MASTER INTERNSHIP REPORT

A non-sequential hierarchy of message-passing models

Prepared by
Davide Ferre'

Supervised by
Cinzia Di Giusto
Etienne Lozes

Master 2 Informatique
at Université Côte d'Azur
Academic Year 2021-2022

Abstract

There is a wide variety of message-passing communication models, ranging from synchronous "rendez-vous" communications to fully asynchronous/out-of-order communications. For large-scale distributed systems, the communication model is determined by the transport layer of the network, and a few classes of orders of message delivery (FIFO, causally ordered) have been identified in the early days of distributed computing. For local-scale message-passing applications, e.g., running on a single machine, the communication model may be determined by the actual implementation of message buffers and by how FIFO queues are used. While large-scale communication models, such as causal ordering, are defined by logical axioms, local-scale models are often defined by an operational semantics. In this work, we connect these two approaches, and we present a unified hierarchy of communication models encompassing both large-scale and local-scale models, based on their non-sequential behaviors. We also show that all the communication models we consider can be axiomatised in the monadic second order logic, and may therefore benefit from several bounded verification techniques based on bounded special treewidth.

1 Introduction

Reasoning about distributed message-passing applications is hard. The degree of asynchrony of the communication model is an important ingredient of the application. One reason is that the communication architecture on which the application runs may vary, and must be accurately specified. Indeed, an approximation of the communication model may hide deadlocks or safety errors, such as unspecified receptions.

In synchronous communication (or rendez-vous communication), send and receive events are viewed as a single event, i.e., a receive event and the corresponding send event happen simultaneously. The idea behind asynchronous communication, instead, is to decouple send and receive events, so that a receive event can happen indefinitely after the corresponding send event. By introducing some additional constraints on the execution of those events, we can obtain new communication models that sit somewhere between synchronous and fully asynchronous communication. In this work we try to clarify and classify some of these communication models. On one hand, we consider communication models that were proposed in the early days of large-scale distributed computing to establish the correctness of some distributed algorithms, such as *causal ordering* [21] for the correctness of Lamport's distributed mutual exclusion algorithm (see also [31] for more examples), or the weaker "FIFO" peer-to-peer assumption. On the other hand, we look at communication models that emerge naturally when considering local-scale message-passing applications, which are based on predictable message buffering supported by local FIFO queues (for instance "mailboxes"). Such communication models have been considered in more recent works (for instance in [3]) and have caused some confusion, specifically regarding the difference between causal ordering and mailboxes [7, 15].

The classification and axiomatisation of large-scale communication models received great attention in the late 90s [9], while the local-scale communication models have only started to be investigated quite recently by Chevrou *et al.* [10], focusing on a *sequential* view of the behaviors of message-passing applications (to be detailed below). At the same time, several works [19, 32, 7, 22] recently addressed the verification of asynchronous message-passing applications by reduction to their synchronous semantics (see also [23] for a seminal work on these questions). These results strongly rely on the ability to safely approximate an asynchronous communication model by a synchronous one. There is therefore a need to clarify how the synchronous-asynchronous spectrum of communication models is organized.

In this work, we start from the sequential hierarchy established by Chevrou *et al.* [10], where a communication model is represented by a class of sequential executions; we revisit this hierarchy with a non-sequential point of view: we define a communication model as a class of *Message Sequence Charts* (MSCs in the following). Our MSCs are a graphical representation of computations of distributed systems, and they are a simplified version of the ITU recommendation [18]. Roughly speaking, a distributed system is composed of processes that can exchange messages. Each process executes a sequence of *send* or *receive actions*. A send event s and a receive event r are said to be *matching* if the message sent at s is the one received at r. In an MSC, such as the one in Fig. 1, each vertical line is called a *process line* and it represents the order in which events are executed by a single process, with time running from top to bottom; arrows are used to represent messages and they connect a send event with the corresponding matching receive. Given a message m_i, we will use $!i$ and $?i$ to denote the corresponding matching send and receive events, respectively. A single process line defines a total order over the events executed by that process, i.e., an event e happens before another event e' if e is higher in the process line; in Fig. 1, if we look at process q we see that ?1 happens before ?2. However, in general MSCs only specify a partial order over events. Consider the events !1 and !2 in Fig. 1, which are executed by two different processes; these two events are *truly concurrent*, meaning that this MSC does not tell us which one is executed first. Even though events on different processes can be concurrent, this is not always the case. For instance, a send event must always happen before the matching receive event. Graphically, this *happens before* relation between events on different processes is represented by a path that follows the direction of the arrows and runs from top to bottom. This will be referred to as a *causal path*, because it estabilishes a causal relation between events. Fig. 1 shows an example of causal path between the events !2 and ?3.

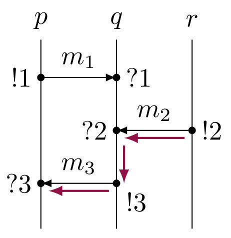


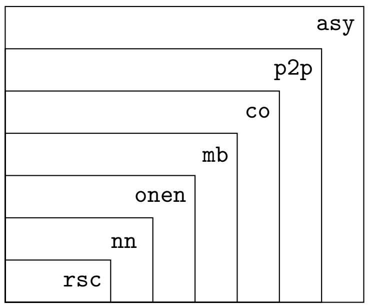
Figure 1: An example of MSC. Red arrows denote a causal path.

As mentioned before, in this work we interpret communication models as classes of MSCs. This non-sequential view of the communication models is arguably the "standard one", rather than the sequential point of view adopted by Chevrou *et al.* It is more relevant for comparing communication models, as some of them, such as causally ordered communications, intrinsically rely on this non-sequential view and the happens-before relation. Large-scale communication models are quite easy to axiomatize in a formal language, such as monadic second order logic (MSO) over MSCs. Local-scale communication models, on the other hand, are easy to define by means of an operational semantics involving FIFO queues, but their axiomatization in MSO is error prone and requires care. The choice of MSO as a specification language for the axiomatisation of these communication models is not incidental: there is a wealth of results on MSO model-checking over MSCs that originated from the work of Genest *et al.* [17, 16], with recent developments by Bollig *et al.* in the context of synchronizability [5].

Our contributions are the following:

- We review the peer-to-peer FIFO (`p2p`), causally ordered (`co`), mailbox (`mb`), FIFO `1−n` (`onen`), FIFO `n−n` (`nn`), asynchronous (`asy`), and synchronous (`rsc`) communication models and propose definitions of these models in terms of classes of MSCs. For the communication models whose intuition stems from an operational semantics, we provide an alternative operational definition, and we show soundness and completeness.

- From these definitions, we deduce a new non-sequential hierarchy of communication models (see Fig. 2a) and establish the strictness of this hierarchy by means of several examples. Surprisingly, the FIFO `1−n` class, that could be thought of as the "dual" of the mailbox class, is a subclass of mailbox class. This strongly contrasts with Chevrou *et al.* sequential



(a) The hierarchy of MSC classes.

	Weakly sync	Weakly k-sync	∃k bounded	∀k bounded
`asy`	unbounded STW	✓	✓	✓
`p2p`	✗ [5]	✓ [5]	✓ [5]	✓ [5]
`co`	✗	✓	✓	✓
`mb`	✓ [5]	✓ [5]	✓ [5]	✓ [5]
`onen`	✓	✓	✓	✓
`nn`	✓	✓	✓	✓

(b) (Un)decidability results for the synchronizability problems.

Figure 2: Main contributions

hierarchy, where FIFO `1−n` and mailbox are incomparable. The comparison between the FIFO `1−n` and mailbox classes is non-trivial in our non-sequential setting, and it motivates the introduction of several alternative characterizations of these communication models.

- We show that all of these communication models can be axiomatised in MSO. This result is rather subtle for mailbox and FIFO `1−n`, and highly non-trivial for FIFO `n−n`. For the latter, we develop a constructive proof based on an algorithm that computes a FIFO `n−n` linearization of an MSC.
- Building on the MSO characterization of these communication models, we derive several new decidability results (cfr. Fig. 2b) for bounded model-checking of systems of communicating finite state machines under various bounded assumptions (existential boundedness, weak synchronizability, etc).

Outline The report is organized as follows. Section 2 describes the communication models we consider. We also recall the notion of MSC and introduce formal definitions for these models, seen as classes of MSCs. In Section 3, we rely on an operational semantics to provide an alternative definition for some of these communication models, which we prove to be sound and complete. Section 4 characterizes the classes of MSCs via MSO logic. In Section 5, we compare all of the considered communication models and show our main result: a strict non-sequential hierarchy of communication models. Finally, Section 6 shows some (un)decidability results for various bounded model-checking problems based on MSO and on the notion of special treewidth. Related works are discussed all along the report in correspondance to specific notions.

2 Asynchronous communication models as classes of MSCs

In this section, we give both informal descriptions and formal definitions of the communication models that will be considered in this work. All of them impose different constraints on the order in which messages can be received.

We will use the following customary conventions: R^+ denotes the transitive closure of a binary relation R, while R^* denotes the transitive and reflexive closure. When R^* is denoted by a symbol suggesting a partial order, like $\leq$, we write e.g. $<$ for R^+. The cardinality of a set A is $|A|$. We assume a finite set of *processes* $\mathbb{P} = \{p, q, \ldots\}$ and a finite set of message contents (or just "message") $\mathbb{M} = \{m, \ldots\}$. Each process may either (asynchronously) send a message to another one, or wait until it receives a message. We therefore consider two kinds of actions. A *send action* is of the form $\mathit{send}(p,q,m)$; it is executed by process p and sends message m to process q. The corresponding *receive action* executed by q is $\mathit{rec}(p,q,m)$. We write $\mathit{Send}(p,q,\_)$ to denote the set $\{\mathit{send}(p,q,m) \mid m \in \mathbb{M}\}$, and $\mathit{Rec}(p,q,\_)$ for the set $\{\mathit{rec}(p,q,m) \mid m \in \mathbb{M}\}$. Similarly, for $p \in \mathbb{P}$, we set $\mathit{Send}(p,\_,\_) = \{\mathit{send}(p,q,m) \mid q \in \mathbb{P}\}$ and $m \in \mathbb{M}\}$, etc. Moreover, $\Sigma_p = \mathit{Send}(p,\_,\_) \cup \mathit{Rec}(\_,p,\_)$ denotes the set of all actions that are executed by p, and $\Sigma = \bigcup_{p\in\mathbb{P}} \Sigma_p$ is the set of all the actions. When p and q are clear from the context, we may write $!i$ (resp. $?i$) instead of $\mathit{send}(p,q,m_i)$ (resp. $\mathit{rec}(p,q,m_i)$).

Fully asynchronous communication In the fully asynchronous communication model (`asy`), messages can be received at any time once they have been sent, and send events are non-blocking. It can be modeled as a bag where all messages are stored and retrieved by processes when necessary (as described in [10] and [3]). It is also referred to as NON-FIFO (cfr. [9]). An MSC that shows a valid computation for the fully asynchronous communication model will be called a fully asynchronous MSC (or simply MSC). An example of such an MSC can be found in Fig. 3a; indeed, even if message m_1 is sent before m_2, process q does not have to receive m_1 first. Below, we give the formal definition of MSC.

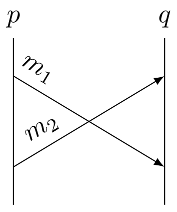
Definition 2.1 (MSC)**.** An MSC over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, where $\mathcal{E}$ is a finite (possibly empty) set of *events*, $\lambda : \mathcal{E} \rightarrow \Sigma$ is a labelling function that associates an action to each event, and $\rightarrow, \lhd$ are binary relations on $\mathcal{E}$ that satisfy the following three conditions. 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.

1. The *process relation* $\rightarrow \subseteq \mathcal{E} \times \mathcal{E}$ relates an event to its immediate successor on the same process: $\rightarrow = \bigcup_{p\in\mathbb{P}} \rightarrow_p$ for some 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 $\mathcal{E}_p$.
2. The *message relation* $\lhd \subseteq \mathcal{E} \times \mathcal{E}$ relates pairs of matching send/receive events:
   (2a) for every pair $(e,f) \in \lhd$, there are processes two p,q and a message m such that $\lambda(e) = \mathit{send}(p,q,m)$ and $\lambda(f) = \mathit{rec}(p,q,m)$.
   (2b) for all $f \in \mathcal{E}$ such that $\lambda(f) = \mathit{rec}(p,q,m)$, there is exactly one $e \in \mathcal{E}$ such that $e \lhd f$.
3. The *happens-before* relation[1] $\leq_{hb}$, defined by $(\rightarrow \cup \lhd)^*$, is a partial order on $\mathcal{E}$.

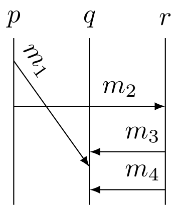
If, for two events e and f, we have that $e \leq_{hb} f$, we say that there is a *causal path* between e and f. Note that the same message m may occur repeatedly on a given MSC, hence the λ labelling function. In most of our examples, we avoid repeating twice a same message, hence events and actions are univocally identified. Definition 2.1 of (fully asynchronous) MSC will serve as a basis on which the other communication models will build on, adding some additional constraints.

According to Condition (2), every receive event must have a matching send event. However, note that, there may be unmatched send events. An unmatched send event represents the scenario in which the recipient is not ready to receive a specific message. This is the case of message m_1 in Fig. 3e. We will always depict unmatched messages with dashed arrows pointing to the time line of the destination process. 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 $\mathit{Unm}(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a send action and there is no } f \in \mathcal{E} \text{ such that } e \lhd f\}$.

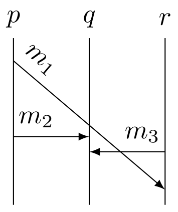
[1]This relation was introduced in [21], and is also referred to as the *happened before* relation, or sometimes *causal relation* or *causality relation*, e.g. in [9, 7] .



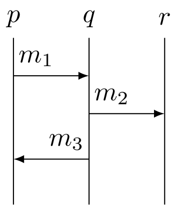
(a) `asy`.



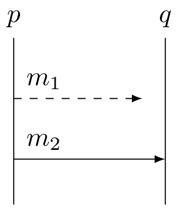
(b) `asy`, `p2p`.



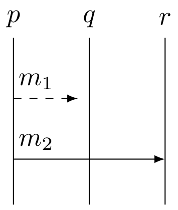
(c) `asy`, `p2p`, `co`, `mb`, `onen`, `nn`.



(d) `rsc`.



(e) `asy`.



(f) `asy`, `p2p`, `co`, `mb`.

Figure 3: Examples of MSCs for various communication models.

Example 2.1. For a set of processes $\mathbb{P} = \{p, q, r\}$ and a set of messages $\mathbb{M} = \{m_1, m_2, m_3\}$, Fig. 1 shows an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ where, for instance, we have $!1 \lhd ?1$, $?1 \ \rightarrow ?2$, and $!2 \ \le_{hb} ?3$. The set of actions is $\Sigma = \{\mathit{send}(p, q, m_1),\ \mathit{send}(r, q, m_2),\ \mathit{send}(q, p, m_3),\ \mathit{rec}(p, q, m_1),\ \mathit{rec}(r, q, m_2),\ \mathit{rec}(q, p, m_3)\}$, or, using the lightweight notation, $\Sigma = \{!1, !2, !3, ?1, ?2, ?3\}$.

Intuitively, a linearization represents the order in which events are executed by the distributed system according to *absolute time*, i.e., as they are seen by an external viewer that has a global view of all the processes. More formally, let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC. A *linearization* of M is a (reflexive) total order $\rightsquigarrow \subseteq \mathcal{E} \times \mathcal{E}$ such that $\le_{hb} \subseteq \rightsquigarrow$. In other words, a linearization of M represents a possible way to schedule its events. For convenience, we will omit the relation $\rightsquigarrow$ when writing a linearization, e.g., !1 !3 !2 ?2 ?3 ?1 is a possible linearization of the MSC in Fig. 3c.

MSCs form a monoid for the following notion of vertical concatenation. 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. The *concatenation* $M_1 \cdot M_2$ of M_1 and M_2 is the MSC $(\mathcal{E}, \rightarrow, \lhd, \lambda)$ where $\mathcal{E}$ is the disjoint union of $\mathcal{E}_1$ and $\mathcal{E}_2$, $\lhd = \lhd_1 \cup \lhd_2$, $\lambda(e) = \lambda_i(e)$ for all $e \in \mathcal{E}_i$ $(i = 1, 2)$; moreover, $\rightarrow = \rightarrow_1 \cup \rightarrow_2 \cup R$, were R is the set of event pairs (e_1, e_2) such that there is a process $p \in \mathbb{P}$ with e_1 the maximal event of $(\mathcal{E}_1)_p$ and e_2 the minimal event of $(\mathcal{E}_2)_p$. Note that $M_1 \cdot M_2$ is indeed an MSC and that concatenation is associative.

Peer-to-peer communication In the peer-to-peer (`p2p`) communication model, any two messages sent from one process to another are always received in the same order as they are sent. A straightforward implementation would be connecting processes pairwise with FIFO channels. For this reason, alternative names for this communication model are FIFO 1−1 [10] or simply FIFO [2, 9, 30]. MSCs that show valid computations for the `p2p` communication model will be called `p2p`-MSCs. The MSC shown in Fig. 3a is not a `p2p`-MSC, as m_1 cannot be received after m_2. Fig. 3b 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 are sent.

Definition 2.2 (`p2p`-MSCs)**.** A `p2p`-MSC is an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ where, for any two send events s and s' such that $\lambda(s) = \mathit{Send}(p, q, \_)$, $\lambda(s') = \mathit{Send}(p, q, \_)$, and $s \rightarrow^+ s'$, one of the following holds

- either $s, s' \in \mathit{Matched}(M)$ with $s \lhd r$ and $s' \lhd r'$ and $r \rightarrow^{+} r'$,
- or $s' \in \mathit{Unm}(M)$.

Note that, according to this definition, we cannot have two messages m_1 and m_2, both sent by p to q in that order, such that m_1 is unmatched and m_2 is matched; unmatched message m_1 excludes the reception of any later message. For this reason, the MSC shown in Fig. 3e is not `p2p`. On the other hand, the MSC in Fig. 3f is `p2p` because the two messages are not sent by and addressed to the same process.

Causally ordered communication In the causally ordered (`co`) 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. This type of partial order was introduced by Lamport in [21] with the "happened before" order. Later on, some implementations were proposed in [27, 29, 20]. Fig. 3b shows an example of non-causally ordered MSC; there is a causal path between the sending of m_1 and m_3, hence m_1 should be received before m_3, which is not the case here. On the other hand, the MSC in Fig. 3c is causally ordered; note that the only two messages with the same recipient are m_2 and m_3, but there is no causal path between their respective send events. Below the formal definition of causally ordered MSC (`co`-MSC).

Definition 2.3 (`co`-MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is *causally ordered* if, for any two send events s and s', such that $\lambda(s) = \mathit{Send}(\_, q, \_)$, $\lambda(s') = \mathit{Send}(\_, q, \_)$, and $s \leq_{hb} s'$, we have either:

- $s, s' \in \mathit{Matched}(M)$ with $s \lhd r$ and $s' \lhd r'$, and $r \rightarrow^{*} r'$, or
- $s' \in \mathit{Unm}(M)$.

Note that in a `co`-MSC we cannot have two send events s and s' addressed to the same process, such that s is unmatched, s' is matched, and $s \leq_{hb} s'$.

Mailbox communication In the mailbox (`mb`) communicating model, any two messages sent to a process must be received in the same order as they are sent (according to absolute time). 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 receives m_1 before m_2, then m_1 must have been sent before m_2. Essentially, `mb` coordinates all the senders of a single receiver. For this reason the model is also called FIFO $n{-}1$ [10]. A high-level implementation of the mailbox communication model could consist in a single incoming FIFO channel for each process p, in which all processes enqueue their messages to p. A low-level implementation can be obtained thanks to a shared real-time clock [12] or a global agreement on the order of events [14, 28]. The MSC shown in Fig. 3b is not a `mb`-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. 3c is a `mb`-MSC; 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). Below the definition of `mb`-MSC.

Definition 2.4 (`mb`-MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a `mb`-*MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events s and s', such that $\lambda(s) = \mathit{Send}(\_, q, \_)$, $\lambda(s') = \mathit{Send}(\_, q, \_)$, and $s \rightsquigarrow s'$

- either $s, s' \in \mathit{Matched}(M)$. Note that $r \rightsquigarrow r'$, since we have that $r \rightarrow^+ r'$,
- or $s' \in \mathit{Unm}(M)$.

Such a linearization will be referred to as a mb-*linearization*. Note that the definition of mb-MSC is based on the *existence* of a linearization with some properties. The same kind of "existential" definition will be used for the remaining communication models. In practice, to claim that an MSC is mb, we just need to find a single valid mb-linearization, regardless of all the others. As with co-MSCs, a mb-MSC cannot have two ordered send events s and s' addressed to the same process, such that s is unmatched, s' is matched. The message related to s would indeed block the buffer and prevent all subsequent receptions included the receive event matching s'. At this stage, the difference between co-MSCs and mb-MSCs might be unclear. Section 5 will clarify how all the classes of MSCs that we introduce are related to each other.

FIFO 1−n communication The FIFO 1−n (onen) communicating model is the dual of mb, it coordinates a sender with all the receivers. Any two messages sent by a process must be received in the same order (in absolute time) as they are sent. These two messages might be received by different processes and the two receive events might be concurrent. A high-level implementation of the FIFO 1−n communication model could consist in a single outgoing FIFO channel for each process, which is shared by all the other processes. A send event would then push a message on the outgoing FIFO channel. The MSC shown in Fig. 3b is not a onen-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. 3c shows an example of onen-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.

Definition 2.5 (onen-MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a onen-*MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events s and s', such that $\lambda(s) = \mathit{Send}(p, \_, \_)$, $\lambda(s') = \mathit{Send}(p, \_, \_)$, and $s \rightarrow^+ s'$ (which implies $s \rightsquigarrow s'$)

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

Such a linearization will be referred to as a onen-*linearization*. Note that a onen-MSC cannot have two send events s and s', executed by the same process, such that s is unmatched, s' is matched, and $s \rightarrow^+ s'$; indeed, it would not be possible to find a onen-linearization, according to Definition 2.5. The MSCs shown in Fig. 3e and Fig. 3f are clearly not onen-MSCs.

FIFO n−n communication In the FIFO n−n (nn) communicating model, messages are globally ordered and delivered according to their emission order. Any two messages must be received in the same order as they are sent, in absolute time. These two messages might be sent or received by any process and the two send or receive events might be concurrent. The FIFO n−n coordinates all the senders with all the receivers. A high-level implementation of the FIFO n−n communication model could consist in a single FIFO channel shared by all processes. It is considered also in [3] where it is called many-to-many (denoted *-*). However, as underlined in [10], such an implementation would be inefficient and unrealistic. The MSC shown in Fig. 3b is clearly not a nn-MSC; if we consider messages m_1 and m_2 we have that, in every linearization, $!1 \leq_{hb} !2$ and $?2 \leq_{hb} ?1$. This violates the constraints imposed by the FIFO n−n communication model. The MSC in Fig. 3c is a nn-MSC because we are able to find a linearization that satisfies the FIFO n−n constraint, e.g. !1 !2 !3 ?1 ?2 ?3.

Definition 2.6 (nn-MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a nn-*MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events s and s', such that $s \rightsquigarrow s'$

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

Such a linearization will be referred to as a nn-*linearization*. Note that, in a nn-linearization, unmatched messages can be sent only after all matched messages have been sent. As a consequence, a nn-MSC cannot have an unmatched send event s and a matched send event s', such that $s \leq_{hb} s'$; indeed, s would appear before s' in every linearization, and we would not be able to find a nn-linearization. The MSCs shown in Fig. 3e and Fig. 3f are both not FIFO n−n, since we have unmatched messages that are sent before matched messages.

RSC communication The Realizable with Synchronous Communication (rsc) communication model imposes the existence of a scheduling such that any send event is immediately followed by its corresponding receive event. It was introduced in [9], and it is the asynchronous model that comes closest to synchronous communication. The MSC in Fig. 3d is the only example of rsc-MSC: for instance linearization !1 ?1 !2 ?2 !3 ?3 respects the constraints of the rsc communication model.

Definition 2.7 (rsc-MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is an 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.

Such a linearization will be referred to as an rsc-*linearization*.

Classes of MSCs We denote by $\mathsf{MSC}_{\mathtt{asy}}$ (resp. $\mathsf{MSC}_{\mathtt{p2p}}$, $\mathsf{MSC}_{\mathtt{co}}$, $\mathsf{MSC}_{\mathtt{mb}}$, $\mathsf{MSC}_{\mathtt{onen}}$, $\mathsf{MSC}_{\mathtt{nn}}$, $\mathsf{MSC}_{\mathtt{rsc}}$) the sets of all MSCs (resp. p2p-MSCs, co-MSCs, mb-MSCs, onen-MSCs, nn-MSCs, rsc-MSCs) over the given sets $\mathbb{P}$ and $\mathbb{M}$. Note that we do not differentiate between isomorphic MSCs.

3 Asynchronous communication models as classes of executions

In the previous section, we defined several communication models as classes of MSCs. In this section, we provide alternative definitions these communication models fitting closer to the way these communication models are defined in other works. We establish the equivalence of the two alternative definitions for each concerned model. We also recall the notion of execution and Chevrou *et al.* sequential hierarchy of communication models seen as set of executions.

We consider networks of processes formed by a bunch of FIFO queues that store the messages in transit. Formally, a *queuing network* is a tuple $\mathfrak{n} = (\mathfrak{Q}, \mathfrak{buf})$ such that $\mathfrak{Q}$ is a finite set of queue identifiers, and $\mathfrak{buf} : \mathbb{P} \times \mathbb{P} \rightarrow \mathfrak{Q}$ assigns a queue to each pair of processes. A queuing network $(\mathfrak{Q}, \mathfrak{buf})$ is p2p if $\mathfrak{Q} = \mathbb{P} \times \mathbb{P}$ and $\mathfrak{buf}$ is the identity. The queuing network $(\mathfrak{Q}, \mathfrak{buf})$ is mb if $\mathfrak{Q} = \mathbb{P}$ and $\mathfrak{buf}(p, q) = q$; it is called onen if $\mathfrak{Q} = \mathbb{P}$ and $\mathfrak{buf}(p, q) = p$. Finally, it is called nn if $\mathfrak{Q} = \{0\}$ and $\mathfrak{buf}(p, q) = 0$ for all $p, q \in \mathbb{P}$.

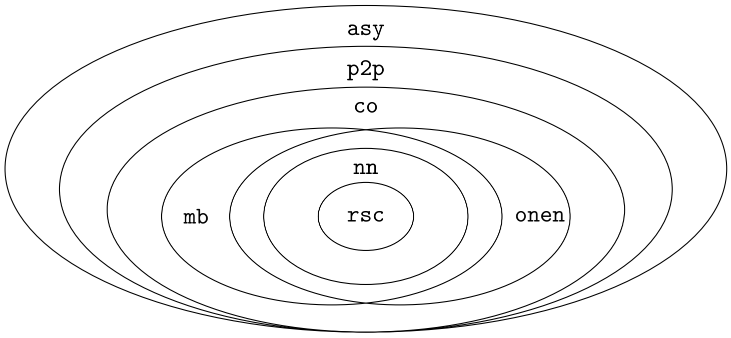


Figure 4: Hierarchy of communication models based on sets of executions (taken from [10])

Configurations, executions, and operational semantics A *configuration* of the queuing network $(\mathfrak{Q}, \mathfrak{buf})$ is a tuple $\gamma = (w_i)_{i \in \mathfrak{Q}} \in (\mathbb{M}^*)^{\mathfrak{Q}}$, where for each queue identifier i, the queue content w_i is a finite sequence of messages. The *initial configuration* $\gamma_\emptyset$ is the one in which all queues are empty, i.e. $w_i = \epsilon$ for all $i \in \mathfrak{Q}$. A *step* is a tuple (γ, a, γ'), (later written $\gamma \xrightarrow{a} \gamma'$) where $\gamma = (w_i)_{i \in \mathfrak{Q}}$, $\gamma' = (w'_i)_{i \in \mathfrak{Q}}$, a is an action, and the following holds:

- if $a = \mathit{send}(p, q, m)$, then $b'_i = b_i \cdot m$ and $b'_j = b_j$ for all $j \in \mathfrak{Q} \setminus \{i\}$, where $i = \mathfrak{buf}(p, q)$.
- if $a = \mathit{rec}(p, q, m)$, then $b_i = m \cdot b'_i$ and $b'_j = b_j$ for all $j \in \mathfrak{Q} \setminus \{i\}$, where $i = \mathfrak{buf}(p, q)$.

An *execution* of the queuing network $(\mathfrak{Q}, \mathfrak{buf})$ is a finite sequence of actions $e = a_1 a_2 \ldots a_n$ such that $\gamma_\emptyset \xrightarrow{a_1}\xrightarrow{a_2} \ldots \xrightarrow{a_n} \gamma$ for some configuration γ. The execution is `p2p` (resp. `mb`, `onen`, `nn`) if its queuing network is.

Example 3.1. The execution

$$\mathit{send}(p, q, m_1) \cdot \mathit{send}(q, r, m_2) \cdot \mathit{rec}(q, r, m_2) \cdot \mathit{rec}(p, q, m_1)$$

is `p2p`, `mb`, and `onen`, but it is not `nn` (because m_2 is received before m_1).

Example 3.2. The execution

$$\mathit{send}(p, q, m_1) \cdot \mathit{send}(r, q, m_2) \cdot \mathit{rec}(r, q, m_2)$$

is `p2p` and `onen`, but it is neither `mb` nor `nn` (because m_2 "overtakes" m_1). Note that in the final configuration m_1 is still in the queue (m_1 is "unmatched").

Consider a network $\mathfrak{n}$ with two queue identifiers i_1 and i_2, and let $\mathfrak{n}'$ be the network obtained by merging the two queues i_1 and i_2 in a same queue. Then $\mathfrak{n}'$ imposes more constraints than $\mathfrak{n}$ on the sequence of actions it admits, and any $\mathfrak{n}'$-execution also is an $\mathfrak{n}$-execution. From this observation, it follows that the communication models we considered define the hierarchy of executions depicted in Fig. 4. We refer to [10] for clarifying how the asynchronous, `rsc`, and `co` communication models may also be defined as sets of executions and fit in this hierarchy; we also refer to [10] for examples illustrating the strictness of this hierarchy.

To conclude this brief discussion on queuing networks and executions, we clarify in what sense the operational semantics we introduced in this section are sound and complete with respect to the axiomatic definitions of the communication models we gave in Section 2. Remember that a linearization $\rightsquigarrow$ of a MSC defines a total order on its events, and therefore an execution.

Fact 3.1. A MSC M is p2p (resp. mb, onen, nn) if and only if there exists a linearization $\rightsquigarrow$ of M that induces a p2p execution (resp. a mb, onen, nn execution).

Note that, moreover, a MSC M is p2p if and only if *all* of its linearizations induce p2p executions.

4 MSO definability

We have introduced seven different communication models and the corresponding classes of MSCs. Here, we show that all of these classes are MSO-definable, i.e. for every communication model com, there is a Monadic Second Order Logic formula φ_{com} that captures exactly the class $\mathsf{MSC}_{\text{com}}$ of all com-MSCs. The communication models whose definitions are stated as the existence of a linearization enjoying some properties are the most difficult to express in MSO. Indeed, their definition suggests a second-order quantification over a *binary* relation, but MSO is restricted to second-order quantification over *unary* predicates.

We therefore have to introduce alternative definitions (equivalent to those given in Section 2) that are closer to the logic, in order to prove MSO-definability. These alternative definitions will also be heavily used in the following sections for proofs. We first recall the formal definition of MSO logic over MSCs.

Definition 4.1 (MSO logic)**.** The set of MSO formulas over MSCs is given by the grammar $\varphi ::= \mathsf{true} \mid x \rightarrow y \mid x \lhd y \mid \lambda(x) = a \mid x = y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \varphi \mid \neg\varphi$, where $a \in \Sigma$, x and y are first-order variables (taken from an infinite set of variables), interpreted as events of an MSC, and X is a second-order variable, interpreted as a set of events.

We use common abbreviations such as $\wedge$, $\Rightarrow$, $\forall$, etc. For instance, 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. MSCs (a), (b), (c) and (d) of Fig. 3 satisfy the formula. Given a sentence φ, i.e., a formula without free variables, we let $L(\varphi)$ denote the set of asynchronous MSCs that satisfy φ. The formula true therefore describes the whole set of asynchronous MSCs, i.e. $L(\mathsf{true}) = \mathsf{MSC}_{\mathsf{asy}}$. It is folklore 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 (see also Appendix A). We will therefore abusively write formulas of the form $x \rightarrow^+ y$, $x \rightarrow^* y$ or $x \leq_{hb} y$.

Peer-to-peer MSCs The MSO formula that defines $\mathsf{MSC}_{\mathsf{p2p}}$ (i.e. the set of p2p-MSCs) directly follows from Definition 2.2:

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

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_{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 receptions happen in the reverse order, or (ii) s is unmatched and s' is matched.

Causally ordered MSCs As with p2p-MSCs, the MSO-definability of $\mathsf{MSC_{co}}$ follows from Definition 2.3, given in Section 2:

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

where ψ_1 and ψ_2 have been defined above for the p2p case. The property φ_{co} says that there cannot be two send events s and s', with the same recipient, such that $s \leq_{hb} s'$ 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.

Mailbox MSCs For the mailbox communication model, Definition 2.4 cannot be easily translated into an MSO formula. Thus, we introduce an alternative definition of mb-MSC that is closer to MSO logic; in particular, we define an additional binary relation that represents a constraint under the mb semantics, which ensures that messages received by a process are sent in the same order as they are received.

Definition 4.2 (mb alternative)**.** Let an MSC $M = (\mathcal{E}, \to, \lhd, \lambda)$ be fixed, and let $\sqsubset_{\mathtt{mb}} \subseteq \mathcal{E} \times \mathcal{E}$ be defined as $s \sqsubset_{\mathtt{mb}} s'$ if there is $q \in \mathbb{P}$ such that $\lambda(s) \in Send(\_, q, \_)$, $\lambda(s') \in Send(\_, q, \_)$, and either:

- $s \in Matched(M)$ and $s' \in Unm(M)$, or
- $s \lhd r_1$ and $s' \lhd r_2$ for some $r_1, r_2 \in \mathcal{E}_q$ such that $r_1 \to^+ r_2$.

We let $\prec_{\mathtt{mb}} = (\to \cup \lhd \cup \sqsubset_{\mathtt{mb}})^+$. M is a mb-*MSC* if $\preceq_{\mathtt{mb}}$ is a partial order.

The $\sqsubset_{\mathtt{mb}}$ relation expresses that two send events that are not necessarily related by a causal path should be scheduled in a precise order because their matching receptions are in this precise order. If $\preceq_{\mathtt{mb}}$ is a partial order, it means that it is possible to find a linearization $\rightsquigarrow$, such that $\rightsquigarrow \subseteq \preceq_{\mathtt{mb}}$. It is not difficult to see that such a linearization is exactly what we called a mb-linearization in Definition 2.4; we show here that Definitions 4.2 is indeed equivalent to Definition 2.4.

Proposition 4.1. Definition 2.4 and Definition 4.2 of mb-MSC are equivalent.

Proof. ($\Rightarrow$) We show that if M is a mb-MSC, according to Definition 4.2, then it is also a mb-MSC, according to Definition 2.4. By definition of $\preceq_{\mathtt{mb}}$, we must have (i) $s \preceq_{\mathtt{mb}} s'$ for any two matched send events s and s' addressed to the same process, such that $r \to^+ r$, where $s \lhd r$ and $s' \lhd r'$, and (ii) $s \preceq_{\mathtt{mb}} s'$, if s and s' are a matched and an unmatched send event, respectively. If $\preceq_{\mathtt{mb}}$ is a partial order, we can find at least one linearization $\rightsquigarrow$ such that $\preceq_{\mathtt{mb}} \subseteq \rightsquigarrow$; such a linearization satisfies the conditions of Definition 2.4.
($\Leftarrow$) We show that if M is not a mb-MSC, according to Definition 4.2, then it is also not a mb-MSC, according to Definition 2.4. Since $\preceq_{\mathtt{mb}} = (\to \cup \lhd \cup \sqsubset_{\mathtt{mb}})^*$ is not a partial order, $\preceq_{\mathtt{mb}}$ must be cyclic[2]. If $\preceq_{\mathtt{mb}}$ is cyclic, it means that we cannot find a linearization $\rightsquigarrow$ such that $\preceq_{\mathtt{mb}} \subseteq \rightsquigarrow$.

[2] $\preceq_{\mathtt{mb}}$ is reflexive and transitive by definition, if it were also acyclic it would be a partial order

In other words, we cannot find a linearization where (i) $s \rightsquigarrow s'$ for any two matched send events s and s' addressed to the same process, such that $r \rightarrow^+ r$, where $s \lhd r$ and $s' \lhd r'$, and (ii) $s \rightsquigarrow s'$, if s and s' are a matched and an unmatched send event, respectively. It follows that M is not a mb-MSC also according to Definition 2.4. □

The MSO-definability of $\mathsf{MSC}_{\mathtt{mb}}$ follows from Definition 4.2; in particular, note that $\preceq_{\mathtt{mb}}$ is reflexive and transitive by definition, thus we just have to check acyclicity: $\varphi_{\mathtt{mb}} = \neg\exists x.\ x \prec_{\mathtt{mb}} x$ where $x \prec_{\mathtt{mb}} y$ is obtained as the MSO-definable transitive closure of the union of the MSO-definable relations $\rightarrow$, $\lhd$, and $\sqsubset_{\mathtt{mb}}$, where $x \sqsubset_{\mathtt{mb}} y$ may be defined as:

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

FIFO 1−n MSCs As with the mailbox communication model, we give an alternative definition of onen-MSC.

Definition 4.3 (onen alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, let $\sqsubset_{\mathtt{1n}} \subseteq \mathcal{E} \times \mathcal{E}$ be defined as $e_1 \sqsubset_{\mathtt{1n}} e_2$ if there are two events e_1 and e_2, and $p \in \mathbb{P}$ such that either:

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

We let $\preceq_{\mathtt{1n}} = (\rightarrow \cup \lhd \cup \sqsubset_{\mathtt{1n}})^*$. M is a onen-*MSC* if $\preceq_{\mathtt{1n}}$ is a partial order.

The $\sqsubset_{\mathtt{1n}}$ relation ensures that messages sent by a process are sent and received in an order that is suitable for the onen communication. Since $\preceq_{\mathtt{1n}}$ is a partial order, it is possible to find a linearization $\rightsquigarrow$ such that $\rightsquigarrow \subseteq \preceq_{\mathtt{1n}}$. It is not difficult to see that such a linearization is exactly what we called a onen-linearization in Definition 2.5; we show here that Definitions 4.3 is indeed equivalent to Definition 2.5.

Proposition 4.2. Definition 2.5 and Definition 4.3 of onen-MSC are equivalent.

Proof. ($\Rightarrow$) We show that if M is a onen-MSC, according to Definition 4.3, then it is also a onen-MSC, according to Definition 2.5. By definition of $\preceq_{\mathtt{1n}}$, we must have (i) $r \preceq_{\mathtt{1n}} r'$ for any two receive events r and r' whose matched send events s and s' are such that $s \rightarrow^+ s'$, and (ii) $s \preceq_{\mathtt{1n}} s'$, if s and s' are a matched and an unmatched send event executed by the same process, respectively. If $\preceq_{\mathtt{1n}}$ is a partial order, we can find at least one linearization $\rightsquigarrow$ such that $\preceq_{\mathtt{1n}} \subseteq \rightsquigarrow$; such a linearization satisfies the conditions of Definition 2.5.
($\Leftarrow$) We show that if M is not a onen-MSC, according to Definition 4.3, then it is also not a onen-MSC, according to Definition 2.5. Since $\preceq_{\mathtt{1n}} = (\rightarrow \cup \lhd \cup \sqsubset_{\mathtt{1n}})^*$ is not a partial order, $\preceq_{\mathtt{1n}}$ must be cyclic. If $\preceq_{\mathtt{1n}}$ is cyclic, it means that we cannot find a linearization $\rightsquigarrow$ such that $\preceq_{\mathtt{1n}} \subseteq \rightsquigarrow$. In other words, we cannot find a linearization where (i) $r \rightsquigarrow r'$ for any two receive events r and r' whose matched send events s and s' are such that $s \rightarrow^+ s'$, and (ii) $s \rightsquigarrow s'$, if s and s' are a matched and an unmatched send event executed by the same process, respectively. It follows that M is not a onen-MSC also according to Definition 2.5. □

The existence of a MSO formula that defines $\mathsf{MSC_{onen}}$ follows from Definition 4.3 and the MSO definability of $\sqsubset_{\mathtt{1n}}$:

$$x \sqsubset_{\mathtt{1n}} y = \begin{array}{l} \left(\bigvee_{\substack{p \in \mathbb{P} \\ a,b \in Send(p,\_,\_)}} (\lambda(x) = a \ \wedge \ \lambda(y) = b) \ \wedge \ matched(x) \ \wedge \ \neg matched(y) \right) \ \vee \\ \left(\bigvee_{\substack{p \in \mathbb{P} \\ a,b \in Rec(p,\_,\_)}} (\lambda(x) = a \ \wedge \ \lambda(y) = b) \ \wedge \ \exists x'.\exists y'.(x' \lhd x \ \wedge \ y' \lhd y \ \wedge \ x' \rightarrow^{+} y') \right) \end{array}$$

FIFO $\mathtt{n}{-}\mathtt{n}$ MSCs In order to show the MSO-definability of $\mathtt{nn}$-MSCs we give an alternative definition and prove that it is equivalent to Definition 2.6. Unlike mailbox and FIFO $\mathtt{1}{-}\mathtt{n}$ communication models, the equivalence is not trivial.

Definition 4.4 ($\mathtt{nn}$ alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, let $\preceq_{\mathtt{1n/mb}} = (\rightarrow \cup \lhd \cup \sqsubset_{\mathtt{mb}} \cup \sqsubset_{\mathtt{1n}})^*$. We define $\sqsubset_{\mathtt{nn}} \subseteq \mathcal{E} \times \mathcal{E}$, such that $e_1 \sqsubset_{\mathtt{nn}} e_2$ if one of the following holds:

1. $e_1 \preceq_{\mathtt{1n/mb}} e_2$
2. $\lambda(e_1) \in Rec(\_,\_,\_)$, $\lambda(e_2) \in Rec(\_,\_,\_)$, $s_1 \lhd e_1$ and $s_2 \lhd e_2$ for some $s_1, s_2 \in \mathcal{E}$, $s_1 \preceq_{\mathtt{1n/mb}} s_2$ and $e_1 \npreceq_{\mathtt{1n/n1}} e_2$.
3. $\lambda(e_1) \in Send(\_,\_,\_)$, $\lambda(e_2) \in Send(\_,\_,\_)$, $e_1 \lhd r_1$ and $e_2 \lhd r_2$ for some $r_1, r_2 \in \mathcal{E}$, $r_1 \preceq_{\mathtt{1n/mb}} r_2$ and $e_1 \npreceq_{\mathtt{1n/n1}} e_2$.
4. $e_1 \in Matched(M)$, $e_2 \in Unm(M)$, $e_1 \npreceq_{\mathtt{1n/n1}} e_2$.

M is a $\mathtt{nn}$-*MSC* if $\sqsubset_{\mathtt{nn}}$ is acyclic.

We now show that Definition 2.6 and Definition 4.4 are equivalent. The proof is not trivial and we will need some preliminary results.

Proposition 4.3. 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, $r_1 \sqsubset_{\mathtt{nn}} r_2 \implies s_1 \sqsubset_{\mathtt{nn}} s_2$.

Proof. Follows from the definition of $\sqsubset_{\mathtt{nn}}$. We have $r_1 \sqsubset_{\mathtt{nn}} r_2$ if either:

- $r_1 \preceq_{\mathtt{1n/mb}} r_2$. Two cases: either (i) $s_1 \preceq_{\mathtt{1n/mb}} s_2$, or (ii) $s_1 \npreceq_{\mathtt{1n/n1}} s_2$. The first case clearly implies $s_1 \sqsubset_{\mathtt{nn}} s_2$, for rule 1 in the definition of $\sqsubset_{\mathtt{nn}}$. The second too, because of rule 3.
- $r_1 \npreceq_{\mathtt{1n/n1}} r_2$, but $r_1 \sqsubset_{\mathtt{nn}} r_2$. This is only possible if rule 2 in the definition of $\sqsubset_{\mathtt{nn}}$ was used, which implies $s_1 \preceq_{\mathtt{1n/mb}} s_2$ and, for rule 1, $s_1 \sqsubset_{\mathtt{nn}} s_2$.

□

Proposition 4.4. Let M be an MSC. If $\sqsubset_{\mathtt{nn}}$ is cyclic, then M is not a $\mathtt{nn}$-MSC (according to Definition 2.6).

Proof. According to Definition 2.6, an MSC is FIFO $\mathtt{n}{-}\mathtt{n}$ if it has at least one $\mathtt{nn}$-linearization. Note that, because of how it is defined, any $\mathtt{nn}$-linearization is always both a $\mathtt{mb}$ and a $\mathtt{onen}$-linearization. It follows that the cyclicity of $\preceq_{\mathtt{1n/mb}}$ (not $\sqsubset_{\mathtt{nn}}$) implies that M is not FIFO $\mathtt{n}{-}\mathtt{n}$, because it means that we are not even able to find a linearization that is both $\mathtt{mb}$ and FIFO $\mathtt{1}{-}\mathtt{n}$. Moreover, since in a $\mathtt{nn}$-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 $\mathtt{nn}$-MSC always has to satisfy the constraints imposed by the $\sqsubset_{\mathtt{nn}}$ relation. If $\sqsubset_{\mathtt{nn}}$ is cyclic, then for sure there is no $\mathtt{nn}$-linearization for M. □

Algorithm 1 Algorithm for finding a nn-linearization

Input: the EDG of an MSC M.
Output: a valid nn-linearization for M, if M is a nn-MSC.

```
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.
```

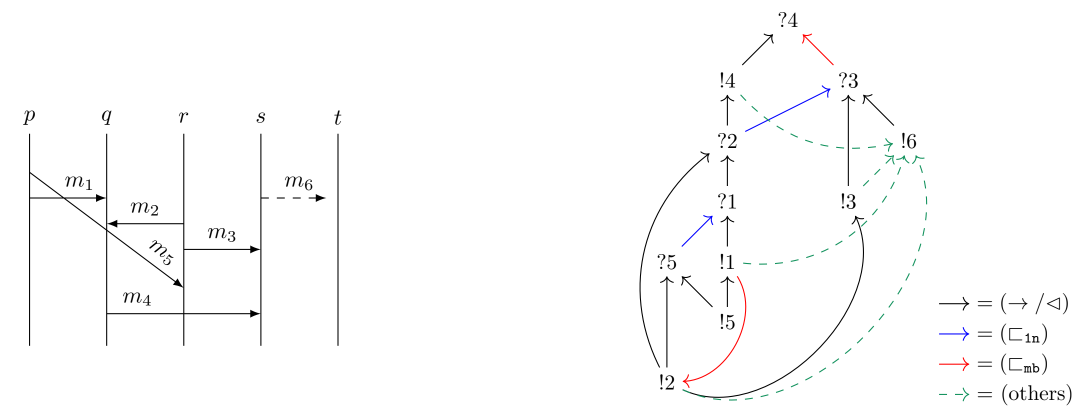
Let the *Event Dependency Graph* (EDG) of a nn-MSC M be a graph that has events as nodes and an edge between any two events e_1 and e_2 if $e_1 \sqsubset_{\mathsf{nn}} e_2$. Algorithm 1, given the EDG of an nn-MSC M, computes a nn-linearization of M. We show that, if $\sqsubset_{\mathsf{nn}}$ is acyclic, this algorithm always terminates correctly. This, along with Proposition 4.4, effectively shows that Definition 2.6 and Definition 4.4 are equivalent.

Example 4.1. Fig. 5 shows an example of nn-MSC and its EDG. We use it to show how the algorithm that builds a nn-linearization works. Note that, for convenience, not all the edges of the EDG have been drawn, but those missing would only connect events for which there is already a path is our drawing; these edges do not have any impact on the execution of the algorithm. We start by applying step 1 on the event !5, which has in-degree 0. The algorithm starts to build a linearization using !5 as the first event, and all the outgoing edges of !5 are removed from the EDG, along with the event itself. Now, !1 has in-degree 0 and we can apply again step 1. The partial linearization becomes !5 !1. Similarly, we can then apply step 1 on !2 and !3 to get the partial linearization !5 !1 !2 !3. At this point, step 1 and 2 cannot be applied, but we can use step 3 on ?5, which gets added to linearization. We then apply step 3 also to ?1 and ?2, followed by step 1 on !4, step 2 on !6 (which is an unmatched send event), and step 3 on ?3 and ?4. Finally, all the events of the MSC have been added to our linearization, which is !5 !1 !2 !3 ?5 ?1 ?2 !4 !6 ?3 ?4. Note that this is a nn-linearization.

We now need to show that (i) if Algorithm 1 terminates correctly (i.e. step 4 is never executed), it returns a nn-linearization, and (ii) if $\sqsubset_{\mathsf{nn}}$ is acyclic, the algorithm always terminates correctly.

Proposition 4.5. Given an MSC M, if Algorithm 1 returns a linearization then it is a nn-linearization.

Proof. Step 2 ensures that 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



p
q
r
s
t
m_1
m_2
m_3
m_4
m_5
m_6
?4
!4
?3
?2
!6
?1
!3
?5
!1
!5
!2
$\rightarrow = (\rightarrow / \lhd)$
$\rightarrow = (\sqsubset_{\mathtt{1n}})$
$\rightarrow = (\sqsubset_{\mathtt{mb}})$
$\dashrightarrow =$ (others)

Figure 5: An MSC and its EDG. In the EDG, only meaningful edges are shown.

send event is added to the linearization only if all the matched send events were already added. $\square$

The proof of termination proceeds by induction on the number of events added to the linearization and relies on the fact that, since $\sqsubset_{\mathsf{nn}}$ is acyclic, the EDG of the MSC is a DAG.

Proposition 4.6. Given an MSC M, Algorithm 1 terminates correctly if $\sqsubset_{\mathsf{nn}}$ is acyclic.

Proof. We want to prove that, if $\sqsubset_{\mathsf{nn}}$ is acyclic, step 4 of the algorithm is never executed, i.e. it terminates correctly. Note that the acyclicity of $\sqsubset_{\mathsf{nn}}$ 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 proceeds 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 $\sqsubset_{\mathsf{nn}}$, we have that $s \sqsubset_{\mathsf{nn}} u$, 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 $r_e \sqsubset_{\mathsf{nn}} r$ implies $s_e \sqsubset_{\mathsf{nn}} s$, according to Proposition 4.3, 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 $\sqsubset_{\mathsf{nn}}$, we have that $s \sqsubset_{\mathsf{nn}} u$, 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 $r_e \sqsubset_{\mathsf{nn}} r$ implies $s_e \sqsubset_{\mathsf{nn}} s$ (see Proposition 4.3), 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 $\sqsubset_{\mathsf{nn}}$ is acyclic, the algorithm always terminates correctly and computes a valid $\mathtt{nn}$-linearization. □

This effectively completes the proof of equivalence for Definition 2.6 and Definition 4.4. Based on Definition 4.4, we can now write the MSO formula for $\mathtt{nn}$-MSCs as $\varphi_{\mathsf{nn}} = \neg\exists x. x \sqsubset^{+}_{\mathsf{nn}} x$, where we can define $x \sqsubset_{\mathsf{nn}} y$ as:

$$x \sqsubset_{\mathsf{nn}} y = \begin{array}{l} \Big(\bigvee_{a,b \in Send(\_,\_,\_)} (\lambda(x) = a \ \wedge \ \lambda(y) = b) \ \wedge \ matched(x) \ \wedge \ \neg matched(y)\Big) \ \vee \\ (x \preceq_{\mathsf{1n/mb}} y) \quad \vee \quad \psi_3 \quad \vee \quad \psi_4 \end{array}$$

and ψ_3, ψ_4 can be specified as:

$$\psi_3 = \begin{array}{l} \bigvee_{a,b \in Rec(\_,\_,\_)} (\lambda(x) = a \ \wedge \ \lambda(y) = b) \ \wedge \\ \exists x'.\exists y'.(x' \lhd x \ \wedge \ y' \lhd y) \ \wedge \ (x' \preceq_{\mathsf{1n/mb}} y') \ \wedge \ \neg(x \preceq_{\mathsf{1n/mb}} y) \end{array}$$

$$\psi_4 = \begin{array}{l} \bigvee_{a,b \in Send(\_,\_,\_)} (\lambda(x) = a \ \wedge \ \lambda(y) = b) \ \wedge \\ \exists x'.\exists y'.(x \lhd x' \ \wedge \ y \lhd y') \ \wedge \ (x' \preceq_{\mathsf{1n/mb}} y') \ \wedge \ \neg(x \preceq_{\mathsf{1n/mb}} y) \end{array}$$

Formulas ψ_3 and ψ_4 encode conditions (2) and (3) in Definition 4.4, respectively. Note that $\preceq_{\mathtt{1n/mb}}$ is MSO-definable, since it is defined as the reflexive transitive closure of the MSO-definable relations $\rightarrow$, $\lhd$, $\sqsubset_{\mathtt{mb}}$, and $\sqsubset_{\mathtt{1n}}$.

Realizable with Synchronous Communication MSCs Following the characterization given in [9, Theorem 4.4], we provide an alternative definition of rsc-MSC that is closer to MSO logic. We first need to introduce the concept of *crown*.

Definition 4.5 (Crown)**.** Let M be an MSC. A *crown* of size k in M is a sequence $\langle (s_i, r_i),\ i \in \{1, \ldots, k\}\rangle$ of pairs of corresponding send and receive events such that

$$s_1 <_{hb} r_2, s_2 <_{hb} r_3, \ldots, s_{k-1} <_{hb} r_k, s_k <_{hb} r_1.$$

Definition 4.6 (rsc alternative)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a rsc-MSC if and only if it does not contain any crown.

The following MSO formula derives directly from previous definition:

$$\Phi_{\mathtt{rsc}} = \neg\exists s_1.\exists s_2.s_1 \propto s_2 \ \wedge \ s_2 \propto^* s_1$$

where $\propto$ is defined as

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

5 Hierarchy of classes of MSCs

As already mentioned, the classes of MSCs for all the seven communication models that we presented form a clear hierarchy, which is shown in Fig. 6. In this section we will provide proofs to support this claim.

First of all, by definition every p2p-MSC is an asy-MSC. Fig. 3a shows an example of MSC that is asynchronous but not p2p, hence we have $\mathsf{MSC}_{\mathtt{p2p}} \subset \mathsf{MSC}_{\mathtt{asy}}$. In the causally ordered communication model, any two messages addressed to the same process are received in an order that matches the causal order in which they are sent. In particular, it is easy to see that each co-MSC is also a p2p-MSC, since for any two messages sent by a process p to another process q, the two send events are causally ordered; below the formal proof.

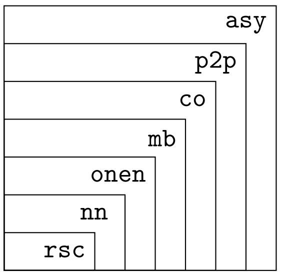


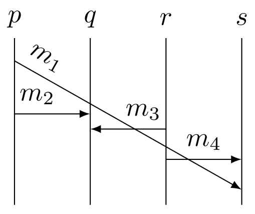
Figure 6: MSC classes.

Proposition 5.1. Every co-MSC is a p2p-MSC.

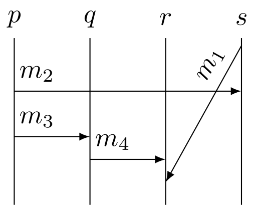
Proof. According to Definition 2.3, and MSC is co if, for any two send events s and s', such that $\lambda(s) = Send(\_, q, \_)$, $\lambda(s') = Send(\_, q, \_)$, and $s \leq_{hb} s'$, we have either (i) $s, s' \in Matched(M)$ and $r \rightarrow^* r'$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$, or (ii) $s' \in Unm(M)$. The conditions imposed by the Definition 2.2 of p2p are clearly satified by any co-MSC; in particular, note that $s \rightarrow^+ s'$ implies $s \leq_{hb} s'$. □

The MSC shown in Fig. 3b is p2p, but not co, hence we can conclude that $\mathsf{MSC}_{\mathtt{co}} \subset \mathsf{MSC}_{\mathtt{p2p}}$. We now show that each mb-MSC is a co-MSC.

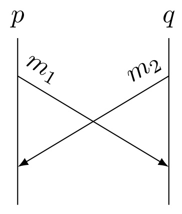
Proposition 5.2. Every mb-MSC is a co-MSC.



(a) `co`, not `mb`.



(b) `onen`, not `nn`.



(c) `nn`, not `rsc`.

Figure 7: Examples of MSCs for various communication models.

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

Fig. 7a shows an example of `co`-MSC that is not `mb`. It is causally ordered because we cannot find two messages, addressed to the same process, such that the corresponding send events are causally related; on the contrary, the MSC is not `mb` because we have $!4 \sqsubset_{\mathtt{mb}} !1$ and $!2 \sqsubset_{\mathtt{mb}} !3$, which lead to a cyclic dependency, e.g. $!1 \rightarrow !2 \sqsubset_{\mathtt{mb}} !3 \rightarrow !4 \sqsubset_{\mathtt{mb}} !1$. This example and Proposition 5.2 prove that $\mathsf{MSC}_{\mathtt{mb}} \subset \mathsf{MSC}_{\mathtt{co}}$.

The relation between `onen`-MSCs and `mb`-MSCs is not as straightforward as those seen so far. We start by only considering MSCs without unmatched messages.

Proposition 5.3. Every `onen`-MSC without unmatched messages is a `mb`-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 FIFO `1−n`. Suppose M is an asynchronous MSC, but not mailbox. There must be a cycle ξ such that $e \prec_{\mathtt{mb}} e$, for some event e. We can always explicitly write a cycle $e \prec_{\mathtt{mb}} e$ only using $\sqsubset_{\mathtt{mb}}$ and $<_{hb}$. For instance, there might be a cycle $e \prec_{\mathtt{mb}} e$ because we have that $e \sqsubset_{\mathtt{mb}} f <_{hb} g \sqsubset_{\mathtt{mb}} h \sqsubset_{\mathtt{mb}} i <_{hb} e$. Consider any two adjacent events s_1 and s_2 in the cycle ξ, where ξ has been written using only $\sqsubset_{\mathtt{mb}}$ and $<_{hb}$, and we never have two consecutive $\leq_{hb}$. This is always possible, since $a \leq_{hb} b \leq_{hb} c$ is written as $a \leq_{hb} c$. We have two cases:

1. $s_1 \sqsubset_{\mathtt{mb}} s_2$. We know, by definition of $\sqsubset_{\mathtt{mb}}$, 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. $s_1 <_{hb} s_2$. Since M is asynchronous by hyphotesis, ξ has to contain at least one $\sqsubset_{\mathtt{mb}}$. If that was not the case, $\leq_{hb}$ would also be cyclic and M would not be an asynchronous MSC. Recall that we also wrote ξ in such a way that we do not have two consecutive $\leq_{hb}$. 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. $s_1 \lhd r_1 \leq_{hb} s_2$. In particular, note that $r_1 \leq_{hb} r_2$.

   (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 $s_1 \rightarrow^+ s_k \lhd r_k \leq_{hb} s_2 \lhd r_2$, 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 $r_1 \sqsubset_{\mathtt{1n}} r_k$, according to the FIFO $\mathtt{1}-\mathtt{n}$ semantics. In particular, note that we have $r_1 \sqsubset_{\mathtt{1n}} r_k \leq_{hb} r_2$.

In both case (a) and (b), we conclude that $r_1 \preceq_{\mathtt{1n}} r_2$.

Notice that, for either cases, a relation between two send events s_1 and s_2 (i.e. $s_1 \sqsubset_{\mathtt{mb}} s_2$ or $s_1 \leq_{hb} s_2$) always implies a relation between the respective receive events r_1 and r_2, according to the FIFO $\mathtt{1}-\mathtt{n}$ semantics. It follows that ξ, which is a cycle for the $\preceq_{\mathtt{mb}}$ relation, always implies a cycle for the $\preceq_{\mathtt{1n}}$ relation (and if $\preceq_{\mathtt{1n}}$ is cyclic, M is not a $\mathtt{onen}$-MSC), as shown by the following example. Let M be a non-mailbox MSC, and suppose we have a cycle $s_1 \sqsubset_{\mathtt{mb}} s_2 \sqsubset_{\mathtt{mb}} s_3 \leq_{hb} s_4 \sqsubset_{\mathtt{mb}} s_5 \leq_{hb} s_1$. $s_1 \sqsubset_{\mathtt{mb}} s_2$ falls into case (1), so it implies $r_1 \rightarrow^+ r_2$. The same goes for $s_2 \sqsubset_{\mathtt{mb}} r_3$, which implies $r_2 \rightarrow^+ r_3$. $s_3 \leq_{hb} s_4$ falls into case (2), and implies that $r_3 \preceq_{\mathtt{1n}} r_4$. $s_4 \sqsubset_{\mathtt{mb}} s_5$ falls into case (1) and it implies $r_4 \rightarrow^+ r_5$. $s_5 \leq_{hb} s_1$ falls into case (2) and implies that $r_5 \preceq_{\mathtt{1n}} r_1$. Putting all these implications together, we have that $r_1 \rightarrow^+ r_2 \rightarrow^+ r_3 \preceq_{\mathtt{1n}} r_4 \rightarrow^+ r_5 \preceq_{\mathtt{1n}} r_1$, which is a cycle for $\preceq_{\mathtt{1n}}$. Note that, given any cycle for $\preceq_{\mathtt{mb}}$, we are always able to apply this technique to obtain a cycle for $\preceq_{\mathtt{1n}}$. □

The opposite direction is also true.

Proposition 5.4. Every $\mathtt{mb}$-MSC without unmatched messages is a $\mathtt{onen}$-MSC.

Proof. We show that the contrapositive is true, i.e. if an MSC is not FIFO $\mathtt{1}-\mathtt{n}$ (and it does not have unmatched messages), it is also not mailbox. Suppose M is an asynchronous MSC, but not FIFO $\mathtt{1}-\mathtt{n}$. There must be a cycle ξ such that $e \preceq_{\mathtt{1n}} e$, for some event e. Recall that $\preceq_{\mathtt{1n}} = (\rightarrow \cup \sqsubset_{\mathtt{1n}} \cup \sqsubset_{\mathtt{mb}})^*$ and $\leq_{hb} = (\rightarrow \cup \sqsubset_{\mathtt{1n}})^*$. We can always explicitely write a cycle $e \preceq_{\mathtt{1n}} e$ only using $\sqsubset_{\mathtt{1n}}$ and $\leq_{hb}$. For instance, there might be a cycle $e \preceq_{\mathtt{1n}} e$ because we have that $e \sqsubset_{\mathtt{1n}} f \leq_{hb} g \sqsubset_{\mathtt{1n}} h \sqsubset_{\mathtt{1n}} i \leq_{hb} e$. Consider any two adiacent events r_1 and r_2 in the cycle ξ, where ξ has been written using only $\sqsubset_{\mathtt{1n}}$ and $\leq_{hb}$, and we never have two consecutive $\leq_{hb}$. We have two cases:

1. $r_1 \sqsubset_{\mathtt{1n}} r_2$. By definition of $\sqsubset_{\mathtt{1n}}$, r_1 and r_2 must be two receive events, since we are not considering unmatched send events, and $s_1 \rightarrow^+ s_2$, where s_1 and s_2 are the send events that match with r_1 and r_2, respectively.

2. $r_1 \leq_{hb} r_2$. Since M is asynchronous by hyphotesis, ξ has to contain at least one $\sqsubset_{\mathtt{1n}}$; recall that we also wrote ξ in such a way that we do not have two consecutive $\leq_{hb}$. It is not difficult to see that r_1 and r_2 have to be receive events, since they belong to ξ. Let s_1 and s_2 be the two send events such that $s_1 \lhd r_1$ and $s_2 \lhd r_2$. We have two cases:

   (a) s_2 is in the causal path between r_1 and r_2, i.e. $s_1 \lhd r_1 \leq_{hb} s_2 \lhd r_2$. In particular, note that $s_1 \leq_{hb} s_2$.

   (b) s_2 is not in the causal path between r_1 and r_2, hence there must be a message m_k received by the same process that executes r_2, such that $r_1 \leq_{hb} s_k \lhd r_k \rightarrow^+ r_2$, where r_k is the send event of m_k. Since messages m_k and m_2 are received by the same process and $r_k \rightarrow^+ r_2$, we should have $s_k \sqsubset_{\mathtt{mb}} s_2$, according to the mailbox semantics. In particular, note the we have $s_1 \leq_{hb} s_k \sqsubset_{\mathtt{mb}} s_2$.

   In both case (a) and (b), we conclude that $s_1 \preceq_{\mathtt{mb}} s_2$.

Notice that, for either cases, a relation between two receive events r_1 and r_2 implies a relation between the respective send events s_1 and s_2, according to the mailbox semantics. It follows that ξ, which is a cycle for the $\preceq_{\mathtt{1n}}$ relation, always implies a cycle for the $\preceq_{\mathtt{mb}}$ relation. □

Interestingly enough, Proposition 5.3 and 5.4 show that the classes of $\texttt{mb}$-MSCs and $\texttt{onen}$-MSCs coincide if we do not allow unmatched messages. This changes when we add unmatched messages into the mix. However, Proposition 5.3 still holds.

Proposition 5.5. Every $\texttt{onen}$-MSC is a $\texttt{mb}$-MSC.

Proof. Let M be an asynchronous MSC. The proof proceeds as for Proposition 5.3, but unmatched messages introduce some additional cases. Consider any two adjacent events s_1 and s_2 in a cycle ξ for $\prec_{\texttt{mb}}$, where ξ has been written using only $\sqsubset_{\texttt{mb}}$ and $<_{hb}$, and we never have two consecutive $<_{hb}$. These are some additional cases:

3. $u_1 \sqsubset_{\texttt{mb}} s_2$, where u_1 is the send event of an unmatched message. This case never happens because of how $\sqsubset_{\texttt{mb}}$ is defined.

4. $u_1 \leq_{hb} u_2$, 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 $u_1 \leq_{hb} u_2 \sqsubset_{\texttt{mb}} s_3$. However, $u_2 \sqsubset_{\texttt{mb}} s_3$ falls into case (3), which can never happen.

5. $u_1 \leq_{hb} s_2$, 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 $u_1 \rightarrow^+ s_k \lhd r_k \leq_{hb} s_2 \lhd r_2$[3], 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 m_1 is unmatched, we should have $s_k \sqsubset_{\texttt{1n}} u_1$, according to the FIFO $\texttt{1}-\texttt{n}$ semantics, but $u_1 \rightarrow^+ s_k$. It follows that if ξ contains $u_1 \leq_{hb} s_2$, we can immediately conclude that M is not a $\texttt{onen}$-MSC.

6. $s_1 \sqsubset_{\texttt{mb}} u_2$, 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 $s_1 \sqsubset_{\texttt{mb}} u_2 \leq_{hb} s_3$; we cannot have $u_2 \sqsubset_{\texttt{mb}} s_3$, because of case (3). $u_2 \leq_{hb} s_3$ falls into case (5), so we can conclude that M is not a $\texttt{onen}$-MSC.

We showed that cases (3) and (4) can never happen, whereas (5) and (6) imply that M is not FIFO $\texttt{1}-\texttt{n}$. If we combine them with the cases described in Proposition 5.3 we have the full proof. ☐

The MSC in Fig. 3f shows a simple example of an MSC with unmatched messages that is $\texttt{mb}$ but not $\texttt{onen}$. This, along with Proposition 5.5, effectively shows that $\mathsf{MSC}_{\texttt{onen}} \subset \mathsf{MSC}_{\texttt{mb}}$.

In the FIFO $\texttt{n}-\texttt{n}$ communication model, any two messages must be received in the same order as they are sent. It is quite intuitive to observe that each $\texttt{nn}$-MSC is a $\texttt{onen}$-MSC, because each $\texttt{nn}$-linearization is also a $\texttt{onen}$-linearization; below the formal proof.

Proposition 5.6. Every $\texttt{nn}$-MSC is a $\texttt{onen}$-MSC.

Proof. Consider Definition 2.6 and Definition 2.5. They are identical, except for the fact that in the FIFO $\texttt{n}-\texttt{n}$ case we consider any two send events, and not just those that are sent by a same process. This is enough to show that each $\texttt{nn}$-linearization is also a $\texttt{onen}$-linearization and, therefore, each $\texttt{nn}$-MSC is a $\texttt{onen}$-MSC. ☐

Fig. 7b shows an example of MSC that is FIFO $\texttt{1}-\texttt{n}$ but not FIFO $\texttt{n}-\texttt{n}$; in particular, note that for messages m_1 and m_4 we have $!1 \leq_{hb} !4$ and $?4 \rightarrow ?1$, so there cannot be a $\texttt{nn}$-linearization, but it is possible to find a $\texttt{onen}$-linearization, such as !1 !2 ?2 !3 ?3 !4 ?4 ?1. This example, along with Proposition 5.6, shows that $\mathsf{MSC}_{\texttt{nn}} \subset \mathsf{MSC}_{\texttt{onen}}$. In the $\texttt{rsc}$ model, every send event is

[3]Note that we can have $m_k = m_2$

immediately followed by its corresponding receive event. $\mathtt{rsc}$ is then a special case of FIFO $\mathtt{n{-}n}$ communication, and every $\mathtt{rsc}$-MSC is a $\mathtt{nn}$-MSC because a $\mathtt{rsc}$-linearization is always also a $\mathtt{nn}$-linearization; below the formal proof.

Proposition 5.7. Every $\mathtt{rsc}$-MSC is a $\mathtt{nn}$-MSC.

Proof. Consider Definition 2.7 and Definition 2.6. Let us pick an $\mathtt{rsc}$-linearization $\leadsto$. If every send event is immediately followed by its matching receive event, and we do not have unmatched messages, then $\leadsto$ is also a $\mathtt{nn}$-linearization; note that, for any two send events s and s' such that $s \leadsto s'$, we also have $r \leadsto r'$, where $s \lhd r$ and $s' \lhd r'$. It follows that each $\mathtt{rsc}$-MSC is a $\mathtt{nn}$-MSC. ☐

Besides, Fig. 7c shows an example of MSC that is FIFO $\mathtt{n{-}n}$ but not $\mathtt{rsc}$, therefore $\mathsf{MSC}_{\mathtt{rsc}} \subset \mathsf{MSC}_{\mathtt{nn}}$.

6 Application: synchronizability and model-checking

In this section, we show how the MSO characterization of all the communication models we considered induces several decidability results for synchronizability and bounded model-checking problems on systems of communicating finite state machines.

A communicating finite state machine is a finite state automaton labeled with send and receive actions; a system $\mathcal{S}$ is a finite collection of such machines. An MSC M is an asynchronous behavior of $\mathcal{S}$ if every process time line of M is accepted by its corresponding process automaton (see Appendix B.1 for a formal definition of these notions). We write $L_{\mathtt{asy}}(\mathcal{S})$ to denote the set of asynchronous behaviors of $\mathcal{S}$, and we write $L_{\mathrm{com}}(\mathcal{S})$ to denote the restriction of $L_{\mathtt{asy}}$ to com MSCs, i.e. $L_{\mathrm{com}}(\mathcal{S}) = L_{\mathtt{asy}}(\mathcal{S}) \cap \mathsf{MSC}_{\mathrm{com}}$.

In general, even simple verification problems, such as control-state reachability, are undecidable for communicating systems [8], under all communication models (except $\mathtt{rsc}$, which we won't consider anymore from now on). They may become decidable if we restrict the behaviors, which motivates the following definition of generic *bounded model-checking* problem for $\{\mathtt{asy}, \mathtt{p2p}, \mathtt{co}, \mathtt{mb}, \mathtt{onen}, \mathtt{nn}\}$. Let $\mathcal{C}$ be a class of MSCs. The $\mathcal{C}$-bounded model-checking problem for a communication model com is: given a system $\mathcal{S}$ and a MSO specification φ, decide whether $L_{\mathrm{com}}(\mathcal{S}) \cap \mathcal{C} \subseteq L(\varphi)$. In the remainder, we consider classes $\mathcal{C}$ of MSCs that share the same intuition that they only contain "almost synchronous" MSCs. So the bounded model-checking problem corresponds to an under-approximation of the standard model-checking problem where the system is assumed to be "almost synchronous". The question of the completeness of this under-approximaton, i.e. whether $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$, will be referred to as the "synchronizability problem".

Bollig *et al.* [5] introduced a general framework that allows to derive decidability results for the bounded model-checking and synchronizability problems for various classes of MSCs $\mathcal{C}$. The communication model is however a fixed parameter in their framework. In this section, we roughly manage to make this framework parametric in the communication model, up to several technical restrictions. The first restriction is that the communication model, combined with the bounding class $\mathcal{C}$, should enforce a bounded treewidth of the MSCs, which is not always the case. The second restriction is that a key lemma in the framework of Bollig *et al.* relied on the existence of "borderline violations", which was granted by a form of prefix closure of the MSCs of a given class. However, this prefix closure property does not hold for all communication models, and these models must be treated with specific techniques.

Special treewidth and bounded model-checking *Special treewidth* (STW) is a graph measure that somehow indicates how close a graph is to a tree (we may also use Courcelle's classical *treewidth* instead [11]). This measure or similar ones are commonly employed in verification. For instance, treewidth and split-width have been used in [25] and [13, 1], respectively, to reason about graph behaviors generated by pushdown and queue systems. Note that an MSC is a graph where the nodes are the events and the edges are represented by the $\rightarrow$ and the $\lhd$ relations.

There are several ways to define the special treewidth of a graph. We adopt the following game-based definition from [4], and, to make it more concrete, reformulate it directly on MSCs instead of graphs. Adam and Eve play a turn based "decomposition game" on an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$. M Eve starts to play and does a move, which consists in the following steps:

1. marking some events of M, resulting in the *marked MSC fragment* (M, U'), where $U' \subseteq \mathcal{E}$ is the subset of marked events,

2. removing edges whose both endpoints are marked, in such a way that the resulting MSC is disconnected (i.e. there are at least two different connected components),

3. splitting (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.

Once Eve does her move, it is Adam's turn. Adam simply chooses one of the two marked MSC fragments, either (M_1, U_1) or (M_2, U_2). Now it is again Eve's turn, and she has to do a move on the marked MSC fragment that was chosen by Adam. The game continues in alternating turns between the two players until they reach a point where all the events on the current marked MSC fragment are marked. For $k \in \mathbb{N}$, we say that the game is k-winning for Eve if she has a strategy that allows her, independently of Adam's moves, to end the game in a way that every marked MSC fragment visited during the game has at most $k + 1$ marked events. The goal of Eve is to keep k as low as possible. Fig. 8 shows an example of a 3-winning game for the MSC in Fig. 3b.

The special treewidth of an MSC is the least k such that the associated game is k-winning for Eve (see for instance [4]). The set of MSCs whose special treewidth is at most k is denoted by $\mathsf{MSC}^{k\text{-stw}}$. It is easy to check that trees have a special treewidth of 1.

Example 6.1. Let M the MSC of the Fig. 3b. In this example, we show that M has a special treewidth of at most 3, since Eve is able to find a strategy that leads to a 3-winning game. We use colors to mark events. Eve starts by marking 4 events. The edges whose both endpoints are marked can be removed (dotted edges in the figure) and the graph becomes disconnected. Eve then splits the graph in 2 and Adam has to choose. Suppose the Adam picks the subgraph with the red and yellow events already marked (top branch in the figure). Eve can mark the third event and, by doing so, the game ends. Suppose Adam chooses the subgraph with the blue and green events (bottom branch). Eve marks the two nodes in the bottom, removes 3 edges, and splits the graph in two. Note that one of the two subgraphs already has all events marked, so Adam zpicks the other one (top branch). Eve simply marks the missing event and the game ends. This is a 3-winning game for Eve since, independently of Adam's choices, we have at most 4 marked event at each step.

Courcelle's theorem implies that the following problems is decidable: given a MSO formula φ and $k \geq 1$, decide whether φ holds for all MSCs $M \in \mathsf{MSC}^{k\text{-stw}}$. Therefore, a direct consequence of Courcelle's theorem and of our MSO characterization of the communication models is that bounded-model-checking is decidable[4].

[4]cfr. proof in Appendix B.2

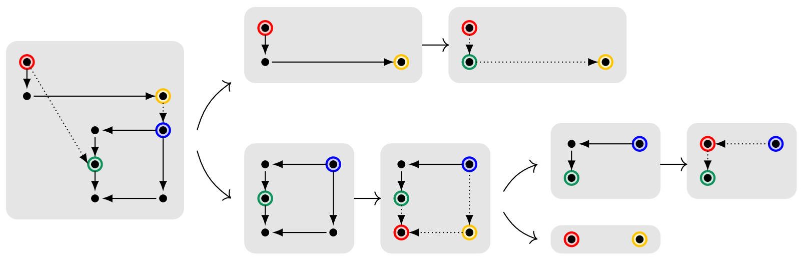
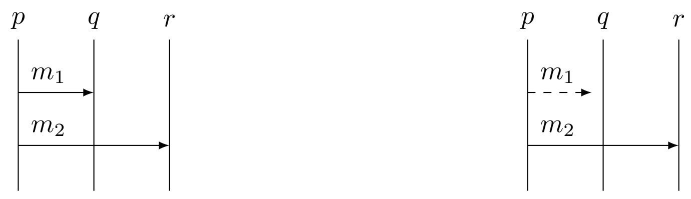


Figure 8: Decomposition game for the MSC of Fig. 3b. This is a 3-winning game for Eve.



(a) A `nn`-MSC M.

(b) A prefix of M.

Figure 9: A `nn`-MSC and a prefix that is neither a `onen`-MSC nor a `nn`-MSC.

Theorem 6.1. Let com $\in \{\texttt{asy}, \texttt{co}, \texttt{p2p}, \texttt{mb}, \texttt{onen}, \texttt{nn}, \texttt{rsc}\}$ and $k \geq 1$ be fixed. Then the following problem is decidable: given a system $\mathcal{S}$ and a MSO specification φ, decide whether $L_{\text{com}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-stw}} \subseteq L(\varphi)$.

The synchronizability problem Theorem 6.1 remains true if instead of $\mathsf{MSC}^{k\text{-stw}}$ we bound the model-checking problem with a class $\mathcal{C}$ of MSCs that is both treewidth bounded and MSO definable. The synchronizability problem (SP, for short) consists in deciding whether this bounded model-checking is complete, i.e. whether all the behaviors generated by a given communicating system are included in this class $\mathcal{C}$, i.e. whether $L_{\text{com}}(\mathcal{S}) \subseteq \mathcal{C}$.

Definition 6.1. Let a communication model com and a class $\mathcal{C}$ of MSCs be fixed. The (com, $\mathcal{C}$)-synchronizability problem is defined as follows: given a system $\mathcal{S}$, decide whether $L_{\text{com}}(\mathcal{S}) \subseteq \mathcal{C}$.

In [5] the authors show that, for com = `p2p` and com = `mb`, the (com, $\mathcal{C}$)-synchronizability problem is decidable for several classes $\mathcal{C}$. We generalize their result to other communication models under a general assumption on the bounding class $\mathcal{C}$.

Theorem 6.2. For any com $\in \{\texttt{asy}, \texttt{p2p}, \texttt{co}, \texttt{mb}, \texttt{onen}, \texttt{nn}\}$ and for all class of MSCs $\mathcal{C}$, if $\mathcal{C}$ is STW-bounded and MSO-definable, then the (com, $\mathcal{C}$)-synchronizability problem is decidable.

The proof of Theorem 6.2 resembles the proof of [6, Theorem 11], and the main technical argument of the existence of a "borderline violation" remains (see [6, Lemma 9]). However, the existence of a borderline violation is more subtle to establish, because the $\mathsf{MSC}_{\texttt{onen}}$ and $\mathsf{MSC}_{\texttt{nn}}$ are not prefixed-closed (see Fig. 9). A way to solve this technical problem is to consider a more strict notion of prefix. All details of the proof of Theorem 6.2 can be found in Appendix B.3.

In the remainder, we investigate which combinations of com and $\mathcal{C}$ fit the hypotheses of this theorem. We review the classes of weakly synchronous and weakly k-synchronous inspired by [7],

	Weakly sync	Weakly k-sync	∃k bounded	∀k bounded
`asy`	unbounded STW	✓	✓	✓
`p2p`	✗ [5]	✓ [5]	✓ [5]	✓ [5]
`co`	✗	✓	✓	✓
`mb`	✓ [5]	✓ [5]	✓ [5]	✓ [5]
`onen`	✓	✓	✓	✓
`nn`	✓	✓	✓	✓

Figure 10: Table summarising the (un)decidability results for the synchronizability problems (each combination of a communication model com and a class $\mathcal{C}$ of MSCs is a different synchronizability problem). The symbol ✗ stands for undecidability and unbounded special treewidth of $\mathsf{MSC}_{\mathrm{com}} \cap \mathcal{C}$, whereas ✓ stands for decidability and bounded STW of $\mathsf{MSC}_{\mathrm{com}} \cap \mathcal{C}$. Unbounded STW stands for unbounded STW of $\mathsf{MSC}_{\mathrm{com}} \cap \mathcal{C}$ (but not necessarily undecidability).

and the classes of existentially k-bounded and universally k-bounded MSCs [17]. Fig. 10 summarizes the decidability results of the $(\mathrm{com}, \mathcal{C})$-synchronizability problem for each combination of com and $\mathcal{C}$ we will consider in the next sections.

Weakly synchronous MSCs We first introduce the class of weakly synchronous MSCs. This is a generalization of k-synchronous MSCs studied in [7]. We say an MSC is weakly synchronous if it can be chunked into *exchanges*, where an exchange is an MSC that allows one to schedule all send events before all receive events.

Definition 6.2 (Exchange)**.** Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC. We say that M is an *exchange* if $\mathit{SendEv}(M)$ is a $\leq_{hb}$-downward-closed set.

In other words, an exchange is an MSC M where no send event depends on a receive event. If that is the case, we can find a linearization for M where all the send events are executed before the receive events. Remember that $M_1 \cdot M_2$ denote the vertical concatenation of MSCs (see Section 2).

Definition 6.3 (Weakly synchronous)**.** We say that $M \in \mathsf{MSC}$ is *weakly synchronous* if it is of the form $M = M_1 \cdot M_2 \cdots M_n$ such that every M_i is an exchange.

Example 6.2. Consider the MSC M_1 in Fig. 11. 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 interleaved, but they constitute an exchange, as all sends can be scheduled before all the receptions.

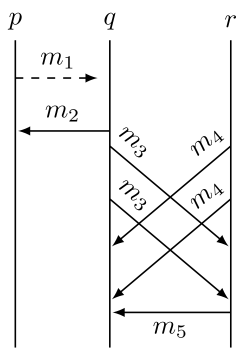


Figure 11: M_1

In [5] it is shown that, for the class of weakly synchronous MSCs, the synchronizability problem is undecidable for com = `p2p`, but decidable for com = `p2p`. Here we investigate the decidability of weak synchronizability for the other communication models. We first show that weak synchronizability is undecidable for causally ordered communication. The proof is an adaptation of the one given in [6, Theorem 20] for the `p2p` case (cfr. Appendix B.4).

Proposition 6.1. The following problem is undecidable: given a communicating system $\mathcal{S}$, is every MSC in $L_{\mathsf{co}}(\mathcal{S})$ weakly synchronous?

For `onen` and FIFO `n−n`, on the other hand, weak synchronizability is decidable.

Proposition 6.2. Let com $\in \{\texttt{onen}, \texttt{nn}\}$. The following problem is decidable: given a communicating system $\mathcal{S}$, is every MSC in $L_{\text{com}}(\mathcal{S})$ weakly synchronous?

Proof. We will consider com $=$ `onen`; the proof for com $=$ `nn` is similar. We would like to know if every MSC in $L_{\texttt{onen}}(\mathcal{S})$ is in the class of weakly synchronous MSCs. Since every MSC in $L_{\texttt{onen}}(\mathcal{S})$ is a `onen`-MSC, we can equivalently restrict the problem to the class of weakly synchronous MSCs that are also `onen`-MSCs. Let $\mathcal{C}$ be the class of `onen` weakly synchronous MSCs; we show that $\mathcal{C}$ is MSO-definable and STW-bounded, which implies the decidability of SP for Theorem 6.2. The class of weakly synchronous MSCs was shown to be MSO-definable in [5]; to be precise, their characterization is for `p2p` weakly synchronous MSCs (since their definition of MSC is equivalent to our definition of `p2p`-MSC), but it also works for (asynchronous) weakly synchronous MSCs. We showed in Section 4 that $\mathsf{MSC}_{\texttt{onen}}$ is MSO-definable; it follows that the class of `onen` weakly synchronous MSCs is also MSO-definable (we just take the conjuction of the the two formulas). The class of `mb` weakly synchronous MSCs was shown to be STW-bounded in [5], and since $\mathsf{MSC}_{\texttt{onen}} \subset \mathsf{MSC}_{\texttt{mb}}$, we also have that the class of `mb` weakly synchronous MSCs has a bounded special treewidth. $\square$

Weakly k-synchronous MSCs We consider now weakly k-synchronous MSCs ([5]), which are the weakly synchronous MSCs such that the number of messages sent per exchange is at most k.

Definition 6.4 (k-exchange)**.** Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC and $k \in \mathbb{N}$. M is a *k-exchange* if M is an exchange and $|SendEv(M)| \leq k$.

Definition 6.5 (Weakly k-synchronous)**.** Let $k \in \mathbb{N}$. $M \in \mathsf{MSC}$ is weakly k-synchronous if it is of the form $M = M_1 \cdot M_2 \cdots M_n$ such that every M_i is a k-exchange.

Example 6.3. MSC M_2 in Fig. 12 is weakly 1-synchronous, as it can be decomposed into three 1-exchanges (the decomposition is depicted by the horizontal dashed lines).

As for weakly synchronous MSCs, the class of weakly k-synchronous MSCs was already shown to be MSO-definable and STW-bounded in [5], and these results still hold even for our definition of MSC. A direct application of Theorem 6.2 shows that, for weakly k-synchronous MSCs, SP is decidable for all communication models.

Proposition 6.3. Let com $\in \{\texttt{asy}, \texttt{p2p}, \texttt{co}, \texttt{mb}, \texttt{onen}, \texttt{nn}\}$. The following problem is decidable: given a communicating system $\mathcal{S}$, is every MSC in $L_{\text{com}}(\mathcal{S})$ weakly k-synchronous?

Proof. The class $\mathcal{C}$ of weakly k-synchronous MSCs is MSO-definable and STW-bounded, therefore the result follows from Theorem 6.2. $\square$

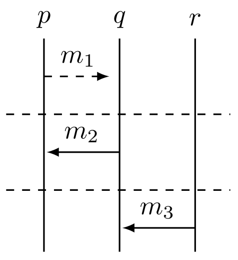


Figure 12: M_2

Existentially bounded MSCs We move now to existentially k-bounded MSCs, first introduced by Lohrey and Muschol [24], that form a relevant class of MSC for extending the Büchi-Elgot-Trakhthenbrot theorem from words to MSCs [17, 16]. Existentially bounded MSCs represent the behavior of systems that can be realized with bounded channels. We stick to the original definition of Lohrey and Muscholl of k-bounded MSCs, where k represents the bound on the number of messages in transit from a given process to another, so that globally there may be up to $k|\mathbb{P}|^2$ in transit.[5] Intuitively, we say that an MSC is existentially k-bounded if it admits a linearization where, at any moment in time, and for all pair of processes p, q, there are no more than k messages in transit from p to q. Such a linearization will be referred to as a *k-bounded linearization.* We give formal definitions below.

Definition 6.6. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$ and $k \in \mathbb{N}$. A linearization $\leadsto$ of M is called *k-bounded* if, for all $e \in \mathit{SendEv}(M)$, with $\lambda(e) = \mathit{send}(p, q, m)$, we have

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

where $\#_A(R, e) = |\{f \in \mathcal{E} \mid (f, e) \in R \text{ and } \lambda(f) \in A\}|$. For instance, $\#_{\mathit{Send}(p,q,\_)}(\leadsto, e)$ denotes the number of send events from p to q that occured before e according to $\leadsto$. Note that, since $\leadsto$ in reflexive, e itself is counted in $\#_{\mathit{Send}(p,q,\_)}(\leadsto, e)$.

Definition 6.7 (Existentially bounded MSC)**.** Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathtt{asy}}$ and $k \in \mathbb{N}$. M is *existentially k-bounded* ($\exists k$-bounded) if it has a k-bounded linearization.

We now look at the definitions of `p2p` $\exists k$-bounded MSCs and causally ordered $\exists k$-bounded, which are quite straightforward.

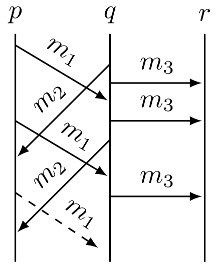


Figure 13: M_3

Example 6.4. MSC M_3 in Fig. 13 is existentially 1-bounded, as witnessed by the linearization !2 !1 !3 ?3 ?1 !1 ?2 !3 ?3 . . . Note that M_3 is not weakly synchronous as we cannot divide it into exchanges.

Definition 6.8. 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 6.9. 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 the other communication models, the definitions are not as straightforward. For instance, the definition of `mb` $\exists k$-bounded MSC should require that there exists a k-bounded linearization that is also a `mb`-linearization, not just any linearization. Recall that an MSC is a `mb`-MSC if it has at least one `mb`-linearization, which represents a sequence of events that can be executed by a `mb` system. Following this intuition, we want one of these `mb`-linearizations to be k-bounded, not just any linearization.

Definition 6.10. An MSC M is `mb` *existentially k-bounded* (`mb`-$\exists k$-bounded) if it has a k-bounded `mb`-linearization.

Definition 6.11. An MSC M is `onen` *existentially k-bounded* (onen-$\exists k$-bounded) if it has a k-bounded `onen`-linearization.

Definition 6.12. An MSC M is `nn` *existentially k-bounded* (nn-$\exists k$-bounded) if it has a k-bounded `nn`-linearization.

We show that each of the $\exists k$-bounded classes of MSCs presented so far is MSO-definable and STW-bounded. We then derive the decidability of *SP* in a similar way to what we did in the proof of Proposition 6.2 for weakly synchronous MSCs.

[5]It may look surprising in our general context to count messages in transit this way, but it can be shown that, up to picking a different value for the bound k, it is equivalent to the possibly more intuitive definition based on counting all messages in transit, independently of their sender and receiver.

MSO-definability We start by investigating the MSO-definability of all the variants of $\exists k$-bounded MSCs, we begin with the most general class of $\exists k$-bounded MSCs. Following the approach taken in [24], 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)[6]:

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

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 be 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.

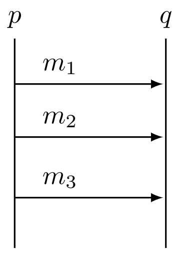


Figure 14: M_4.

Example 6.5. Consider MSC M_4 in Fig 14. Suppose we want to look for a 2-bounded linearization. For $k=2$, we have $?1 \longmapsto_2 !3$; if we find a valid linearization that respect the $\longmapsto_2$ relation, then it is 2-bounded, e.g., !1 !2 ?1 !3 ?2 ?3 (note that ?1 is executed before !3). On the other hand, the linearization !1 !2 !3 ?1 ?2 ?3 is not 2-bounded, since ?1 is executed after !3.

In [24] it was shown that an MSC is $\exists k$-bounded if and only if the relation $\leq_{hb} \cup \longmapsto_k$ is acyclic. Since $\leq_{hb}$ and acyclicity are both MSO-definable, it suffices to find an MSO formula that defines $\longmapsto_k$ to claim the MSO-definability of $\exists k$-bounded MSCs. Unfortunately, $\longmapsto_k$ is not MSO-definable because MSO logic cannot be used to "count" for an arbitrary i. For this reason, we introduce a similar MSO-definable binary relation $\hookrightarrow_k$, and we show that an MSC M is $\exists k$-bounded MSC iff $\leq_{hb} \cup \hookrightarrow_k$ 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):

- 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 for the matched send events among $s_1, \ldots, s_k, s$.

Proposition 6.4. An MSC M is $\exists k$-bounded if and only if $\leq_{hb} \cup \hookrightarrow_k$ is acyclic and, for each channel (p,q), there are at most k unmatched send events.

Proof. ($\Rightarrow$) Suppose M is $\exists k$-bounded, i.e. it has at least one 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 know that $\leq_{hb} \subseteq \rightsquigarrow$, and we will show that also $\hookrightarrow_k \subseteq \rightsquigarrow$. This implies that $\leq_{hb} \cup \hookrightarrow_k$ is acyclic, otherwise we would not be able to find a linearization $\rightsquigarrow$ that respects both $\leq_{hb}$ and $\hookrightarrow_k$. Suppose, by contradiction, that $\hookrightarrow_k \nsubseteq \rightsquigarrow$, 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 k-bounded). We reached a contradiction, hence $\hookrightarrow_k \subseteq \rightsquigarrow$ and $\leq_{hb} \cup \hookrightarrow_k$ is acyclic.

($\Leftarrow$) Suppose $\leq_{hb} \cup \hookrightarrow_k$ is acyclic and, for each channel (p,q), there are at most k unmatched send events. If $\leq_{hb} \cup \hookrightarrow_k$ is acyclic, we are able to find one linearization $\rightsquigarrow$ for the partial order $(\leq_{hb} \cup \hookrightarrow_k)^*$. We show that this linearization is k-bounded. By contradiction, suppose $\rightsquigarrow$ is not 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. Two cases:

[6]Recall that (p, q) is a channel where messages are sent by p and received by q.

- 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 $(\leq_{hb} \cup \hookrightarrow_k)^*$, if $r \hookrightarrow_k s$.

$\square$

According to Proposition 6.4, we can write the MSO formula the defines $\exists k$-bounded MSCs as

$$\Psi_{\exists k} = acyclic(\leq_{hb} \cup \hookrightarrow_k) \wedge \neg \left(\exists s_1 \dots s_{k+1}. s_1 \rightarrow^+ \dots \rightarrow^+ s_{k+1} \ \wedge \ allSends\_pq(k+1) \wedge allUnm\right)$$

$$allSends\_pq(t) = \bigvee_{p \in \mathbb{P}, q \in \mathbb{P}} \ \bigwedge_{s \in s_1, \dots, s_t} \ \bigvee_{a \in Send(p,q,\_)} (\lambda(s) = a)$$

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

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

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

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. Recall that we introduced the $\hookrightarrow_k$ relation because the $\longmapsto_k$ relation introduced in [24] was not MSO-definable for asynchronous communication. However, when considering `p2p` communication but also all of the other communication models, because of the hierarchy shown in Section 5, $\longmapsto_k$ becomes MSO-definable; the FIFO behavior ensures that, for any channel (p,q), the i-th matched send event of p matches with the i-th receive event of q. This allows us to define $r \longmapsto_k s$ as:

$$r \longmapsto_k s = \exists s_1. \dots \exists s_k. (allSends\_p\_q(k) \ \wedge \ s_1 \rightarrow s_2 \rightarrow \dots \rightarrow s_k \rightarrow s \ \wedge \ s_1 \lhd r)$$

Recall that an MSC M is `mb`-$\exists k$-bounded if it has a linearization that is both `mb` and $\exists k$-bounded. A linearization $\rightsquigarrow$ is `mb` if M is `mb` and $\rightsquigarrow$ is a linear extension of the partial order $\preceq_{\mathtt{mb}}$, i.e. $\preceq_{\mathtt{mb}} \subseteq \rightsquigarrow$. A linearization $\rightsquigarrow$ is $\exists k$-bounded if $\longmapsto_k \subseteq \rightsquigarrow$. It follows that a linearization $\longmapsto_k$ is `mb`-$\exists k$-bounded if $(\preceq_{\mathtt{mb}} \cup \longmapsto_k) \ \subseteq \rightsquigarrow$. Such a linerization exists only if $\preceq_{\mathtt{mb}} \cup \longmapsto_k$ is acyclic. If $\preceq_{\mathtt{mb}} \cup \longmapsto_k$ is acyclic, its transitive closure always exists and it is a partial order, hence we are always able to find a linear extension. The characterization for `onen`-$\exists k$-bounded MSCs and `nn`-$\exists k$-bounded is very similar. Summing up:

Proposition 6.5. An MSC M is `mb`-$\exists k$-bounded iff the relation $\preceq_{\mathtt{mb}} \cup \longmapsto_k$ is acyclic.
An MSC M is onen-$\exists k$-bounded iff the relation $\preceq_{\mathtt{1n}} \cup \longmapsto_k$ is acyclic.
An MSC M is nn-$\exists k$-bounded iff the relation $\sqsubset_{\mathtt{nn}} \cup \longmapsto_k$ is acyclic.

The MSO-definability of all the variants of $\exists$k-bounded MSCs directly follows from Proposition 6.5, since all of these relations were shown to be MSO-definable (Section 4).

Special treewidth In [4, 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 class of Concurrent Behaviours with Matching (CBM), but the result is still valid since $\mathsf{MSC_{asy}} \subset \mathsf{CBM}$. The special treewidth of the other classes of $\exists k$-bounded MSCs is also bounded, since they are clearly subclasses of $\exists k$-bounded MSCs.

Universally bounded MSCs An MSC is existentially k-bounded if it has a k-bounded linearization. An MSC is universally k-bounded MSCs if all of its linearizations are k-bounded, hence the name "universally". This class of MSCs was also introduced in [24].

Definition 6.13 (Universally bounded MSC). Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC_{asy}}$ and $k \in \mathbb{N}$. M is *universally k-bounded* ($\forall k$-bounded) if all of its linearizations are k-bounded.

Definition 6.14. 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.

Definition 6.15. 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.

As for the existential case, the definitions for the other communication models are not as straightforward. For instance, the definition of `mb` $\forall k$-bounded MSC should require that all the `mb`-linearizations of the MSC are k-bounded, but we say nothing about linearizations that are not `mb`. The same goes for the FIFO `1−n` and FIFO `n−n` communication models.

Definition 6.16. 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.

Definition 6.17. An MSC M is `onen` *universally k-bounded* (`onen`-$\forall k$-bounded) if it is a `onen`-MSC and all of its `onen`-linearizations are k-bounded.

MSO-definability In this section, we will investigate the MSO-definability of all the variants of universally k-bounded MSCs that we discussed. In [24], it is shown that an MSC M is universally k-bounded if and only if $\longmapsto_k \subseteq \leq_{hb}$. In other words, $r \longmapsto_k s \Rightarrow r \leq_{hb} s$ for any two events r and s. This is equivalent to saying that every linearization $\rightsquigarrow$ of M respects the $\longmapsto_k$ relation, since $\longmapsto_k \subseteq \leq_{hb} \subseteq \rightsquigarrow$. We already saw that $\longmapsto_k$ is not MSO-definable when communication is asynchronous, hence we will use the $\hookrightarrow_k$ relation to give the following alternative characterization of universally k-bounded MSCs.

Proposition 6.6. An MSC M is $\forall k$-bounded if and only if $\hookrightarrow_k \subseteq \leq_{hb}$ and, for each channel (p,q), there are at most k unmatched send events.

Proof. ($\Rightarrow$) Suppose M is $\forall k$-bounded, which by definition means that all of its linearizations are k-bounded. Firstly, notice that every MSC that has more than k unmatched send events in any channel cannot be an $\forall k$-bounded MSC (not even $\exists k$-bounded). By contradiction, suppose that $\hookrightarrow_k \nsubseteq \leq_{hb}$, i.e. there are two events r and s such that $r \hookrightarrow_k s$ and $r \nleq_{hb} s$. If $r \nleq_{hb} s$, we either have that $s \leq_{hb} r$ or that s and r are incomparable w.r.t. $\leq_{hb}$; note that, in both cases, M must have one linearization where s is executed before r[7]. The existence of such a linearization implies that M is not $\forall k$-bounded.

($\Leftarrow$) Suppose $\hookrightarrow_k \subseteq \leq_{hb}$ and, for each channel (p,q), there are at most k unmatched send events. By definition, every linearization $\rightsquigarrow$ of M is such that $\leq_{hb} \subseteq \rightsquigarrow$; it follows that $\hookrightarrow_k \subseteq \rightsquigarrow$, which means that every linearization of M is k-bounded, i.e. M is $\forall k$-bounded. □

[7]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.

It follows that $\texttt{p2p}$-$\forall k$-bounded and $\texttt{co}$-$\forall k$-bounded MSCs are MSO-definable by definition, since $\texttt{p2p}$-MSCs, $\texttt{co}$-MSCs, and universally k-bounded MSCs are all MSO-definable. We already showed that $\longmapsto_k$ is MSO-definable when considering $\texttt{p2p}$ communication. The characterization for the other communication models is similar to that given in [24], but it uses the proper relation for each communication model.

Proposition 6.7. An MSC M is $\texttt{mb}$-$\forall k$-bounded if and only if $\longmapsto_k \subseteq \preceq_{\texttt{mb}}$.
An MSC M is onen-$\forall k$-bounded if and only if $\longmapsto_k \subseteq \preceq_{\texttt{1n}}$.
An MSC M is nn-$\forall k$-bounded if and only if $\longmapsto_k \subseteq \sqsubset_{\texttt{nn}}$.

Proof. We only show it for the $\texttt{mb}$ communication model. The proof for the other communication models works the same way. Consider an MSC M and a $k \in \mathbb{N}$.

($\Leftarrow$) Suppose $\longmapsto_k \subseteq \preceq_{\texttt{mb}}$. For every mailbox linearization $\leadsto$ of M we have that $\preceq_{\texttt{mb}} \subseteq \leadsto$. This implies $\longmapsto_k \subseteq \leadsto$, that is to say every mailbox linearization is k-bounded.

($\Rightarrow$) Suppose M is a $\texttt{mb}$-$\forall k$-bounded MSC. By definition, every mailbox linearization $\leadsto$ of M is k-bounded, i.e. $\longmapsto_k \subseteq \leadsto$, and we have $\preceq_{\texttt{mb}} \subseteq \leadsto$, according to the definition of mailbox linearization. Moreover, we also know that $\preceq_{\texttt{mb}} \cup \longmapsto_k$ is acyclic, since M is $\exists k$-bounded and by definition every $\texttt{mb}$-$\forall k$-bounded MSC is also a $\texttt{mb}$-$\exists k$-bounded MSC. Suppose now, by contradiction, that $\longmapsto_k \nsubseteq \preceq_{\texttt{mb}}$. Thus, there must be at least two events r and s such that $r \longmapsto_k s$ and $r \not\preceq_{\texttt{mb}} s$; we also have $s \not\preceq_{\texttt{mb}} r$ because of the acyclicity of $\preceq_{\texttt{mb}} \cup \longmapsto_k$ (we cannot have the cycle $r \longmapsto_k s \preceq_{\texttt{mb}} r$). Consider a mailbox linearization $\leadsto$ of M, such that $s \leadsto r$. Note that such a mailbox linearization always exists, since r and s are incomparable w.r.t. the partial order $\preceq_{\texttt{mb}}$. This mailbox linearization does not respect $\longmapsto_k$ (because we have $s \leadsto r$ and $r \longmapsto_k s$), so it is not k-bounded. This is a contradiction, since we assumed that M was a $\texttt{mb}$-$\forall k$-bounded MSC. It has to be that $\longmapsto_k \subseteq \preceq_{\texttt{mb}}$. □

Using Proposition 6.7, we can now easily write the MSO formulas that define these variants of universally k-bounded MSCs.

$$\Phi_{mb\text{-}\forall k\text{-}b} = \neg\exists r.\exists s.(r \longmapsto_k s \wedge \neg(r \preceq_{\texttt{mb}} s))$$
$$\Phi_{onen\text{-}\forall k\text{-}b} = \neg\exists r.\exists s.(r \longmapsto_k s \wedge \neg(r \preceq_{\texttt{1n}} s))$$
$$\Phi_{nn\text{-}\forall k\text{-}b} = \neg\exists r.\exists s.(r \longmapsto_k s \wedge \neg(r \sqsubset_{\texttt{nn}} s))$$

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.

7 Conclusion

We studied seven different communication models and their corresponding classes of MSCs, we characterized each of these classes with MSO logic, and draw the hierarchy of these communication models. These results were then applied to deal with (un)decidability of some verification problems.

To refine the picture, we could consider other logics like FO+TC or LCPDL, and other communication models, such as the FIFO-based implementation of the causally ordered communication model proposed in [26], which we expect to sit somewhere between mailbox and causally ordered within the hierarchy that we presented. Moreover, as shown by Fig. 10, the decidability of the synchronizability problem for weakly synchronous MSCs and fully asynchronous communication is not entailed by our techniques, and could be studied in future works.

References

[1] C. Aiswarya, Paul Gastin, and K. Narayan Kumar. Verifying communicating multi-pushdown systems via split-width. In *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014*, volume 8837 of *Lecture Notes in Computer Science*, pages 1–17, Sydney, Australia, 2014. Springer.

[2] Ozalp Babaoglu and Keith Marzullo. Consistent global states of distributed systems: Fundamental concepts and mechanisms. *Distributed Systems*, 53, 1993.

[3] Samik Basu and Tevfik Bultan. On deciding synchronizability for asynchronously communicating systems. *Theor. Comput. Sci.*, 656:60–75, 2016.

[4] Benedikt Bollig and Paul Gastin. Non-sequential theory of distributed systems. *CoRR*, abs/1904.06942:1–82, 2019.

[5] Benedikt Bollig, Cinzia Di Giusto, Alain Finkel, Laetitia Laversa, Étienne Lozes, and Amrita Suresh. A unifying framework for deciding synchronizability. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021*, volume 203 of *LIPIcs*, pages 14:1–14:18, Virtual Conference, 2021. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[6] Benedikt Bollig, Cinzia Di Giusto, Alain Finkel, Laetitia Laversa, Étienne Lozes, and Amrita Suresh. A unifying framework for deciding synchronizability (extended version). Technical report, HAL, 2021. available at `https://hal.archives-ouvertes.fr/hal-03278370/document`.

[7] Ahmed Bouajjani, Constantin Enea, Kailiang Ji, and Shaz Qadeer. On the completeness of verifying message passing programs under bounded asynchrony. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, July 14-17, 2018, Proceedings, Part II*, volume 10982 of *Lecture Notes in Computer Science*, pages 372–391, Oxford, UK, 2018. Springer.

[8] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.

[9] Bernadette Charron-Bost, Friedemann Mattern, and Gerard Tel. Synchronous, asynchronous, and causally ordered communication. *Distributed Comput.*, 9(4):173–191, 1996.

[10] Florent Chevrou, Aurélie Hurault, and Philippe Quéinnec. On the diversity of asynchronous communication. *Formal Aspects Comput.*, 28(5):847–879, 2016.

[11] Bruno Courcelle. Special tree-width and the verification of monadic second-order graph properties. In *FSTTCS*, volume 8 of *LIPIcs*, pages 13–29, Chennai, India, 2010. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[12] Flaviu Cristian and Christof Fetzer. The timed asynchronous distributed system model. *IEEE Transactions on Parallel and Distributed systems*, 10(6):642–657, 1999.

[13] Aiswarya Cyriac, Paul Gastin, and K. Narayan Kumar. MSO decidability of multi-pushdown systems via split-width. In Maciej Koutny and Irek Ulidowski, editors, *Concurrency Theory - 23rd International Conference, CONCUR 2012, September 4-7, 2012. Proceedings*, volume

7454 of *Lecture Notes in Computer Science*, pages 547–561, Newcastle upon Tyne, UK, 2012. Springer.

[14] Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys (CSUR)*, 36(4):372–421, 2004.

[15] Cinzia Di Giusto, Laetitia Laversa, and Étienne Lozes. On the k-synchronizability of systems. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Proceedings*, volume 12077 of *Lecture Notes in Computer Science*, pages 157–176, Virtual conference, 2020. Springer.

[16] Blaise Genest, Dietrich Kuske, and Anca Muscholl. On communicating automata with bounded channels. *Fundamenta Informaticae*, 80(1-3):147–167, 2007.

[17] Blaise Genest, Anca Muscholl, and Dietrich Kuske. A Kleene theorem for a class of communicating automata with effective algorithms. In *International Conference on Developments in Language Theory*, pages 30–48, Auckland, New Zealand, 2004. Springer, Springer.

[18] ITU-T. Recommendation itu-t z.120: Message sequence chart (msc). Technical report, International Telecommunication Union, Geneva, February 2011.

[19] Bernhard Kragl, Shaz Qadeer, and Thomas A. Henzinger. Synchronizing the asynchronous. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018*, volume 118 of *LIPIcs*, pages 21:1–21:17, Beijing, China, 2018. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

[20] Ajay D Kshemkalyani and Mukesh Singhal. Necessary and sufficient conditions on information for causal message ordering and their optimal implementation. *Distributed Computing*, 11(2):91–111, 1998.

[21] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[22] Julien Lange and Nobuko Yoshida. Verifying asynchronous interactions via communicating session automata. In *Computer Aided Verification - 31st International Conference, CAV 2019, July 15-18, 2019, Proceedings, Part I*, pages 97–117, New York City, NY, USA, 2019. Springer.

[23] Richard J. Lipton. Reduction: A method of proving properties of parallel programs. *Commun. ACM*, 18(12):717–721, 1975.

[24] Markus Lohrey and Anca Muscholl. Bounded MSC communication. In Mogens Nielsen and Uffe Engberg, editors, *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, April 8-12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 295–309, Grenoble, France, 2002. Springer.

[25] P. Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, January 26-28, 2011*, pages 283–294, Austin, TX, USA, 2011. ACM.

[26] Friedemann Mattern and Stefan Fünfrocken. A non-blocking lightweight implementation of causal order message delivery. In Kenneth P. Birman, Friedemann Mattern, and André Schiper, editors, *Theory and Practice in Distributed Systems, International Workshop, September 5-9, 1994, Selected Papers*, volume 938 of *Lecture Notes in Computer Science*, pages 197–213, Dagstuhl Castle, Germany, 1994. Springer.

[27] Larry L Peterson, Nick C Buchholz, and Richard D Schlichting. Preserving and using context information in interprocess communication. *ACM Transactions on Computer Systems (TOCS)*, 7(3):217–246, 1989.

[28] Michel Raynal. Communication and agreement abstractions for fault-tolerant asynchronous distributed systems. *Synthesis Lectures on Distributed Computing Theory*, 1(1):1–273, 2010.

[29] André Schiper, Jorge Eggli, and Alain Sandoz. A new algorithm to implement causal ordering. In Jean-Claude Bermond and Michel Raynal, editors, *Distributed Algorithms, 3rd International Workshop, September 26-28, 1989, Proceedings*, volume 392 of *Lecture Notes in Computer Science*, pages 219–232, Nice, France, 1989. Springer.

[30] Gerard Tel. *Introduction to distributed algorithms.* Cambridge university press, Cambridge, 2000.

[31] Robbert van Renesse. Causal controversy at le mont st.-michel. *ACM SIGOPS Oper. Syst. Rev.*, 27(2):44–53, 1993.

[32] Klaus von Gleissenthall, Rami Gökhan Kici, Alexander Bakst, Deian Stefan, and Ranjit Jhala. Pretend synchrony: synchronous verification of asynchronous distributed programs. *PACMPL*, 3(POPL):59:1–59:30, 2019.

A MSO-definable properties

In this section we give MSO formulas for some MSO-definable properties that are used throughout the report.

Transitive Closure Given a binary relation R, we can express its reflexive transitive closure R^* in MSO as

$$x\, R^*\, y = \forall X.(x \in X \;\wedge\; \mathit{forward\_closed}(X)) \implies y \in X$$

$$\mathit{forward\_closed}(X) = \forall z.\forall t.(z \in X \;\wedge\; z\, R\, t) \implies t \in X$$

The transitive (but not necessarily reflexive) closure of R can also be expressed as

$$x\, R^+\, y = \forall X.\; \big(\forall z, t\; (z \in X \cup \{x\} \wedge z\, R\, t) \implies t \in X\big) \implies y \in X$$

Acyclicity Given a binary relation R, we can use MSO to express the acyclicity of R, or equivalently, the fact that its transitive closure R^+ is irreflexive.

$$\Phi_{acyclic} = \neg\exists x.(x\, R^+\, x).$$

B Additional material for Section 6

B.1 Communicating finite state machines

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 channels from $\mathbb{C}$.

Definition B.1. A *system of communicating finite state machines* over the set $\mathbb{P}$ of rocesses and the set $\mathbb{M}$ of messages is a tuple $\mathcal{S} = (A_p)_{p\in\mathbb{P}}$. For each $p \in \mathbb{P}$, $A_p = (Loc_p, \delta_p, \ell_p^0)$ is a finite transition system where Loc_p is a finite set of local (control) states, $\delta_p \subseteq Loc_p \times \Sigma_p \times Loc_p$ is the transition relation, and $\ell_p^0 \in Loc_p$ 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 $a \in \mathit{Send}(\_,\_,m) \cup \mathit{Rec}(\_,\_,m)$.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC. A *run* of $\mathcal{S}$ on M is a mapping $\rho : \mathcal{E} \to \bigcup_{p\in\mathbb{P}} \delta_p$ 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 $\mathit{source}(\rho(e)) = \ell_p^0$.

We write $L_{\mathsf{asy}}(\mathcal{S})$ to denote the set of MSCs M that admit a run of $\mathcal{S}$. Intuitively, $L_{\mathsf{asy}}(\mathcal{S})$ is the set of all asynchronous behaviors of $\mathcal{S}$.

B.2 Proof of Theorem 6.1

Theorem 6.1. Let com $\in \{\mathtt{asy}, \mathtt{co}, \mathtt{p2p}, \mathtt{mb}, \mathtt{onen}, \mathtt{nn}, \mathtt{rsc}\}$ and $k \geq 1$ be fixed. Then the following problem is decidable: given a system $\mathcal{S}$ and a MSO specification φ, decide whether $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi)$.

Proof. Let com, $\mathcal{C}$, $\mathcal{S}$, and φ be fixed. We showed in Section 4 that there is a MSO formula φ_{com} that defines $\mathsf{MSC}_{\mathrm{com}}$. There is also a MSO formula $\varphi_{\mathcal{S}}$ such that $L_{\mathtt{asy}}(\mathcal{S}) = L(\varphi_{\mathcal{S}})$.[8] Putting everything together, we have

$$\begin{aligned}
& L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-stw}} \subseteq L(\varphi) \\
\iff \quad & L_{\mathtt{asy}}(\mathcal{S}) \cap \mathsf{MSC}_{\mathrm{com}} \cap \mathsf{MSC}^{k\text{-stw}} \subseteq L(\varphi) \\
\iff \quad & L(\varphi_{\mathcal{S}}) \cap L(\varphi_{\mathrm{com}}) \cap \mathsf{MSC}^{k\text{-stw}} \subseteq L(\varphi) \\
\iff \quad & \mathsf{MSC}^{k\text{-stw}} \subseteq L(\varphi \vee \neg\varphi_{\mathrm{com}} \vee \neg\varphi_{\mathcal{S}})\,.
\end{aligned}$$

The latter is decidable by Courcelle's theorem [11]. ◻

B.3 Proof of Theorem 6.2

In order to prove Theorem 6.2, we first need to introduce some concepts and give preliminary proofs.

Definition B.2 (Prefix)**.** Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$ and consider $E \subseteq \mathcal{E}$ such that E is $\leq_{hb}$-*downward-closed*, i.e, for all $(e, f) \in \leq_{hb}$ such that $f \in E$, we also have $e \in E$. Then, the MSC $M' = (E, \rightarrow \cap (E \times E), \lhd \cap (E \times E), \lambda')$, where λ' is the restriction of $\mathcal{E}$ to E, is called a *prefix* of M.

If we consider a set E that is $\preceq_{\mathtt{1n}}$-*downward-closed*, we call M' a $\mathtt{onen}$-*prefix*. If the set E is $\sqsubset_{\mathtt{nn}}$-*downward-closed*, we call M' a $\mathtt{nn}$-*prefix*. Note that every $\mathtt{onen}$ or $\mathtt{nn}$-prefix is also a prefix, since $\leq_{hb} \subseteq \preceq_{\mathtt{1n}}$ and $\leq_{hb} \subseteq \sqsubset_{\mathtt{nn}}$.

Note that the empty MSC is a prefix of M. We denote the set of prefixes of M by $\mathit{Pref}(M)$, whereas $\mathit{Pref}_{\mathtt{onen}}(M)$ and $\mathit{Pref}_{\mathtt{nn}}(M)$ are used for the FIFO $\mathtt{1{-}n}$ and the FIFO $\mathtt{n{-}n}$ variants, respectively. This is extended to sets $L \subseteq \mathsf{MSC}$ as expected, letting $\mathit{Pref}(L) = \bigcup_{M \in L} \mathit{Pref}(M)$.

Proposition B.1. For com $\in \{\mathtt{asy}, \mathtt{p2p}, \mathtt{co}, \mathtt{mb}\}$, every prefix of a com-MSC is a com-MSC.

Proof. For com $= \mathtt{asy}$ it is true by definition. For com $= \{\mathtt{p2p}, \mathtt{mb}\}$ it was already shown to be true in [5], so we just consider com $= \mathtt{co}$. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathtt{co}}$ 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 $\mathtt{co}$-MSC. There must be two distinct $s, s' \in \mathcal{E}_0$ such that $\lambda(s) = \mathit{Send}(\_, q, \_)$, $\lambda(s') = \mathit{Send}(\_, q, \_)$, $s \leq_{hb}^{(M_0)} s'$ 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 $\mathtt{co}$-a 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. ◻

Note that this proposition is not true for the FIFO $\mathtt{1{-}n}$ and the FIFO $\mathtt{n{-}n}$ communication models. Fig. 9 shows an example of $\mathtt{nn}$-MSC with a prefix that is neither a $\mathtt{nn}$-MSC nor a $\mathtt{onen}$-MSC.

Proposition B.2. Every $\mathtt{onen}$-prefix of a $\mathtt{onen}$-MSC is a $\mathtt{onen}$-MSC.

Proof. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathtt{onen}}$ and let $M_0 = (\mathcal{E}_0, \rightarrow_0, \lhd_0, \lambda_0)$ be a $\mathtt{onen}$-prefix of M, where $\mathcal{E}_0 \subseteq \mathcal{E}$. Firstly, the $\preceq_{\mathtt{1n}}$-downward-closeness of $\mathcal{E}_0$ guarantees that M_0 is still an MSC. We need to prove that it is a $\mathtt{onen}$-MSC. By contradiction, suppose that M_0 is not a $\mathtt{onen}$-MSC. Then,

[8] The formula simply encodes the existence of a run of $\mathcal{S}$ on the MSC using a MSO variable X_l for each control state l, with the meaning that X_l is the set of events before which the local communicating automaton was in state l. See [4, Theorem 3.4] for a detailed proof.

there are distinct $e, f \in \mathcal{E}_0$ such that $e \preceq_{\mathtt{1n}}^{(M_0)} f \preceq_{\mathtt{1n}}^{(M_0)} e$, where $\preceq_{\mathtt{1n}}^{(M_0)} = (\rightarrow_0 \cup \lhd_0 \cup \sqsubset_{\mathtt{1n}}^{(M_0)})^*$. As $\mathcal{E}_0 \subseteq \mathcal{E}$, we have that $\rightarrow_0 \subseteq \rightarrow$, $\lhd_0 \subseteq \lhd$, $\sqsubset_{\mathtt{1n}}^{(M_0)} \subseteq \sqsubset_{\mathtt{1n}}$. Clearly, $\preceq_{\mathtt{1n}}^{(M_0)} \subseteq \preceq_{\mathtt{1n}}$, so $e \preceq_{\mathtt{1n}} f \preceq_{\mathtt{1n}} e$. This implies that M is not a $\mathtt{onen}$-MSC, because $\preceq_{\mathtt{1n}}$ is cyclic, which is a contradiction. Hence M_0 is a $\mathtt{onen}$-MSC. □

Proposition B.3. Every $\mathtt{nn}$-prefix of a $\mathtt{nn}$-MSC is a $\mathtt{nn}$-MSC.

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

The next lemma is about the prefix closure of a communicating system and it follows from Proposition B.1.

Proposition B.4. For all com $\in \{\mathtt{asy}, \mathtt{p2p}, \mathtt{mb}, \mathtt{co}\}$, $L_{\text{com}}(\mathcal{S})$ is prefix-closed: $\mathit{Pref}(L_{\text{com}}(\mathcal{S})) \subseteq L_{\text{com}}(\mathcal{S})$.

Similar results also hold for the FIFO $\mathtt{1}-\mathtt{n}$ and FIFO $\mathtt{n}-\mathtt{n}$ communication models.

Proposition B.5. $L_{\mathtt{onen}}(\mathcal{S})$ is $\mathtt{onen}$-prefix-closed: $\mathit{Pref}_{\mathtt{onen}}(L_{\mathtt{onen}}(\mathcal{S})) \subseteq L_{\mathtt{onen}}(\mathcal{S})$.

Proof. Given a system $\mathcal{S}$, we have that $L_{\mathtt{onen}}(\mathcal{S}) = L_{\mathtt{p2p}}(\mathcal{S}) \cap \mathsf{MSC}_{\mathtt{onen}}$. Note that, because of how we defined a $\mathtt{onen}$-prefix, we have that $\mathit{Pref}_{\mathtt{onen}}(L_{\mathtt{onen}}(\mathcal{S})) = \mathit{Pref}(L_{\mathtt{onen}}(\mathcal{S})) \cap \mathsf{MSC}_{\mathtt{onen}}$. Moreover, $\mathit{Pref}(L_{\mathtt{onen}}(\mathcal{S})) \subseteq \mathit{Pref}(L_{\mathtt{p2p}}(\mathcal{S}))$, and $\mathit{Pref}(L_{\mathtt{onen}}(\mathcal{S})) \subseteq L_{\mathtt{p2p}}(\mathcal{S})$ for Proposition B.4. Putting everything together, $\mathit{Pref}_{\mathtt{onen}}(L_{\mathtt{onen}}(\mathcal{S})) \subseteq L_{\mathtt{p2p}}(\mathcal{S}) \cap \mathsf{MSC}_{\mathtt{onen}} = L_{\mathtt{onen}}(\mathcal{S})$. □

Proposition B.6. $L_{\mathtt{nn}}(\mathcal{S})$ is $\mathtt{nn}$-prefix-closed:

$$\mathit{Pref}_{\mathtt{nn}}(L_{\mathtt{nn}}(\mathcal{S})) \subseteq L_{\mathtt{nn}}(\mathcal{S}).$$

Proof. Given a system $\mathcal{S}$, we have that $L_{\mathtt{nn}}(\mathcal{S}) = L_{\mathtt{p2p}}(\mathcal{S}) \cap \mathsf{MSC}_{\mathtt{nn}}$. Note that, because of how we defined a $\mathtt{nn}$-prefix, we have that $\mathit{Pref}_{\mathtt{nn}}(L_{\mathtt{nn}}(\mathcal{S})) = \mathit{Pref}(L_{\mathtt{nn}}(\mathcal{S})) \cap \mathsf{MSC}_{\mathtt{nn}}$. Moreover, $\mathit{Pref}(L_{\mathtt{nn}}(\mathcal{S})) \subseteq \mathit{Pref}(L_{\mathtt{p2p}}(\mathcal{S}))$, and $\mathit{Pref}(L_{\mathtt{nn}}(\mathcal{S})) \subseteq L_{\mathtt{p2p}}(\mathcal{S})$ for Proposition B.4. Putting everything together, $\mathit{Pref}_{\mathtt{nn}}(L_{\mathtt{nn}}(\mathcal{S})) \subseteq L_{\mathtt{p2p}}(\mathcal{S}) \cap \mathsf{MSC}_{\mathtt{nn}} = L_{\mathtt{nn}}(\mathcal{S})$. □

In this last part we prove a series of statements to conclude that, when we have a STW-bounded class $\mathcal{C}$, the synchronizability problem can be reduced to bounded model-checking, which we showed to be decidable in Theorem 6.1.

Proposition B.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 $(\mathit{Pref}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset$.

Proof. Already proved in [5], but we adapt the proof to our setting. 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 $\leq_{hb}$-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 $\leq_{hb}$-maximal events. Let e be a $\leq_{hb}$-maximal event of M', 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

$k+1$ colors for M''. Let us design a winning strategy with $k+3$ colors 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 $k+3$ colors). □

We have similar results also for the FIFO `1−n` and FIFO `n−n` communication models.

Proposition B.8. Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. For all $M \in \mathsf{MSC}_{\mathtt{onen}} \setminus \mathcal{C}$, we have

$$(\mathit{Pref}_{\mathtt{onen}}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset.$$

Proof. Let k and $\mathcal{C}$ be fixed, and let $M \in \mathsf{MSC}_{\mathtt{onen}} \setminus \mathcal{C}$ be fixed. If the empty MSC is not in $\mathcal{C}$, then we are done, since it is a valid `onen`-prefix of M and it is in $\mathsf{MSC}^{(k+2)\text{-stw}} \setminus \mathcal{C}$. Otherwise, let $M' \in \mathit{Pref}_{\mathtt{onen}}(M) \setminus \mathcal{C}$ such that, for all $\preceq_{\mathtt{1n}}$-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 $\preceq_{\mathtt{1n}}$-maximal events. Let e be $\preceq_{\mathtt{1n}}^{(M')}$-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 Proposition B.7. □

Proposition B.9. Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. For all $M \in \mathsf{MSC}_{\mathtt{nn}} \setminus \mathcal{C}$, we have

$$(\mathit{Pref}_{\mathtt{nn}}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset.$$

Proof. Let k and $\mathcal{C}$ be fixed, and let $M \in \mathsf{MSC}_{\mathtt{nn}} \setminus \mathcal{C}$. If the empty MSC is not in $\mathcal{C}$, then we are done, since it is a valid `nn`-prefix of M and it is in $\mathsf{MSC}^{(k+2)\text{-stw}} \setminus \mathcal{C}$. Otherwise, let $M' \in \mathit{Pref}_{\mathtt{nn}}(M) \setminus \mathcal{C}$ such that, for all $\sqsubset_{\mathtt{nn}}^{(M)}$-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 $\sqsubset_{\mathtt{nn}}^{(M)}$-maximal events. Let e be $\sqsubset_{\mathtt{nn}}^{(M')}$-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 Proposition B.7. □

The following proposition is the last ingredient that we need to prove Theorem 6.2.

Proposition B.10. Let $\mathcal{S}$ be a communicating system, com $\in$ {`asy`, `p2p`, `co`, `mb`, `onen`, `nn`, `rsc`}, $k \in \mathbb{N}$, and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. Then, $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$ iff $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$.

Proof. For com $\in$ {`asy`, `p2p`, `co`, `mb`}, the proposition follows from Proposition B.7. For com $\in$ {`onen`, `nn`}, it follows from Proposition B.8 and Proposition B.9, respectively. □

Theorem 6.2. For any com $\in$ {`asy`, `p2p`, `co`, `mb`, `onen`, `nn`} and for all class of MSCs $\mathcal{C}$, if $\mathcal{C}$ is STW-bounded and MSO-definable, then the (com, $\mathcal{C}$)-synchronizability problem is decidable.

Proof. According to Proposition B.10, we have $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$ iff $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$. The latter is decidable according to Theorem 6.1. □

B.4 Proof of Proposition 6.1

Proposition 6.1. The following problem is undecidable: given a communicating system $\mathcal{S}$, is every MSC in $L_{\mathsf{co}}(\mathcal{S})$ weakly synchronous?

The proof is very similar to the one of [6, Theorem 20] for the `p2p` case. We do the same reduction from the Post correspondence problem. The original proof considered a `p2p` system $\mathcal{S}$ with four machines (P1, P2, V1, V2), where we have unidirectional communication channels from provers (P1 and P2) to verifiers (V1 and V2). In particular notice that all the possible behaviors of $\mathcal{S}$ are causally ordered, i.e. $L_{\mathtt{p2p}}(\mathcal{S}) \subseteq \mathsf{MSC}_{\mathsf{co}}$; 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[9], which implies that causal ordering is already ensured by any possible `p2p` behavior of $\mathcal{S}$. The rest of the proof is identical to the `p2p` case.

[9]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.