)$ can be calculated from m and $(\prec\restriction_{\lambda^{-1}(p)})_{p \in P}$ in time $O(|M|^2)$, the following proposition follows from Lemma 3.1(2):

Proposition 3.3. $\forall_{\text{loc}}$-MSC-BOUNDED *can be solved in time* $O(n^2)$.

Surprisingly, if we consider *global* boundedness instead of local boundedness, the existential variant of the problem becomes NP-complete:

Theorem 3.4. $\exists_{\text{glob}}$-MSC-BOUNDED *is NP-complete.*

Proof. Membership in NP is obvious. In order to prove NP-hardness, we construct for a finite dag $G = (V, E)$ a finite MSC $M(G) = (\mathcal{E}, P, \lambda, t, m, \prec)$ such that G can be b-pebbled without the move rule if and only if $M(G)$ is $\exists_{\text{glob}}^{(b+1)}$-bounded. With each node $v \in V$ we associate the set of processes $P_v = \{p_{\text{in}}(v), p(v), p_{\text{out}}(v)\} \cup \{p(v, w) \mid (v, w) \in E\}$, the set of all processes is then $P = \bigcup_{v \in V} P_v$. The set $\mathcal{E}$ of events consists of $V \cup \overline{V}$ (where $\overline{V} = \{\overline{v} \mid v \in V\}$) plus some additional events (see Figure 3). We have $\lambda(v) = p_{\text{in}}(v)$ and $\lambda(\overline{v}) = p_{\text{out}}(v)$. For each node v there is a message $(v, \overline{v})$ from process $p_{\text{in}}(v)$ to process $p_{\text{out}}(v)$. This messages crosses the chain of the two messages from $p_{\text{in}}(v)$ to $p(v)$ and from $p(v)$ to $p_{\text{out}}(v)$. The basic idea is that a pebble on node v corresponds to the fact that the send event v was already executed, whereas the corresponding receive $\overline{v}$ wasn't yet executed. Furthermore for each edge $(u, v) \in E$ we have exactly one message from process $p(u, v)$ to $p_{\text{in}}(v)$ and back from $p_{\text{in}}(v)$ to $p(u, v)$. Finally if $(v, w_1), \ldots, (v, w_n)$ are all outgoing edges of node v (listed in an arbitrary order) then there is exactly one message from process $p_{\text{out}}(v)$ to $p(v, w_1)$ and back and exactly one message from process $p(v, w_i)$ to $p(v, w_{i+1})$ and back $(1 \leq i \leq n-1)$. The order of the events on the processes in P_v is shown in Figure 3, where we show an example where $(u_1, v), (u_2, v), (v, w_1), (v, w_2)$, and (v, w_3) are the adjacent edges of v. The process names labeling message arrows specify the source, resp. target process of the message. Note that the resulting visual order $\prec$ is indeed acyclic: Messages from $p_{\text{in}}(v)$ to $p_{\text{out}}(v)$, from $p_{\text{in}}(w)$ to $p(v, w)$, from $p(v, w_{i+1})$ to $p(v, w_i)$, and from $p(v, w_1)$ to $p_{\text{out}}(v)$, respectively, cannot be involved in a cycle (going along these edges, one finally arrives at the maximal event $\overline{v}$). If the remaining messages yield a cycle, then this cycle would result from a cycle in the dag G (for instance the send of the message from process $p(u, v)$ to $p_{\text{in}}(v)$ must precede the send of the message from process $p(v, w)$ to $p_{\text{in}}(w)$, with $(u, v), (v, w)$ edges of the dag G). Moreover, $M(G)$ respects the FIFO-restriction, in fact this was the only reason for introducing process $p(v)$. The crucial point of our construction is that the restriction $\prec\restriction_{V \cup \overline{V}}$ of the visual order $\prec$ of $M(G)$ to the set of events $V \cup \overline{V} \subseteq \mathcal{E}$ is precisely the transitive closure of the relation $\tilde{E}$ from Lemma 2.4.

Claim 1: If G can be b-pebbled without the move rule then $M(G)$ is $\exists_{\text{glob}}^{(b+1)}$-bounded.

Assume that G can be b-pebbled without the move rule by a sequence of moves. We translate each move into a sequence of events, such that the resulting

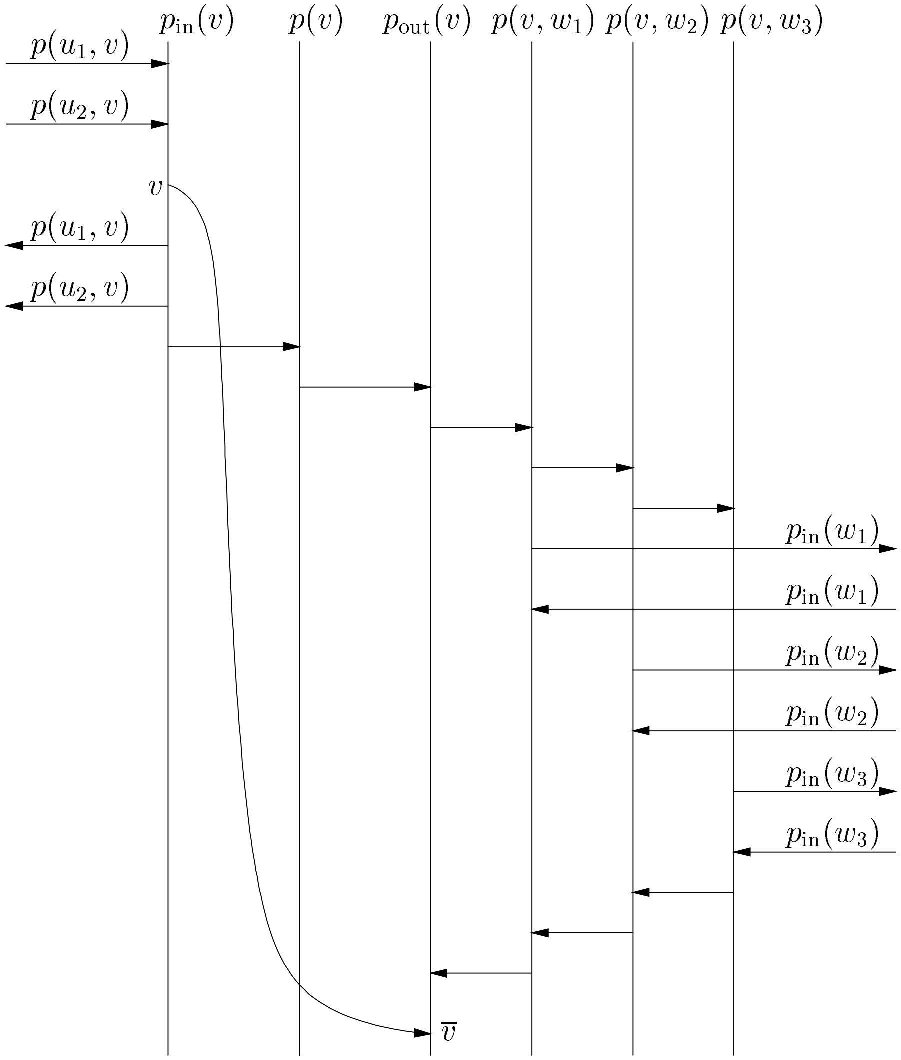

Fig. 3. Communication between the processes in P_v in the MSC $M(G)$

sequence of events is a linearization of M which is globally-bounded by $b+1$. Consider a move (C_1, C_2). If $C_2 = C_1 \dot{\cup} \{v\}$, i.e., node v is pebbled in the move, then we execute the following sequence of events:

(1) Send and immediately receive the message from process $p(u_i, v)$ to $p_{\text{in}}(v)$ for $1 \le i \le k$, where $(u_1, v), \ldots, (u_k, v)$ are all incoming edges of node v.
(2) Execute send event v on process $p_{\text{in}}(v)$.
(3) Send and immediately receive the message from process $p_{\text{in}}(v)$ to $p(u_i, v)$ for $1 \le i \le k$.
(4) Send and immediately receive the message from process $p_{\text{in}}(v)$ to $p(v)$, followed by the message from $p(v)$ to $p_{\text{out}}(v)$.
(5) Send and immediately receive the message from process $p_{\text{out}}(v)$ to $p(v, w_1)$ followed by the messages from $p(v, w_i)$ to $p(v, w_{i+1})$ for $1 \le i < n$, where $(v, w_1), \ldots, (v, w_n)$ are all outgoing edges of node v.

Of course if v has in-degree 0 (resp. out-degree 0) then (1) and (3) (resp. (5)) disappear. On the other hand if $C_2 = C_1 \setminus \{v\}$, i.e., a pebble is taken from node v in the move, then we execute the following sequence of events:

(1) Send and immediately receive the message from process $p(v, w_{i+1})$ to $p(v, w_i)$ for $n > i \geq 1$, where $(v, w_1), \ldots, (v, w_n)$ are all outgoing edges of node v.
(2) Send and immediately receive the message from process $p(v, w_1)$ to $p_{\mathrm{out}}(v)$.
(3) Execute the receive event $\overline{v}$ on process $p_{\mathrm{out}}(v)$.

Claim 2: If $M(G)$ is $\exists_{\mathrm{glob}}^{(b+1)}$-bounded then G can be b-pebbled without the move rule.

Let L be a linearization of $M(G)$, which is globally-bounded by $b+1$ such that furthermore the number of prefixes K of L that satisfy $|K|_{t^{-1}(S)} - |K|_{t^{-1}(R)} = b+1$ is minimal among all linearizations of $M(G)$ that are globally-bounded by $b+1$. Clearly such an L exists. Let $\pi(L)$ be the projection of the word L onto $V \cup \overline{V} \subseteq \mathcal{E}$. Since of course $\pi(L)$ is a linearization of $\prec\restriction_{V \cup \overline{V}}$, and hence a linearization of the dag $\tilde{G}$, by Lemma 2.4 it suffices to prove the following claim:

Claim 3: For every prefix k of $\pi(L)$ it holds $|k|_V - |k|_{\overline{V}} \leq b$.

Clearly we have $|k|_V - |k|_{\overline{V}} \leq b+1$ for every prefix k of $\pi(L)$. In order to prove the claim let us assume that $|\pi(L_1 v)|_V - |\pi(L_1 v)|_{\overline{V}} = b+1$, where $v \in V$ and $L = L_1 v L_2$. Let $L_2 = e L_3$, where $e \in \mathcal{E}$ (note that we must have $L_2 \neq \epsilon$), thus $L = L_1 v e L_3$. If e would be a send event then $|L_1 v e|_{t^{-1}(S)} - |L_1 v e|_{t^{-1}(R)} \geq b+2$, a contradiction. Thus e must be a receive event. If $e \notin \overline{V}$ then, since the corresponding send event belongs to L_1, we would have $|L_1 v|_{t^{-1}(S)} - |L_1 v|_{t^{-1}(R)} \geq b+2$ (note that already $|\pi(L_1 v)|_{t^{-1}(S)} - |\pi(L_1 v)|_{t^{-1}(R)} = b+1$). Thus $e = \overline{u}$ for some $u \in V$. We cannot have $v \prec \overline{u}$, since by the construction of $M(G)$ this would imply that a non-empty sequence of events occurs between v and $\overline{u}$. It follows that $L' = L_1 \overline{u} v L_3$ is also a linearization of $M(G)$ that is globally-bounded by $b+1$. Since furthermore the number of prefixes K of L' such that $|K|_{t^{-1}(S)} - |K|_{t^{-1}(R)} = b+1$ is smaller than for L, we have a contradiction. This proves claim 3 and the theorem. □

For universal-global-boundedness we can obtain a polynomial time solution using flow theory:

Theorem 3.5. $\forall_{\mathrm{glob}}$-MSC-BOUNDED *can be solved in time* $O(n^2 \log(n))$.

Proof. In order to check universal-global-boundedness we consider the complementary problem, namely whether given a finite MSC M and $b \in \mathbb{N}$ there exists a configuration C of M such that $\mathrm{gus}(C, M) > b$. This question can be answered in polynomial time using the min-flow max-cut theorem, see e.g. [9]. More precisely we construct from M a dag as follows: View M as a dag, where the nodes are the events of M, and the edges are the messages of M plus pairs of events (e, f) such that e immediately precedes f on some process. To this dag we add two nodes σ and τ. We add an edge from σ to each minimal event of M, and similarly we add an edge from each maximal event of M to τ. Let us call the resulting dag G and let E be its edge set. Note that G contains precisely $|M|$ nodes of degree 3 and two nodes of degree $|P|$ (namely σ and τ). Thus, $|E| = \frac{3}{2} \cdot |M| + |P| \in O(|M|)$. To each edge (v, w) of G we assign an *upper capacity* $c_{v,w}$ and a *lower capacity* $\ell_{v,w}$ as follows: All edges receive the upper capacity ∞. For all messages $(s, m(s))$ of M we let $\ell_{s,m(s)} = 1$, whereas for all other edges (v, w) of G we let $\ell_{v,w} = 0$. By the min-flow max-cut theorem the minimum value of a (σ, τ)-flow of G is equal to the maximum of $\sum_{(v,w)\in E\cap S\times T} \ell_{v,w} - \sum_{(v,w)\in E\cap T\times S} c_{v,w}$, where the maximum is taken over all partitions $\{S, T\}$ of the nodes of G with $\sigma \in S$ and $\tau \in T$. By the choice of the capacities this is precisely the maximum of $\mathrm{gus}(C, M)$, taken over all configurations C of M. Finally, since all upper capacities are ∞, we can use [19] in order to compute $\max(\sum_{(v,w)\in E\cap S\times T} \ell_{v,w} - \sum_{(v,w)\in E\cap T\times S} c_{v,w})$ in time $O(n \log(n) r)$, where $n = |M| + 2$ is the number of nodes of G and $r \in O(|M|)$ is the number of edges in a transitive reduction of G. $\square$

We note also that before computing the minimal flow in the previous proof, we may reduce the graph G as follows: If two nodes v and w are such that v immediately precedes w on some process and either v has out-degree one, or w has in-degree one, then the edge (v, w) can be contracted to a single node. This reduction step can be iterated as long as possible. The resulting graph may be considerably smaller than G.

4 Bounded communication in HMSCs

The following result follows easily from Theorem 3.4.

Proposition 4.1. $\exists_{\mathrm{glob}}$-HMSC-BOUNDED *is NP-complete.*

Proof. The lower bound follows directly from Theorem 3.4. For the upper bound note that an HMSC $H = (V, \rightarrow, P, \mu, v)$ is $\exists^b_{\mathrm{glob}}$-bounded if and only if for every node $u \in V$ the MSC $\mu(u)$ is $\exists^b_{\mathrm{glob}}$-bounded. $\square$

Analogously to Proposition 4.1 it follows that $\exists_{\mathrm{loc}}$-HMSC-BOUNDED can be solved in linear time.

For the universal-boundedness question for HMSCs we need the concept of the *communication graph* $G(M)$ of a finite MSC $M = (\mathcal{E}, P, \lambda, t, m, \prec)$. It is defined as $G(M) = (P, \mapsto)$, where $p \mapsto q$ if and only if there exists a message $(s, m(s))$ in M with $\lambda(s) = p$ and $\lambda(m(s)) = q$. We say that M is *locally strongly connected* if $G(M)$ is locally strongly connected (i.e., $G(M)$ is a disjoint union of strongly connected subgraphs). Finally an HMSC $H = (V, \rightarrow, P, \mu, v)$ is locally strongly connected if for every cycle $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1$ of $(V, \rightarrow)$ the MSC $\mu(v_1)\mu(v_2)\cdots\mu(v_n)$ is locally strongly connected. The notion of a locally strongly connected HMSC should not be confused with the related notion of a bounded HMSC [2] (called locally synchronized in [15]): an HMSC $H = (\mathcal{E}, P, \lambda, t, m, \prec)$ is called *bounded* if for every cycle $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1$ of $(V, \rightarrow)$, the restriction of the communication graph $G = G(\mu(v_1)\mu(v_2)\cdots\mu(v_n))$ to the non-isolated nodes of G is strongly connected. The first example from Figure 2 is not bounded but locally strongly connected. The second one is both bounded and locally strongly connected. Finally, the third HMSC is neither bounded nor locally strongly connected.

It is easy to see that H is locally strongly connected if and only if for all simple cycles $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n \rightarrow v_1$ (i.e, $v_i \neq v_j$ for $i \neq j$) the MSC $\mu(v_1)\mu(v_2)\cdots\mu(v_n)$ is locally strongly connected. For this just note that if we have cycles $v_1 \rightarrow \cdots \rightarrow v_n \rightarrow v_1$ and $w_1 \rightarrow \cdots \rightarrow w_m \rightarrow w_1$ with $v_1 = w_1$ then $G = G(\mu(v_1)\cdots\mu(v_n)\mu(w_2)\cdots\mu(w_m)\mu(v_1))$ is the union of the two communication graphs $G(\mu(v_1)\cdots\mu(v_n))$ and $G(\mu(w_1)\cdots\mu(w_m))$. Thus if both of them are locally strongly connected then the same holds for G. It was shown in [13] that an HMSC H is $\forall^b_{\mathrm{loc}}$-bounded for some b if and only if H is locally strongly connected. The fact that H is locally strongly connected if H is $\forall^b_{\mathrm{loc}}$-bounded by some b is quite easy to see. Lemma 4.2 below allows us to present a simpler proof of the other direction of the result of [13]. Moreover, it allows to consider paths of polynomial length for establishing the existence of configurations of a given buffer size.

Proposition 4.2. *Let the HMSC $H = (V, \rightarrow, P, \mu, v)$ be locally strongly connected. Let $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_m$ be a path in $(V, \rightarrow)$ and let C be a configuration of the MSC $M = \mu(u_1)\cdots\mu(u_m)$. Then there exists a path $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n$ in $(V, \rightarrow)$ and a configuration D of the MSC $N = \mu(v_1)\cdots\mu(v_n)$ such that $n \leq |P| \cdot |V|$, $\mathrm{gus}(C, M) = \mathrm{gus}(D, N)$, and $\mathrm{lus}(C, M) = \mathrm{lus}(D, N)$.*

Proof. Let $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_m$ be a path in $(V, \rightarrow)$ and let C be a configuration of $M = \mu(u_1)\cdots\mu(u_m)$. If $m \leq |P| \cdot |V|$, then we are ready. Thus, assume that $m > |P| \cdot |V|$. The idea is to find a loop within the path $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_m$ such that all send events $s \in C$ produced in this loop are matched within C. Then we may shorten the path by this loop. This can be iterated until the length of the path is at most $|P| \cdot |V|$.

Formally, assume that $P = \{1, \ldots, |P|\}$. For each $i \in P$ let e_i be the maximal event of M which is located on process i and which belongs to C, if this event exists. Otherwise e_i is undefined. By reordering the processes suitably, we may assume that $e_1, \ldots, e_{k-1}$ are undefined and event e_i belongs to the MSC $\mu(u_{m_i})$ for $k \leq i \leq |P|$, where $1 \leq m_k \leq m_{k+1} \leq \cdots \leq m_{|P|} \leq m$. Let $m_0 = m_1 = \cdots = m_{k-1} = 0$. Since $m > |P| \cdot |V|$ either $m > m_{|P|}$, or $m_{\ell+1} - m_\ell > |V|$ for some $0 \leq \ell < |P|$. In the first case we may take the path $u_1 \to u_2 \to \cdots \to u_{m-1}$ for $v_1 \to \cdots \to v_n$. Now assume that $m_{\ell+1} - m_\ell > |V|$ for some $0 \leq \ell < |P|$. Then there exist $m_\ell < i < j \leq m_{\ell+1}$ such that $u_i = u_j$. Note that $\mu(u_i)\mu(u_{i+1}) \cdots \mu(u_{j-1}) \cap C$ contains precisely the events of $\mu(u_i)\mu(u_{i+1}) \cdots \mu(u_{j-1})$ that are located on some process $p \in \{\ell+1, \ldots, |P|\}$. Let G be the communication graph of the factor $\mu(u_i)\mu(u_{i+1}) \cdots \mu(u_{j-1})$ of M. Then G must be locally strongly connected. Assume that there exists a message (s, r) in $\mu(u_i)\mu(u_{i+1}) \cdots \mu(u_{j-1})$ such that $s \in C$ and $r \notin C$. Since $i > m_\ell$ and $j - 1 < m_{\ell+1}$, the send s is located on some process $p \geq \ell + 1$ and the receive r is located on some process $q \leq \ell$. Since G is locally strongly connected, there exists a directed path from process $q \leq \ell$ to process $p \geq \ell+1$ in G. Thus there exists a message (s', r') in $\mu(u_i)\mu(u_{i+1}) \cdots \mu(u_{j-1})$ such that s' is located on some process $q' \leq \ell$ and r' is located on some process $p' \geq \ell+1$. It follows that $r' \in C$ and $s' \notin C$. But this is a contradiction to the fact that C is a configuration of M. Thus the message (s, r) cannot exist. It follows that in the shortened MSC $N = \mu(u_1) \cdots \mu(u_{i-1})\mu(u_j) \cdots \mu(u_m)$, the set of all events that belong to C form a configuration D such that $\mathrm{gus}(C, M) = \mathrm{gus}(D, N)$ and $\mathrm{lus}(C, M) = \mathrm{lus}(D, N)$. Moreover, $u_1 \to \cdots \to u_{i-1} \to u_j \to \cdots \to u_m$ is a path in $(V, \to)$. $\square$

For an HMSC $H = (V, \to, P, \mu, v)$ let $\sigma(H) = \max\{\frac{|\mu(u)|}{2} \mid u \in V\}$, which is the maximal number of send events in one of the finite MSCs $\mu(u)$, $u \in V$.

Corollary 4.3. *Let the HMSC $H = (V, \to, P, \mu, v)$ be locally strongly connected. Then H is $\forall^b_{\text{glob}}$-bounded (and hence also $\forall^b_{\text{loc}}$-bounded) for some $b \leq |P| \cdot |V| \cdot \sigma(H)$. Furthermore if $b \in \mathbb{N}$ is minimal such that H is $\forall^b_{\text{glob}}$-bounded (resp. $\forall^b_{\text{loc}}$-bounded) then there exist a path $v_1 \to v_2 \to \cdots \to v_n$ in $(V, \to)$ and a configuration C of the MSC $M = \mu(v_1) \cdots \mu(v_n)$ such that $n \leq |P| \cdot |V|$ and* $\mathrm{gus}(C, M) = b$ *(resp.* $\mathrm{lus}(C, M) = b$*).*

Proof. For the first statement assume that there exist a path $v = u_1 \to u_2 \to \cdots \to u_n$ in $(V, \to)$ and a configuration C in the MSC $M = \mu(u_1) \cdots \mu(u_n)$ such that $\mathrm{gus}(C, M) > |P| \cdot |V| \cdot \sigma(H)$. Furthermore let n be minimal with the properties above. Since each of the MSCs $\mu(u_i)$ can only contribute $\sigma(H)$ messages (s, r) with $s \in C$ and $r \notin C$, it follows that $n > |P| \cdot |V|$, which by Proposition 4.2 gives us a contradiction to the minimality of n. The second statement follows immediately from Proposition 4.2. $\square$

We should remark that Theorem 2.8 of [13], which corresponds to the first statement of Corollary 4.3, is formulated in terms of regular MSC-expressions instead of HMSCs. Using the notation of [13], we can prove the following statement in the same fashion as Corollary 4.3.

Lemma 4.4. *Let L be a set of finite MSCs over some fixed set of processes P. Assume that each $M \in L$ is locally strongly connected and $\forall^b_{\text{glob}}$-bounded (resp. $\forall^b_{\text{loc}}$-bounded). Then every MSC in L^* is $\forall^{|P|\cdot b}_{\text{glob}}$-bounded (resp. $\forall^{|P|\cdot b}_{\text{loc}}$-bounded).*

Theorem 4.5. $\forall_{\text{glob}}$-HMSC-BOUNDED *is coNP-complete.*

Proof. We show that the complementary problem is NP-complete:

INPUT: HMSC H and positive integer b.

QUESTION: Is there an MSC $M \in \text{msc}(H)$ which is not $\forall^b_{\text{glob}}$-bounded?

An NP-algorithm that solves this problem proceeds as follows: Let us fix $H = (V, \to, P, \mu, v)$. First we guess in $(V, \to)$ a simple cycle $u_1 \to u_2 \to \cdots \to u_m \to u_1$ and a path $v_1 \to v_2 \to \cdots \to v_n$ with $n \leq |P| \cdot |V|$ together with a configuration C in the MSC $M = \mu(v_1) \cdots \mu(v_n)$. Then the algorithm outputs "yes" if and only if either the communication graph of the MSC $\mu(u_1)\mu(u_2) \cdots \mu(u_m)$ is not locally strongly connected or $\text{gus}(C, M) \geq b + 1$. We claim that this NP-algorithm is correct. Clearly if the algorithm outputs "yes" then there exists an MSC in $\text{msc}(H)$ which is not $\forall^b_{\text{glob}}$-bounded (recall that we assume that every $u \in V$ is accessible from the initial node v). On the other hand assume that there exists an MSC in $\text{msc}(H)$ which is not $\forall^b_{\text{glob}}$-bounded. Either H is not locally strongly connected, which can be detected by the algorithm, or there exists some $b' > b$ such that H is $\forall^{b'}_{\text{glob}}$-bounded, but not $\forall^{b'-1}_{\text{glob}}$-bounded. By Corollary 4.3 there exists a path $v_1 \to v_2 \to \cdots \to v_n$ with $n \leq |P| \cdot |V|$ together with a configuration C in the MSC $M = \mu(v_1) \cdots \mu(v_n)$ such that $\text{gus}(C, M) = b' \geq b + 1$. Again both this path and the configuration can be detected by the algorithm.

In order to prove NP-hardness, we reduce SAT to our problem. A construction similar to the following one was also used in [16]. Let $\{x_1, \ldots, x_n\}$ be a set of propositional variables, and let $C = \{C_1, \ldots, C_m\}$ be a set of clauses, where each clause C_i consists of variables and negated variables. We construct an HMSC $H = (V, \to, P, \mu, v)$ such that C is satisfiable if and only if there exists $M \in \text{msc}(H)$ which is not $\forall^{m-1}_{\text{glob}}$-bounded. Let $V = \{v, v_1, \overline{v}_1, \ldots, v_n, \overline{v}_n\}$ and $\to\, = \{(v, v_1), (v, \overline{v}_1)\} \cup \{(v_i, v_{i+1}), (v_i, \overline{v}_{i+1}), (\overline{v}_i, v_{i+1}), (\overline{v}_i, \overline{v}_{i+1}) \mid 1 \leq i < n\}$. The set of processes is $P = \{c_i, c'_i \mid 1 \leq i \leq m\}$. It remains to define the MSCs $\mu(u)$ for $u \in V$. The MSC $\mu(v)$ is empty. The MSC $\mu(v_i)$ contains a message from process c_j to process c'_j and back from c'_j to c_j if $x_i \in C_j$. Similarly the MSC $\mu(\overline{v}_i)$ contains a message from process c_j to process c'_j and back from c'_j

to c_j if $\overline{x}_i \in C_j$. No other messages are present. It follows that C is satisfiable if and only if there exists an MSC $M \in \mathrm{msc}(H)$ such that for every $1 \leq j \leq m$ the projection of M onto the processes c_j and c'_j is a non-empty iteration of the MSC that sends a message from c_j to c'_j and back. This holds if and only if there exists an MSC $M \in \mathrm{msc}(H)$ that is not $\forall_{\mathrm{glob}}^{m-1}$-bounded. $\square$

It should be noted that Theorem 4.5 holds no matter whether the buffer bound $b \in \mathbb{N}$ is represented in unary or binary. Our lower bound proof holds also for the unary representation, whereas the upper bound proof holds for the binary representation. Furthermore note that the HMSC H used for the lower-bound proof is based on an acyclic graph $(V, \rightarrow)$, thus H defines a finite set of MSCs.

Theorem 4.6. $\forall_{\mathrm{loc}}$-HMSC-BOUNDED *is coNP-complete. Moreover this problem is coNP-complete even if the input parameter b is fixed to $b = 1$.*

Proof. Membership in coNP follows in exactly the same way as in Theorem 4.5. In order to show coNP-hardness we will reduce NAE-SAT (not-all-equal-SAT), see e.g. [4, p.259], to the complement of our problem. We consider a collection of m clauses $C = \{C_1, \ldots, C_m\}$ each of size three, over variables $\{x_1, \ldots, x_n\}$ and we want to find out whether there is a variable assignment such that for each clause C_i, the literals of C_i do not all have the same value. We will construct an HMSC H such that for some fixed channel (A, B) of H there is an execution with more than one send in the corresponding buffer if and only if there is an assignment as above for C. For every channel different from (A, B) each execution of H will contain at most one message for that buffer, so channels different from (A, B) will be universally bounded by 1.

The graph underlying H is similar to the one from Theorem 4.5. The node set is $V = \{v, v_1, \overline{v}_1, \ldots, v_n, \overline{v}_n, v'\}$, and $\rightarrow = \{(v, v_1), (v, \overline{v}_1), (v_n, v'), (\overline{v}_n, v')\} \cup \{(v_i, v_{i+1}), (v_i, \overline{v}_{i+1}), (\overline{v}_i, v_{i+1}), (\overline{v}_i, \overline{v}_{i+1}) \mid 1 \leq i < n\}$. Again, vertex v_i stands for x_i true, whereas $\overline{v}_i$ stands for x_i false. The HMSC H uses processes A, B and processes $P_{j,1}, P_{j,2}, P_{j,3}, P_{j,4}, N_{j,1}, N_{j,2}, N_{j,3}, N_{j,4}$ for $j \in \{1, \ldots, m\}$ ranging over the clauses. We denote as P_j-group the processes in $P_{j,1}, P_{j,2}, P_{j,3}, P_{j,4}$, and as N_j-group the processes in $N_{j,1}, N_{j,2}, N_{j,3}, N_{j,4}$. The initial node v contains a message from A to B, followed by one message from B to each of $P_{j,1}$ and $N_{j,1}$. Each node v_i contains a message in the P_j-group for every clause C_j where x_i occurs positively, and a message in the N_j-group for every clause C_j where x_i occurs negatively. Precisely, v_i contains a message from $P_{j,k}$ to $P_{j,k+1}$, if x_i is the k-th literal of C_j, $k \in \{1, 2, 3\}$, and it contains a message from $N_{j,\ell}$ to $N_{j,\ell+1}$ if $\overline{x_i}$ is the ℓ-th literal of C_j, $\ell \in \{1, 2, 3\}$. Here, it is important that the ordering of the literals in each clause respects the order $x_1, \ldots, x_n$ of the variables that is determined by the graph $(V, \rightarrow)$. For the nodes $\overline{v}_i$ we switch the roles of P_j and N_j. The final node v' is labeled by messages from each of $P_{j,4}$, $N_{j,4}$ to A, followed by a message from A to B.

Note that paths from v to v' correspond precisely to variable assignments. For instance, let $C_1 = \overline{x_1} \vee x_2 \vee x_5$ and $C_2 = x_1 \vee x_3 \vee \overline{x_4}$. Node v_1 will be labeled by a message from $N_{1,1}$ to $N_{1,2}$ and a message from $P_{2,1}$ to $P_{2,2}$, whereas $\overline{v_1}$ will be labeled by a message from $P_{1,1}$ to $P_{1,2}$ and a message from $N_{2,1}$ to $N_{2,2}$.

For a given assignment/path from the initial node v to v' we note that the first receive of channel type $R(A, B)$ precedes in the visual order the second send of channel type $S(A, B)$ if and only if there is some j and a $\prec$-path either from $P_{j,1}$ to $P_{j,4}$, or from $N_{j,1}$ to $N_{j,4}$ (notice that there is no message between N-processes and P-processes). For instance, the assignment corresponding to the path $v, v_1, v_2, v_3, \overline{v_4}, v_5, v'$ yields for the second clause a message from $P_{2,1}$ to $P_{2,2}$ in node v_1, a message from $P_{2,2}$ to $P_{2,3}$ in node v_3, and a message from $P_{2,3}$ to $P_{2,4}$ in node $\overline{v_4}$. For the first clause the path yields a message from $N_{1,1}$ to $N_{1,2}$ (node v_1), a message from $P_{1,2}$ to $P_{1,3}$ (node v_2), and a message from $P_{1,3}$ to $P_{1,4}$ (node v_5).

Hence, the first receive r of type $R(A, B)$ precedes in the visual order the second send s of type $S(A, B)$ if and only if there is a clause C_j in which all literals have the same value under the variable assignment corresponding to the chosen path from v to v'. But this is exactly the case where C is not satisfied as an NAE-SAT instance. Moreover, r precedes s in a given execution of H if and only if the MSC corresponding to this execution is $\forall^1_{\text{loc}}$-bounded. $\square$

Let us remark that a simple extension of the construction from the previous proof also shows that it is coNP-complete, whether a given HMSC is $\forall^b_{\text{loc}}$-bounded for some b, i.e., whether it is locally strongly connected. For this we have to add an edge from the final node v' back to the initial node v. Furthermore we have to add confirm messages that ensure that only the buffer (A, B) may contain an arbitrary number of undelivered messages. For this we simply confirm each message from a process p to q where $(p, q) \neq (A, B)$ directly by a message from q back to p.

5 Local boundedness and nested MSCs

A *nested MSC (nMSC)* is a sequence $M = (M_k)_{1 \leq k \leq m}$ of modules M_k. Each module M_k is defined as an MSC to which we add references to modules M_i with $k < i \leq m$, by specifying the start and end of each reference to M_i on the process lines belonging to M_i. We use the definition of [5], where messages are restricted to be matched on the same hierarchy level , i.e., within the same module (in particular, we don't consider ports), but they can cross references to submodules, see Figure 4.

In principle, a message may also cross a module in the other direction, i.e., the

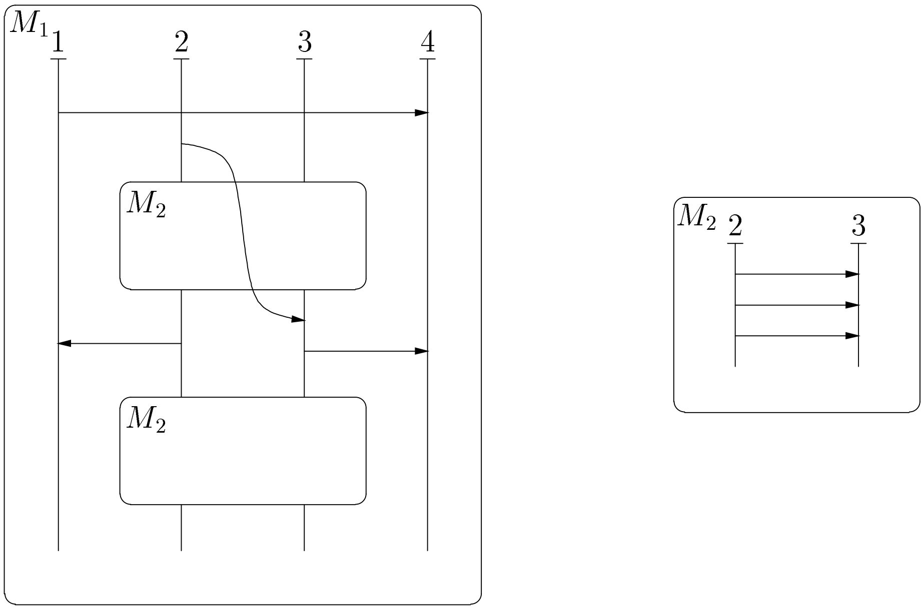

Fig. 4. An nested MSC, where a message crosses a submodule

send event succeeds the end of the module on the sending process, whereas the receive precedes the start of the module on the receiving process. It has to be only guaranteed that the overall picture is acyclic, which can be checked in polynomial time using standard techniques for hierarchically defined graphs [10].

Each module M_k of M can be expanded to a finite MSC $\mathrm{flat}(M_k)$ by replacing inductively in M_k each reference to a module M_i $(i > k)$ by the MSC $\mathrm{flat}(M_i)$. Finally we define $\mathrm{flat}(M) = \mathrm{flat}(M_1)$. Every reference to some module M_i $(i > 1)$ contributes events to $\mathrm{flat}(M)$, and we say that these events *result from a reference to* M_i. The events of $\mathrm{flat}(M)$ that are defined directly in the main module M_1 are called *top level events*. For instance, the two events of channel type $S(1, 4)$ and $R(1, 4)$, respectively, in the above nMSC are top level events. Let $P(M_i)$ be the set of processes of $\mathrm{flat}(M_i)$. Note that $\mathrm{flat}(M)$ may be of exponential size in the description of M.

The definition above is analogous to the notion of straight-line expressions, where any expression may use in its definition sub-expressions that were previously defined. In fact, it is easy to calculate from M a straight-line expression for every $p \in P$ that represents the projection $\pi_p(M)$ of $\mathrm{flat}(M)$ to process p. Moreover, any event e of $\mathrm{flat}(M)$ can be represented by its (binary coded) position $\mathrm{pos}(e)$ in $\pi_{\lambda(e)}(M)$ (here λ refers to $\mathrm{flat}(M)$).

Lemma 5.1. *The following computations can be done in polynomial time:*

- *Compute* $\mathrm{pos}(m(e))$ *from* $\mathrm{pos}(e)$*, where* e *is a send event.*
- *Given* $\mathrm{pos}(e)$ *and a channel type* ct*, compute* $\mathrm{pos}(f)$*, where* f *is the smallest*

event with $\lambda(f) = \lambda(e)$*,* $\mathrm{pos}(f) > \mathrm{pos}(e)$*, and* $\mathrm{ct}(f) = \mathrm{ct}$ *(if existing).*
- *Given* i *and a channel type* ct*, calculate the position of the* i*-th event of channel type* ct *(if existing).*

All computations in the previous lemma can be reduced to simple arithmetic.

If a module M_k contains a reference to a module M_i, $k < i$, then $P(M_i) \subseteq P(M_k)$. Thus, if M_k contains a message (s, r) from p to q that crosses a reference to module M_i, $i > k$, (i.e., p, q are processes of M_i and s precedes the beginning of M_i on p, whereas r succeeds the end of M_i on q), then M_i cannot contain any message from p to q, unless the FIFO-restriction is violated. For instance, the nMSC above does not satisfy the FIFO-restriction. Checking the FIFO-restriction is feasible in polynomial time.

We show in this section that both versions (existential and universal) of the local-boundedness problem for nMSCs can be solved in polynomial time, provided that the nMSC M satisfies the FIFO-restriction, i.e., $\mathrm{flat}(M)$ satisfies the FIFO-restriction. Of course, the algorithms must exploit the hierarchy, since nMSCs can be exponentially more succinct than the MSCs they define (i.e., a module M_k of M may have exponentially many copies in the MSC $\mathrm{flat}(M)$).

For our further considerations let us fix a channel bound b and an nMSC $M = (M_k)_{1 \leq k \leq m}$. For $1 \leq k \leq m$ let $\stackrel{k}{\rightsquigarrow}$ (resp. $\prec^k$) be the $\rightsquigarrow_b$-relation (resp. visual order) associated with $\mathrm{flat}(M_k)$. Recall that $r \rightsquigarrow_b s$ was defined in Section 3 for r the j-th receive of type $R(p, q)$, and s the $(j + b)$-th send of type $S(p, q)$. Furthermore let $\prec \; = \; \prec^1$, $\rightsquigarrow \; = \; \stackrel{1}{\rightsquigarrow}$, and $P = P(M_1)$. Note that given $\mathrm{pos}(r)$ we can calculate $\mathrm{pos}(s)$ with $r \rightsquigarrow s$ (in case s exists) in polynomial time. The following lemma is easy to show.

Lemma 5.2. *Let the nMSC* $M = (M_k)_{1 \leq k \leq m}$ *satisfy the FIFO-restriction. Let* $\mathcal{E}$ *be the set of events of* $\mathrm{flat}(M)$ *that result from a reference to some module* M_k*,* $k > 1$*. Then* $\rightsquigarrow \cap \, (\mathcal{E} \times \mathcal{E}) \; = \stackrel{k}{\rightsquigarrow}$*.*

Note that Lemma 5.2 does not hold for the case where the nMSC violates the FIFO-restriction. In this case different occurrences of the MSC $\mathrm{flat}(M_k)$ in $\mathrm{flat}(M)$ may have different local $\rightsquigarrow$-relations depending on their context. Consider for instance the nMSC from the above example and let $b = 2$. In the first occurrence of $\mathrm{flat}(M_2)$ there is a $\stackrel{2}{\rightsquigarrow}$-edge from the first receive of type $R(2, 3)$ to the second send of type $S(2, 3)$, which is due to the crossing message in M_1. On the other hand, in the second occurrence of $\mathrm{flat}(M_2)$ there is a $\stackrel{2}{\rightsquigarrow}$-edge from the first receive of type $R(2, 3)$ to the third send of type $S(2, 3)$.

Lemma 5.3. *Suppose that the relation $\rightsquigarrow \cup \prec$ contains a cycle. Then for some $k \leq |P|$ there exists a cycle of the form $r_1 \rightsquigarrow s_1 \prec r_2 \rightsquigarrow s_2 \prec \cdots \prec r_k \rightsquigarrow s_k \prec r_1$.*

The proof of the previous lemma is simple: we consider a cycle of $\rightsquigarrow \cup \prec$ that uses a minimal number of $\rightsquigarrow$-edges $r_i \rightsquigarrow s_i$. Then $\lambda(s_i) \neq \lambda(s_j)$, for any $i \neq j$, because otherwise we could shorten the cycle.

For the rest of this section it is useful to add for every reference A in some module M_i and every process p that is used by A a new local event before (resp. after) the beginning (resp. end) of A. These local events do not have a corresponding send or receive. Note that the new events that we have added to the main module M_1 are top level events.

Theorem 5.4. *The following problem can be solved in polynomial time:*

INPUT: nMSC M satisfying the FIFO-restriction and positive integer b.

QUESTION: Is $\mathrm{flat}(M)$ $\exists^b_{\mathrm{loc}}$-bounded?

Proof. By Lemma 3.1(1) it suffices to verify that the transitive closure of the relation $\prec \cup \rightsquigarrow$ associated with the MSC $\mathrm{flat}(M)$ is acyclic. Of course, we cannot explicitly generate the $\rightsquigarrow$-edges, since there can be exponentially many $\rightsquigarrow$-edges leading out of a copy of M_i within M_k, or vice versa. More precisely, there may be b $\rightsquigarrow$-edges leading out of a reference, which is an exponential number due to the binary coding of b. By Lemma 5.3 it suffices to look for a cycle containing at most $|P|$ new $\rightsquigarrow$-edges.

Our algorithm will first check recursively, whether for some module M_i, $i \geq 2$, there exists a cycle in the relation $\prec^i \cup \overset{i}{\rightsquigarrow}$. By Lemma 5.2 this would result in a cycle in $\prec \cup \rightsquigarrow$. Thus assume that for every submodule M_i, $\prec^i \cup \overset{i}{\rightsquigarrow}$ is acyclic. Now, if $\prec \cup \rightsquigarrow$ contains a cycle nevertheless, then this cycle has to visit a top level event of M (this is due to the new local events marking the beginning and end of references). Hence it suffices to show how to calculate a suitable representation of the set $\mathrm{Succ}_{(\prec \cup \rightsquigarrow)^+}(e) = \{f \mid (e, f) \in (\prec \cup \rightsquigarrow)^+\}$ in polynomial time for every top level event e. Then we just have to check whether $e \in \mathrm{Succ}_{(\prec \cup \rightsquigarrow)^+}(e)$. Our representation of $\mathrm{Succ}_{(\prec \cup \rightsquigarrow)^+}(e)$ cannot consist in an enumeration of this set, because it may be of exponential size. Instead, we represent $\mathrm{Succ}_{(\prec \cup \rightsquigarrow)^+}(e)$ by a tuple $(\ell_p)_{p \in P}$ of positions, one for each process p. The position ℓ_p corresponds to the first event on process p that belongs to the set $\mathrm{Succ}_{(\prec \cup \rightsquigarrow)^+}(e)$ (if this event does not exist, then $\ell_p = \infty$).

Let us first describe how we can compute the set $\mathrm{Succ}_{\prec}(e) = \{f \mid e \prec f\}$ of $\prec$-successors of e for any given (not necessarily top level) event e of $\mathrm{flat}(M)$, which is represented by $\mathrm{pos}(e)$. Note that if $e \prec f$, then there exists a chain

$e = e_1 \preceq f_1 \prec e_2 \prec f_2 \prec \cdots \prec f_t \prec e_{t+1} \preceq f_{t+1} = f$ with $\lambda(e_i) = \lambda(f_i)$, $m(f_i) = e_{i+1}$, and $t < |P|$. Here $\preceq$ denotes the reflexive closure of the visual order $\prec$. The computation of $\mathrm{Succ}_\prec(e)$ can be performed by induction on t. We start by setting $\ell_{\lambda(e)} = \mathrm{pos}(f)$, where f is the direct successor of e on process $\lambda(e)$, and $\ell_{\lambda(m(e))} = \mathrm{pos}(m(e))$ in case e is a send event. All ℓ_p that are not defined in this way are set to ∞. For the inductive step we determine for all $\ell_p < \infty$ and all processes $q \neq p$ the first send s of type $S(p,q)$ with $\mathrm{pos}(s) > \ell_p$, and we compute the minimum between $\mathrm{pos}(m(s))$ and ℓ_q.

In order to compute $\mathrm{Succ}_{(\prec\cup\rightsquigarrow)^+}(e)$ for a top level event e, we start with $\mathrm{Succ}_{(\prec\cup\rightsquigarrow)^+}(e) = \mathrm{Succ}_\prec(e)$, represented by the tuple of positions $(\ell_p)_{p\in P}$. For the inductive step we determine for all $\ell_q < \infty$ and all processes $p \neq q$ the first receive r of type $R(p,q)$ with $\mathrm{pos}(r) > \ell_q$, and we compute the send s of type $S(p,q)$ with $r \rightsquigarrow s$. Then we compute for each such s the set $\mathrm{Succ}_\prec(s)$ and we build the minima with $(\ell_p)_{p\in P}$ on every process. By Lemma 5.3 this step has to be repeated only $|P|$ times.

The above computation of the set $\mathrm{Succ}_{(\prec\cup\rightsquigarrow)^+}(e)$ relies heavily on the FIFO-restriction: We use the fact that in order to get a better approximation for ℓ_q, it suffices to consider the *earliest* message from p to q, where the corresponding send succeeds position ℓ_p, and similarly for $\rightsquigarrow$-edges. Finally, note that by Lemma 5.1 all computations can be done in polynomial time. $\square$

Theorem 5.5. *The following problem can be solved in polynomial time:*

INPUT: nMSC M satisfying the FIFO-restriction and positive integer b.

QUESTION: Is $\mathrm{flat}(M)$ $\forall^b_{\mathrm{loc}}$*-bounded?*

Proof. We apply Lemma 3.1(2), that is, we check whether $\rightsquigarrow \subseteq \prec$. First we check this property inductively for every $\mathrm{flat}(M_i)$ with $i \geq 2$. Assume that for all $i \geq 2$ we have $\overset{i}{\rightsquigarrow} \subseteq \prec^i$ (otherwise we can conclude with Lemma 5.2 that $\rightsquigarrow \subseteq \prec$ does not hold). Now we calculate for every top level event e the future $\mathrm{Succ}_\prec(e)$ and past $\mathrm{Pred}_\prec(e) = \{f \mid f \prec e\}$, see the proof of Theorem 5.4. Recall that $\mathrm{Succ}_\prec(e)$ is represented as a tuple $(\ell^e_p)_{p\in P}$ of minimal positions. Analogously, $\mathrm{Pred}_\prec(e)$ is represented as a tuple of maximal positions $(k^e_p)_{p\in P}$ (i.e., $k^e_p = \mathrm{pos}(f)$, where f is the maximal event in $\lambda^{-1}(p) \cap \mathrm{Pred}_\prec(e)$). Due to the local events that mark the beginning and end of every reference, we have $e \prec f$ for events e and f if and only if $e \in \mathrm{Pred}_\prec(t)$ and $f \in \mathrm{Suc}_\prec(t)$ for some top level event t.

Since we already know that $\overset{i}{\rightsquigarrow} \subseteq \prec^i$ for every $i \geq 2$, it follows that $\rightsquigarrow \subseteq \prec$ if and only if the following three conditions hold:

(1) For every top level receive r of M we have $s \in \mathrm{Succ}_{\prec}(r)$ for the unique send s with $r \rightsquigarrow s$.
(2) For every top level send s of M we have $r \in \mathrm{Pred}_{\prec}(s)$ for the unique receive r with $r \rightsquigarrow s$.
(3) Let A, B be two different references in M_1. For all r, s with $r \rightsquigarrow s$ and r resulting from A (s resulting from B, resp.) there exists a top level event e with $r \in \mathrm{Pred}_{\prec}(e)$ and $s \in \mathrm{Succ}_{\prec}(e)$ (hence, $r \prec s$).

(1) and (2) can be easily verified using Lemma 5.1. For (3) let us fix different references A and B in M_1 and a channel (p, q) such that A and B both use p and q. For every top level event e let α_q^e be the position on process q of the smallest receive of channel type $R(p, q)$ that results from reference A and that is larger than position k_q^e (i.e., this receive does not belong to $\mathrm{Pred}_{\prec}(e)$). If this receive does not exist, then we set $\alpha_q^e = \infty$. Similarly let β_p^e be the position on process p of the largest send of channel type $S(p, q)$ that results from reference B and that is smaller than position ℓ_p^e (i.e., this send does not belong to $\mathrm{Succ}_{\prec}(e)$). We set $\beta_p^e = -\infty$, if this send does not exist. Next let α_q (resp. β_p) be the position of the last receive (resp. first send) of channel type $R(p, q)$ (resp. $S(p, q)$) that results from reference A (resp. B). All these positions can be computed in polynomial time. What we have to check is whether there exist $r \rightsquigarrow s$ such that

- $\lambda(s) = p$, $\lambda(r) = q$,
- $\mathrm{pos}(r) \leq \alpha_q$, $\mathrm{pos}(s) \geq \beta_p$, and
- for every top level event e, either $\mathrm{pos}(r) \geq \alpha_q^e$ or $\mathrm{pos}(s) \leq \beta_p^e$.

Then not $r \prec s$, but r (resp. s) results from reference A (resp. B). To check this we calculate the number of receives of channel type $R(p, q)$ up to position α_q and α_q^e, respectively, on process q. Let us denote these numbers by m_q and m_q^e, respectively. Similarly let n_p and n_p^e denote the number sends of channel type $S(p, q)$ up to position β_p and β_p^e, respectively, on process p. Now it suffices to check whether there exists x such that $x \leq m_q$, $x + b \geq n_p$, and for all top level-events e, either $x \geq m_q^e$ or $x + b \leq n_p^e$, which is of course easy to do. $\square$

6 Fixed number of processes

In practice, the set of processes of an MSC can be much smaller than the number of messages. Hence we are interested in the complexity of our problems when the number of processes is fixed. The main result of this section states that for a fixed number of processes all the variants of the channel boundedness problem can be solved in polynomial time (more precisely in nondeterministic logspace).

Theorem 6.1. *Let P be a fixed set of processes. The following problem is in NL:*

INPUT: MSC M over the set of processes P and positive integer b.

QUESTION: Is M $\exists^b_{\text{glob}}$-bounded?

Proof. Let $M = (\mathcal{E}, P, \lambda, t, m, \prec)$. Our NL-algorithm guesses a sequence $\emptyset = C_1, C_2, \ldots, C_n = \mathcal{E}$ of configurations of M which forms an execution of M. Note that each configuration C_i can be stored using $|P|$ pointers, one for each process in P. Since P is fixed we need only logarithmic space for this. Each time, a new configuration C_i is computed non-deterministically from C_{i-1}, we calculate $\text{gus}(C_i, M)$ from $\text{gus}(C_{i-1}, M)$, check whether $\text{gus}(C_i, M) \leq b$, and forget the old configuration C_{i-1}. Since we may assume that $b \leq |\mathcal{E}|$ (which we can check at the beginning) we can write down $\text{gus}(C_i, M)$ in logarithmic space. □

Theorem 6.2. *Let P be a fixed set of processes. The following problem is in NL:*

INPUT: HMSC H over the set of processes P and positive integer b.

QUESTION: Is H $\forall^b_{\text{glob}}$-bounded (resp. $\forall^b_{\text{loc}}$-bounded)?

Proof. Since NL is closed under complement [8,20], it suffices to check in NL whether an HMSC $H = (V, \to, P, \mu, v)$ is not $\forall^b_{\text{glob}}$-bounded (resp. $\forall^b_{\text{loc}}$-bounded). First we show that it can be verified in NL whether H is not locally strongly connected. For this we guess a node $u \in V$ and a cycle $u_1 \to u_2 \to \cdots \to u_m \to u_1$ in $(V, \to)$ with $u_1 = u$. While guessing this cycle, only the current node u_i and the communication graph of the MSC $\mu(u_1)\mu(u_2)\cdots\mu(u_i)$ are stored. Since P is fixed, we need only constant space in order to store this graph. The communication graph of the MSC $\mu(u_1)\mu(u_2)\cdots\mu(u_{i+1})$ can be easily constructed in logspace from the graph for the MSC $\mu(u_1)\mu(u_2)\cdots\mu(u_i)$. At the end we just have to check whether the communication graph of the whole cycle is not locally strongly connected.

Now in order to check whether H is not $\forall^b_{\text{glob}}$-bounded, we first nondeterministically branch into two cases. In the first case we check in NL whether H is not locally strongly connected, which is possible by the preceding paragraph. In the second case we first test whether $b \leq |P| \cdot |V| \cdot \sigma(H)$ (recall that $\sigma(H)$ is the maximal number of sends in an MSC $\mu(u)$, $u \in V$). If not we reject, otherwise we have to check whether there exists a path $v_1 \to v_2 \to \cdots \to v_n$ in $(V, \to)$ together with a configuration C in the MSC $M = \mu(v_1)\cdots\mu(v_n)$ such that $\text{gus}(C, M) > b$. The correctness of this procedure follows from Corollary 4.3. It remains to prove that it can be implemented in NL. Assume that

$P = \{1, \ldots, |P|\}$. We will guess the path $v_1 \to v_2 \to \cdots \to v_n$, where only the current node v_i will be stored. While guessing this path, we will also guess the configuration C and thereby accumulate $\mathrm{gus}(C, M)$ in a variable g. As soon as g reaches a value larger than $b \leq |P| \cdot |V| \cdot \sigma(H)$ we can immediately accept, thus the binary coding of g can be stored in logspace. Of course we cannot store the whole configuration C in logspace, thus we have to guess C in a "local way". For this we use a variable Q which stores a subset of $P = \{1, \ldots, |P|\}$ (thus Q only needs constant space). The set Q will store the processes p on which the maximal event in $\lambda^{-1}(p) \cap C$ was already executed (here λ refers to the MSC M).

Assume that during the execution of our NL-algorithm we go from node v_{i-1} to node v_i. We update the variables Q and g as follows: First we guess an arbitrary set $P' \subseteq P \backslash Q$ such that for every $p \in P'$, the MSC $\mu(v_i)$ contains an event on process p. We set $Q := Q \cup P'$ and guess for each process $p \in P'$ an event e_p of $\mu(v_i)$ which is located on process p. Let C_i be the set of all events e of $\mu(v_i)$ that are either located on some process in $P \backslash Q$ (here Q refers already to the updated value), or such that e is located on some process $p \in P'$ and lies in the past of e_p. We can easily check in deterministic logspace whether C_i is a configuration of $\mu(v_i)$, for this it is not necessary to construct C_i explicitly (which would not be possible in logspace). If C_i is not a configuration of $\mu(v_i)$ then we reject. Otherwise we can easily compute in logspace the value $\mathrm{gus}(C_i, \mu(v_i))$. If $g + \mathrm{gus}(C_i, \mu(v_i)) > b$ we immediately accept, otherwise we update g by $g := g + \mathrm{gus}(C_i, \mu(v_i))$ and proceed to the next node v_{i+1}.

Using Corollary 4.3, we see that H is not $\forall^b_{\mathrm{glob}}$-bounded if and only if some execution of our NL-algorithm accepts. In order to check whether an HMSC H is not $\forall^b_{\mathrm{loc}}$-bounded we can proceed similarly. □

7 Summary and open problems

Table 1 summarizes our results for boundedness problems for finite MSCs and HMSCs, for which we precisely determined the tractable boundedness problems. Concerning nested MSCs we have shown that the two local-boundedness problems can be decided in polynomial time. The precise complexity of the two global-boundedness problems remains open for nested MSCs. An NP-lower bound for existential-global-boundedness follows trivially from the NP-lower bound for finite MSCs. Concerning the upper bound we can only prove membership in PSPACE. For universal-global-boundedness we can prove membership in coNP for nMSCs, but the existence of a polynomial time algorithm remains open. Another interesting problem might be to investigate the complexity of boundedness problems for a fixed buffer-bound b, which means that b does not contribute to the input size. One might expect that the complexity

finite MSCs	∃	∀
global	NP-complete	P
local	P	P
local (nMSC)	P	P

HMSCs	∃	∀
global	NP-complete	coNP-complete
local	P	coNP-complete

Table 1

of boundedness problems decreases under this restriction.

Acknowledgment: We thank the anonymous referees for their useful comments and suggestions for improvement.

References

[1] R. Alur, G. J. Holzmann, and D. Peled. An analyzer for message sequence charts. *Software - Concepts and Tools*, 17(2):70–77, 1996.

[2] 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.

[3] 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.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–completeness.* Freeman, 1979.

[5] B. Genest and A. Muscholl. Pattern matching and membership for hierarchical message sequence charts. In S. Rajsbaum, editor, *In Proceedings of the 5th Latin American Symposium on Theoretical Informatics (LATIN 2002), Cancun (Mexico)*, number 2286 in Lecture Notes in Computer Science, pages 326–340. Springer, 2002.

[6] 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.

[7] 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.

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

[9] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.

[10] T. Lengauer and E. Wanke. The correlation between the complexities of the nonhierarchical and hierarchical versions of graph problems. *Journal of Computer and System Sciences*, 44:63–93, 1992.

[11] M. Lohrey and A. Muscholl. Bounded MSC communication. In M. Nielsen and U. Engberg, editors, *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2002), Grenoble (France)*, number 2303 in Lecture Notes in Computer Science, pages 295–309. Springer, 2002.

[12] P. Madhusudan and B. Meenakshi. Beyond Message Sequence Graphs. In R. Hariharan et al., eds., *Proceedings of FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science, 21st Conference, Bangalore (India)*, number 2245 in Lecture Notes in Computer Science, pages 256-267. Springer, 2001.

[13] R. Morin. On regular message sequence chart languages and relationships to Mazurkiewicz trace theory. In F. Honsell and M. Miculan, editors, *Proceedings of the 4th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2001), Genova (Italy)*, number 2030 in Lecture Notes in Computer Science, pages 332–346. Springer, 2001.

[14] 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.

[15] 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.

[16] A. Muscholl, D. Peled, and Z. Su. Deciding properties for message sequence charts. In M. Nivat, editor, *Proceedings of the 4th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'98), Lisbon (Portugal)*, number 1378 in Lecture Notes in Computer Science, pages 226–242. Springer, 1998.

[17] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.

[18] R. Sethi. Complete register allocation problems. *SIAM Journal on Computing*, 4(3):226–248, 1975.

[19] K. Simon. On minimum flow and transitive reduction. In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming (ICALP 88), Tampere (Finland)*, number 317 in Lecture Notes in Computer Science, pages 535–546. Springer, 1988.

[20] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988.

[21] P. van Emde Boas and J. van Leeuwen. Move rules and trade-offs in the pebble game. In *Proceedings of the 4th GI Conference*, number 67 in Lecture Notes in Computer Science, pages 101–112. Springer, 1979.