Stage M2

Davide Ferre'

June 9, 2022

1 Useful stuff

1.1 Message Sequence Charts

Assume a finite set of processes $\mathbb{P}$ and a finite set of messages $\mathbb{M}$. The set of (p2p) channels is $\mathbb{C} = \{(p,q) \in \mathbb{P} \times \mathbb{P} \mid p \neq q\}$. A send action is of the form $\mathit{send}(p,q,m)$ where $(p,q) \in \mathbb{C}$ and $m \in \mathbb{M}$. It is executed by p and sends message m to q. The corresponding receive action, executed by q, is $\mathit{rec}(p,q,m)$. For $(p,q) \in \mathbb{C}$, let $\mathit{Send}(p,q,\_) = \{\mathit{send}(p,q,m) \mid m \in \mathbb{M}\}$ and $\mathit{Rec}(p,q,\_) = \{\mathit{rec}(p,q,m) \mid m \in \mathbb{M}\}$. For $p \in \mathbb{P}$, we set $\mathit{Send}(p,\_,\_) = \{\mathit{send}(p,q,m) \mid q \in \mathbb{P}\setminus\{p\}$ and $m \in \mathbb{M}\}$, etc. Moreover, $\Sigma_p = \mathit{Send}(p,\_,\_) \cup \mathit{Rec}(\_,p,\_)$ will denote the set of all actions that are executed by p. Finally, $\Sigma = \bigcup_{p\in\mathbb{P}} \Sigma_p$ is the set of all the actions.

Peer-to-peer MSCs. A *p2p MSC* (or simply *MSC*) over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ where $\mathcal{E}$ is a finite (possibly empty) set of *events* and $\lambda : \mathcal{E} \rightarrow \Sigma$ is a labeling function. For $p \in \mathbb{P}$, let $\mathcal{E}_p = \{e \in \mathcal{E} \mid \lambda(e) \in \Sigma_p\}$ be the set of events that are executed by p. We require that $\rightarrow$ (the *process relation*) is the disjoint union $\bigcup_{p\in\mathbb{P}} \rightarrow_p$ of relations $\rightarrow_p \subseteq \mathcal{E}_p \times \mathcal{E}_p$ such that $\rightarrow_p$ is the direct successor relation of a total order on $\mathcal{E}_p$. For an event $e \in \mathcal{E}$, a set of actions $A \subseteq \Sigma$, and a relation $R \subseteq \mathcal{E} \times \mathcal{E}$, let $\#_A(R, e) = |\{f \in \mathcal{E} \mid (f, e) \in R \text{ and } \lambda(f) \in A\}|$. We require that $\lhd \subseteq \mathcal{E} \times \mathcal{E}$ (the *message relation*) satisfies the following:

(1) for every pair $(e, f) \in \lhd$, there is a send action $\mathit{send}(p,q,m) \in \Sigma$ such that $\lambda(e) = \mathit{send}(p,q,m)$, $\lambda(f) = \mathit{rec}(p,q,m)$, and $\#_{\mathit{Send}(p,q,\_)}(\rightarrow^+, e) = \#_{\mathit{Rec}(p,q,\_)}(\rightarrow^+, f)$,

(2) for all $f \in \mathcal{E}$ such that $\lambda(f)$ is a receive action, there is $e \in \mathcal{E}$ such that $e \lhd f$.

Finally, letting $\leq_M = (\rightarrow \cup \lhd)^*$, we require that $\leq_M$ is a partial order. For convenience, we will simply write $\leq$ when M is clear from the context.

Condition (1) above ensures that every (p2p) channel (p,q) behaves in a FIFO manner. By Condition (2), every receive event has a matching send event. Note that, however, there may be unmatched send events in an MSC. We let $\mathit{SendEv}(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a send action}\}$, $\mathit{RecEv}(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a receive action}\}$, $\mathit{Matched}(M) = \{e \in \mathcal{E} \mid \text{there is } f \in \mathcal{E} \text{ such that } e \lhd f\}$, and $\mathit{Unm}(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a send action and there is no } f \in \mathcal{E} \text{ such that } e \lhd f\}$. We do not distinguish isomorphic MSCs and let $\mathsf{MSC}_{\mathsf{p2p}}$ be the set of all MSCs over the given sets $\mathbb{P}$ and $\mathbb{M}$.

Example 1.1. For a set of processes $\mathbb{P} = \{p,q,r\}$ and a set of messages $\mathbb{M} = \{m_1, m_2, m_3, m_4\}$, $M_1 = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is an MSC where, for example, $e_2 \lhd e_2'$ and $e_3' \rightarrow e_4$. The dashed arrow means that the send event e_1 does not have a matching receive, so $e_1 \in \mathit{Unm}(M_1)$. Moreover, $e_2 \leq_{M_1} e_4$, but $e_1 \not\leq_{M_1} e_4$. We can find a total order $\rightsquigarrow \supseteq \leq_{M_1}$ such that $e_1 \rightsquigarrow e_2 \rightsquigarrow e_2' \rightsquigarrow e_3 \rightsquigarrow e_3' \rightsquigarrow e_4 \rightsquigarrow e_4'$. We call $\rightsquigarrow$ a linearization, which is formally defined below.

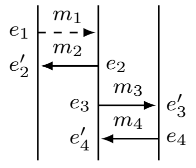


Figure 1: MSC M_1

Mailbox MSCs. For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, we define an additional binary relation that represents a constraint under the mailbox semantics, where each process has only one incoming channel. Let $\sqsubset_M \subseteq \mathcal{E} \times \mathcal{E}$ be defined by: $e_1 \sqsubset_M e_2$ if there is $q \in \mathbb{P}$ such that $\lambda(e_1) \in \mathit{Send}(\_,q,\_)$, $\lambda(e_2) \in \mathit{Send}(\_,q,\_)$, and one of the following holds:

- $e_1 \in \mathit{Matched}(M)$ and $e_2 \in \mathit{Unm}(M)$, or
- $e_1 \lhd f_1$ and $e_2 \lhd f_2$ for some $f_1, f_2 \in \mathcal{E}_q$ such that $f_1 \rightarrow^+ f_2$.

We let $\preceq_M = (\rightarrow \cup \lhd \cup \sqsubset_M)^*$. Note that $\leq_M \subseteq \preceq_M$. We call $M \in \mathsf{MSC}_{\mathsf{p2p}}$ a *mailbox MSC* if $\preceq_M$ is a partial order. Intuitively, this means that events can be scheduled in a way that corresponds to the mailbox semantics, i.e., with one incoming channel per process. Following the terminology in Bouajjani et al. 2018, we also say that a mailbox MSC satisfies *causal delivery*. The set of mailbox MSCs $M \in \mathsf{MSC}_{\mathsf{p2p}}$ is denoted by $\mathsf{MSC}_{\mathsf{mb}}$.

Example 1.2. MSC M_1 is a mailbox MSC. Indeed, even though the order $\leadsto$ defined in Example 1.1 does not respect all mailbox constraints, particularly the fact that $e_4 \sqsubset_{M_1} e_1$, there is a total order $\leadsto \supseteq \preceq_{M_1}$ such that $e_2 \leadsto e_3 \leadsto e_3' \leadsto e_4 \leadsto e_1 \leadsto e_2' \leadsto e_4'$. We call $\leadsto$ a mailbox linearization, which is formally defined below.

Linearizations, Prefixes, and Concatenation. Consider $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$. A *p2p linearization* (or simply *linearization*) of M is a (reflexive) total order $\leadsto \subseteq \mathcal{E} \times \mathcal{E}$ such that $\leq_M \subseteq \leadsto$. Similarly, a *mailbox linearization* of M is a total order $\leadsto \subseteq \mathcal{E} \times \mathcal{E}$ such that $\preceq_M \subseteq \leadsto$. That is, every mailbox linearization is a p2p linearization, but the converse is not necessarily true (Example 1.2). Note that an MSC is a mailbox MSC iff it has at least one mailbox linearization.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$ and consider $E \subseteq \mathcal{E}$ such that E is $\leq_M$*-downward-closed*, i.e, for all $(e, f) \in \leq_M$ such that $f \in E$, we also have $e \in E$. Then, the MSC $(E, \rightarrow \cap (E \times E), \lhd \cap (E \times E), \lambda')$, where λ' is the restriction of $\mathcal{E}$ to E, is called a *prefix* of M. In particular, the empty MSC is a prefix of M. We denote the set of prefixes of M by $\mathit{Pref}(M)$. This is extended to sets $L \subseteq \mathsf{MSC}$ as expected, letting $\mathit{Pref}(L) = \bigcup_{M \in L} \mathit{Pref}(M)$.

Lemma 1.1. *Every prefix of a mailbox MSC is a mailbox MSC.*

Proof. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{mb}}$ and $M_0 = (\mathcal{E}_0, \rightarrow_0, \lhd_0, \lambda_0)$ be a prefix of M, i.e., $\mathcal{E}_0 \subseteq \mathcal{E}$. By contradiction, suppose that M_0 is not a mailbox MSC. Then, there are distinct $e, f \in \mathcal{E}_0$ such that $e \preceq_{M_0} f \preceq_{M_0} e$ with $\preceq_{M_0} = (\rightarrow_0 \cup \lhd_0 \cup \sqsubset_{M_0})^*$. As $\mathcal{E}_0 \subseteq \mathcal{E}$, we have that $\rightarrow_0 \subseteq \rightarrow$, $\lhd_0 \subseteq \lhd$, and $\sqsubset_{M_0} \subseteq \sqsubset_M$. Finally, $\preceq_{M_0} \subseteq \preceq_M$ and M is not a mailbox MSC, which is a contradiction. $\square$

Let $M_1 = (\mathcal{E}_1, \rightarrow_1, \lhd_1, \lambda_1)$ and $M_2 = (\mathcal{E}_2, \rightarrow_2, \lhd_2, \lambda_2)$ be two MSCs. Their *concatenation* $M_1 \cdot M_2 = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is defined if, for all $(p, q) \in \mathbb{C}$, $e_1 \in \mathit{Unm}(M_1)$, and $e_2 \in \mathcal{E}_2$ such that $\lambda(e_1) \in \mathit{Send}(p, q, \_)$ and $\lambda(e_2) \in \mathit{Send}(p, q, \_)$, we have $e_2 \in \mathit{Unm}(M_2)$. As expected, $\mathcal{E}$ is the disjoint union of $\mathcal{E}_1$ and $\mathcal{E}_2$, $\lhd = \lhd_1 \cup \lhd_2$, λ is the "union" of λ_1 and λ_2, and $\rightarrow = \rightarrow_1 \cup \rightarrow_2 \cup R$. Here, R contains, for all $p \in \mathbb{P}$ such that $(\mathcal{E}_1)_p$ and $(\mathcal{E}_2)_p$ are non-empty, the pair (e_1, e_2) where e_1 is the maximal p-event in M_1 and e_2 is the minimal p-event in M_2. Note that $M_1 \cdot M_2$ is indeed an MSC and that concatenation is associative.

1.2 Communicating Systems

We now recall the definition of communicating systems (aka communicating finite-state machines or message-passing automata), which consist of finite-state machines A_p (one for every process $p \in \mathbb{P}$) that can communicate through the FIFO channels from $\mathbb{C}$.

Definition 1.1. A *communicating system* over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $\mathcal{S} = (A_p)_{p \in \mathbb{P}}$. For each $p \in \mathbb{P}$, $A_p = (\mathit{Loc}_p, \delta_p, \ell_p^0)$ is a finite transition system where Loc_p is a finite set of local (control) states, $\delta_p \subseteq \mathit{Loc}_p \times \Sigma_p \times \mathit{Loc}_p$ is the transition relation, and $\ell_p^0 \in \mathit{Loc}_p$ is the initial state.

Given $p \in \mathbb{P}$ and a transition $t = (\ell, a, \ell') \in \delta_p$, we let $\mathit{source}(t) = \ell$, $\mathit{target}(t) = \ell'$, $\mathit{action}(t) = a$, and $\mathit{msg}(t) = m$ if $a \in \mathit{Send}(\_, \_, m) \cup \mathit{Rec}(\_, \_, m)$.

There are in general two ways to define the semantics of a communicating system. Most often it is defined as a global infinite transition system that keeps track of the various local control states and all (unbounded) channel contents. As, in this paper, our arguments are based on a graph view of MSCs, we will define the language of $\mathcal{S}$ directly as a set of MSCs. These two semantic views are essentially equivalent, but they have different advantages depending on the context. We refer to Aiswarya and Gastin 2014 for a thorough discussion.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC. A *run* of $\mathcal{S}$ on M is a mapping $\rho : \mathcal{E} \rightarrow \bigcup_{p \in \mathbb{P}} \delta_p$ that assigns to every event e the transition $\rho(e)$ that is executed at e. Thus, we require that (i) for all $e \in \mathcal{E}$, we have $\mathit{action}(\rho(e)) = \lambda(e)$, (ii) for all $(e, f) \in \rightarrow$, $\mathit{target}(\rho(e)) = \mathit{source}(\rho(f))$, (iii) for all $(e, f) \in \lhd$, $\mathit{msg}(\rho(e)) = \mathit{msg}(\rho(f))$, and (iv) for all $p \in \mathbb{P}$ and $e \in \mathcal{E}_p$ such that there is no $f \in \mathcal{E}$ with $f \rightarrow e$, we have $\mathit{source}(\rho(e)) = \ell_p^0$.

Letting run $\mathcal{S}$ directly on MSCs is actually very convenient. This allows us to associate with $\mathcal{S}$ its p2p language and mailbox language in one go. The *p2p language* of $\mathcal{S}$ is $L_{\mathsf{p2p}}(\mathcal{S}) = \{M \in \mathsf{MSC}_{\mathsf{p2p}} \mid$

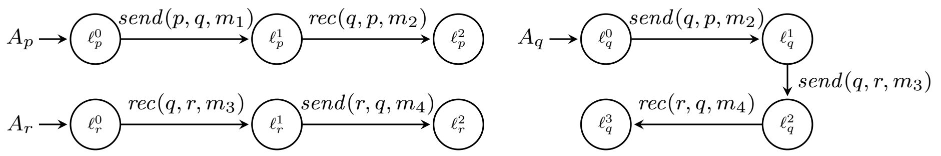


Figure 2: System $\mathcal{S}_1$

there is a run of $\mathcal{S}$ on $M\}$. The *mailbox language* of $\mathcal{S}$ is $L_{\mathsf{mb}}(\mathcal{S}) = \{M \in \mathsf{MSC}_{\mathsf{mb}} \mid$ there is a run of $\mathcal{S}$ on $M\}$.

Note that, following Bouajjani et al. 2018; Di Giusto, Laversa, and Lozes 2020, we do not consider final states or final configurations, as our purpose is to reason about all possible traces that can be *generated* by $\mathcal{S}$. The next lemma is obvious for the p2p semantics and follows from Lemma 1.1 for the mailbox semantics.

Lemma 1.2. *For all* com $\in \{\mathsf{p2p}, \mathsf{mb}\}$, *$L_{\text{com}}(\mathcal{S})$ is prefix-closed: $\mathit{Pref}(L_{\text{com}}(\mathcal{S})) \subseteq L_{\text{com}}(\mathcal{S})$.*

Example 1.3. Fig. 2 depicts $\mathcal{S}_1 = (A_p, A_q, A_r)$ such that MSC M_1 in Fig. 1 belongs to $L_{\mathsf{p2p}}(\mathcal{S}_1)$ and to $L_{\mathsf{mb}}(\mathcal{S}_1)$. There is a unique run ρ of $\mathcal{S}_1$ on M_1. We can see that $(e'_3, e_4) \in \rightarrow$ and $\mathit{target}(\rho(e'_3)) = \mathit{source}(\rho(e_4)) = \ell_r^1$, $(e_2, e'_2) \in \lhd_{M_1}$, and $\mathit{msg}(\rho(e_2)) = \mathit{msg}(\rho(e'_2)) = m_2$.

1.3 Conflict Graph

We now recall the notion of a conflict graph associated to an MSC defined in Bouajjani et al. 2018. This graph is used to depict the causal dependencies between message exchanges. Intuitively, we have a dependency whenever two messages have a process in common. For instance, an $\xrightarrow{SS}$ dependency between message exchanges v and v' expresses the fact that v' has been sent after v, by the same process. This notion is of interest because it was seen in Bouajjani et al. 2018 that the notion of synchronizability in MSCs (which is studied in this paper) can be graphically characterized by the nature of the associated conflict graph. It is defined in terms of linearizations in Di Giusto, Laversa, and Lozes 2020, but we equivalently express it directly in terms of MSCs.

For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ and $e \in \mathcal{E}$, we define the type $\tau(e) \in \{S, R\}$ of e by $\tau(e) = S$ if $e \in \mathit{SendEv}(M)$ and $\tau(e) = R$ if $e \in \mathit{RecEv}(M)$. Moreover, for $e \in \mathit{Unm}(M)$, we let $\mu(e) = e$, and for $(e, e') \in \lhd$, we let $\mu(e) = \mu(e') = (e, e')$.

Definition 1.2 (Conflict graph)**.** The *conflict graph* $\mathsf{CG}(M)$ of an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is the labeled graph $(\mathit{Nodes}, \mathit{Edges})$, with $\mathit{Edges} \subseteq \mathit{Nodes} \times \{S, R\}^2 \times \mathit{Nodes}$, defined by $\mathit{Nodes} = \lhd \cup \mathit{Unm}(M)$ and $\mathit{Edges} = \{(\mu(e), \tau(e)\tau(f), \mu(f)) \mid (e, f) \in \rightarrow^+\}$. In particular, a node of $\mathsf{CG}(M)$ is either a single unmatched send event or a message pair $(e, e') \in \lhd$.

1.4 Logic and Special Tree-Width

Monadic Second-Order Logic. The set of MSO formulas over MSCs (over $\mathbb{P}$ and $\mathbb{M}$) is given by the grammar $\varphi ::= x \rightarrow y \mid x \lhd y \mid \lambda(x) = a \mid x = y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \varphi \mid \neg\varphi$, where $a \in \Sigma$, x and y are first-order variables, interpreted as events of an MSC, and X is a second-order variable, interpreted as a set of events. We assume that we have an infinite supply of variables, and we use common abbreviations such as $\wedge$, $\forall$, etc. The satisfaction relation is defined in the standard way and self-explanatory. For example, the formula $\neg\exists x.(\bigvee_{a \in \mathit{Send}(\_,\_,\_)} \lambda(x) = a \wedge \neg \mathit{matched}(x))$ with $\mathit{matched}(x) = \exists y. x \lhd y$ says that there are no unmatched send events. It is not satisfied by MSC M_1 of Fig. 1, as message m_1 is not received, but by M_4 from Fig. **??**.

Given a sentence φ, i.e., a formula without free variables, we let $L(\varphi)$ denote the set of (p2p) MSCs that satisfy φ. It is worth mentioning that the (reflexive) transitive closure of a binary relation defined by an MSO formula with free variables x and y, such as $x \rightarrow y$, is MSO-definable so that the logic can freely use formulas of the form $x \rightarrow^+ y$ or $x \leq y$ (where $\leq$ is interpreted as $\leq_M$ for the given MSC M). Therefore, the definition of a mailbox MSC can be readily translated into the formula $\varphi_{\mathsf{mb}} = \neg\exists x.\exists y.(\neg(x = y) \wedge x \preceq y \wedge y \preceq x)$ so that we have $L(\varphi_{\mathsf{mb}}) = \mathsf{MSC}_{\mathsf{mb}}$. Here, $x \preceq y$ is obtained as the MSO-definable reflexive transitive closure of the union of the MSO-definable relations $\rightarrow$, $\lhd$, and $\sqsubset$. In particular, we may define $x \sqsubset y$ by :

$$x \sqsubset y = \bigvee_{\substack{q \in \mathbb{P} \\ a,b \in \mathit{Send}(\_,q,\_)}} \lambda(x) = a \;\wedge\; \lambda(y) = b \wedge \left(\begin{array}{ll} & \mathit{matched}(x) \wedge \neg\mathit{matched}(y) \\ \vee & \exists x'.\exists y'.(x \lhd x' \;\wedge\; y \lhd y' \;\wedge\; x' \rightarrow^+ y') \end{array} \right)$$

Special Tree-Width. *Special tree-width* Courcelle 2010, is a graph measure that indicates how close a graph is to a tree (we may also use classical *tree-width* instead). This or similar measures are commonly employed in verification. For instance, tree-width and split-width have been used in Madhusudan and Parlato 2011 and, respectively, Cyriac, Gastin, and Kumar 2012; Aiswarya, Gastin, and Kumar 2014 to reason about graph behaviors generated by pushdown and queue systems. There are several ways to define the special tree-width of an MSC. We adopt the following game-based definition from Bollig and Gastin 2019.

Adam and Eve play a two-player turn based "decomposition game" whose positions are MSCs with some pebbles placed on some events. More precisely, Eve's positions are *marked MSC fragments* (M, U), where $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is an *MSC fragment* (an MSC with possibly some edges from $\lhd$ or $\rightarrow$ removed) and $U \subseteq \mathcal{E}$ is the subset of marked events. Adam's positions are pairs of marked MSC fragments. A move by Eve consists in the following steps:

1. marking some events of the MSC resulting in (M, U') with $U \subseteq U' \subseteq \mathcal{E}$,
2. removing (process and/or message) edges whose endpoints are marked,
3. dividing (M, U) in (M_1, U_1) and (M_2, U_2) such that M is the disjoint (unconnected) union of M_1 and M_2 and marked nodes are inherited.

When it is Adam's turn, he simply chooses one of the two marked MSC fragments. The initial position is $(M, \emptyset)$ where M is the (complete) MSC at hand. A terminal position is any position belonging to Eve such that all events are marked. For $k \in \mathbb{N}$, we say that the game is k-winning for Eve if she has a (positional) strategy that allows her, starting in the initial position and independently of Adam's moves, to reach a terminal position such that, in every single position visited along the play, there are at most $k + 1$ marked events.

Fact 1.3 (Bollig and Gastin 2019)**.** *The special tree-width of an MSC is the least k such that the associated game is k-winning for Eve.*

The set of MSCs whose special tree-width is at most k is denoted by $\mathsf{MSC}^{k\text{-stw}}$.

1.5 Model Checking

In general, even simple verification problems, such as control-state reachability, are undecidable for communicating systems Brand and Zafiropulo 1983. However, they are decidable when we restrict to behaviors of bounded special tree-width, which motivates the following definition of a generic **bounded model-checking problem** for com $\in \{\mathsf{p2p}, \mathsf{mb}\}$:
Input: Two finite sets $\mathbb{P}$ and $\mathbb{M}$, a communicating system $\mathcal{S}$, an MSO sentence φ, and $k \in \mathbb{N}$ (given in unary).
Question: Do we have $L_{\text{com}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-stw}} \subseteq L(\varphi)$?

Fact 1.4 (Bollig and Gastin 2019)**.** *The bounded model-checking problem for* com $= \mathsf{p2p}$ *is decidable. When the formulas φ are from LCPDL, then the problem is solvable in exponential time.*

Note that Bollig and Gastin 2019 does not employ the LCPDL modality jump, but it can be integrated easily. Using φ_{mb} or Φ_{mb}, we obtain the corresponding result for mailbox systems as a corollary:

Theorem 1.5. *The bounded model-checking problem for* com $= \mathsf{mb}$ *is decidable. When the formulas φ are from LCPDL, then the problem is solvable in exponential time.*

1.6 Synchronizability

The above model-checking approach is incomplete in the sense that a positive answer does not imply correctness of the whole system. The system may still produce behaviors of special tree-width greater than k that violate the given property. However, if we know that a system only generates behaviors from a class whose special tree-width is bounded by k, we can still conclude that the system is correct.

This motivates the *synchronizability problem.* Several notions of synchronizability have been introduced in the literature. However, they all amount to asking whether all behaviors generated by a given communicating system have a particular shape, i.e., whether they are all included in a fixed (or given) set of MSCs $\mathcal{C}$. Thus, the synchronizability problem is essentially an inclusion problem, namely $L_{\mathsf{p2p}}(\mathcal{S}) \subseteq \mathcal{C}$ or $L_{\mathsf{mb}}(\mathcal{S}) \subseteq \mathcal{C}$. We show that, for decidability, it is enough to

Table 1: Summary of the decidability of the synchronizability problem in various classes

	PEER-TO-PEER	MAILBOX
Weakly synchronous	Undecidable [Thm. 1.10]	EXPTIME [Thm. 1.9]
Weakly k-synchronous	Decidable Bouajjani et al. 2018; Di Giusto, Laversa, and Lozes 2020 and [Thm. 1	
Strongly k-synchronous	—	Decidable [Thm. ??]
Existentially k-p2p-bounded	Decidable Genest, Kuske, and Muscholl 2007, Prop. 5.5	
Existentially k-mailbox-bounded	—	Decidable [Prop. 1.6]

have that $\mathcal{C}$ is MSO-definable and special-tree-width-bounded (STW-bounded): We call $\mathcal{C} \subseteq \mathsf{MSC}$ (i) *MSO-definable* if there is an MSO-formula φ such that $L(\varphi) = \mathcal{C}$, (ii) *LCPDL-definable* if there is an an LCPDL-formula Φ such that $L(\Phi) = \mathcal{C}$, (iii) *STW-bounded* if there is $k \in \mathbb{N}$ such that $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$.

An important component of the decidability proof is the following lemma, which shows that we can reduce synchronizability wrt. an STW-bounded class to bounded model-checking.

Lemma 1.6. *Let $\mathcal{S}$ be a communicating system,* $\mathsf{com} \in \{\mathsf{p2p}, \mathsf{mb}\}$*, $k \in \mathbb{N}$, and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. Then, $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$ iff $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$.*

The result follows from the following lemma. Note that a similar property was shown in Genest, Kuske, and Muscholl 2007, Proposition 5.4 for the specific class of existentially k-bounded MSCs.

Lemma 1.7. *Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. For all $M \in \mathsf{MSC} \setminus \mathcal{C}$, we have $(\mathit{Pref}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset$.*

We now have all ingredients to state a generic decidability result for synchronizability:

Theorem 1.8. *Fix finite sets $\mathbb{P}$ and $\mathbb{M}$. Suppose* $\mathsf{com} \in \{\mathsf{p2p}, \mathsf{mb}\}$ *and let $\mathcal{C} \subseteq \mathsf{MSC}$ be an MSO-definable and STW-bounded class (over $\mathbb{P}$ and $\mathbb{M}$). The following problem is decidable: Given a communicating system $\mathcal{S}$, do we have $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$?*

Proof. Consider the MSO-formula φ such that $L(\varphi) = \mathcal{C}$, and let $k \in \mathbb{N}$ such that $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. We have $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C} \overset{\text{Lemma 1.6}}{\iff} L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C} \iff L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq L(\varphi)$. The latter can be solved thanks to Fact 1.4 and Theorem 1.5. □

1.7 Application to Concrete Classes of Synchronizability

In this section, we instantiate our general framework by specific classes. Table 1 gives a summary of the results.

1.8 A New General Class: Weakly Synchronous MSCs

We first introduce the class of weakly synchronous MSCs. This is a generalization of synchronous MSCs studied earlier, in Bouajjani et al. 2018; Di Giusto, Laversa, and Lozes 2020, which we shall discuss later. We say an MSC is weakly synchronous if it is breakable into *exchanges* where an exchange is an MSC that allows one to schedule all sends before all receives. Let us define this formally:

Definition 1.3 (exchange)**.** Let $M = (\mathcal{E}, \to, \lhd, \lambda)$ be an MSC. We say that M is an *exchange* if $\mathit{SendEv}(M)$ is a $\leq_M$-downward-closed set.

Definition 1.4 (weakly synchronous)**.** We say that $M \in \mathsf{MSC}$ is *weakly synchronous* if it is of the form $M = M_1 \cdot \ldots \cdot M_n$ such that every M_i is an exchange.

We use the term *weakly* to distinguish from variants introduced later.

Example 1.4. Consider the MSC M_2 in Fig. 3. It is is weakly synchronous. Indeed, m_1, m_2, and m_5 are independent and can be put alone in an exchange. Repetitions of m_3 and m_4 are interlaced, but they constitute an exchange, as we can do all sends and then all receptions.

An easy adaptation of a characterization from Di Giusto, Laversa, and Lozes 2020 yields the following result for weakly synchronous MSCs:

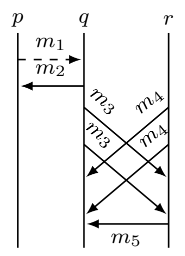


Figure 3: MSC M_2

Proposition 1.1. Let M be an MSC. Then, M is weakly synchronous iff no RS edge occurs on any cyclic path in the conflict graph $\mathsf{CG}(M)$.

It is easily seen that the characterization from Proposition 1.1 is LCPDL-definable:

Corollary 1.8.1. *The sets of weakly synchronous MSCs and weakly synchronous* mailbox *MSCs are LCPDL-definable. Both formulas have polynomial size.*

Moreover, under the mailbox semantics, we can show:

Proposition 1.2. The set of weakly synchronous mailbox MSCs is STW-bounded (in fact, it is included in $\mathsf{MSC}^{4|\mathbb{P}|\text{-stw}}$).

Proof. Let M be fixed, and let us sketch Eve's winning strategy. Let $n = |\mathbb{P}|$.

The first step for Eve is to split M in exchanges. She first disconnects the first exchange from the rest of the graph ($2n$ pebbles are needed), then she disconnects the second exchange from the rest of the graph ($2n$ pebbles needed, plus n pebbles remaining from the first round), and so on for each exchange.

So we are left with designing a winning strategy for Eve with $4n + 1$ pebbles on the graph of an exchange M_0, where initially there are (at most) n pebbles placed on the first event of each process and also (at most) n pebbles placed on the last event of each process. Eve also places (at most) n pebbles on the last send event of each process and also (at most) n pebbles on the first receive event of each process. Eve erases the (at most) n $\rightarrow$-edges between the last send event and the first receive event.

We are now in a configuration that will be our invariant.

Let us fix a mailbox linearization of M_0 and let e be the first send event in this linearization.

- if e is an unmatched send of process p, Eve places her last pebble on the next send event of p (if it exists), let us call it e'. Then Eve erases the $\rightarrow$-edge (e, e'), and now e is completely disconnected, so it can be removed and the pebble can be taken back.

- if $e \lhd e'$, with e' a receive event of process q, then due to the mailbox semantics e' is the first receive event of q, so it has a pebble placed on it. Eve removes the $\lhd$-edge between e and e', then using the extra pebble she disconnects e and places a pebble on the $\rightarrow$-successor of e, then she also disconnects e' and places a pebble on the $\rightarrow$-successor of e'.

After that, we are back to our invariant, so we can repeat the same strategy with the second send event of the linearization, and so on until all edges have been erased. ☐

We obtain the following result as a corollary. Note that it assumes the mailbox semantics.

Theorem 1.9. *The following problem is decidable in exponential time: Given $\mathbb{P}$, $\mathbb{M}$, and a communicating system $\mathcal{S}$ (over $\mathbb{P}$ and $\mathbb{M}$), is every MSC in $L_{\mathsf{mb}}(\mathcal{S})$ weakly synchronous?*

Proof. According to Corollary 1.8.1, we determine the LCPDL formula Φ_{wsmb} such that $L(\Phi_{\mathsf{wsmb}})$ is the set of weakly synchronous mailbox MSCs. Moreover, recall from Proposition 1.2 that the special tree-width of all weakly synchronous mailbox MSCs is bounded by $4|\mathbb{P}|$. By Lemma 1.6, $L_{\mathsf{mb}}(\mathcal{S}) \subseteq L(\Phi_{\mathsf{wsmb}})$ iff $L_{\mathsf{mb}}(\mathcal{S}) \cap \mathsf{MSC}^{(4|\mathbb{P}|+2)\text{-stw}} \subseteq L(\Phi_{\mathsf{wsmb}})$. The latter is an instance of the bounded model-checking problem. As the length of Φ_{wsmb} is polynomial in $|\mathbb{P}|$, we obtain that the original problem is decidable in exponential time by Theorem 1.5. ☐

For the same reasons, the model-checking problem for "weakly synchronous" systems is decidable. Interestingly, a reduction from Post's correspondence problem shows that decidability fails when adopting the p2p semantics:

Theorem 1.10. *The following problem is undecidable: Given finite sets $\mathbb{P}$ and $\mathbb{M}$ as well as a communicating system $\mathcal{S}$, is every MSC in $L_{\mathsf{p2p}}(\mathcal{S})$ weakly synchronous?*

Proof. We show that the control state reachability problem for p2p weakly synchronizable systems is not decidable. This immediately shows that the model-checking problem for p2p weak synchronizable systems is not decidable.

D: Clarify why undecidability of control state reachability implies undecidability of model checking.

With some extra coding, it also shows that the membership problem (decide whether a given system is p2p weakly synchronizable) also is undecidable: indeed, it is enough to add a non weak synchronizable behavior after the control states for which reachability is undecidable: the system will be not weakly synchronizable iff the control states are reached.

We reduce from Post correspondence problem (PCP). Let us recall that a PCP instance consists of N pairs (u_i, v_i) of finite words over an alphabet A, and that PCP undecidability holds already for $N = 7$ and $A = \{0, 1\}$. We let the set of messages be $\{1, \ldots, N\} \uplus A \uplus \{\sharp\}$, and we consider a system with four machines: Prover1, Prover2, Verifier1, and Verifier2. We have unidirectional communication channels from provers to verifiers, so the system is weakly synchronous by construction.

Informally, the system works as follows:

- Prover1 guesses a solution $u_{i_1} \ldots u_{i_m}$ of the PCP instance, and Prover2 also guesses the same solution $v_{i_1} ... v_{i_m}$.
- Prover1 sends $u_{i_1} \ldots u_{i_n}$ to Verifier1 and sends simultaneously $i_1 \ldots i_m$ to Verifier2
- Prover2 sends $v_{i_1} \ldots v_{i_m}$ to Verifier1 and sends simultaneously $i_1 \ldots i_m$ to Verifier 2
- Verifier1 checks that the two words are equal and Verifier2 checks that the sequences of indices are equal.

Let us now formally define these machines. We describe them with regular expressions. For $w = a_1 \cdots a_n$, we write $\mathit{send}^*(p, q, w)$ (resp $\mathit{rec}^*(p, q, w)$) for $\mathit{send}(p, q, a_1) \cdots \mathit{send}(p, q, a_n)$ (resp $\mathit{rec}(p, q, a_1) \cdots \mathit{rec}(p, q, a_n)$). We abbreviate Prover1 as P1, Prover2 as P2, Verifier1 as V1, and Verifier2 as V2

- Prover1 is

$$\Big(\sum_{i=1}^{N} \mathit{send}(P_1, V_1, i)\mathit{send}^*(P_1, V_2, u_i)\Big)^+ \mathit{send}(P_1, V_1, \sharp)\mathit{send}(P_1, V_2, \sharp)$$

- Prover2 is

$$\Big(\sum_{i=1}^{N} \mathit{send}(P_2, V_1, i)\mathit{send}^*(P_2, V_2, v_i)\Big)^+ \mathit{send}(P_2, V_1, \sharp)\mathit{send}(P_2, V_2, \sharp)$$

- Verifier1 is

$$\Big(\sum_{i=1}^{N} \mathit{rec}(P_1, V_1, i)\mathit{rec}(P_2, V_1, i)\Big)^* \mathit{rec}(P_1, V_1, \sharp)\mathit{rec}(P_2, V_1, \sharp)$$

- Verifier2 is

$$\Big(\sum_{a \in \Sigma} \mathit{rec}(P_1, V_2, a)\mathit{rec}(P_2, V_2, a)\Big)^* \mathit{rec}(P_1, V_2, \sharp)\mathit{rec}(P_2, V_2, \sharp)$$

It can be checked that all machines reach their own final state if and only if the PCP instance has a solution. □

1.9 Weakly k-Synchronous MSCs

This negative result for the p2p semantics motivates the study of other classes. In fact, our framework captures several classes introduced in the literature.

Definition 1.5 (k-exchange)**.** Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC and $k \in \mathbb{N}$. We call M a *k-exchange* if M is an exchange and $|\mathit{SendEv}(M)| \leq k$.

Let us now recall the definition from Bouajjani et al. 2018; Di Giusto, Laversa, and Lozes 2020, but (equivalently) expressed directly in terms of MSCs rather than via *executions*. It differs from the weakly synchronous MSCs in that here, we insist on constraining the number of messages sent per exchange to be at most k.

Definition 1.6 (weakly k-synchronous)**.** Let $k \in \mathbb{N}$. We say that $M \in \mathsf{MSC}$ is weakly k-synchronous if it is of the form $M = M_1 \cdot \ldots \cdot M_n$ such that every M_i is a k-exchange.

Example 1.5. MSC M_3 in Fig. 4 is weakly 1-synchronous, as it can be decomposed into three 1-exchanges (the decomposition is depicted by the horizontal dashed lines). We remark that $M_3 \in \mathsf{MSC}_{\mathsf{mb}}$. Note that there is a p2p linearization that respects the decomposition. On the other hand, a mailbox linearization needs to reorganize actions from different MSCs: the sending of m_3 needs to be done before the sending of m_1. Note that M_1 in Fig. 1 is also weakly 1-synchronous.

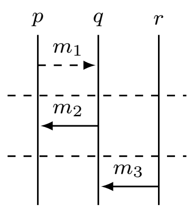


Figure 4: MSC M_3

Proposition 1.3. Let $k \in \mathbb{N}$. The set of weakly k-synchronous p2p (mailbox, respectively) MSCs is effectively MSO-definable.

In fact, MSO-definability essentially follows from the following known theorem:

Theorem 1.11 (Di Giusto, Laversa, and Lozes 2020)**.** *Let M be an MSC. Then, M is weakly k-synchronous iff every SCC in its conflict graph $\mathsf{CG}(M)$ is of size at most k and no RS edge occurs on any cyclic path.*

This property is similar to the graphical characterization of weakly synchronous MSCs, except for the condition that every SCC in the conflict graph is of size at most k. Furthermore, it is easy to establish a bound on the special tree-width:

Proposition 1.4. Let $k \in \mathbb{N}$. The set of MSCs that are weakly k-synchronous have special tree-width bounded by $2k + |\mathbb{P}|$.

Hence, we can conclude that the class of weakly k-synchronous MSCs is MSO-definable and STW-bounded. As a corollary, we get the following (known) decidability result, but via an alternative proof:

Theorem 1.12 (Bouajjani et al. 2018; Di Giusto, Laversa, and Lozes 2020)**.** *For* $\mathsf{com} \in \{\mathsf{p2p}, \mathsf{mb}\}$*, the following problem is decidable: Given finite sets $\mathbb{P}$ and $\mathbb{M}$, a communicating system $\mathcal{S}$, and $k \in \mathbb{N}$, is every MSC in $L_{\mathsf{com}}(\mathcal{S})$ weakly k-synchronous?*

Proof. We proceed similarly to the proof of Theorem 1.9. For the given $\mathbb{P}$, $\mathbb{M}$, and k, we first determine, using Proposition 1.3, the MSO formula φ_k such that $L(\varphi_k)$ is the set of weakly k-synchronous p2p/mailbox MSCs. From Proposition 1.4, we know that the special tree-width of all weakly k-synchronous MSCs is bounded by $2k + |\mathbb{P}|$. By Lemma 1.6, we have $L_{\mathsf{com}}(\mathcal{S}) \subseteq L(\varphi_k)$ iff $L_{\mathsf{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(2k+|\mathbb{P}|+2)\text{-}\mathsf{stw}} \subseteq L(\varphi_k)$. The latter is an instance of the bounded model-checking problem. By Fact 1.4 and Theorem 1.5, we obtain decidability. $\square$

Remark 1.1. The set of weakly k-synchronous MSCs is not directly expressible in LCPDL (the reason is that LCPDL does not have a built-in counting mechanism). However, its *complement* is expressible in the extension of LCPDL with existentially quantified propositions (we need $k+1$ of them). The model-checking problem for this kind of property is still in EXPTIME and, therefore, so is the problem from Theorem 1.12 when k is given in unary. It is very likely that our approach can also be used to infer the PSPACE upper bound from Bouajjani et al. 2018 by showing bounded *path width* and using finite word automata instead of tree automata. Finally, note that the problem to decide whether there exists an integer $k \in \mathbb{N}$ such that all MSCs in $L_{\mathsf{com}}(\mathcal{S})$ are weakly k-synchronous has recently been studied in Giusto, Laversa, and Lozes 2021 and requires different techniques.

Observe also that we can remove the constraint of all the sends preceding all the receives in a k-exchange, and still have decidability. We then have the following definition.

Definition 1.7 (modified k-exchange)**.** Let $M = (\mathcal{E}, \to, \lhd, \lambda)$ be an MSC and $k \in \mathbb{N}$. We call M a *modified k-exchange* if $|SendEv(M)| \leq k$.

We extend this notion to consider modified weakly k-synchronous executions as before, and the graphical characterization of this property is that there are at most k nodes in every SCC of the conflict graph. Hence, this class is also MSO-definable, and since each modified k-exchange has at most $2k$ events, it also has bounded special tree-width.

1.10 Existentially k-Bounded MSCs

Now, we turn to existentially k-bounded MSCs Lohrey and Muscholl 2002; Genest, Muscholl, and Kuske 2004; Genest, Kuske, and Muscholl 2007. Synchronizability has been studied for the p2p case in Genest, Kuske, and Muscholl 2007, so we only consider the mailbox case here. A linearization $\rightsquigarrow$ of an MSC $M = (\mathcal{E}, \to, \lhd, \lambda) \in \mathsf{MSC}$ is called *k-mailbox-bounded* if, for all $e \in \mathit{Matched}(M)$, say with $\lambda(e) = \mathit{send}(p, q, m)$, we have $\#_{\mathit{Send}(\_,q,\_)}(\rightsquigarrow, e) - \#_{\mathit{Rec}(\_,q,\_)}(\rightsquigarrow, e) \leq k$.

Definition 1.8. Let $M = (\mathcal{E}, \to, \lhd, \lambda) \in \mathsf{MSC}$ and $k \in \mathbb{N}$. We call M *existentially k-mailbox-bounded* if it has some mailbox linearization that is k-mailbox-bounded.

Note that every existentially k-mailbox-bounded MSC is a mailbox MSC.

Example 1.6. MSC M_5 in Fig. 5 is existentially 1-mailbox-bounded, as witnessed by the (informally given) linearization $\mathsf{s}(q, p, m_2) \rightsquigarrow \mathsf{s}(p, q, m_1) \rightsquigarrow \mathsf{s}(q, r, m_3) \rightsquigarrow \mathsf{r}(q, r, m_3) \rightsquigarrow \mathsf{r}(p, q, m_1) \rightsquigarrow \mathsf{s}(p, q, m_1) \rightsquigarrow \mathsf{r}(q, p, m_2) \rightsquigarrow \mathsf{s}(q, r, m_3) \ldots$ Note that M_5 is neither weakly nor strongly synchronous as we cannot divide it into exchanges.

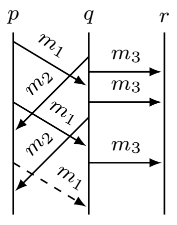


Figure 5: MSC M_5

Proposition 1.5. For all $k \in \mathbb{N}$, the set of existentially k-p2p-bounded MSCs is MSO-definable and STW-bounded.

Proof. The set of existentially k-p2p-bounded MSCs was shown to be MSO-definable (in fact, even FO-definable) in Lohrey and Muscholl 2004. Note that there are minor differences in the definitions (in particular, the fact that we deal with unmatched messages), which, however, do not affect FO-definability. In Bollig and Gastin 2019, Proposition 5.4, page 163, it was shown that their special tree-width is bounded by $k|\mathbb{P}|^2 + |\mathbb{P}|$. $\square$

We obtain the following result as a corollary:

Theorem 1.13. *For* $\mathsf{com} \in \{\mathsf{p2p}, \mathsf{mb}\}$*, the following problem is decidable: Given finite sets $\mathbb{P}$ and $\mathbb{M}$, a communicating system $\mathcal{S}$, and $k \in \mathbb{N}$, is every MSC in $L_{\mathsf{com}}(\mathcal{S})$ existentially k-p2p-bounded?*

Proof. Again, the proof follows exactly the same lines as that or Theorem 1.12, now using Proposition 1.5. $\square$

Note that this is similar to the problem considered in Genest, Kuske, and Muscholl 2007; Kuske and Muscholl 2014, though there is a subtle difference: in Genest, Kuske, and Muscholl 2007; Kuske and Muscholl 2014, there are a notion of deadlock and distinguished final configurations. We define the following relation in order to characterize k-mailbox-bounded MSCs.

Let $k \geq 1$, and let M be a fixed mailbox MSC. Let $\xrightarrow{\text{rev}}_k$ be the binary relation among events of M defined as follows: $r \xrightarrow{\text{rev}}_k s$ if

1. r is a receive event of a process p;
2. let r' be the k-th receive event of process p after r; then $s \lhd r'$.

Lemma 1.14. *M is existential k-mailbox-bounded if and only if $\preceq_M \cup \xrightarrow{rev}_k$ is acyclic.*

Proof. Assume that M is existential k-mailbox-bounded. Let $\rightsquigarrow$ be a mailbox linearisation of M such that for all $e \in \mathit{Matched}(M)$, say with $\lambda(e) = \mathit{send}(p, q, m)$,

$$\#_{\mathit{Send}(-,q,\_)}(\rightsquigarrow, e) - \#_{\mathit{Rec}(-,q,\_)}(\rightsquigarrow, e) \leq k\,.$$

Then $\rightsquigarrow$ is also a linearisation of $(\preceq_M \cup \xrightarrow{\text{rev}}_k)^*$. Indeed, if it was not the case, there would be a pair of events r, s such that $r \xrightarrow{\text{rev}}_k s$ and $s \rightsquigarrow r$. But then we would have

$$\#_{\mathit{Send}(-,q,\_)}(\rightsquigarrow, s) - \#_{\mathit{Rec}(-,q,\_)}(\rightsquigarrow, s) > k\,,$$

and the contradiction. So $\rightsquigarrow$ is a linearisation of $(\preceq_M \cup \xrightarrow{\text{rev}}_k)^*$ and $\preceq_M \cup \xrightarrow{\text{rev}}_k$ is acyclic.

Conversely, assume that $\preceq_M \cup \xrightarrow{\text{rev}}_k$, and let $\rightsquigarrow$ be a linearisation of $(\preceq_M \cup \xrightarrow{\text{rev}}_k)^*$. In particular, $\rightsquigarrow$ is a mailbox linearisation of M. Let us show that for all $s \in \mathit{Matched}(M)$, say with $\lambda(s) = \mathit{send}(p, q, m)$,

$$\#_{\mathit{Send}(-,q,\_)}(\rightsquigarrow, s) - \#_{\mathit{Rec}(-,q,\_)}(\rightsquigarrow, s) \leq k\,.$$

Let $s \in \mathit{Matched}(M)$ be fixed, and let r' be such that $s \lhd r'$. There are two cases:

- $\#_{Rec(-,q,\_)}(\rightarrow, r') \leq k$. Then
$$\#_{Send(-,q,\_)}(\rightsquigarrow, s) \leq k\,,$$
because all sends before s are matched. So
$$\#_{Send(-,q,\_)}(\rightsquigarrow, s) - \#_{Rec(-,q,\_)}(\rightsquigarrow, s) \leq k\,,$$
- $\#_{Rec(-,q,\_)}(\rightarrow, r') \leq k$. Then there is r on process q such that $r \xrightarrow{\text{rev}}_k s$. So $r \rightsquigarrow s$, and there are at most k messages in the buffer of q at the time of event s, or in other words,
$$\#_{Send(-,q,\_)}(\rightsquigarrow, e) - \#_{Rec(-,q,\_)}(\rightsquigarrow, e) \leq k\,.$$

So $\rightsquigarrow$ is a mailbox linearisation with k bounded buffers, and M is existential k-mailbox-bounded. □

Proposition 1.6. For all $k \in \mathbb{N}$, the set of existentially k-mailbox-bounded MSCs is MSO-definable and STW-bounded.

Proof. Let $k \geq 1$ be fixed. Since every existentially k-mailbox-bounded MSCs is also existentially k-p2p-bounded, and since the class of existentially k-p2p-bounded MSCs is STW bounded (cf Proposition 1.5), the class of existentially k-mailbox-bounded MSCs is also STW bounded.

Let us show that it is moreover MSO definable.

By Lemma 1.14, it is enough to show that the acyclicity of $\preceq_M \cup \xrightarrow{\text{rev}}_k$ is MSO definable, and since $\preceq_M$ was already shown MSO definable and acyclicity is easily MSO definable, it is enough to show that $\xrightarrow{\text{rev}}_k$ is MSO definable. It is indeed the case, as demonstrated by this formula
$$\varphi(r,s) = \exists r_1, r_2, \ldots, r_n . r \rightarrow r_1 \rightarrow r_2 \rightarrow \ldots \rightarrow r_n \wedge s \lhd r_n.$$

Finally, let us show that existentially k-mailbox-bounded is also LCPDL definable. This follows from the following formulas:
$$\begin{aligned}
\prec_M =& \quad (\lhd + \rightarrow)^+ \\
R =& \quad \langle \lhd^{-1} \rangle \top \\
\xrightarrow{\text{next R}} =& \quad (\rightarrow \wedge \mathsf{test}(\neg R))^* \cdot (\rightarrow \wedge \mathsf{test}(R)) \\
\xrightarrow{\text{rev}}_k =& \quad (\xrightarrow{\text{next R}})^k . (\lhd)^{-1}. \\
\Phi_{\exists k \text{ mb-bounded}} =& \quad \neg \mathsf{ELoop}\langle (\prec_M + \xrightarrow{\text{rev}}_k)^+ \rangle
\end{aligned}$$
□

This extension is also valid for the p2p definition of existentially k-bounded MSCs, which were addressed in Genest, Kuske, and Muscholl 2007. Finally, our framework can also be adapted to treat universally bounded systems Henriksen et al. 2005; Lohrey and Muscholl 2002.

2 New - Preliminary results

2.1 Message Sequence Charts

Definition 2.1 (Causally ordered MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is *causally ordered* if, for any two send events s and s', such that $\lambda(s) = Send(\_, q, \_)$, $\lambda(s') = Send(\_, q, \_)$, and $s \leq_M s'$, we have either:

- $s, s' \in Matched(M)$ and $r \rightarrow^+ r'$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$.
- $s' \in Unm(M)$.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$ and consider $E \subseteq \mathcal{E}$ such that E is $\leq_M$*-downward-closed*, i.e, for all $(e, f) \in \leq_M$ such that $f \in E$, we also have $e \in E$. Then, the MSC $(E, \rightarrow \cap (E \times E), \lhd \cap (E \times E), \lambda')$, where λ' is the restriction of $\mathcal{E}$ to E, is called a *prefix* of M. In particular, the empty MSC is a prefix of M. We denote the set of prefixes of M by $Pref(M)$. This is extended to sets $L \subseteq \mathsf{MSC}$ as expected, letting $Pref(L) = \bigcup_{M \in L} Pref(M)$.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{1-n}$ and consider $E \subseteq \mathcal{E}$ such that E is $\lessdot_M$*-downward-closed*, i.e, for all $(e, f) \in \lessdot_M$ such that $f \in E$, we also have $e \in E$. Then, the MSC $(E, \rightarrow \cap (E \times E), \lhd \cap$

$(E \times E), \lambda')$, where λ' is the restriction of $\mathcal{E}$ to E, is called a $1{-}n$ *prefix* of M. We denote the set of $1{-}n$ prefixes of M by $\mathit{Pref}_{1-n}(M)$.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{n-n}$ and consider $E \subseteq \mathcal{E}$ such that E is $\bowtie_M$*-downward-closed*, i.e, for all $(e, f) \in \bowtie_M$ such that $f \in E$, we also have $e \in E$. Then, the MSC $(E, \rightarrow \cap (E \times E), \lhd \cap (E \times E), \lambda')$, where λ' is the restriction of $\mathcal{E}$ to E, is called a $n{-}n$ *prefix* of M. We denote the set of $n{-}n$ prefixes of M by $\mathit{Pref}_{n-n}(M)$.

2.2 Communicating Systems

Lemma 2.1. *Every prefix of a p2p MSC is a p2p MSC.*

Proof. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{p2p}}$ and let $M_0 = (\mathcal{E}_0, \rightarrow_0, \lhd_0, \lambda_0)$ be a prefix of M, where $\mathcal{E}_0 \subseteq \mathcal{E}$, $\rightarrow_0 \subseteq \rightarrow$, and $\lhd_0 \subseteq \lhd$. Since M is p2p, we have that for every pair $(e, f) \in \lhd$, such that $\lambda(e) = \mathit{send}(p, q, m)$ and $\lambda(f) = \mathit{rec}(p, q, m)$, $\#_{\mathit{Send}(p,q,\_)}(\rightarrow^+, e) = \#_{\mathit{Rec}(p,q,\_)}(\rightarrow^+, f)$. We show that, for every pair $(e, f) \in \lhd_0$, this property still holds. Since $\lhd_0 \subseteq \lhd$, every pair (e, f) that belongs to $\lhd_0$ also belongs to $\lhd$, and we know that $\#_{\mathit{Send}(p,q,\_)}(\rightarrow^+, e) = \#_{\mathit{Rec}(p,q,\_)}(\rightarrow^+, f)$. Because of the $\leq_M$-downward-closeness of $\mathcal{E}_0$, it is easy to see that $\#_{\mathit{Send}(p,q,\_)}(\rightarrow^+, e) = \#_{\mathit{Send}(p,q,\_)}(\rightarrow_0^+, e)$ and $\#_{\mathit{Rec}(p,q,\_)}(\rightarrow^+, f) = \#_{\mathit{Rec}(p,q,\_)}(\rightarrow_0^+, f)$. ☐

Lemma 2.2. *Every prefix of a causally ordered MSC is a causally ordered MSC.*

Proof. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{co}}$ and let $M_0 = (\mathcal{E}_0, \rightarrow_0, \lhd_0, \lambda_0)$ be a prefix of M. By contradiction, suppose that M_0 is not a causally ordered MSC. There must be two distinct $s, s' \in \mathcal{E}_0$ such that $\lambda(s) = \mathit{Send}(\_, q, \_)$, $\lambda(s') = \mathit{Send}(\_, q, \_)$, $s \leq_{M_0} s'$ and either (i) $r' \rightarrow^+ r$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$, or (ii) $s \in \mathit{Unm}(M_0)$ and $s' \in \mathit{Matched}(M_0)$. In both cases, M would also not be a causally ordered MSC, since $\mathcal{E}_0 \subseteq \mathcal{E}$, $\rightarrow_0 \subseteq \rightarrow$, and $\lhd_0 \subseteq \lhd$. This is a contradiction, thus M_0 has to be causally ordered. ☐

Lemma 2.3. *Every $1{-}n$ prefix of a $1{-}n$ MSC is a $1{-}n$ MSC.*

Proof. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{1-n}$ and let $M_0 = (\mathcal{E}_0, \rightarrow_0, \lhd_0, \lambda_0)$ be a $1{-}n$ prefix of M, where $\mathcal{E}_0 \subseteq \mathcal{E}$. Firstly, the $\lessdot_M$-downward-closeness of $\mathcal{E}_0$ guarantees that M_0 is still an MSC. We need to prove that it is a $1{-}n$ MSC. By contradiction, suppose that M_0 is not a $1{-}n$ MSC. Then, there are distinct $e, f \in \mathcal{E}_0$ such that $e \lessdot_{M_0} f \lessdot_{M_0} e$, where $\lessdot_{M_0} = (\rightarrow_0 \cup \lhd_0 \cup \blacktriangleleft_{M_0})^*$. As $\mathcal{E}_0 \subseteq \mathcal{E}$, we have that $\rightarrow_0 \subseteq \rightarrow$, $\lhd_0 \subseteq \lhd$, $\blacktriangleleft_{M_0} \subseteq \blacktriangleleft_M$. Clearly, $\lessdot_{M_0} \subseteq \lessdot_M$, so $e \lessdot_M f \lessdot_M e$. This implies that M is not a $1{-}n$ MSC, because $\lessdot_M$ is cyclic, which is a contradiction. Hence M_0 is a $1{-}n$ MSC. ☐

Note that every $1{-}n$ prefix is also a prefix.

Lemma 2.4. *Every $n{-}n$ prefix of a $n{-}n$ MSC is a $n{-}n$ MSC.*

Proof. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{n-n}$ and let $M_0 = (\mathcal{E}_0, \rightarrow_0, \lhd_0, \lambda_0)$ be a $n{-}n$ prefix of M, where $\mathcal{E}_0 \subseteq \mathcal{E}$. Firstly, the $\bowtie_M$-downward-closeness of $\mathcal{E}_0$ guarantees that M_0 is still an MSC. We need to prove that it is a $n{-}n$ MSC. By contradiction, suppose that M_0 is not a $n{-}n$ MSC. Then, there are distinct $e, f \in \mathcal{E}_0$ such that $e \bowtie_{M_0} f \bowtie_{M_0} e$. As $\mathcal{E}_0 \subseteq \mathcal{E}$, we have that $\rightarrow_0 \subseteq \rightarrow$, $\lhd_0 \subseteq \lhd$, $\ltimes_0 \subseteq \ltimes$. Clearly, $\bowtie_{M_0} \subseteq \bowtie_M$, so $e \bowtie_M f \bowtie_M e$. This implies that M is not a $n{-}n$ MSC, because $\bowtie_M$ is cyclic, which is a contradiction. Hence M_0 is a $n{-}n$ MSC. ☐

Note that every $n{-}n$ prefix is also a prefix.

Lemma 1.2 can be easily extendend to com = co.

Lemma 2.5. *For all* com $\in \{\mathsf{p2p}, \mathsf{mb}, \mathsf{co}\}$, *$L_{\mathrm{com}}(\mathcal{S})$ is prefix-closed: $\mathit{Pref}(L_{\mathrm{com}}(\mathcal{S})) \subseteq L_{\mathrm{com}}(\mathcal{S})$.*

Proof. Follows from Lemma 2.2. ☐

Lemma 2.6. *$L_{1-n}(\mathcal{S})$ is $1{-}n$ prefix-closed: $\mathit{Pref}_{1-n}(L_{1-n}(\mathcal{S})) \subseteq L_{1-n}(\mathcal{S})$.*

Proof. Given a system $\mathcal{S}$, we have that $L_{1-n}(\mathcal{S}) = L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}_{1-n}$. Note that, because of how we defined a $1{-}n$ prefix, we have that $\mathit{Pref}_{1-n}(L_{1-n}(\mathcal{S})) = \mathit{Pref}(L_{1-n}(\mathcal{S})) \cap \mathsf{MSC}_{1-n}$. Moreover, $\mathit{Pref}(L_{1-n}(\mathcal{S})) \subseteq \mathit{Pref}(L_{\mathsf{p2p}}(\mathcal{S}))$, and $\mathit{Pref}(L_{1-n}(\mathcal{S})) \subseteq L_{\mathsf{p2p}}(\mathcal{S})$ for Lemma 1.2. Putting everything together, $\mathit{Pref}_{1-n}(L_{1-n}(\mathcal{S})) \subseteq L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}_{1-n} = L_{1-n}(\mathcal{S})$. ☐

Lemma 2.7. *$L_{n-n}(\mathcal{S})$ is $n{-}n$ prefix-closed: $\mathit{Pref}_{n-n}(L_{n-n}(\mathcal{S})) \subseteq L_{n-n}(\mathcal{S})$.*

Proof. Given a system $\mathcal{S}$, we have that $L_{n-n}(\mathcal{S}) = L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}_{n-n}$. Note that, because of how we defined a $n{-}n$ prefix, we have that $\mathit{Pref}_{n-n}(L_{n-n}(\mathcal{S})) = \mathit{Pref}(L_{n-n}(\mathcal{S})) \cap \mathsf{MSC}_{n-n}$. Moreover, $\mathit{Pref}(L_{n-n}(\mathcal{S})) \subseteq \mathit{Pref}(L_{\mathsf{p2p}}(\mathcal{S}))$, and $\mathit{Pref}(L_{n-n}(\mathcal{S})) \subseteq L_{\mathsf{p2p}}(\mathcal{S})$ for Lemma 1.2. Putting everything together, $\mathit{Pref}_{n-n}(L_{n-n}(\mathcal{S})) \subseteq L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}_{n-n} = L_{n-n}(\mathcal{S})$. ☐

2.3 Model Checking

Proposition 2.1. The set $\mathsf{MSC}_{\mathsf{co}}$ of causally odered MSCs is MSO-definable.

Proof. Given an MSC M, it is causally ordered if it satisfies the MSO formula

$$\varphi_{\mathsf{co}} = \neg\exists s.\exists s'. \left(\bigvee_{\substack{q\in\mathbb{P} \\ a,b\in Send(\_,q,\_)}} \lambda(s) = a \;\wedge\; \lambda(s') = b \;\wedge\; s \leq_M s' \;\wedge\; (\psi_1 \vee \psi_2) \right)$$

where ψ_1 and ψ_2 are

$$\psi_1 = \exists r.\exists r'. \begin{pmatrix} s \lhd r & \wedge \\ s' \lhd r' & \wedge \\ r' \rightarrow^+ r & \end{pmatrix} \qquad \psi_2 = (\neg matched(s) \wedge matched(s'))$$

$$matched(x) = \exists y.x \lhd y$$

The property φ_{co} says that there cannot be two send events s and s', with the same recipient, such that $s \leq_M s'$ and either (i) their corresponding receive events r and r' happen in the opposite order, i.e. $r' \rightarrow^+ r$, or (ii) s is unmatched and s' is matched. The set $\mathsf{MSC}_{\mathsf{co}}$ of causally ordered MSCs is therefore MSO-definable as $\mathsf{MSC}_{\mathsf{co}} = L(\varphi_{\mathsf{co}})$.

□

Knowing that $\mathsf{MSC}_{\mathsf{co}}$ is MSO-definable, Theorem 1.5 can be restated for com $= \mathsf{co}$.

Theorem 2.8. *The bounded model-checking problem for* com $= \mathsf{co}$ *is decidable.*

Proof. By Proposition 2.1, $\mathsf{MSC}_{\mathsf{co}} = L(\varphi_{\mathsf{co}})$. Given a system $\mathcal{S}$, we have that $L_{\mathsf{co}}(\mathcal{S}) = L_{\mathsf{p2p}}(\mathcal{S}) \cap L(\varphi_{\mathsf{co}})$. Therefore, we can rewrite the bounded model checking problem for com $= \mathsf{co}$ as

$$\begin{aligned} & L_{\mathsf{co}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap L(\varphi_{\mathsf{co}}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \cup L(\neg\varphi_{\mathsf{co}}) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi \vee \neg\varphi_{\mathsf{co}})\,. \end{aligned}$$

The latter is decidable due to Fact 1.4. □

Theorem 2.9. *The bounded model-checking problem for* com $= 1{-}\mathsf{n}$ *is decidable.*

Proof. By Proposition ??, $\mathsf{MSC}_{1-n} = L(\varphi_{1-n})$. Given a system $\mathcal{S}$, we have that $L_{1-n}(\mathcal{S}) = L_{\mathsf{p2p}}(\mathcal{S}) \cap L(\varphi_{1-n})$. Therefore, we can rewrite the bounded model checking problem for com $= 1{-}\mathsf{n}$ as

$$\begin{aligned} & L_{1-n}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap L(\varphi_{1-n}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \cup L(\neg\varphi_{1-n}) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi \vee \neg\varphi_{1-n})\,. \end{aligned}$$

The latter is decidable due to Fact 1.4. □

Theorem 2.10. *The bounded model-checking problem for* com $= \mathsf{n}{-}\mathsf{n}$ *is decidable.*

Proof. By Proposition ??, $\mathsf{MSC}_{n-n} = L(\varphi_{n-n})$. Given a system $\mathcal{S}$, we have that $L_{n-n}(\mathcal{S}) = L_{\mathsf{p2p}}(\mathcal{S}) \cap L(\varphi_{n-n})$. Therefore, we can rewrite the bounded model checking problem for com $= \mathsf{n}{-}\mathsf{n}$ as

$$\begin{aligned} & L_{n-n}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap L(\varphi_{n-n}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi) \cup L(\neg\varphi_{n-n}) \\ \iff\; & L_{\mathsf{p2p}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi \vee \neg\varphi_{n-n})\,. \end{aligned}$$

The latter is decidable due to Fact 1.4. □

Table 2: Summary of the decidability of the synchronizability problem for different classes of MSCs.

	P2P	Causally ordered	Mailbox
Weakly synchronous	Undecidable [Thm. 1.10]	Undecidable [Thm. 2.20]	EXPTIME [Thm. 1.9]
Weakly k-synchronous	Decidable [Thm. 2.21]		
Existentially k-bounded	Decidable	Decidable	Decidable

2.4 Synchronizability

Lemma 2.11. *Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. For all $M \in \mathsf{MSC} \setminus \mathcal{C}$, we have $(\mathit{Pref}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset$.*

Proof. Let k and $\mathcal{C}$ be fixed, and let $M \in \mathsf{MSC} \setminus \mathcal{C}$ be fixed. If the empty MSC is not in $\mathcal{C}$, then we are done, since it is a valid prefix of M and it is in $\mathsf{MSC}^{(k+2)\text{-stw}} \setminus \mathcal{C}$. Otherwise, let $M' \in \mathit{Pref}(M) \setminus \mathcal{C}$ such that, for all $\leq_M$-maximal events e of M', removing e (along with its adjacent edges) gives an MSC in $\mathcal{C}$. In other words, M' is the "shortest" prefix of M that is not in $\mathcal{C}$. We obtain such an MSC by successively removing $\leq_M$-maximal events. Let e be $\leq_{M'}$-maximal and let $M'' = M' \setminus \{e\}$. Since M' was taken minimal in terms of number of events, $M'' \in \mathcal{C}$. So Eve has a winning strategy with $k+1$ pebbles for M''. Let us design a winning strategy with $k+3$ pebbles for Eve for M', which will show the claim.

Observe that the event e occurs at the end of the timeline of a process (say p), and it is part of at most two edges:

- one with the previous p-event (if any)
- one with the corresponding send event (if e is a receive event)

Let e_1, e_2 be the two neighbours of e. The strategy of Eve is the following: in the first round, mark e, e_1, e_2, then erase the edges (e_1, e) and (e_2, e), then split the remaining graph in two parts: M'' on the one side, and the single node graph $\{e\}$ on the other side. Then Eve applies its winning strategy for M'', except that initially the two events e_1, e_2 are marked (so she may need up to $k+3$ pebbles). □

Lemma 2.12. *Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. For all $M \in \mathsf{MSC}_{1-n} \setminus \mathcal{C}$, we have $(\mathit{Pref}_{1-n}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset$.*

Proof. Let k and $\mathcal{C}$ be fixed, and let $M \in \mathsf{MSC}_{1-n} \setminus \mathcal{C}$ be fixed. If the empty MSC is not in $\mathcal{C}$, then we are done, since it is a valid $1-n$ prefix of M and it is in $\mathsf{MSC}^{(k+2)\text{-stw}} \setminus \mathcal{C}$. Otherwise, let $M' \in \mathit{Pref}_{1-n}(M) \setminus \mathcal{C}$ such that, for all $\lessdot_M$-maximal events e of M', removing e (along with its adjacent edges) gives an MSC in $\mathcal{C}$. In other words, M' is the "shortest" prefix of M that is not in $\mathcal{C}$. We obtain such an MSC by successively removing $\lessdot_M$-maximal events. Let e be $\lessdot_{M'}$-maximal and let $M'' = M' \setminus \{e\}$. Since M' was taken minimal in terms of number of events, $M'' \in \mathcal{C}$. The proof proceeds exactly as the proof of Lemma 2.11. □

Lemma 2.13. *Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. For all $M \in \mathsf{MSC}_{n-n} \setminus \mathcal{C}$, we have $(\mathit{Pref}_{n-n}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset$.*

Proof. Let k and $\mathcal{C}$ be fixed, and let $M \in \mathsf{MSC}_{n-n} \setminus \mathcal{C}$ be fixed. If the empty MSC is not in $\mathcal{C}$, then we are done, since it is a valid $n-n$ prefix of M and it is in $\mathsf{MSC}^{(k+2)\text{-stw}} \setminus \mathcal{C}$. Otherwise, let $M' \in \mathit{Pref}_{n-n}(M) \setminus \mathcal{C}$ such that, for all $\bowtie_M$-maximal events e of M', removing e (along with its adjacent edges) gives an MSC in $\mathcal{C}$. In other words, M' is the "shortest" prefix of M that is not in $\mathcal{C}$. We obtain such an MSC by successively removing $\bowtie_M$-maximal events. Let e be $\bowtie_{M'}$-maximal and let $M'' = M' \setminus \{e\}$. Since M' was taken minimal in terms of number of events, $M'' \in \mathcal{C}$. The proof proceeds exactly as the proof of Lemma 2.11. □

Note that Lemma 1.6 can be extended to com $= \mathsf{co}$, since Lemma 1.7 does not depend on the kind of communication used by the system.

Lemma 2.14. *Let $\mathcal{S}$ be a communicating system, com $\in \{\mathsf{p2p}, \mathsf{mb}, \mathsf{co}\}$, $k \in \mathbb{N}$, and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. Then, $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$ iff $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$.*

Lemma 2.15. *Let $\mathcal{S}$ be a communicating system, $k \in \mathbb{N}$, and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. Then, $L_{1-n}(\mathcal{S}) \subseteq \mathcal{C}$ iff $L_{1-n}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$.*

Proof. Follows from Lemma 2.6 and Lemma 2.12. □

Lemma 2.16. *Let $\mathcal{S}$ be a communicating system, $k \in \mathbb{N}$, and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. Then, $L_{n-n}(\mathcal{S}) \subseteq \mathcal{C}$ iff $L_{n-n}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$.*

Proof. Follows from Lemma 2.7 and Lemma 2.13. □

Theorem 1.8 can also be extended to com = co.

Theorem 2.17. *Fix finite sets $\mathbb{P}$ and $\mathbb{M}$. Suppose com $\in \{\mathsf{p2p}, \mathsf{mb}, \mathsf{co}\}$ and let $\mathcal{C} \subseteq \mathsf{MSC}$ be an MSO-definable and STW-bounded class (over $\mathbb{P}$ and $\mathbb{M}$). The following problem is decidable: Given a communicating system $\mathcal{S}$, do we have $L_{\text{com}}(\mathcal{S}) \subseteq \mathcal{C}$?*

Proof. Same as the proof for Theorem 1.8, but using Lemma 2.14 in place of Lemma 1.6, and Theorem 2.8 in place of Theorem 1.5. □

Theorem 2.18. *Fix finite sets $\mathbb{P}$ and $\mathbb{M}$. Let $\mathcal{C} \subseteq \mathsf{MSC}$ be an MSO-definable and STW-bounded class (over $\mathbb{P}$ and $\mathbb{M}$). The following problem is decidable: Given a communicating system $\mathcal{S}$, do we have $L_{1-n}(\mathcal{S}) \subseteq \mathcal{C}$?*

Proof. Same as the proof for Theorem 1.8, but using Lemma 2.15 in place of Lemma 1.6, and Theorem 2.9 in place of Theorem 1.5. □

Theorem 2.19. *Fix finite sets $\mathbb{P}$ and $\mathbb{M}$. Let $\mathcal{C} \subseteq \mathsf{MSC}$ be an MSO-definable and STW-bounded class (over $\mathbb{P}$ and $\mathbb{M}$). The following problem is decidable: Given a communicating system $\mathcal{S}$, do we have $L_{n-n}(\mathcal{S}) \subseteq \mathcal{C}$?*

Proof. Same as the proof for Theorem 1.8, but using Lemma 2.16 in place of Lemma 1.6, and Theorem 2.10 in place of Theorem 1.5. □

2.4.1 Weakly synchronous causally ordered MSCs

Corollary 1.8.1 can be extended to com = co.

Proposition 2.2. The set of weakly synchronous *causally ordered* MSCs is MSO-definable.

Proof. Both the sets of weakly synchronous MSCs and of causally ordered MSCs are MSO-definable, as shown by Corollary 1.8.1 and Proposition 2.1. Recall that any LCPDL-definable property is also MSO-definable. It suffices to take the conjuction of the two respective MSO formulas. □

Theorem 2.20. *The following problem is undecidable: Given finite sets $\mathbb{P}$ and $\mathbb{M}$ as well as a communicating system $\mathcal{S}$, is every MSC in $L_{\mathsf{co}}(\mathcal{S})$ weakly synchronous?*

Proof. The proof is essentially identical to the p2p case. We do the same reduction from the Post correspondence problem. Recall from the proof of Theorem 1.10 that we consider a system $\mathcal{S}$ with four machines (P1, P2, V1, V2), where we have unidirectional communication channels from provers to verifiers. In particular notice that all the possible behaviours of $\mathcal{S}$ are causally ordered, i.e. $L_{\mathsf{p2p}}(\mathcal{S}) \subseteq \mathsf{MSC}_{\mathsf{co}}$; according to how we built our system $\mathcal{S}$, it is impossible to have a pair of causally-related send events of P1 and P2[1], which implies that causal ordering is already ensured by any possible p2p behaviour of $\mathcal{S}$. The rest of the proof is identical to the p2p case. □

Corollary 2.20.1. *The set of weakly synchronous causally ordered MSCs has unbounded special tree-width.*

Proof. Suppose that the set of weakly synchronous causally ordered MSCs is STW-bounded. By Proposition 2.2 and Theorem 2.17, we have that the syncronicity problem for the class of weakly synchronous causally ordered MSCs would be decidable. This is a contradiction, since Theorem 2.20 states that this problem is undecidable. □

[1]There is no channel between P1 and P2, and we only have unidirectional communication channels from provers to verifiers; it is impossible to have a causal path between two send events of P1 and P2.

2.4.2 Weakly k-synchronous causally ordered MSCs

Proposition 2.3. The set of weakly k-synchronous causally ordered MSCs is MSO-definable.

Proof. Both the sets of weakly k-synchronous MSCs and of causally ordered MSCs are MSO-definable, as shown by Proposition 1.3 and Proposition 2.1. It suffices to take the conjuction of the two respective MSO formulas. $\square$

Theorem 1.8 can be easily extended to $\text{com} = \mathsf{co}$.

Theorem 2.21. *For* $\text{com} \in \{\mathsf{p2p}, \mathsf{mb}, \mathsf{co}\}$, *the following problem is decidable: Given finite sets* $\mathbb{P}$ *and* $\mathbb{M}$, *a communicating system* $\mathcal{S}$, *and* $k \in \mathbb{N}$, *is every MSC in* $L_{\text{com}}(\mathcal{S})$ *weakly* k*-synchronous?*

Proof. By Proposition 2.3 and Proposition 1.4 we have that the class of causally ordered k-synchronous MSCs is MSO-definable and STW-bounded[2]. Theorem 2.17 ends the proof. $\square$

2.4.3 Existentially k causally ordered bounded MSCs

Definition 2.2. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$ and $k \in \mathbb{N}$. A linearization $\rightsquigarrow$ of M is called *k-bounded* if, for all $e \in \mathit{Matched}(M)$, with $\lambda(e) = \mathit{send}(p, q, m)$, we have

$$\#_{\mathit{Send}(p,q,\_)}(\rightsquigarrow, e) - \#_{\mathit{Rec}(p,q,\_)}(\rightsquigarrow, e) \leq k.$$

Recall that $\#_{\mathit{Send}(p,q,\_)}(\rightsquigarrow, e)$ denotes the number of send events from p to q that occured before e, according to $\rightsquigarrow$.

Definition 2.3. An MSC is said to be *existentially p2p bounded* ($\exists k$-$p2p$-bounded) if it has a k-bounded linearization.

Definition 2.4. An MSC is said to be *existentially k causally ordered bounded* ($\exists k$-co-bounded) if it is causally ordered and it has a k-bounded linearization.

Note that every existentially k causally ordered bounded MSC is an existentially k-p2p-bounded MSC.

Proposition 2.4. For all $k \in \mathbb{N}$, the set of $\exists k$-co-bounded MSCs is MSO-definable and STW-bounded.

Proof. Let $\mathsf{MSC}_{p2p\text{-}\exists k\text{-}\text{-}b}$ and $\mathsf{MSC}_{\mathsf{co}\text{-}\exists k\text{-}\text{-}b}$ be the set of existentially k-p2p-bounded MSCs and the set of existentially k causally ordered bounded MSCs, respectively. $\mathsf{MSC}_{p2p\text{-}\exists k\text{-}\text{-}b}$ was shown to be both MSO-definable (in Lohrey and Muscholl 2004) and STW-bounded (in Bollig and Gastin 2019, Proposition 5.4, page 163). $\mathsf{MSC}_{\mathsf{co}\text{-}\exists k\text{-}\text{-}b}$ also has to be STW-bounded, since we have $\mathsf{MSC}_{\mathsf{co}\text{-}\exists k\text{-}\text{-}b} \subseteq \mathsf{MSC}_{p2p\text{-}\exists k\text{-}\text{-}b}$. Note that, by definition, $\mathsf{MSC}_{\mathsf{co}\text{-}\exists k\text{-}\text{-}b} = \mathsf{MSC}_{p2p\text{-}\exists k\text{-}\text{-}b} \cap \mathsf{MSC}_{\mathsf{co}}$. Since both $\mathsf{MSC}_{p2p\text{-}\exists k\text{-}\text{-}b}$ and $\mathsf{MSC}_{\mathsf{co}}$ can be defined by an MSO formula, the latter according to Proposition 2.1, $\mathsf{MSC}_{\mathsf{co}\text{-}\exists k\text{-}\text{-}b}$ is also MSO-definable[3]. $\square$

Theorem 2.22. *The following problem is decidable: Given finite sets* $\mathbb{P}$ *and* $\mathbb{M}$, *a communicating system* $\mathcal{S}$, *and* $k \in \mathbb{N}$, *is every MSC in* $L_{\mathsf{co}}(\mathcal{S})$ $\exists k$*-*co*-bounded?*

Proof. Directly follows from 2.4 and 2.17. $\square$

3 Sketches

3.1 MSCs and partial order

D: Give intuition of what is an MSC and how MSCs can be used to represent graphically the behaviour of a system in terms of send/receive events. Talk about partial order on MSCs (causal relation/paths) and linearizations.

D: Clarify terminology: 'message delivery' and 'receive event'

[2]Note that Proposition 1.4 is independent from the type of communication.

[3]Suppose $\varphi_{\exists k\text{-}p2p\text{-}b}$ is the MSO formula for $\mathsf{MSC}_{p2p\text{-}\exists k\text{-}\text{-}b}$, and φ_{co} is the MSO formula for $\mathsf{MSC}_{\mathsf{co}}$. Then, $\mathsf{MSC}_{\mathsf{co}\text{-}\exists k\text{-}\text{-}b}$ is defined by $\varphi_{\exists k\text{-}\mathsf{co}\text{-}b} = \varphi_{\exists k\text{-}p2p\text{-}b} \wedge \varphi_{\mathsf{co}}$

3.2 Communication architectures and variants

A communicating system, intended as a set of finite-state machines that can exchange messages through channels, may use different *communication architectures*. We will consider the following:

- *Fully asynchronous*: a fully asychronous architecture (or simply asynchronous from now on) can be modeled as a collection of channels between each pair (M_1, M_2) of machines, such that M_1 can send messages to M_2. Following this description, given two machines M_1 and M_2 that can exchange messages in both directions, we have two channels: one for the messages sent by M_1 to M_2, and the other for the messages sent by M_2 to M_1. The channels behave as *bags*, which means that they do not guarantee any specific order on the delivery of messages. This architecture is equivalent to the "Fully asynchronous" communication model in Chevrou, Hurault, and Quéinnec 2016.

  D: Is it true? For the set of MSCs yes, it should be.

- *FIFO* 1−1 *(p2p)*: this is a variant of the asynchronous architecture in which channels operate in FIFO mode (i.e. as queues). This means that messages between a couple of peers are delivered in their send order, whereas messages from/to different peers are delivered independently. We will use the term *peer-to-peer* (p2p) as a synonym of FIFO 1−1. This architecture is equivalent to the "FIFO 1−1" communication model in Chevrou, Hurault, and Quéinnec 2016.

- *Causally orderered*: this can be described as a more specific variant of the p2p architecture. In a causally orderered architecture, the communicating system ensures that messages are delivered according to the causality of their emissions. In other words, if a message m_1 is causally sent before a message m_2 (i.e. there exists a causal path from the first emission to the second one), then a peer cannot receive m_2 before m_1. Channels still operate in FIFO mode, but the delivery of some messages might be delayed to enforce causal ordering. This architecture is equivalent to the "Causally ordered" communication model in Chevrou, Hurault, and Quéinnec 2016. In literature, several implementations of causal ordering have been proposed. For instance, the algorithm described in Schiper, Eggli, and Sandoz 1989 makes use of the logical vector clocks introduced by Mattern-Fidge Fidge 1988; Mattern 1989 to enforce causal ordering.

- *FIFO* n−1 *(mailbox)*: in a mailbox architecture, each machine merges all of its incoming messages (from any source) into a unique queue. In other words, we can model it as each machine P_i having a single incoming FIFO channel, which is shared by the other machines that can send messages to P_i. A send event consists in adding the message at the end of the queue of the destination peer. This architecture is equivalent to the "FIFO n−1" communication model in Chevrou, Hurault, and Quéinnec 2016.

- *FIFO* 1−n *(mailbox)*: in a 1−n architecture, the messages sent from a single machine are always delivered in their send order, independently of the recipients. Its implementation is not expensive: each machine has a unique queue to store sent messages. The recipients fetch messages from this queue and acknowledge their reception. After the acknowledgement, the next message in the queue can be fetched. This architecture is equivalent to the "FIFO 1−n" communication model in Chevrou, Hurault, and Quéinnec 2016.

- *FIFO* n−n: this architecture can be modeled as a unique shared FIFO channel. Messages are globally ordered and delivered according to the their emission order. As noted in Chevrou, Hurault, and Quéinnec 2016, this model is generally unrealistic and its implementations are inefficient.

For convenience, we will refer to a system that uses the p2p architecture simply as a p2p system. The same shorthand will be used for the other communication architectures. Note that, for each of these communicating architectures, there may be other equivalent ways of describing/modeling them.

3.3 Definitions

A Message Sequence Chart (MSC), such as the one in Fig. **??**, provides a visual description of the behaviour of a distributed system. In this section, we start by formally defining the most generic class of Message Sequence Charts (MSCs), which we call asynchronous MSCs. More specialized classes of MSCs, such as *p2p* MSCs, will also be discussed. Intuitively, we say that an MSC M is asynchronous if there is an asynchronous system $\mathcal{S}$ that can exhibit the behaviour described by M.

Definition 3.1 (Asynchronous MSC)**.** An *asynchronous MSC* (or simply MSC) over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, where $\mathcal{E}$ is a finite (possibly empty) set of *events* and $\lambda : \mathcal{E} \rightarrow \Sigma$ is a labeling function that associates an action to each event. For $p \in \mathbb{P}$, let $\mathcal{E}_p = \{e \in \mathcal{E} \mid \lambda(e) \in \Sigma_p\}$ be the set of events that are executed by p. We require that $\rightarrow$ (the *process relation*) is the disjoint union $\bigcup_{p\in\mathbb{P}} \rightarrow_p$ of relations $\rightarrow_p \subseteq \mathcal{E}_p \times \mathcal{E}_p$ such that $\rightarrow_p$ is the direct successor relation of a total order on $\mathcal{E}_p$. For an event $e \in \mathcal{E}$, a set of actions $A \subseteq \Sigma$, and a relation $R \subseteq \mathcal{E} \times \mathcal{E}$, let $\#_A(R, e) = |\{f \in \mathcal{E} \mid (f, e) \in R \text{ and } \lambda(f) \in A\}|$. We require that $\lhd \subseteq \mathcal{E} \times \mathcal{E}$ (the *message relation*) satisfies the following:

(1) for every pair $(e, f) \in \lhd$, there is a send action $send(p, q, m) \in \Sigma$ such that $\lambda(e) = send(p, q, m)$, $\lambda(f) = rec(p, q, m)$.

(2) for all $f \in \mathcal{E}$ such that $\lambda(f)$ is a receive action, there is exactly one $e \in \mathcal{E}$ such that $e \lhd f$.

Finally, letting $\leq_M = (\rightarrow \cup \lhd)^*$, we require that $\leq_M$ is a partial order. For convenience, we simply write $\leq$ when M is clear from the context. We will refer to $\leq$ as the *causal ordering* or *happens-before* relation. If, for two events e and f, we have that $e \leq f$, we will equivalently say that there is a *causal path* between e and f.

According to Condition (2), every receive event must have a matching send event. Note that, however, there may be unmatched send events. We let $SendEv(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a send action}\}$, $RecEv(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a receive action}\}$, $Matched(M) = \{e \in \mathcal{E} \mid \text{there is } f \in \mathcal{E} \text{ such that } e \lhd f\}$, and $Unm(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a send action and there is no } f \in \mathcal{E} \text{ such that } e \lhd f\}$. We do not distinguish isomorphic MSCs and let $\mathsf{MSC}_{\mathsf{asy}}$ be the set of all the asynchronous MSCs over the given sets $\mathbb{P}$ and $\mathbb{M}$.

Linearizations. Consider $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{asy}}$. A *linearization* of M is a (reflexive) total order $\rightsquigarrow \subseteq \mathcal{E} \times \mathcal{E}$ such that $\leq_M \subseteq \rightsquigarrow$. In other words, a linearization of M is a total order that respects the happens-before relation $\leq_M$ defined over M.

D: Provide example of linearization.

Asynchronous MSCs are the widest class of MSCs that we will deal with. By introducing additional constraints, we are able to define other classes of MSCs that exclusively describe the behaviours of p2p systems, mailbox systems, and so on.

D: Provide example of asynchronous MSC that is not p2p.

As in the asynchronous case, we say that M is a p2p MSC if there is a p2p system that can produce the behaviour described by M. We give here the formal definition of p2p MSC, which also considers the possibility of having unmatched messages (i.e. messages that are sent but not received).

D: Why unmatched messages? What do they represent? When an automata does not have the receive action for a message that was already sent (see examples at the end of concur paper).

Definition 3.2 (Peer-to-peer MSCs)**.** A *p2p MSC* (or simply *MSC*) is an asynchronous MSC where we require that, for every pair $(e, f) \in \lhd$, such that $\lambda(e) = send(p, q, m)$, $\lambda(f) = rec(p, q, m)$, we have $\#_{Send(p,q,\_)}(\rightarrow^+, e) = \#_{Rec(p,q,\_)}(\rightarrow^+, f)$.

The additional constraint satisfied by p2p MSCs ensures that channels operate in FIFO mode; when a process q receives a message from a process p, it must have already received all the messages that were previously sent to him by p. Let $\mathsf{MSC}_{\mathsf{p2p}}$ denote the set of all the p2p MSCs over two given sets $\mathbb{P}$ and $\mathbb{M}$. Note that, by definition, every p2p MSC is an asynchronous MSC. The idea is that we are always able to find an asynchronous system that *can* exhibit the behaviour described by a p2p MSC; after all, the channels of an asynchronous system do not have to follow any specific behaviour, so they can indeed happen to operate as if they were queues. Example **??** shows that the opposite direction is generally not true, an asynchronous MSC is not always a p2p MSC. It follows that $\mathsf{MSC}_{\mathsf{p2p}} \subset \mathsf{MSC}_{\mathsf{asy}}$.

We will now consider the class of MSCs for which there is a causally ordered system that can produce their behaviour. Intuitively, an MSC is causally ordered if all the messages sent to the same process are received in an order which is consistent with the causal ordering of the corresponding send events. Below the formal definition, which also considers unmatched messages.

D: Provide example of an MSC that is not causally ordered

Definition 3.3 (Causally ordered MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is *causally ordered* if, for any two send events s and s', such that $\lambda(s) = Send(\_, q, \_)$, $\lambda(s') = Send(\_, q, \_)$, and $s \leq_M s'$, we have either:

- $s, s' \in Matched(M)$ and $r \rightarrow^* r'$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$.
- $s' \in Unm(M)$.

By definition, every causally ordered MSC is a p2p MSC. This is not surprising, considering that a causally ordered system is essentially a p2p system with an additional constraint on the delivery of messages; indeed, we are always able to find a p2p system that *can* exhibit the behaviour described by a causally ordered MSC. Let $\mathsf{MSC}_{\mathsf{co}}$ denote the set of all the causally ordered MSCs over two given sets $\mathbb{P}$ and $\mathbb{M}$. Example **??** shows that a p2p MSC is not always a causally ordered MSC. It follows that $\mathsf{MSC}_{\mathsf{co}} \subset \mathsf{MSC}_{\mathsf{p2p}}$.

Moving on to the mailbox semantics, we say that M is a mailbox MSC if there is a mailbox system that can exhibit the behaviour described by M.

D: Provide example of MSC which is not mailbox

Definition 3.4 (Mailbox MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *mailbox MSC* if it has a linearization $\leadsto$ where, for any two send events s and s', such that $\lambda(s) = Send(\_, q, \_)$, $\lambda(s') = Send(\_, q, \_)$, and $s \leadsto s'$, we have either:

- $s, s' \in Matched(M)$ and $r \leadsto r'$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$.
- $s' \in Unm(M)$.

Such a linearization will be referred to as a *mailbox linearization*, and the symbol $\overset{\text{mb}}{\leadsto}$ will be used to denote one. Let $\mathsf{MSC}_{\mathsf{mb}}$ denote the set of all the mailbox MSCs over two given sets $\mathbb{P}$ and $\mathbb{M}$. By definition, every mailbox MSC is a p2p MSC. Conversely, Example **??** shows a p2p MSC which is not a mailbox MSC. It follows that $\mathsf{MSC}_{\mathsf{mb}} \subset \mathsf{MSC}_{\mathsf{p2p}}$. We show here that each mailbox MSC is also a causally ordered MSC.

D: Provide example of causally ordered MSC which is not mailbox (Figure 2.18 of Laetitia's thesis).

Proposition 3.1. Every mailbox MSC is a causally ordered MSC.

Proof. Let M be a mailbox MSC and $\leadsto$ a mailbox linearization of it. Recall that a linearization has to respect the happens-before partial order over M, i.e. $\leq_M \subseteq \leadsto$. Consider any two send events s and s', such that $\lambda(s) = Send(\_, q, \_)$, $\lambda(s') = Send(\_, q, \_)$ and $s \leq_M s'$. Since $\leq_M \subseteq \leadsto$, we have that $s \leadsto s'$ and, by the definition of mailbox linearization, either (i) $s' \in Unm(M)$, or (ii) $s, s' \in Matched(M)$, $s \lhd r$, $s' \lhd r'$ and $r \leadsto r'$. The former clearly respects the definition of causally ordered MSC, so let us focus on the latter. Note that r and r' are two receive events executed by the same process, hence $r \leadsto r'$ implies $r \rightarrow^+ r'$. It follows that M is a causally ordered MSC. □

Moving on to the $1{-}n$ semantics, we say that M is a $1{-}n$ MSC if there is a $1{-}n$ system that can exhibit the behaviour described by M.

Definition 3.5 ($1{-}n$ MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a $1{-}n$ *MSC* if it has a linearization $\leadsto$ where, for any two send events s and s', such that $\lambda(s) = Send(p, \_, \_)$, $\lambda(s') = Send(p, \_, \_)$, and $s \rightarrow^+ s'$ (which implies $s \leadsto s'$), we have either:

- $s, s' \in Matched(M)$ and $r \leadsto r'$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$.
- $s' \in Unm(M)$.

Such a linearization will be referred to as a $1-n$ *linearization*. Note that the definition is very similar to the mailbox case, but here s and s' are two send events executed by the same process. Let MSC_{1-n} denote the set of all the $1-n$ MSCs over two given sets $\mathbb{P}$ and $\mathbb{M}$. By definition, every $1-n$ MSC is a p2p MSC. Conversely, Example **??** shows a p2p MSC which is not a $1-n$ MSC. It follows that $\mathsf{MSC}_{\mathsf{mb}} \subset \mathsf{MSC}_{\mathsf{p2p}}$. We show here that each $1-n$ MSC is also a causally ordered MSC, which is not as intuitive as the mailbox case.

D: Provide example of MSC which is not $1-n$

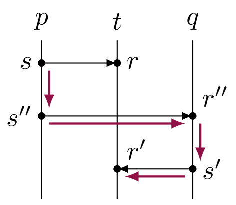
D: Should I still leave this proof if we showed that every $1-n$ MSC is a mailbox MSC?

Proposition 3.2. Every $1-n$ MSC is a causally ordered MSC.

Proof. By contradiction. Suppose that M is a $1-n$ MSC, but not a causally ordered MSC. Since M is not causally ordered, there must be two send events s and s' such that $\lambda(s) = \mathit{Send}(\_, q, \_)$, $\lambda(s') = \mathit{Send}(\_, q, \_)$, $s \leq_M s'$, and we have either:

1. $s, s' \in \mathit{Matched}(M)$ and $r' \rightarrow^* r$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$.
2. $s \in \mathit{Unm}(M)$ and $s' \in \mathit{Matched}(M)$.

We need to show that both of these scenarios lead to a contradiction. (1) Suppose s and s' are executed by the same process. Since M is a $1-n$ MSC, there must be a linearization $\rightsquigarrow$ such that $r \rightsquigarrow r'$, but this is clearly impossible since we have $r' \rightarrow^* r$. Suppose now that s and s' are executed by two different processes p and q. We know by hypothesis that $s \leq_M s'$, i.e. there is a causal path of events $P = s \sim a \sim \cdots \sim s' \sim r'$ from s to r', where $\sim$ is either $\rightarrow$ or $\lhd$. Refer to the first example in Figure 6 for a visual representation (P is drawn in purple). To have a causal path P, there must be a send event s'' that is executed by p after s and that is part of P, along with its receipt r'' (i.e. $P = s \leq_M s'' \lhd r'' \leq_M s' \lhd r'$). We clearly have $r'' \rightsquigarrow r'$ for any linearization of M, because $r'' \leq_M r'$ (they are both in the causal path P and r'' happens before r). Since M is a $1-n$ MSC, there has to be a linearization $\rightsquigarrow$ where $r \rightsquigarrow r''$, because s and s'' are send events executed by the same process. It follows that M should have a linearization were $r \rightsquigarrow r'' \rightsquigarrow r'$, but this is not possible because of the hypothesis that $r' \rightarrow^* r$. This is a contradiction. (2) Suppose s and s' are executed by the same process. It is trivial to see, by definition, that M cannot be a $1-n$ MSC. Suppose now that s and s' are executed by two different processes p and q, and consider the same send event s'' as before (executed by p). Refer to the second example in Figure 6 for a visual representation. Since s'' is matched, we have two events s and s'', sent by the same process p, that are unmatched and matched, respectively. Clearly, M cannot be a $1-n$ MSC. $\square$



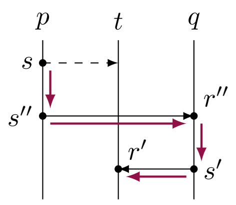


Figure 6: Two examples of $1-n$ MSCs.

D: Provide example of causally ordered MSC which is not $1-n$ (same as mailbox! Figure 2.18 of Laetitia's thesis).

Definition 3.6 ($1-n$ alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, we define an additional binary relation that represents a constraint under the $1-n$ semantics, which ensures that messages sent from the same process are received in the same order. Let $\blacktriangleleft_M \subseteq \mathcal{E} \times \mathcal{E}$ be defined as $e_1 \blacktriangleleft_M e_2$ if there are two events e_1 and e_2, and $p \in \mathbb{P}$ such that either:

- $\lambda(e_1) \in \mathit{Send}(p, \_, \_)$, $\lambda(e_2) \in \mathit{Send}(p, \_, \_)$, $e_1 \in \mathit{Matched}(M)$, and $e_2 \in \mathit{Unm}(M)$, or
- $\lambda(e_1) \in \mathit{Rec}(p, \_, \_)$, $\lambda(e_2) \in \mathit{Rec}(p, \_, \_)$, $s_1 \lhd e_1$ and $s_2 \lhd e_2$ for some $s_1, s_2 \in \mathcal{E}_p$, and $s_1 \rightarrow^+ s_2$.

We let $\lessdot_M = (\rightarrow \cup \lhd \cup \blacktriangleleft_M)^*$. Note that $\leq_M \subseteq \lessdot_M$. We call $M \in \mathsf{MSC}_{\mathsf{asy}}$ a $1-n$ *MSC* if $\lessdot_M$ is a partial order.

D: Should I also include the proof that every mailbox MSC without unmatched messages is $1-n$? Do we need it?

Proposition 3.3. Every $1-n$ MSC without unmatched messages is a mailbox MSC.

Proof. We show that the contrapositive is true, i.e. if an MSC is not mailbox (and it does not have unmatched messages), it is also not $1-n$. Suppose M is an asynchronous MSC, but not mailbox. There must be a cycle ξ such that $e \preceq e$, for some event e. Recall that $\preceq = (\rightarrow \cup \lhd \cup \sqsubset)^*$ and $\leq = (\rightarrow \cup \lhd)^*$. We can always explicitely write a cycle $e \preceq e$ only using $\sqsubset$ and $\leq$. For instance, there might be a cycle $e \preceq e$ because we have that $e \sqsubset f \leq g \sqsubset h \sqsubset i \leq e$. Consider any two adiacent events s_1 and s_2 in the cycle ξ, where ξ has been written using only $\sqsubset$ and $\leq$, and we never have two consecutive $\leq$[4]. We have two cases:

1. $s_1 \sqsubset s_2$. We know, by definition of $\sqsubset$, that s_1 and s_2 must be two send events and that $r_1 \rightarrow^+ r_2$, where r_1 and r_2 are the receive events that match with s_1 and s_2, respectively (we are not considering unmatched messages by hypothesis).

2. $s_1 \leq s_2$. Since M is asynchronous by hyphotesis, ξ has to contain at least one $\sqsubset$[5]; recall that we also wrote ξ in such a way that we do not have two consecutive $\leq$. It is not difficult to see that s_1 and s_2 have to be send events, since they belong to ξ. We have two cases:

   (a) r_1 is in the causal path, i.e. $s_1 \lhd r_1 \leq s_2$. In particular, note that $r_1 \leq r_2$.

   (b) r_1 is not in the causal path, hence there must be a message m_k sent by the same process that sent s_1, such that $s_1 \rightarrow^+ s_k \lhd r_k \leq s_2 \lhd r_2$, where s_k and r_k are the send and receive events associated with m_k, respectively. Since messages m_1 and m_k are sent by the same process and $s_1 \rightarrow^+ s_k$, we should have $r_1 \blacktriangleleft r_k$, according to the $1-n$ semantics. In particular, note the we have $r_1 \blacktriangleleft r_k \leq r_2$.

   In both case (a) and (b), we conclude that $r_1 \lessdot r_2$. Recall that $\lessdot = (\rightarrow \cup \lhd \cup \blacktriangleleft_M)^*$.

Notice that, for either cases, a relation between two send events s_1 and s_2 (i.e. $s_1 \sqsubset s_2$ or $s_1 \leq s_2$) always implies a relation between the respective receive events r_1 and r_2, according to the $1-n$ semantics. It follows that ξ, which is a cycle for the $\preceq$ relation, always implies a cycle for the $\lessdot$ relation[6], as shown by the following example. Let M be a non-mailbox MSC, and suppose we have a cycle $s_1 \sqsubset s_2 \sqsubset s_3 \leq s_4 \sqsubset s_5 \leq s_1$. $s_1 \sqsubset s_2$ falls into case (1), so it implies $r_1 \rightarrow^+ r_2$. The same goes for $s_2 \sqsubset r_3$, which implies $r_2 \rightarrow^+ r_3$. $s_3 \leq s_4$ falls into case (2), and implies that $r_3 \lessdot r_4$. $s_4 \sqsubset s_5$ falls into case (1) and it implies $r_4 \rightarrow^+ r_5$. $s_5 \leq s_1$ falls into case (2) and implies that $r_5 \lessdot r_1$. Putting all these implications together, we have that $r_1 \rightarrow^+ r_2 \rightarrow^+ r_3 \lessdot r_4 \rightarrow^+ r_5 \lessdot r_1$, which is a cycle for $\lessdot$. Note that, given any cycle for $\preceq$, we are always able to apply this technique to obtain a cycle for $\lessdot$. □

Proposition 3.3 remains true even if we consider unmatched messages.

Proposition 3.4. Every $1-n$ MSC is a mailbox MSC.

Proof. Let M be an asynchronous MSC. The proof proceeds in the same way as the one of Proposition 3.3, but unmatched messages introduce some additional cases. Consider any two adiacent events s_1 and s_2 in a cycle ξ for $\preceq$, where ξ has been written using only $\sqsubset$ and $\leq$, and we never have two consecutive $\leq$. These are some additional cases:

3. $u_1 \sqsubset s_2$, where u_1 is the send event of an unmatched message. This case never happens because of how $\sqsubset$ is defined.

4. $u_1 \leq u_2$, where u_1 and u_2 are both send events of unmatched messages. Since both u_1 and u_2 are part of the cycle ξ, there must be an event s_3 such that $u_1 \leq u_2 \sqsubset s_3$. However, $u_2 \sqsubset s_3$ falls into case (3), which can never happen.

5. $u_1 \leq s_2$, where u_1 is the send event of an unmatched message and s_2 is the send event of a matched message. Since we have a causal path between u_1 and s_2, there has to be a message m_k, sent by the same process that sent m_1, such that $u_1 \rightarrow^+ s_k \lhd r_k \leq s_2 \lhd r_2$[7], where s_k and r_k are the send and receive events associated with m_k, respectively. Since messages m_1 and

[4] This is always possible, since $a \leq b \leq c$ is written as $a \leq c$.
[5] If that was not the case, $\leq$ would also be cyclic and M would not be an asynchronous MSC.
[6] If $\lessdot$ is cyclic, M is not a $1-n$ MSC.
[7] Note that we can have $m_k = m_2$

m_k are sent by the same process and m_1 is unmatched, we should have $s_k \blacktriangleleft u_1$, according to the $1-n$ semantics, but $u_1 \rightarrow^+ s_k$. It follows that if ξ contains $u_1 \leq s_2$, we can immediately conclude that M is not a $1-n$ MSC.

6. $s_1 \sqsubset u_2$, where s_1 is the send event of a matched message and u_2 is the send event of an unmatched message. Since both s_1 and u_2 are part of a cycle, there must be an event s_3 such that $s_1 \sqsubset u_2 \leq s_3$; we cannot have $u_2 \sqsubset s_3$, because of case (3). $u_2 \leq s_3$ falls into case (5), so we can conclude that M is not a $1-n$ MSC.

We showed that cases (3) and (4) can never happen, whereas cases (5) and (6) both imply that M is not $1-n$. If we combine them with the cases described in Proposition 3.3 we have the full proof. □

Definition 3.7 ($n-n$ MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *$n-n$ MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events s and s', such that $s \rightsquigarrow s'$, we have either:

- $s, s' \in \mathit{Matched}(M)$ and $r \rightsquigarrow r'$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$.
- $s' \in \mathit{Unm}(M)$.

Such a linearization will be referred to as a $n-n$ *linearization*. Intuitively, with an $n-n$ MSC we are always able to schedule events in such a way that messages are received in the same order as they were sent, and unmatched messages are sent only after all matched messages are sent. By definition, every $n-n$ MSC is a $1-n$ MSC.

Definition 3.8 ($n-n$ alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, let $\ltimes_M = (\rightarrow \cup \lhd \cup \sqsubset_M \cup \blacktriangleleft_M)^*$. We define an additional binary relation $\bowtie_M \subseteq \mathcal{E} \times \mathcal{E}$, such that for two events e_1 and e_2 we have $e_1 \bowtie_M e_2$ if one of the following holds:

1. $e_1 \ltimes_M e_2$
2. $\lambda(e_1) \in \mathit{Rec}(\_,\_,\_)$, $\lambda(e_2) \in \mathit{Rec}(\_,\_,\_)$, $s_1 \lhd e_1$ and $s_2 \lhd e_2$ for some $s_1, s_2 \in \mathcal{E}$, $s_1 \ltimes_M s_2$ and $e_1 \not\ltimes_M e_2$.
3. $\lambda(e_1) \in \mathit{Send}(\_,\_,\_)$, $\lambda(e_2) \in \mathit{Send}(\_,\_,\_)$, $e_1 \lhd r_1$ and $e_2 \lhd r_2$ for some $r_1, r_2 \in \mathcal{E}$, $r_1 \ltimes_M r_2$ and $e_1 \not\ltimes_M e_2$.
4. $e_1 \in \mathit{Matched}(M)$, $e_2 \in \mathit{Unm}(M)$, $e_1 \not\ltimes_M e_2$.

Note that $\preceq_M \subseteq \ltimes_M$, $<_M \subseteq \ltimes_M$, and $\ltimes_M \subseteq \bowtie_M$. We call $M \in \mathsf{MSC_{asy}}$ a *$n-n$ MSC* if $\bowtie_M$ is acyclic.

D: Initially I said "We call $M \in \mathsf{MSC_{asy}}$ a *$n-n$ MSC* if $\bowtie_M$ is a partial order", but I don't think it is true, since $\bowtie_M$ does not have to be transitive.

It is not trivial to see that Definition 3.8 is equivalent to Definition 3.7. To show that, we need some preliminary results and definitions.

Proposition 3.5. Let M be an MSC. Given two matched send events s_1 and s_2, and their respective receive events r_1 and r_2, $r_1 \bowtie_M r_2 \implies s_1 \bowtie_M s_2$.

Proof. Follows from the definition of $\bowtie_M$. We have $r_1 \bowtie_M r_2$ if either:

- $r_1 \ltimes_M r_2$. Two cases: either (i) $s_1 \ltimes_M s_2$, or (ii) $s_1 \not\ltimes_M s_2$. The first case clearly implies $s_1 \bowtie_M s_2$, for rule 1 in the definition of $\bowtie_M$. The second too, because of rule 3.
- $r_1 \not\ltimes_M r_2$, but $r_1 \bowtie_M r_2$. This is only possible if rule 2 in the definition of $\bowtie_M$ was used, which implies $s_1 \ltimes_M s_2$ and, for rule 1, $s_1 \bowtie_M s_2$.

□

Proposition 3.6. Let M be an MSC. If $\bowtie_M$ is cyclic, then M is not $n-n$.

Proof. Since a $n-n$ MSC is always both a mailbox and a $1-n$ MSC, it is clear that the cyclicity of $\ltimes_M$ implies that M is not $n-n$, because it means that we are not even able to find a linearization that is both mailbox and $1-n$. Moreover, since in a $n-n$ linearization the order in which messages are sent matches the order in which they are received, and unmatched send events can be executed only after matched send events, a $n-n$ MSC always has to satisfy the constraints imposed by the $\bowtie_M$ relation. If $\bowtie_M$, then for sure there is no $n-n$ linearization for M. □

Let the *Event Dependency Graph* (EDG) of a $n{-}n$ MSC M be a graph that has events as nodes and an edge between two events e_1 and e_2 if $e_1 \bowtie_M e_2$. We now present an algorithm that, given the EDG of an $n{-}n$ MSC M, computes a $n{-}n$ linearization of M. We then show that, if $\bowtie_M$ is acyclic (i.e. it is a partial order), this algorithm always terminates correctly. This, along with Proposition 3.6, effectively shows that Definition 3.7 and Definition 3.8 are equivalent.

Algorithm for finding a $n{-}n$ linearization The input of this algorithm is the EDG of an MSC M, and it outputs a valid $n{-}n$ linearization for M, if M is $n{-}n$. The algorithm works as follows:

1. If there is a matched send event s with in-degree 0 in the EDG, add s to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 2.

2. If there are no matched send events in the EDG and there is an unmatched send event s with in-degree 0 in the EDG, add s to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 3.

3. If there is a receive event r with in-degree 0 in the EDG, such that r is the receive event of the first message whose sent event was already added to the linearization, add r to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 4.

4. Throw an error and terminate.

5. If all the events of M were added to the linearization, return the linearization and terminate. Otherwise, go back to step 1.

We now need to show that (i) if this algorithm terminates correctly (i.e. step 4 is never executed), it returns a $n{-}n$ linearization, and (ii) if $\bowtie_M$ is acyclic, the algorithm always terminates correctly.

Proposition 3.7. *If the above algorithm returns a linearization for an MSC M, it is a $n{-}n$ linearization.*

Proof. Note that, because of how step 2 works, the order (in the linearization) in which matched messages are sent is the same as the order in which they are received. Moreover, according to step 3, an unmatched send events is added to the linearization only if all the matched send events were already added. □

Proposition 3.8. *Given an MSC M, the above algorithm always terminates correctly if $\bowtie_M$ is acyclic.*

Proof. We want to prove that, if $\bowtie_M$ is acyclic, step 4 of the algorithm is never executed, i.e. it terminates correctly. Note that the acyclicity of $\bowtie_M$ implies that the EDG of M is a DAG. Moreover, at every step of the algorithm we remove nodes and edges from the EDG, so it still remains a DAG. The proof goes by induction on the number of events added to the linearization. Base case: no event has been added to the linearization yet. Since the EDG is a DAG, there must be an event with in-degree 0. In particular, this has to be a send event (a receive event depends on its respective send event, so it cannot have in-degree 0). If it is a matched send event, step 1 is applied. If there are no matched send events, step 2 is applied on an unmatched send. We show that it is impossible to have an unmatched send event of in-degree 0 if there are still matched send events in the EDG, so either step 1 or 2 are applied in the base case. Let s be one of those matched send events and let u be an unmatched send. Because of rule 4 in the definition of $\bowtie_M$, we have that $s \bowtie_M u$, which implies that u cannot have in-degree 0 if s is still in the EDG.
Inductive step: we want to show that we are never going to execute step 4. In particular, Step 4 is executed when none of the first three steps can be applied. This happens when there are no matched send events with in-degree 0 and one of the following holds:

- *There are still matched send events in the EDG with in-degree > 0, there are no unmatched messages with in-degree 0, and there is no receive event r with in-degree 0 in the EDG, such that r is the receive event of the first message whose sent event was already added to the linearization.* Since the EDG is a DAG, there must be at least one receive event with in-degree 0. We want to show that, between these receive events with in-degree 0, there is also the receive event r of the first message whose send event was added to the linearization, so that we can apply step 3 and step 4 is not executed. Suppose, by contradiction, that r has in-degree > 0, so it depends on other events. For any maximal chain in the EDG that contains one of these events, consider the first event e, which clearly has in-degree 0. In particular, e cannot be a send event, because we would have applied step 1 or step 2. Hence, e can only be a receive event for a send event that was not the first added to the linearization (and whose respective receive still has not been added). However, this is also impossible, since $r_e \bowtie_M r$ implies $s_e \bowtie_M s$, and we could not have added s to the linearization before s_e. Because we got to a contradiction, the hypothesis that r has in-degree > 0 must be false, and we can indeed apply step 3.

- *There are still matched send events in the EDG with in-degree > 0, there is at least one unmatched message with in-degree 0, and there is no receive event r with in-degree 0 in the EDG, such that r is the receive event of the first message whose sent event was already added to the linearization.* We show that it is impossible to have an unmatched send event of in-degree 0 if there are still matched send events in the EDG. Let s be one of those matched send events and let u be an unmatched send. Because of rule 4 in the definition of $\bowtie_M$, we have that $s \bowtie_M u$, which implies that u cannot have in-degree 0 if s is still in the EDG.
- *There are no more matched send events in the EDG, there are no unmatched messages with in-degree 0, and there is no receive event r with in-degree 0 in the EDG, such that r is the receive event of the first message whose sent event was already added to the linearization.* Very similar to the first case. Since the EDG is a DAG, there must be at least one receive event with in-degree 0. We want to show that, between these receive events with in-degree 0, there is also the receive event r of the first message whose send event was added to the linearization, so that we can apply step 3 and step 4 is not executed. Suppose, by contradiction, that r has in-degree > 0, so it depends on other events. For any maximal chain in the EDG that contains one of these events, consider the first event e, which clearly has in-degree 0. In particular, e cannot be a send event, because by hypothesis there are no more send events with in-degree 0 in the EDG. Hence, e can only be a receive event for a send event that was not the first added to the linearization (and whose respective receive still has not been added). However, this is also impossible, since $r_e \bowtie_M r$ implies $s_e \bowtie_M s$, and we could not have added s to the linearization before s_e. Because we got to a contradiction, the hypothesis that r has in-degree > 0 must be false, and we can indeed apply step 3.

We showed that, if $\bowtie_M$ is acyclic, the algorithm always terminates correctly and computes a valid $n{-}n$ linearization. □

Definition 3.9 (RSC MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *RSC MSC* if it has no unmatched send events and there is a linearization $\rightsquigarrow$ where any matched send event is immediately followed by its respective receive event.

Following the characterization given in Charron-Bost, Mattern, and Tel 1996, Theorem 4.4, we also give an alternative but equivalent definition of RSC MSC.

Definition 3.10. Let M be an MSC. A crown of size k in M is a sequence $\langle (s_i, r_i),\ i \in \{1, \dots, k\} \rangle$ of pairs of corresponding send and receive events such that

$$s_1 <_M r_2, s_2 <_M r_3, \dots, s_{k-1} <_M r_k, s_k <_M r_1.$$

D: Specify somewhere that $<_M = (\rightarrow \cup \lhd)^+$.

Definition 3.11 (RSC alternative)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *RSC MSC* if and only if it does not contain any crown.

Proposition 3.3 shows that $\mathsf{MSC}_{1-n} \subset \mathsf{MSC}_{\mathsf{mb}}$. The classes of MSCs that we presented form a hierarchy, namely $\mathsf{MSC}_{1-n} \subset \mathsf{MSC}_{\mathsf{mb}} \subset \mathsf{MSC}_{\mathsf{co}} \subset \mathsf{MSC}_{\mathsf{p2p}} \subset \mathsf{MSC}_{\mathsf{asy}}$, as shown by Fig. 7.

3.4 Monadic Second-Order Logic

The set of MSO formulas over (asynchronous) MSCs (over $\mathbb{P}$ and $\mathbb{M}$) is given by the grammar $\varphi ::= \mathit{true} \mid x \rightarrow y \mid x \lhd y \mid \lambda(x) = a \mid x = y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \varphi \mid \neg\varphi$, where $a \in \Sigma$, x and y are first-order variables, interpreted as events of an MSC, and X is a second-order variable, interpreted as a set of events. We assume that we have an infinite supply of variables, and we use

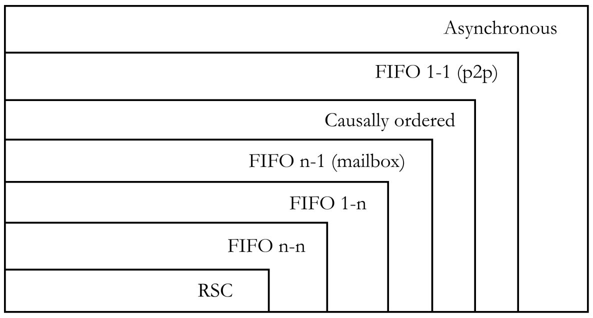


Figure 7: The hierarchy of MSC classes.

common abbreviations such as $\wedge$, $\Rightarrow$, $\forall$, etc. The satisfaction relation is defined in the standard way and self-explanatory. For example, the formula $\neg\exists x.(\bigvee_{a\in Send(\_,\_,\_)} \lambda(x) = a \;\wedge\; \neg matched(x))$ with $matched(x) = \exists y.x \lhd y$ says that there are no unmatched send events. It is not satisfied by MSC M_1 of Fig. 1, as message m_1 is not received, but by M_4 from Fig. ??.

Given a sentence φ, i.e., a formula without free variables, we let $L(\varphi)$ denote the set of MSCs that satisfy φ. Since we have defined the set of MSO formulas over asynchronous MSCs, the formula $\varphi_{\mathsf{asy}} = true$ clearly describes the set of asynchronous MSCs, i.e. $L(\varphi_{\mathsf{asy}}) = \mathsf{MSC}_{\mathsf{asy}}$. It is worth mentioning that the (reflexive) transitive closure of a binary relation defined by an MSO formula[8] with free variables x and y, such as $x \to y$, is MSO-definable so that the logic can freely use formulas of the form $x \to^+ y$, $x \to^* y$ or $x \leq y$ (where $\leq$ is interpreted as $\leq_M$ for the given MSC M).

Peer-to-peer MSCs The set of p2p MSCs is MSO-definable as

$$\varphi_{\mathsf{p2p}} = \neg\exists s.\exists s'.\left(\bigvee_{p\in\mathbb{P},q\in\mathbb{P}} \;\bigvee_{a,b\in Send(p,q,\_)} (\lambda(s) = a \;\wedge\; \lambda(s') = b) \;\wedge\; s \to^+ s' \;\wedge\; (\psi_1 \vee \psi_2)\right)$$

where ψ_1 and ψ_2 are

$$\psi_1 = \exists r.\exists r'.\begin{pmatrix} s \lhd r & \wedge \\ s' \lhd r' & \wedge \\ r' \to^+ r & \end{pmatrix} \qquad \psi_2 = (\neg matched(s) \wedge matched(s'))$$

$$matched(x) = \exists y.x \lhd y$$

The property φ_{p2p} says that there cannot be two matched send events s and s', with the same sender and receiver, such that either (i) $s \to^+ s'$ and their receipts happen in the reverse order, or (ii) s is unmatched and s' is matched. In other words, it ensures that channels operate in FIFO mode, where an unmatched messages blocks the receipt of all the subsequent messages on that channel. The set $\mathsf{MSC}_{\mathsf{p2p}}$ is therefore MSO-definable as $\mathsf{MSC}_{\mathsf{p2p}} = L(\varphi_{\mathsf{p2p}})$.

Causally ordered MSCs Given an MSC M, it is causally ordered if and only if it satisfies the MSO formula

$$\varphi_{\mathsf{co}} = \neg\exists s.\exists s'.\left(\bigvee_{q\in\mathbb{P}} \;\bigvee_{a,b\in Send(\_,q,\_)} (\lambda(s) = a \;\wedge\; \lambda(s') = b) \;\wedge\; s \leq_M s' \;\wedge\; (\psi_1 \vee \psi_2)\right)$$

where ψ_1 and ψ_2 are the same formulas used for p2p.

The property φ_{co} says that there cannot be two send events s and s', with the same recipient, such that $s \leq_M s'$ and either (i) their corresponding receive events r and r' happen in the opposite order, i.e. $r' \to^+ r$, or (ii) s is unmatched and s' is matched. The set $\mathsf{MSC}_{\mathsf{co}}$ of causally ordered MSCs is therefore MSO-definable as $\mathsf{MSC}_{\mathsf{co}} = L(\varphi_{\mathsf{co}})$.

[8]See Section 3.6.4 for details.

Mailbox MSCs Given an MSC M, it is a mailbox MSC if and only if it satisfies the MSO formula

$$\varphi_{\mathsf{mb}} = \varphi_{\mathsf{p2p}} \ \wedge\ \neg\exists x.\exists y.(\neg(x = y) \wedge x \preceq_M y \wedge y \preceq_M x)$$

The set $\mathsf{MSC}_{\mathsf{mb}}$ of mailbox MSCs is therefore MSO-definable as $\mathsf{MSC}_{\mathsf{mb}} = L(\varphi_{\mathsf{mb}})$.

D: This does not match my definition of mailbox MSC, should I also give the alternative definition (the one in the concur paper) or rewrite everything according to my definition? Give both definitions and prove that they are equivalent

$1{-}n$ **MSCs** Following Definition 3.6, an MSC M is a $1{-}n$ MSC if and only if it satisfies the MSO formula

$$\varphi_{1-n} = \neg\exists x.\exists y.(\neg(x = y) \wedge x \lessdot_M y \wedge y \lessdot_M x)$$

Recall that $\lessdot_M$ is the union of the MSO-definable relations $\rightarrow$, $\lhd$, and $\blacktriangleleft_M$. In particular, we can define $x \blacktriangleleft_M y$ as

$$x \blacktriangleleft_M y = \begin{array}{l} \Big(\bigvee_{\substack{p\in\mathbb{P} \\ a,b\in \mathit{Send}(p,\_,\_)}} (\lambda(x) = a \ \wedge\ \lambda(y) = b) \ \wedge\ \mathit{matched}(x) \ \wedge\ \neg\mathit{matched}(y)\Big) \ \vee \\ \Big(\bigvee_{\substack{p\in\mathbb{P} \\ a,b\in \mathit{Rec}(p,\_,\_)}} (\lambda(x) = a \ \wedge\ \lambda(y) = b) \ \wedge\ \exists x'.\exists y'.(x' \lhd x \ \wedge\ y' \lhd y \ \wedge\ x' \rightarrow^+ y')\Big) \end{array}$$

The MSO formula for $x \blacktriangleleft_M y$ closely follows Definition 3.6. The set MSC_{1-n} of $1{-}n$ MSCs is therefore MSO-definable as $\mathsf{MSC}_{1-n} = L(\varphi_{1-n})$.

$n{-}n$ **MSCs** Following Definition 3.8, an MSC M is a $n{-}n$ MSC if and only if it satisfies the MSO formula

$$\varphi_{n-n} = \neg\exists x.\exists y.(\neg(x = y) \wedge x \bowtie_M y \wedge y \bowtie_M x)$$

In particular, we can define $x \bowtie_M y$ as

$$x \bowtie_M y = \begin{array}{l} \Big(\bigvee_{a,b\in \mathit{Send}(\_,\_,\_)} (\lambda(x) = a \ \wedge\ \lambda(y) = b) \ \wedge\ \mathit{matched}(x) \ \wedge\ \neg\mathit{matched}(y)\Big) \ \vee \\ (x \ltimes_M y) \quad \vee \quad \psi_3 \quad \vee \quad \psi_4 \end{array}$$

where ψ_3 and ψ_4 are defined as

$$\psi_3 = \begin{array}{l} \bigvee_{a,b\in \mathit{Rec}(\_,\_,\_)} (\lambda(x) = a \ \wedge\ \lambda(y) = b) \ \wedge \\ \exists x'.\exists y'.(x' \lhd x \ \wedge\ y' \lhd y) \ \wedge\ (x' \ltimes_M y') \ \wedge\ \neg(x \ltimes_M y) \end{array}$$

$$\psi_4 = \begin{array}{l} \bigvee_{a,b\in \mathit{Send}(\_,\_,\_)} (\lambda(x) = a \ \wedge\ \lambda(y) = b) \ \wedge \\ \exists x'.\exists y'.(x \lhd x' \ \wedge\ y \lhd y') \ \wedge\ (x' \ltimes_M y') \ \wedge\ \neg(x \ltimes_M y) \end{array}$$

The MSO formula for $x \ltimes_M y$ closely follows Definition 3.8. The set MSC_{n-n} of $n{-}n$ MSCs is therefore MSO-definable as $\mathsf{MSC}_{n-n} = L(\varphi_{n-n})$.

RSC **MSCs** Following Definition 3.11, an MSC M is a RSC MSC if and only if it satisfies the MSO formula

$$\Phi_{\mathsf{RSC}} = \neg\exists s_1.\exists s_2.s_1 \propto s_2 \ \wedge\ s_2 \propto^* s_1$$

where $\propto$ is defined as

$$s_1 \propto s_2 = \bigvee_{e\in \mathit{Send}(\_,\_,\_)} (\lambda(s_1) = e) \ \wedge\ s_1 \neq s_2 \ \wedge\ \exists r_2.(s_1 < r_2 \ \wedge\ s_2 \lhd r_2)$$

3.5 Existentially bounded MSCs

D: Should we say for all sends, also unmatched? Shouldn't it be $<$ instead of $\leq$?

Definition 3.12. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$ and $k \in \mathbb{N}$. A linearization $\rightsquigarrow$ of M is called *k-bounded* if, for all $e \in \mathit{Matched}(M)$, with $\lambda(e) = \mathit{send}(p, q, m)$, we have

$$\#_{\mathit{Send}(p,q,\_)}(\rightsquigarrow, e) - \#_{\mathit{Rec}(p,q,\_)}(\rightsquigarrow, e) \leq k.$$

Recall that $\#_{\mathit{Send}(p,q,\_)}(\rightsquigarrow, e)$ denotes the number of send events from p to q that occured before e, according to $\rightsquigarrow$. Intuitively, a linearization is k-bounded if, at any moment in time, there are no more than k messages in any channel.

Definition 3.13 (Existentially bounded MSC)**.** Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{asy}}$ and $k \in \mathbb{N}$. We call M *existentially k-bounded* if it has a k-bounded linearization.

Let $\mathsf{MSC}_{\exists k\text{--}b}$ be the set of existentially k-bounded MSCs, for a given $k \in \mathbb{N}$.

Definition 3.14. An MSC M is *p2p existentially k-bounded* (p2p-$\exists k$-bounded) if it is a p2p MSC and it is also existentially k-bounded.

Definition 3.15. An MSC M is *causally orderered existentially k-bounded* (co-$\exists k$-bounded) if it is a causally ordered MSC and it is also existentially k-bounded.

When moving on to mailbox MSCs, the definition of mailbox existentially k-bounded MSC should require that there exists a k-bounded linearization that is also a mailbox linearization, not just any linearization. Recall that an MSC is a mailbox MSC if it has at least one mailbox linearization, which represents a sequence of events that can be executed by a mailbox system. Following this intuition, we want one of these mailbox linearizations to be k-bounded, because *non*-mailbox linearizations cannot be executed by a mailbox system.

Definition 3.16. An MSC M is *mailbox existentially k-bounded* (mb-$\exists k$-bounded) if it is a mailbox MSC and it has a k-bounded mailbox linearization.

D: This paragraph depends on how we choose to formally define the mailbox communication model... we could go for a (i) single incoming channel, or (ii) just an enforcing of the delivery of messages by the transport layer.

It should be noted that, for a k-bounded mailbox linearization, it is not necessarily true that at any time we have at most k messages in each channel. Recall that in the mailbox communication architecture every process has a single incoming channel, but the Definition 3.12 of k-bounded linearization considers the number of pending messages between each pair (p, q) of processes. Let n be the number of processes. We can say that, for a k-bounded mailbox linearization, we have at most $k(n-1)$ messages in each channel at any moment (because each process can have at most k pending messages coming from any of the other $n-1$ processes).

D: An example would be nice.

Definition 3.17. An MSC M is $1{-}n$ *existentially k-bounded* (1n-$\exists k$-bounded) if it is a $1{-}n$ MSC and it has a k-bounded $1{-}n$ linearization.

3.5.1 MSO-definability

In this section, we will investigate the MSO-definability of all the variants of $\exists k$-bounded MSCs that we introduced, starting from the asynchronous $\exists k$-bounded MSCs.

Following the approach taken in Lohrey and Muscholl 2002, we introduce a binary relation $\longmapsto_k$ ($\rightsquigarrow_b$ in their work) associated with a given bound k and an MSC M. Let $k \geq 1$ and M be a fixed MSC: we have $r \longmapsto_k s$ if, for some $i \geq 1$ and some channel (p,q)[9]:

1. r is the i-th receive event (executed by q).
2. s is the $(i+k)$-th send event (executed by p).

Note that, for any two events s and r such that $r \longmapsto_k s$, every linearization of M in which r is executed after s cannot k-bounded. Intuitively, we can read $r \longmapsto_k s$ as "r has to be executed before s in a k-bounded linearization". A linearization $\rightsquigarrow$ that respects $\longmapsto_k$ (i.e. $\longmapsto_k \subseteq \rightsquigarrow$) is k-bounded.

D: Should I prove it?

D: An example would be nice.

In Lohrey and Muscholl 2002 it was shown that an MSC is $\exists k$-bounded if and only if the relation $\leq_M \cup \longmapsto_k$ is acyclic[10]. Since $\leq_M$ and acyclicity are both MSO-definable, it suffices to find an MSO formula that defines the $\longmapsto_k$ relation to claim the MSO-definability of $\exists k$-bounded MSCs. Unfortunately, this relation is not MSO-definable for asynchronous MSCs, because MSO logic cannot be used to "count" for an arbitrary i. For this reason, we introduce another similar MSO-definable binary relation $\hookrightarrow_k$, and we show that with this new relation we still have that an MSC M is $\exists k$-bounded MSC iff $\leq_M \cup \hookrightarrow_k$ is acyclic and another condition holds. Let $k \geq 1$ and M be a fixed MSC: we have $r \hookrightarrow_k s$ if, for some $i \geq 1$ and some channel (p,q):

[9] Recall that (p, q) is a channel where messages are sent by p and received by q.
[10] A binary relation is acyclic if its transitive closure is antisymmetric.

- There are $k+1$ send events $(s_1, \ldots, s_k, s)$, where at least one is matched, such that $s_1 \rightarrow^+ \ldots \rightarrow^+ s_k \rightarrow^+ s$.
- r is the first receive event among the receive events for the matched sends among $s_1, \ldots, s_k, s$.

Proposition 3.9. An MSC M is $\exists k$-bounded if and only if $\leq_M \cup \hookrightarrow_k$ is acyclic and, for each channel (p,q), there are at most k unmatched send events.

Proof. ($\Rightarrow$) Suppose M is $\exists k$-bounded, which by definition means there is at least one $\exists k$-bounded linearization $\rightsquigarrow$. Firstly, notice that every MSC that has more than k unmatched send events in any channel cannot be an $\exists k$-bounded MSC. We already know that $\leq_M \subseteq \rightsquigarrow$, and we will show show that also $\hookrightarrow_k \subseteq \rightsquigarrow$. This implies that $\leq_M \cup \hookrightarrow_k$ is acyclic, otherwise we would not be able to find a linearization $\rightsquigarrow$ that respects both $\leq_M$ and $\hookrightarrow_k$. Suppose, by contradiction, that $\hookrightarrow_k \not\subseteq \rightsquigarrow$, i.e. there are two events r and s such that $r \hookrightarrow_k s$ and $s \rightsquigarrow r$. By definition of $\hookrightarrow_k$, there are k send events in a channel (p,q) that are executed before s, and whose respective receive events happens after r. If s is executed before r in the linearization, there will be $k+1$ messages in channel (i.e. $\rightsquigarrow$ is not a $\exists k$-bounded linearization). We reached a contradiction, hence $\hookrightarrow_k \subseteq \rightsquigarrow$ and $\leq_M \cup \hookrightarrow_k$ is acyclic.
($\Leftarrow$) Suppose $\leq_M \cup \hookrightarrow_k$ is acyclic and, for each channel (p,q), there are at most k unmatched send events. If $\leq_M \cup \hookrightarrow_k$ is acyclic, we are able to find at least one linearization $\rightsquigarrow$ for the partial order $(\leq_M \cup \hookrightarrow_k)^*$. We want to show that this linearization is $\exists k$-bounded. By contradiction, suppose $\rightsquigarrow$ is not $\exists k$-bounded, i.e. we are able to find $k+1$ send events $s_1 \rightarrow^+ \ldots \rightarrow^+ s_k \rightarrow^+ s$ on a channel (p,q), such that s is executed before any of the respective receive events takes place. There are two possible scenarios:

- Suppose all the $k+1$ send events are unmatched. This is impossible, since we supposed that there are at most k unmatched send events for any channel.
- Suppose there is at least one matched send event between the $k+1$ sends. Let the first matched send event be s_i and let r be the receive event that is executed first among the receive events for these $k+1$ sends. By hypothesis, $s \rightsquigarrow r$. However, according to the definition of $\hookrightarrow_k$, we must have $r \hookrightarrow_k s$. We reached a contradiction, since we cannot have that s happens before r in a linearization for the partial order $(\leq_M \cup \hookrightarrow_k)^*$, if $r \hookrightarrow_k s$.

□

According to Proposition 3.9, we can write the MSO formula the defines $\exists k$-bounded MSCs as

$$\Psi_{\exists k} = acyclic(\leq_M \cup \hookrightarrow_k) \;\wedge\; \neg\left(\exists s_1 \ldots s_{k+1}.s_1 \rightarrow^+ \ldots \rightarrow^+ s_{k+1} \;\wedge\; allSends\_p\_q \;\wedge\; allUnm\right)$$

$$allSends\_p\_q = \bigvee_{p \in \mathbb{P}, q \in \mathbb{P}} \; \bigwedge_{s \in s_1, \ldots, s_{k+1}} \; \bigvee_{a \in Send(p,q,\_)} (\lambda(s) = a)$$

$$allUnm = \bigwedge_{s \in s_1, \ldots, s_{k+1}} (\neg matched(s))$$

where $acyclic(\leq_M \cup \hookrightarrow_k)$ is an MSO formula that checks the acyclicity of the $\leq_M \cup \hookrightarrow_k$ relation, and the $\hookrightarrow_k$ relation can be defined as

$$r \hookrightarrow_k s = \exists s_1 \ldots s_{k+1}. \left(\begin{array}{l} s_1 \rightarrow^+ \ldots \rightarrow^+ s_{k+1} \;\wedge\; allSends\_p\_q \;\wedge \\ \exists r.(\bigvee_{s \in s_1, \ldots, s_{k+1}} s \lhd r) \;\wedge\; \bigwedge_{e \in s_1, \ldots, s_{k+1}} (\exists f. e \lhd f \implies r \rightarrow^* f) \end{array} \right)$$

In particular, we can define $r \longmapsto_k s$ as

D: This might need some tweaking depending on if we want to count or not unmatched messages

$$r \longmapsto_k s = \exists s_1 . \ldots \exists s_k . \left(allDistSend\_p\_q(k) \;\wedge\; s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_k \rightarrow s \;\wedge\; s_1 \lhd r\right)$$

It follows that, given $k \in \mathbb{N}$, the set of existentially k-bounded MSCs is MSO-definable. Causally ordered and p2p existentially k-bounded MSCs are clearly MSO-definable by definition, since we already showed that p2p MSCs, causally ordered MSCs, and existentially k-bounded MSCs are all MSO-definable. The set of mailbox existentially k-bounded MSCs was already shown to be MSO-definable in Bollig, Giusto, et al. 2021, but they use a slightly different definition of mailbox existentially k-bounded MSC. With their definition, a $\exists k$-mb-bounded MSC has at most k messages in any incoming channel at any moment, whereas according to our definition the threshold is $k(n-1)$, were n is the number of processes. In particular, they introduce a binary relation $\xrightarrow{\text{rev}}_k \supseteq \longmapsto_k$ and they show that an MSC M is $\exists k$-mb-bounded iff $\preceq_M \cup \xrightarrow{\text{rev}}_k$ is acyclic. With our definition, an MSC is $\exists k$-mb-bounded iff $\preceq_M \cup \longmapsto_k$ is acyclic. This can easily be shown by simply replacing $\xrightarrow{\text{rev}}_k$ with $\longmapsto_k$ in the proof given in Bollig, Giusto, et al. 2021. We already talked about the MSO-defininability of $\preceq_M$, $\longmapsto_k$, and acyclicity, so the set of mailbox existentially k-bounded MSCs is also MSO-definable.

D: Should I rewrite the proof?

3.5.2 Special treewidth

In Bollig and Gastin 2019, Lemma 5.37 it was shown that the special treewidth of existentially k-bounded MSCs is bounded by $k\,|\mathbb{P}|^2$, for $k \geq 1$. Actually, STW-boundness was shown for the more general CBM class (Concurrent Behaviours with Matching), but the result is still valid since $\mathsf{MSC}_{\mathsf{asy}} \subset \mathsf{CBM}$. The special treewidth of both p2p-$\exists k$-bounded and co-$\exists k$-bounded MSCs is also bounded, since every p2p or causally ordered existentially k-bounded MSC is existentially k-bounded by definition. Similarly, mb-$\exists k$-bounded MSCs also have a bounded special treewidth, since it is trivial to see that every existentially mailbox k-bounded MSC is existentially k-bounded.

D: Should I rewrite the proof more clearly?

3.6 Universally bounded MSCs

Definition 3.18 (Universally bounded MSC)**.** Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{asy}}$ and $k \in \mathbb{N}$. We call M *universally k-bounded* if all of its linearizations are k-bounded.

Let $\mathsf{MSC}_{\forall k\text{-}b}$ be the set of universally k-bounded MSCs.

Definition 3.19. An MSC M is *p2p universally k-bounded* (p2p-$\forall k$-bounded) if it is a p2p MSC and it is also universally k-bounded.

Let $\mathsf{MSC}_{p2p\text{-}\forall k\text{--}b}$ be the set of p2p universally k-bounded MSCs.

Definition 3.20. An MSC M is *causally orderered universally k-bounded* (co-$\forall k$-bounded) if it is a causally ordered MSC and it is also universally k-bounded.

Let $\mathsf{MSC}_{\mathsf{co}\text{-}\forall k\text{-}b}$ be the set of causally ordered universally k-bounded MSCs.

Definition 3.21. An MSC M is *mailbox universally k-bounded* (mb-$\forall k$-bounded) if it is a mailbox MSC and all of its mailbox linearizations are k-bounded.

Let $\mathsf{MSC}_{\mathsf{mb}\text{-}\forall k\text{-}b}$ be the set of mailbox universally k-bounded MSCs.

Definition 3.22. An MSC M is *$1{-}n$ universally k-bounded* (1n-$\forall k$-bounded) if it is a $1{-}n$ MSC and all of its $1{-}n$ linearizations are k-bounded.

Let $\mathsf{MSC}_{\mathsf{mb}\text{-}\forall k\text{-}b}$ be the set of mailbox universally k-bounded MSCs.

3.6.1 Hierarchy

D: Investigate hierarchy of universally k-bounded MSCs.

In this section we will investigate the relations between the various classes of universally k-bounded MSCs that we introduced. From their definition, it is quite straightforward to see that $\mathsf{MSC}_{\mathsf{co}\text{-}\forall k\text{-}b} \subseteq \mathsf{MSC}_{p2p\text{-}\forall k\text{--}b} \subseteq \mathsf{MSC}_{\forall k\text{-}b}$. The set of mailbox universally k-bounded MSCs, however, does not fit in this hierarchy. Recall that an MSC is mb-$\forall k$-bounded if all of its mailbox linearizations are k-bounded, but the definition does not say anything about non-mailbox linearizations. It can be the case that an MSC has a bound k for its mailbox linearizations, but a higher bound k' for non-mailbox linearizations. Fig. 8 shows an MSC M which is mb-$\forall 1$-bounded, but $\forall 2$-bounded. According to the mailbox semantics, a mailbox linearization of M has to respect the order $!m_1 \sqsubset_M !m_3 \sqsubset_M !m_4$. Note that all mailbox linearizations are 1-bounded, but we are able to find a non-mailbox linearization that is 2-bounded, such as $!m_1 \rightsquigarrow !m_4 \rightsquigarrow ?m_1 \rightsquigarrow !m_2 \rightsquigarrow ?m_2 \rightsquigarrow !m_3 \rightsquigarrow !m_4 \rightsquigarrow ?m_3 \rightsquigarrow ?m_4$.

For a given $k \in \mathbb{N}$, Fig 9 gives a visual representation of how the diffent variants of universally k-bounded MSCs are related.

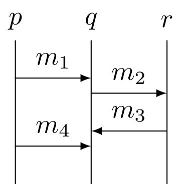


Figure 8: Example of MSC which is mailbox universally 1-bounded, but not universally 1-bounded (it is universally 2-bounded).

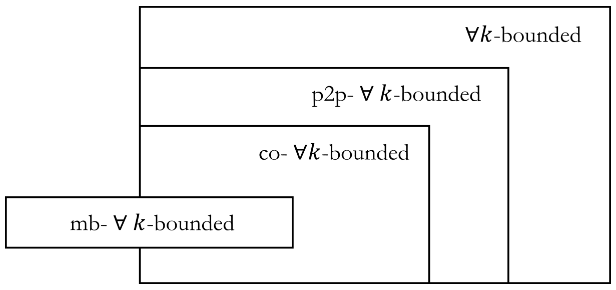


Figure 9: The relation between different variants of universally k-bounded MSCs, given a $k \in \mathbb{N}$.

3.6.2 MSO-definability

In this section, we will investigate the MSO-definability of all the variants of universally k-bounded MSCs that we introduced.

In Lohrey and Muscholl 2002, it is shown that an MSC M is universally k-bounded if and only if $\longmapsto_k \subseteq \leq_M$. In other words, $r \longmapsto_k s \Rightarrow r \leq_M s$ for any two events r and s. This is equivalent to saying that every linearization $\rightsquigarrow$ of M respects the $\longmapsto_k$ relation, since $\longmapsto_k \subseteq \leq_M \subseteq \rightsquigarrow$. We already showed that $\longmapsto_k$ is MSO-definable. The MSO formula that defines universally k-bounded MSCs can be written as

$$\Phi_{\forall k\text{-}b} = \neg\exists r.\exists s.(r \longmapsto_k s \wedge \neg(r \leq_M s))$$

Causally ordered and p2p universally k-bounded MSCs are clearly MSO-definable by definition, since we already showed that p2p MSCs, causally ordered MSCs, and universally k-bounded MSCs are all MSO-definable. To show MSO-definability of mailbox universally k-bounded MSCs we first prove the following property.

Proposition 3.10. An MSC M is mailbox universally k-bounded if and only if $\longmapsto_k \subseteq \preceq_M$.

Proof. Consider an MSC M and a $k \in \mathbb{N}$.
($\Leftarrow$) Suppose $\longmapsto_k \subseteq \preceq_M$. For every mailbox linearization $\overset{\text{mb}}{\rightsquigarrow}$ of M we have that $\preceq_M \subseteq \overset{\text{mb}}{\rightsquigarrow}$. This implies $\longmapsto_k \subseteq \overset{\text{mb}}{\rightsquigarrow}$, that is to say every mailbox linearization is k-bounded.
($\Rightarrow$) Suppose M is a mailbox universally k-bounded MSC. By definition, every mailbox linearization $\overset{\text{mb}}{\rightsquigarrow}$ of M is k-bounded, i.e. $\longmapsto_k \subseteq \overset{\text{mb}}{\rightsquigarrow}$, and we have $\preceq_M \subseteq \overset{\text{mb}}{\rightsquigarrow}$, according to the definition of mailbox linearization. Moreover, we also know that $\preceq_M \cup \longmapsto_k$ is acyclic, since M is existentially k-bounded[11]. Suppose now, by contradiction, that $\longmapsto_k \not\subseteq \preceq_M$. Thus, there must be at least two events r and s such that $r \longmapsto_k s$ and $r \not\preceq_M s$; we also have $s \not\preceq_M r$ because of the acyclicity of $\preceq_M \cup \longmapsto_k$ (we cannot have the cycle $r \longmapsto_k s \preceq_M r$). Consider a mailbox linearization $\overset{\text{mb}}{\rightsquigarrow}$ of M, such that $s \overset{\text{mb}}{\rightsquigarrow} r$. Note that such a mailbox linearization always exists, since r and s are incomparable w.r.t. the partial order $\preceq_M$ (i.e. $s \not\preceq_M r$)[12]. This mailbox linearization does not respect $\longmapsto_k$ (because we have $s \overset{\text{mb}}{\rightsquigarrow} r$ and $r \longmapsto_k s$), so it is not k-bounded. This is a contradiction, since we assumed that M was a mailbox universally k-bounded MSC. It has to be that $\longmapsto_k \subseteq \preceq_M$. □

Using Proposition 3.10, we can now easily write the MSO formula that defines mailbox universally k-bounded MSCs as

$$\Phi_{\forall k\text{-}mb\text{-}b} = \neg\exists r.\exists s.(r \longmapsto_k s \wedge \neg(r \preceq_M s))$$

[11] Every mailbox universally k-bounded MSC is also a mailbox existentially k-bounded MSC by definition.
[12] If two elements a and b of a set are incomparable w.r.t. a partial order $\leq$, it is always possible to find a total order of the elements (that respects $\leq$) where a comes before b, or viceversa.

3.6.3 Special treewidth

All the variants of universally k-bounded MSCs that we presented have a bounded special treewidth. This directly follows from the STW-boundness of the existential counterparts, since every universally k-bounded MSC is existentially k-bounded by definition.

3.6.4 MSO additional content

Transitive Closure

D: Wrong definition, change with Lozes'

Given a set Σ and binary relation $\rightarrow \subseteq \Sigma \times \Sigma$ that is irreflexive, antisymmetric, and transitive (i.e., $\rightarrow$ is a strict partial order), we can express its reflexive transitive closure $\rightarrow^*$ in MSO as

$$x \rightarrow^* y = \exists X.(x \in X \wedge y \in X \wedge \forall z.(z \in X \implies z = y \ \vee \ \exists k.(k \in X \wedge z \rightarrow k)))$$

Acyclicity Given a binary relation $\rightarrow$, the acyclicity of $\rightarrow$ can be expressed with an MSO formula. Recall that, given a binary relation $\rightarrow$, it is acyclic if and only if its transitive closure $\rightarrow^+$ is antisymmetric. The MSO formula of acyclicity directly follows from this definition:

$$\Phi_{acyclic} = \neg\exists x.\exists y.(x \rightarrow^+ y \ \wedge \ y \rightarrow^+ x).$$

1 Introduction

- Interleaving based semantics VS partial order/graph based semantics
- Synchronous and asynchronous communication
- The problem of synchronizability

2 Preliminaries/Basics

- Communicating systems (communicating finite-state automata with bag channels)
- MSCs and conflict graph
- Monadic Second-Order logic on MSCs
- (Language of a system as a set of MSCs)
- (Model checking and synchronizability)

3 Asynchronous communication models overview

- Overview of asynchronous variants
- High-level description of each variant along with references to implementations (if existing)
- (Definitions based on linearization, intuitive)
- (Language of a system with a given communication model as a set of MSCs)
- Hint of hierarchy result

4 Asynchronous communication models operational semantics

- TODO...

5 Asynchronous communication models as classes of MSCs, MSO-definability

- Definition of MSC class for each communication model (alternative definitions)
- MSO-definability of each class

6 Equivalence of the two definitions

- TODO...

7 Hierarchy of asynchronous classes of MSCs

...

8 An application: special treewidth and decidability of the synchronizability problem

- The synchronizability problem
- Special treewidth and how the results regarding the hierarchy are useful for detecting STW-boundness of certain classes
- MSO-decidability and STW-boundess tables

9 Conclusion

10 (2) Preliminaries

D: This is just a copy-paste of some sections of the concur paper, they need to be changed accordingly. I would also like to add in the beginning some intuitive ideas on (i) what are MSCs (2) why we use them to describe distributed systems computation (3) what we mean by absolute time and executions/linearizations.

10.1 Message Sequence Charts

Assume a finite set of processes $\mathbb{P}$ and a finite set of messages $\mathbb{M}$. The set of (p2p) channels is $\mathbb{C} = \{(p,q) \in \mathbb{P} \times \mathbb{P} \mid p \neq q\}$. A send action is of the form $send(p,q,m)$ where $(p,q) \in \mathbb{C}$ and $m \in \mathbb{M}$. It is executed by p and sends message m to q. The corresponding receive action, executed by q, is $rec(p,q,m)$. For $(p,q) \in \mathbb{C}$, let $Send(p,q,\_) = \{send(p,q,m) \mid m \in \mathbb{M}\}$ and $Rec(p,q,\_) = \{rec(p,q,m) \mid m \in \mathbb{M}\}$. For $p \in \mathbb{P}$, we set $Send(p,\_,\_) = \{send(p,q,m) \mid q \in \mathbb{P}\setminus\{p\}$ and $m \in \mathbb{M}\}$, etc. Moreover, $\Sigma_p = Send(p,\_,\_) \cup Rec(\_,p,\_)$ will denote the set of all actions that are executed by p. Finally, $\Sigma = \bigcup_{p \in \mathbb{P}} \Sigma_p$ is the set of all the actions.

Peer-to-peer MSCs. A *p2p MSC* (or simply *MSC*) over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ where $\mathcal{E}$ is a finite (possibly empty) set of *events* and $\lambda : \mathcal{E} \rightarrow \Sigma$ is a labeling function. For $p \in \mathbb{P}$, let $\mathcal{E}_p = \{e \in \mathcal{E} \mid \lambda(e) \in \Sigma_p\}$ be the set of events that are executed by p. We require that $\rightarrow$ (the *process relation*) is the disjoint union $\bigcup_{p \in \mathbb{P}} \rightarrow_p$ of relations $\rightarrow_p \subseteq \mathcal{E}_p \times \mathcal{E}_p$ such that $\rightarrow_p$ is the direct successor relation of a total order on $\mathcal{E}_p$. For an event $e \in \mathcal{E}$, a set of actions $A \subseteq \Sigma$, and a relation $R \subseteq \mathcal{E} \times \mathcal{E}$, let $\#_A(R,e) = |\{f \in \mathcal{E} \mid (f,e) \in R$ and $\lambda(f) \in A\}|$. We require that $\lhd \subseteq \mathcal{E} \times \mathcal{E}$ (the *message relation*) satisfies the following:

(1) for every pair $(e,f) \in \lhd$, there is a send action $send(p,q,m) \in \Sigma$ such that $\lambda(e) = send(p,q,m)$, $\lambda(f) = rec(p,q,m)$, and $\#_{Send(p,q,\_)}(\rightarrow^+, e) = \#_{Rec(p,q,\_)}(\rightarrow^+, f)$,

(2) for all $f \in \mathcal{E}$ such that $\lambda(f)$ is a receive action, there is $e \in \mathcal{E}$ such that $e \lhd f$.

Finally, letting $\leq_M = (\rightarrow \cup \lhd)^*$, we require that $\leq_M$ is a partial order. For convenience, we will simply write $\leq$ when M is clear from the context.

Condition (1) above ensures that every (p2p) channel (p,q) behaves in a FIFO manner. By Condition (2), every receive event has a matching send event. Note that, however, there may be unmatched send events in an MSC. We let $SendEv(M) = \{e \in \mathcal{E} \mid \lambda(e)$ is a send action$\}$, $RecEv(M) = \{e \in \mathcal{E} \mid \lambda(e)$ is a receive action$\}$, $Matched(M) = \{e \in \mathcal{E} \mid$ there is $f \in \mathcal{E}$ such that $e \lhd f\}$, and $Unm(M) = \{e \in \mathcal{E} \mid \lambda(e)$ is a send action and there is no $f \in \mathcal{E}$ such that $e \lhd f\}$. We do not distinguish isomorphic MSCs and let $\mathsf{MSC}_{\mathsf{p2p}}$ be the set of all MSCs over the given sets $\mathbb{P}$ and $\mathbb{M}$.

Example 10.1. For a set of processes $\mathbb{P} = \{p,q,r\}$ and a set of messages $\mathbb{M} = \{m_1, m_2, m_3, m_4\}$, $M_1 = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is an MSC where, for example, $e_2 \lhd e_2'$ and $e_3' \rightarrow e_4$. The dashed arrow means that the send event e_1 does not have a matching receive, so $e_1 \in Unm(M_1)$. Moreover, $e_2 \leq_{M_1} e_4$, but $e_1 \not\leq_{M_1} e_4$. We can find a total order $\rightsquigarrow \supseteq \leq_{M_1}$ such that $e_1 \rightsquigarrow e_2 \rightsquigarrow e_2' \rightsquigarrow e_3 \rightsquigarrow e_3' \rightsquigarrow e_4 \rightsquigarrow e_4'$. We call $\rightsquigarrow$ a linearization, which is formally defined below.

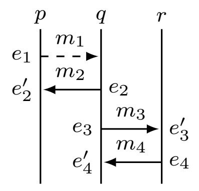


Figure 10: MSC M_1

Mailbox MSCs. For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, we define an additional binary relation that represents a constraint under the mailbox semantics, where each process has only one incoming channel. Let $\sqsubset_M \subseteq \mathcal{E} \times \mathcal{E}$ be defined by: $e_1 \sqsubset_M e_2$ if there is $q \in \mathbb{P}$ such that $\lambda(e_1) \in Send(\_,q,\_)$, $\lambda(e_2) \in Send(\_,q,\_)$, and one of the following holds:

- $e_1 \in Matched(M)$ and $e_2 \in Unm(M)$, or
- $e_1 \lhd f_1$ and $e_2 \lhd f_2$ for some $f_1, f_2 \in \mathcal{E}_q$ such that $f_1 \rightarrow^+ f_2$.

We let $\preceq_M = (\rightarrow \cup \lhd \cup \sqsubset_M)^*$. Note that $\leq_M \subseteq \preceq_M$. We call $M \in \mathsf{MSC}_{\mathsf{p2p}}$ a *mailbox MSC* if $\preceq_M$ is a partial order. Intuitively, this means that events can be scheduled in a way that corresponds to the mailbox semantics, i.e., with one incoming channel per process. Following the terminology in Bouajjani et al. 2018, we also say that a mailbox MSC satisfies *causal delivery.* The set of mailbox MSCs $M \in \mathsf{MSC}_{\mathsf{p2p}}$ is denoted by $\mathsf{MSC}_{\mathsf{mb}}$.

Example 10.2. MSC M_1 is a mailbox MSC. Indeed, even though the order $\rightsquigarrow$ defined in Example 1.1 does not respect all mailbox constraints, particularly the fact that $e_4 \sqsubset_{M_1} e_1$, there is a total order $\rightsquigarrow \supseteq \preceq_{M_1}$ such that $e_2 \rightsquigarrow e_3 \rightsquigarrow e_3' \rightsquigarrow e_4 \rightsquigarrow e_1 \rightsquigarrow e_2' \rightsquigarrow e_4'$. We call $\rightsquigarrow$ a mailbox linearization, which is formally defined below.

Linearizations, Prefixes, and Concatenation. Consider $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$. A *p2p linearization* (or simply *linearization*) of M is a (reflexive) total order $\rightsquigarrow \subseteq \mathcal{E} \times \mathcal{E}$ such that $\leq_M \subseteq \rightsquigarrow$. Similarly, a *mailbox linearization* of M is a total order $\rightsquigarrow \subseteq \mathcal{E} \times \mathcal{E}$ such that $\preceq_M \subseteq \rightsquigarrow$. That is, every mailbox linearization is a p2p linearization, but the converse is not necessarily true (Example 1.2). Note that an MSC is a mailbox MSC iff it has at least one mailbox linearization.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$ and consider $E \subseteq \mathcal{E}$ such that E is $\leq_M$*-downward-closed*, i.e, for all $(e, f) \in \leq_M$ such that $f \in E$, we also have $e \in E$. Then, the MSC $(E, \rightarrow \cap (E \times E), \lhd \cap (E \times E), \lambda')$, where λ' is the restriction of $\mathcal{E}$ to E, is called a *prefix* of M. In particular, the empty MSC is a prefix of M. We denote the set of prefixes of M by $\mathit{Pref}(M)$. This is extended to sets $L \subseteq \mathsf{MSC}$ as expected, letting $\mathit{Pref}(L) = \bigcup_{M \in L} \mathit{Pref}(M)$.

Lemma 10.1. *Every prefix of a mailbox MSC is a mailbox MSC.*

Proof. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{mb}}$ and $M_0 = (\mathcal{E}_0, \rightarrow_0, \lhd_0, \lambda_0)$ be a prefix of M, i.e., $\mathcal{E}_0 \subseteq \mathcal{E}$. By contradiction, suppose that M_0 is not a mailbox MSC. Then, there are distinct $e, f \in \mathcal{E}_0$ such that $e \preceq_{M_0} f \preceq_{M_0} e$ with $\preceq_{M_0} = (\rightarrow_0 \cup \lhd_0 \cup \sqsubset_{M_0})^*$. As $\mathcal{E}_0 \subseteq \mathcal{E}$, we have that $\rightarrow_0 \subseteq \rightarrow$, $\lhd_0 \subseteq \lhd$, and $\sqsubset_{M_0} \subseteq \sqsubset_M$. Finally, $\preceq_{M_0} \subseteq \preceq_M$ and M is not a mailbox MSC, which is a contradiction. ☐

Let $M_1 = (\mathcal{E}_1, \rightarrow_1, \lhd_1, \lambda_1)$ and $M_2 = (\mathcal{E}_2, \rightarrow_2, \lhd_2, \lambda_2)$ be two MSCs. Their *concatenation* $M_1 \cdot M_2 = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is defined if, for all $(p, q) \in \mathbb{C}$, $e_1 \in \mathit{Unm}(M_1)$, and $e_2 \in \mathcal{E}_2$ such that $\lambda(e_1) \in \mathit{Send}(p, q, \_)$ and $\lambda(e_2) \in \mathit{Send}(p, q, \_)$, we have $e_2 \in \mathit{Unm}(M_2)$. As expected, $\mathcal{E}$ is the disjoint union of $\mathcal{E}_1$ and $\mathcal{E}_2$, $\lhd = \lhd_1 \cup \lhd_2$, λ is the "union" of λ_1 and λ_2, and $\rightarrow = \rightarrow_1 \cup \rightarrow_2 \cup R$. Here, R contains, for all $p \in \mathbb{P}$ such that $(\mathcal{E}_1)_p$ and $(\mathcal{E}_2)_p$ are non-empty, the pair (e_1, e_2) where e_1 is the maximal p-event in M_1 and e_2 is the minimal p-event in M_2. Note that $M_1 \cdot M_2$ is indeed an MSC and that concatenation is associative.

10.2 Communicating Systems

We now recall the definition of communicating systems (aka communicating finite-state machines or message-passing automata), which consist of finite-state machines A_p (one for every process $p \in \mathbb{P}$) that can communicate through the FIFO channels from $\mathbb{C}$.

Definition 10.1. A *communicating system* over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $\mathcal{S} = (A_p)_{p \in \mathbb{P}}$. For each $p \in \mathbb{P}$, $A_p = (\mathit{Loc}_p, \delta_p, \ell_p^0)$ is a finite transition system where Loc_p is a finite set of local (control) states, $\delta_p \subseteq \mathit{Loc}_p \times \Sigma_p \times \mathit{Loc}_p$ is the transition relation, and $\ell_p^0 \in \mathit{Loc}_p$ is the initial state.

Given $p \in \mathbb{P}$ and a transition $t = (\ell, a, \ell') \in \delta_p$, we let $\mathit{source}(t) = \ell$, $\mathit{target}(t) = \ell'$, $\mathit{action}(t) = a$, and $\mathit{msg}(t) = m$ if $a \in \mathit{Send}(\_, \_, m) \cup \mathit{Rec}(\_, \_, m)$.

There are in general two ways to define the semantics of a communicating system. Most often it is defined as a global infinite transition system that keeps track of the various local control states and all (unbounded) channel contents. As, in this paper, our arguments are based on a graph view of MSCs, we will define the language of $\mathcal{S}$ directly as a set of MSCs. These two semantic views are essentially equivalent, but they have different advantages depending on the context. We refer to Aiswarya and Gastin 2014 for a thorough discussion.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC. A *run* of $\mathcal{S}$ on M is a mapping $\rho : \mathcal{E} \rightarrow \bigcup_{p \in \mathbb{P}} \delta_p$ that assigns to every event e the transition $\rho(e)$ that is executed at e. Thus, we require that (i) for all $e \in \mathcal{E}$, we have $\mathit{action}(\rho(e)) = \lambda(e)$, (ii) for all $(e, f) \in \rightarrow$, $\mathit{target}(\rho(e)) = \mathit{source}(\rho(f))$, (iii) for all $(e, f) \in \lhd$, $\mathit{msg}(\rho(e)) = \mathit{msg}(\rho(f))$, and (iv) for all $p \in \mathbb{P}$ and $e \in \mathcal{E}_p$ such that there is no $f \in \mathcal{E}$ with $f \rightarrow e$, we have $\mathit{source}(\rho(e)) = \ell_p^0$.

Letting run $\mathcal{S}$ directly on MSCs is actually very convenient. This allows us to associate with $\mathcal{S}$ its p2p language and mailbox language in one go. The *p2p language* of $\mathcal{S}$ is $L_{\mathsf{p2p}}(\mathcal{S}) = \{M \in \mathsf{MSC}_{\mathsf{p2p}} \mid$ there is a run of $\mathcal{S}$ on $M\}$. The *mailbox language* of $\mathcal{S}$ is $L_{\mathsf{mb}}(\mathcal{S}) = \{M \in \mathsf{MSC}_{\mathsf{mb}} \mid$ there is a run of $\mathcal{S}$ on $M\}$.

Note that, following Bouajjani et al. 2018; Di Giusto, Laversa, and Lozes 2020, we do not consider final states or final configurations, as our purpose is to reason about all possible traces that can be *generated* by $\mathcal{S}$. The next lemma is obvious for the p2p semantics and follows from Lemma 1.1 for the mailbox semantics.

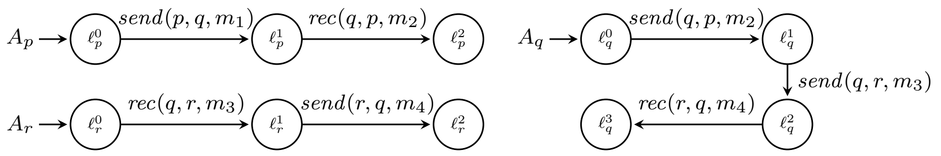


Figure 11: System $\mathcal{S}_1$

Lemma 10.2. *For all* com $\in \{\mathsf{p2p}, \mathsf{mb}\}$*,* $L_{\mathsf{com}}(\mathcal{S})$ *is prefix-closed:* $\mathit{Pref}(L_{\mathsf{com}}(\mathcal{S})) \subseteq L_{\mathsf{com}}(\mathcal{S})$*.*

Example 10.3. Fig. 2 depicts $\mathcal{S}_1 = (A_p, A_q, A_r)$ such that MSC M_1 in Fig. 1 belongs to $L_{\mathsf{p2p}}(\mathcal{S}_1)$ and to $L_{\mathsf{mb}}(\mathcal{S}_1)$. There is a unique run ρ of $\mathcal{S}_1$ on M_1. We can see that $(e_3', e_4) \in \rightarrow$ and $\mathit{target}(\rho(e_3')) = \mathit{source}(\rho(e_4)) = \ell_r^1$, $(e_2, e_2') \in \lhd_{M_1}$, and $\mathit{msg}(\rho(e_2)) = \mathit{msg}(\rho(e_2')) = m_2$.

10.3 Conflict Graph

We now recall the notion of a conflict graph associated to an MSC defined in Bouajjani et al. 2018. This graph is used to depict the causal dependencies between message exchanges. Intuitively, we have a dependency whenever two messages have a process in common. For instance, an $\xrightarrow{SS}$ dependency between message exchanges v and v' expresses the fact that v' has been sent after v, by the same process. This notion is of interest because it was seen in Bouajjani et al. 2018 that the notion of synchronizability in MSCs (which is studied in this paper) can be graphically characterized by the nature of the associated conflict graph. It is defined in terms of linearizations in Di Giusto, Laversa, and Lozes 2020, but we equivalently express it directly in terms of MSCs.

For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ and $e \in \mathcal{E}$, we define the type $\tau(e) \in \{S, R\}$ of e by $\tau(e) = S$ if $e \in \mathit{SendEv}(M)$ and $\tau(e) = R$ if $e \in \mathit{RecEv}(M)$. Moreover, for $e \in \mathit{Unm}(M)$, we let $\mu(e) = e$, and for $(e, e') \in \lhd$, we let $\mu(e) = \mu(e') = (e, e')$.

Definition 10.2 (Conflict graph)**.** The *conflict graph* $\mathsf{CG}(M)$ of an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is the labeled graph $(\mathit{Nodes}, \mathit{Edges})$, with $\mathit{Edges} \subseteq \mathit{Nodes} \times \{S, R\}^2 \times \mathit{Nodes}$, defined by $\mathit{Nodes} = \lhd \cup \mathit{Unm}(M)$ and $\mathit{Edges} = \{(\mu(e), \tau(e)\tau(f), \mu(f)) \mid (e, f) \in \rightarrow^+\}$. In particular, a node of $\mathsf{CG}(M)$ is either a single unmatched send event or a message pair $(e, e') \in \lhd$.

10.4 Logic and Special Tree-Width

Monadic Second-Order Logic. The set of MSO formulas over MSCs (over $\mathbb{P}$ and $\mathbb{M}$) is given by the grammar $\varphi ::= x \rightarrow y \mid x \lhd y \mid \lambda(x) = a \mid x = y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \varphi \mid \neg\varphi$, where $a \in \Sigma$, x and y are first-order variables, interpreted as events of an MSC, and X is a second-order variable, interpreted as a set of events. We assume that we have an infinite supply of variables, and we use common abbreviations such as $\wedge$, $\forall$, etc. The satisfaction relation is defined in the standard way and self-explanatory. For example, the formula $\neg\exists x.(\bigvee_{a \in \mathit{Send}(\_,\_,\_)} \lambda(x) = a \ \wedge \ \neg \mathit{matched}(x))$ with $\mathit{matched}(x) = \exists y. x \lhd y$ says that there are no unmatched send events. It is not satisfied by MSC M_1 of Fig. 1, as message m_1 is not received, but by M_4 from Fig. **??**.

Given a sentence φ, i.e., a formula without free variables, we let $L(\varphi)$ denote the set of (p2p) MSCs that satisfy φ. It is worth mentioning that the (reflexive) transitive closure of a binary relation defined by an MSO formula with free variables x and y, such as $x \rightarrow y$, is MSO-definable so that the logic can freely use formulas of the form $x \rightarrow^+ y$ or $x \leq y$ (where $\leq$ is interpreted as $\leq_M$ for the given MSC M). Therefore, the definition of a mailbox MSC can be readily translated into the formula $\varphi_{\mathsf{mb}} = \neg\exists x.\exists y.(\neg(x = y) \wedge x \preceq y \wedge y \preceq x)$ so that we have $L(\varphi_{\mathsf{mb}}) = \mathsf{MSC}_{\mathsf{mb}}$. Here, $x \preceq y$ is obtained as the MSO-definable reflexive transitive closure of the union of the MSO-definable relations $\rightarrow$, $\lhd$, and $\sqsubset$. In particular, we may define $x \sqsubset y$ by :

$$x \sqsubset y = \bigvee_{\substack{q \in \mathbb{P} \\ a,b \in \mathit{Send}(\_,q,\_)}} \lambda(x) = a \ \wedge \ \lambda(y) = b \wedge \left(\begin{array}{ll} & \mathit{matched}(x) \wedge \neg \mathit{matched}(y) \\ \vee & \exists x'.\exists y'.(x \lhd x' \ \wedge \ y \lhd y' \ \wedge \ x' \rightarrow^+ y') \end{array} \right)$$

Special Tree-Width. *Special tree-width* Courcelle 2010, is a graph measure that indicates how close a graph is to a tree (we may also use classical *tree-width* instead). This or similar measures are commonly employed in verification. For instance, tree-width and split-width have been used in Madhusudan and Parlato 2011 and, respectively, Cyriac, Gastin, and Kumar 2012; Aiswarya, Gastin, and Kumar 2014 to reason about graph behaviors generated by pushdown and queue systems. There are several ways to define the special tree-width of an MSC. We adopt the following game-based definition from Bollig and Gastin 2019.

Adam and Eve play a two-player turn based "decomposition game" whose positions are MSCs with some pebbles placed on some events. More precisely, Eve's positions are *marked MSC fragments* (M, U), where $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is an *MSC fragment* (an MSC with possibly some edges from $\lhd$ or $\rightarrow$ removed) and $U \subseteq \mathcal{E}$ is the subset of marked events. Adam's positions are pairs of marked MSC fragments. A move by Eve consists in the following steps:

1. marking some events of the MSC resulting in (M, U') with $U \subseteq U' \subseteq \mathcal{E}$,
2. removing (process and/or message) edges whose endpoints are marked,
3. dividing (M, U) in (M_1, U_1) and (M_2, U_2) such that M is the disjoint (unconnected) union of M_1 and M_2 and marked nodes are inherited.

When it is Adam's turn, he simply chooses one of the two marked MSC fragments. The initial position is $(M, \emptyset)$ where M is the (complete) MSC at hand. A terminal position is any position belonging to Eve such that all events are marked. For $k \in \mathbb{N}$, we say that the game is k-winning for Eve if she has a (positional) strategy that allows her, starting in the initial position and independently of Adam's moves, to reach a terminal position such that, in every single position visited along the play, there are at most $k+1$ marked events.

Fact 10.3 (Bollig and Gastin 2019)**.** *The special tree-width of an MSC is the least k such that the associated game is k-winning for Eve.*

The set of MSCs whose special tree-width is at most k is denoted by $\mathsf{MSC}^{k\text{-}\mathsf{stw}}$.

11 (3) Asynchronous communication models overview

In synchronous communication, send and receive elements are essentially viewed as a single entity, i.e. a receive event is always executed immediately after the corresponding send event. The whole idea behind (fully) asynchronous communication is to decouple send and receive events, so that a receive event can happen indefinitely after its corresponding send event. However, by introducing some additional contraints on asynchronous communication, we can obtain new communication models that sit somewhere between synchronous and fully asynchronous communication. In this section, we will present 7 different asynchronous communication models, which were already introduced in Chevrou, Hurault, and Quéinnec 2016 (even though they do not consider unmatched messages). A major difference is that in Chevrou, Hurault, and Quéinnec 2016 these communication models are addressed from a linearizations standpoint, whereas we are interested in MSCs. Recall that a single MSC can have several possible linearizations. The work in Chevrou, Hurault, and Quéinnec 2016 describes the properties that a single linearization must satisfy in order to be realizable by a system that uses a given communication model. On the other hand, we are interested in understanding if a given MSC describes a computation that can be realized by a system that uses some communication model CM. In other words, given a MSC we want to know if it has at least one linearization that respects the constraints imposed by CM. If that is the case, the MSC represents a behaviour that can be exhibited by a system that uses CM as a communication model. These are two fundamentally dissimilar problems; at the end of this section we provide an example to clarify the difference. In our work, we are going to formally characterize the classes of MSCs which represent valid computations for all of these 7 asynchronous communication model. We also show how these classes form a well-defined hierarchy, which does not correspond entirely to that found in Chevrou, Hurault, and Quéinnec 2016.

We model a distributed system as a set of concurrent Finite-State Machines (FSMs) that exchange messages asynchronously through channels. Each FSM models a single machine/process of the system and transitions are labeled with "send" and "receive" operations, which specify the sender and the receiver of a message. In our work we consider only point-to-point communication, that is, messages that have exactly one sender and one receiver. The role of the communication model is to impose an order on the reception of messages, according to its specification. For instance, the delivery of a message could be delayed or even prevented by a communication model CM, so as to ensure that messages are received in an order that is valid for CM. The 7 communication models that we address all impose different constraints on the order in which messages can be received.

11.1 Fully asynchronous

In the fully asychronous communication model (or simply asynchronous) messages can be received at any time once they have been sent. In asynchronous communication, send events are non-blocking, i.e. the sender of a message does not have to wait for it to be delivered to the recipient, in order to resume normal operations. Fig. 12 shows a computation that can be executed by a system that uses asynchronous communication; indeed, even if m_1 is sent before m_2, q does not have to receive m_1 first. For convenience, we will refer to a system that uses asynchronous communication simply as an asynchronous system. In a similar way, an MSC such that in Fig. 12 will be referred to as an asynchronous MSC, since it represents a valid computation for an asynchronous system. The same jargon will also be used for all the other communication models. We will call $\mathsf{MSC}_{\mathsf{asy}}$ the set of all asynchronous MSCs.

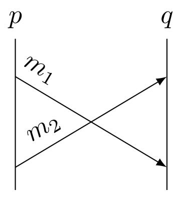


Figure 12: An asynchronous MSC.

11.2 FIFO 1−1 (p2p)

In the FIFO 1−1 communication model, any two messages sent from one process p to another process q are always received in the same order as they were sent. In the most classical definition of Communicating Finite-State Machine, processes are connected pairwise by FIFO channels, i.e. messages are delivered by channels in the order in which they were sent[13]. This definition of Communicating Finite-State Machines clearly uses the FIFO 1−1 communication model, since we have FIFO channels between processes that take care of delivering messages in the correct order. The FIFO 1−1 communication model is referred to as p2p in Bollig, Giusto, et al. 2021, and we will also use this terminology. The MSC shown in Fig. 12 is clearly not a FIFO 1−1 MSC; both m_1 and m_2 are sent by and to the same process, so the receive order must match the send order, which is not the case here. Fig. 13 shows an example of p2p MSC; the only two messages sent by and to the same process are m_3 and m_4, which are received in the same order as they have been sent. An example of linearization is !1 !2 ?2 !3 !4 ?3 ?1 ?4. Let $\mathsf{MSC}_{\mathsf{p2p}}$ be the set of p2p MSCs.

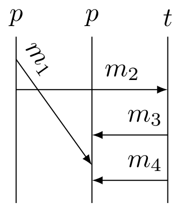


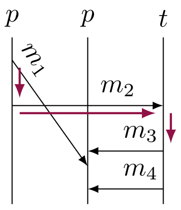
Figure 13: A p2p MSC.

11.3 Causally ordered

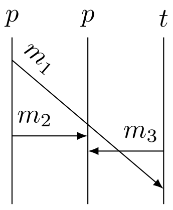
In the causally ordered communication model, messages are delivered to a process according to the causality of their emissions. In other words, if there are two messages m_1 and m_2 with the same recipient, such that m_1 is causally sent before m_2 (i.e. there exists a causal path from the first send to the second one), then m_1 must be received before m_2. Fig. 14a, which is identical to Fig.13, shows an example of non-causally ordered MSC; there is a causal path between the sending of m_1 and m_3 (highlighted with red arrows), hence m_1 should be received before m_3 according to the causally ordered communication model, which is not the case. On the other hand, the second MSC shown in Fig. 14 is causally ordered; note that the only two messages with the same recipient

[13]Please note that our definition of Communicating Finite-State Machine is different from the classical one. FIFO channels are replaced by bag channels.

are m_2 and m_3, but there is no causal path between their respective send events (i.e. the causally ordered communication model does not introduce any new constraint that must be satisfied). Let $\mathsf{MSC}_{\mathsf{co}}$ be the set of causally ordered MSCs.



(a) Asynchronous, p2p, not causally ordered, not mailbox, not FIFO $1{-}n$, not FIFO $n{-}n$, not RSC.



(b) Asynchronous, p2p, causally ordered, mailbox, FIFO $1{-}n$, FIFO $n{-}n$, not RSC.

Figure 14: Two examples of MSCs.

11.4 FIFO $n{-}1$ (mailbox)

In the FIFO $n{-}1$ communicating model, any two messages sent to a process q must be received in the same order as they have been sent (according to absolute time). Note that these two messages might be sent by different processes and the two send events might be concurrent (i.e. there is no causal path between them). In other words, if a process q receives m_1 before m_2, then m_1 must have been sent before m_2 in absolute time. Essentially, the FIFO $n{-}1$ coordinates all the senders of a single receiver. A high-level implementation of the mailbox communication model could consist in a single incoming FIFO channel for each process, which is shared by all the other processes. A send event would consist in pushing the message on the shared FIFO channel. The MSC shown in Fig. 14a is not a mailbox MSC; m_1 and m_3 have the same recipient, but they are not received in the same order as they are sent. The MSC in Fig. 14b is mailbox; indeed, we are able to find a linearization that respects the mailbox constraints, such as !1 !2 !3 ?2 ?3 ?1 (note that m_2 is both sent and received before m_3). Such a linearization will be referred to as a *mailbox linearization*. At this stage, the difference between the class of causally ordered MSCs and the class of mailbox MSCs might not be clear. We will clarify later how all these classes of MSCs are related to each other. Let $\mathsf{MSC}_{\mathsf{mb}}$ be the set of mailbox MSCs.

11.5 FIFO $1{-}n$

The FIFO $1{-}n$ communicating model is the dual of FIFO $n{-}1$, it coordinates a sender with all the receivers. Any two messages sent by a process p must be received in the same order (in absolute time) as they have been sent. Note that these two messages might be received by different processes and the two receive events might be concurrent (i.e. there is no causal path between them). In other words, if a process p sends m_1 before m_2, then m_1 must be received before m_2 in absolute time. A high-level implementation of the FIFO $1{-}n$ communication model could consist in a single outgoing FIFO channel for each process P_i, which is shared by all the other processes. A send event would consist in pushing the message on the outgoing FIFO channel. The MSC shown in Fig. 14a is not a FIFO $1{-}n$ MSC; m_1 and m_2 are sent in this order by the same process, but they are received in the opposite order (note that there is a causal path between the reception of m_2 and the reception of m_1, so ?2 happens before ?1 in every linearization of this MSC). Fig. 14b shows an example of FIFO $1{-}n$ MSC; m_1 is sent before m_2 by the same process, and we are able to find a linearization where m_1 is received before m_2, such as !1 !2 !3 ?1 ?2 ?3. Such a linearization will be referred to as a $1{-}n$ *linearization*. Let MSC_{1-n} be the set of $1{-}n$ MSCs.

11.6 FIFO $n{-}n$

In the FIFO $n{-}n$ communicating model, messages are globally ordered and delivered according to the their emission order. Any two messages must be received in the same order as they have been sent, in absolute time. Note that these two messages might be received by different processes and the two receive events might be concurrent (i.e. there is no causal path between them). In other words, if a message m_1 is sent before m_2 in absolute time, then m_1 must be received before m_2 in absolute time. The FIFO $n{-}n$ coordinates all the senders with all the receivers. A high-level implementation of the FIFO $1{-}n$ communication model could consist in a single FIFO channel shared by all processes. The MSC shown in Fig. 14a is clearly not a FIFO $n{-}n$ MSC; if we consider messages m_1 and m_2 we have that, in every linearization, !1 happens before !2 and ?2 happens before ?1. This violates the constraints imposed by the FIFO $n{-}n$ communication model. The MSC in Fig. 14b is $n{-}n$ because we are able to find a linearization that satisfies the $n{-}n$ communication model, e.g. !1 !2 !3 ?1 ?2 ?3. Such a linearization will be referred to as an $n{-}n$ *linearization*. Let MSC_{n-n} be the set of $n{-}n$ MSCs.

11.7 Realizable with Synchronous Communication (RSC)

The RSC communication model imposes that a send event is always immediately followed by its corresponding receive event. In an execution of a system that uses the RSC communication model it is impossible to find an event that is executed between a send and its corresponding receive. An asynchronous distributed system that implements the RSC communication model effectively behaves as a synchronous system. None of the MSCs shown in Fig. 14 is a RSC MSC; indeed, for both of them it is impossible to find a linearization where each send event is immediately followed by the corresponding receive event. The MSC shown in Fig. 15 is an example of RSC MSC; we can easily find a linearization that respects the constraints of the RSC communication model, such as !1 ?1 !2 ?2 !3 ?3. Such a linearization will be referred to as an RSC *linearization*. Let $\mathsf{MSC}_{\mathsf{RSC}}$ be the set of RSC MSCs.

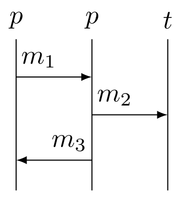


Figure 15: A RSC MSC.

11.8 Hierarchy of MSC classes

We find of particular interest to study the relation between the classes of MSCs for all of these communication models. For instance, it is trivial to see that each p2p MSC is also asynchronous (i.e. $\mathsf{MSC}_{\mathsf{p2p}} \subseteq \mathsf{MSC}_{\mathsf{asy}}$); p2p communication is equivalent to asynchronous communication plus the FIFO constraints, therefore every MSC that respects the p2p communication model is also valid for the asynchronous communication model. It is also quite straightforward to notice that every causally ordered MSC is a p2p MSC. In Section **??** we prove that all these classes of MSCs form a very neat hierarchy, which is graphically shown in Fig. 16.

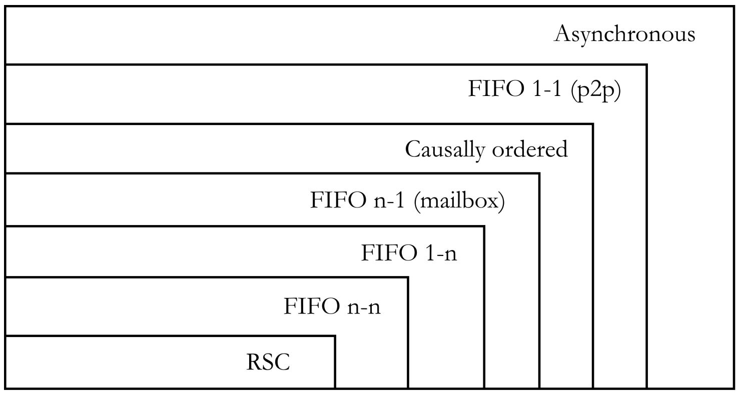


Figure 16: The hierarchy of MSC classes.

References

Aiswarya, C. and Paul Gastin (2014). "Reasoning About Distributed Systems: WYSIWYG (Invited Talk)". In: *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India.* Ed. by Venkatesh

Raman and S. P. Suresh. Vol. 29. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 11–30. DOI: 10.4230/LIPIcs.FSTTCS.2014.11. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2014.11.

Aiswarya, C., Paul Gastin, and K. Narayan Kumar (2014). "Verifying Communicating Multi-pushdown Systems via Split-Width". In: *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014*. Vol. 8837. Lecture Notes in Computer Science. Springer, pp. 1–17.

Bollig, Benedikt and Paul Gastin (2019). "Non-Sequential Theory of Distributed Systems". In: *CoRR* abs/1904.06942. arXiv: 1904.06942. URL: http://arxiv.org/abs/1904.06942.

Bollig, Benedikt, Cinzia Di Giusto, et al. (2021). "A Unifying Framework for Deciding Synchronizability". In: *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*. Ed. by Serge Haddad and Daniele Varacca. Vol. 203. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 14:1–14:18. DOI: 10.4230/LIPIcs.CONCUR.2021.14. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2021.14.

Bouajjani, Ahmed et al. (2018). "On the Completeness of Verifying Message Passing Programs Under Bounded Asynchrony". In: *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*. Ed. by Hana Chockler and Georg Weissenbacher. Vol. 10982. Lecture Notes in Computer Science. Springer, pp. 372–391. DOI: 10.1007/978-3-319-96142-2_23. URL: https://doi.org/10.1007/978-3-319-96142-2%5C_23.

Brand, Daniel and Pitro Zafiropulo (1983). "On Communicating Finite-State Machines". In: *J. ACM* 30.2, pp. 323–342. DOI: 10.1145/322374.322380. URL: http://doi.acm.org/10.1145/322374.322380.

Charron-Bost, Bernadette, Friedemann Mattern, and Gerard Tel (1996). "Synchronous, Asynchronous, and Causally Ordered Communication". In: *Distributed Comput.* 9.4, pp. 173–191. DOI: 10.1007/s004460050018. URL: https://doi.org/10.1007/s004460050018.

Chevrou, Florent, Aurélie Hurault, and Philippe Quéinnec (2016). "On the diversity of asynchronous communication". In: *Formal Aspects Comput.* 28.5, pp. 847–879. DOI: 10.1007/s00165-016-0379-x. URL: https://doi.org/10.1007/s00165-016-0379-x.

Courcelle, Bruno (2010). "Special tree-width and the verification of monadic second-order graph properties". In: *FSTTCS*. Vol. 8. LIPIcs, pp. 13–29.

Cyriac, Aiswarya, Paul Gastin, and K. Narayan Kumar (2012). "MSO Decidability of Multi-Pushdown Systems via Split-Width". In: *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*. Ed. by Maciej Koutny and Irek Ulidowski. Vol. 7454. Lecture Notes in Computer Science. Springer, pp. 547–561. DOI: 10.1007/978-3-642-32940-1_38. URL: https://doi.org/10.1007/978-3-642-32940-1%5C_38.

Di Giusto, Cinzia, Laetitia Laversa, and Étienne Lozes (2020). "On the k-synchronizability of Systems". In: *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Proceedings*. Ed. by Jean Goubault-Larrecq and Barbara König. Vol. 12077. Lecture Notes in Computer Science. Springer, pp. 157–176. DOI: 10.1007/978-3-030-45231-5_9. URL: https://doi.org/10.1007/978-3-030-45231-5%5C_9.

Fidge, Colin J (1988). "Timestamps in Message-Passing Systems That Preserve the Partial Ordering,"" in: *Proc. 11th Austral. Comput. Sci. Conf. (ACSC '88)*, pp. 56–66.

Genest, Blaise, Dietrich Kuske, and Anca Muscholl (2007). "On Communicating Automata with Bounded Channels". In: *Fundamenta Informaticae* 80.1-3, pp. 147–167.

Genest, Blaise, Anca Muscholl, and Dietrich Kuske (2004). "A Kleene Theorem for a Class of Communicating Automata with Effective Algorithms". In: *Developments in Language Theory, 8th International Conference, DLT 2004, Auckland, New Zealand, December 13-17, 2004, Proceedings*. Ed. by Cristian Calude, Elena Calude, and Michael J. Dinneen. Vol. 3340. Lecture Notes in Computer Science. Springer, pp. 30–48. DOI: 10.1007/978-3-540-30550-7_4. URL: https://doi.org/10.1007/978-3-540-30550-7%5C_4.

Giusto, Cinzia Di, Laetitia Laversa, and Étienne Lozes (2021). "Guessing the buffer bound for k-synchronizability". In: *Implementation and Application of Automata - 25th International Conference, CIAA 2021, Proceedings*. Lecture Notes in Computer Science. To appear. Springer.

Henriksen, Jesper G. et al. (2005). "A theory of regular MSC languages". In: *Information and Computation* 202.1, pp. 1–38. ISSN: 0890-5401.

Kuske, Dietrich and Anca Muscholl (2014). *Communicating automata*.

Lohrey, Markus and Anca Muscholl (2002). "Bounded MSC Communication". In: *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS*

2002 Grenoble, France, April 8-12, 2002, Proceedings. Ed. by Mogens Nielsen and Uffe Engberg. Vol. 2303. Lecture Notes in Computer Science. Springer, pp. 295–309. DOI: 10.1007/3-540-45931-6_21. URL: https://doi.org/10.1007/3-540-45931-6%5C_21.

Lohrey, Markus and Anca Muscholl (2004). "Bounded MSC communication". In: *Inf. Comput.* 189.2, pp. 160–181. DOI: 10.1016/j.ic.2003.10.002. URL: https://doi.org/10.1016/j.ic.2003.10.002.

Madhusudan, P. and Gennaro Parlato (2011). "The tree width of auxiliary storage". In: *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*. Ed. by Thomas Ball and Mooly Sagiv. ACM, pp. 283–294.

Mattern, Friedemann (1989). "Virtual time and global states of distributed systems." In: *Proc. Workshop on Parallel and Distributed Algorithms,* North-Holland / Elsevier, pp. 215–226.

Schiper, André, Jorge Eggli, and Alain Sandoz (1989). "A New Algorithm to Implement Causal Ordering". In: *Distributed Algorithms, 3rd International Workshop, Nice, France, September 26-28, 1989, Proceedings*. Ed. by Jean-Claude Bermond and Michel Raynal. Vol. 392. Lecture Notes in Computer Science. Springer, pp. 219–232. DOI: 10.1007/3-540-51687-5_45. URL: https://doi.org/10.1007/3-540-51687-5%5C_45.