A Partial Order View of Message-Passing Communication Models

CINZIA DI GIUSTO, Université Côte d'Azur, CNRS, I3S, France
DAVIDE FERRÉ, Université Côte d'Azur, CNRS, I3S, France
LAETITIA LAVERSA, Université Côte d'Azur, CNRS, I3S, France
ETIENNE LOZES, Université Côte d'Azur, CNRS, I3S, France

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 concurrent behaviors. We also show that all the communication models we consider can be axiomatized in the monadic second order logic, and may therefore benefit from several bounded verification techniques based on bounded special treewidth.

CCS Concepts: • **Theory of computation** → **Verification by model checking**; **Modal and temporal logics**; **Distributed computing models**.

Additional Key Words and Phrases: Asynchronous Communication, Message-passing, Monadic Second order Logic, Bounded Model Checking, Bounded Treewidth

ACM Reference Format:
Cinzia Di Giusto, Davide Ferré, Laetitia Laversa, and Etienne Lozes. 2023. A Partial Order View of Message-Passing Communication Models. *Proc. ACM Program. Lang.* 7, POPL, Article 55 (January 2023), 27 pages. https://doi.org/10.1145/3571248

1 INTRODUCTION

Reasoning about distributed message-passing applications is notoriously hard. One reason is that the communication architecture 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 (or rendez-vous) communication, send and receive events are viewed as a single event, i.e., a receive and the corresponding send event happen simultaneously. The idea behind asynchronous communication, instead, is to decouple send and receive events, so that a receive can happen indefinitely after the corresponding send. A prominent model of systems with asynchronous communication is the one of communicating finite state machines, where each agent is a finite

Authors' addresses: Cinzia Di Giusto, Université Côte d'Azur, CNRS, I3S, France, cinzia.di-giusto@univ-cotedazur.fr; Davide Ferré, Université Côte d'Azur, CNRS, I3S, France, ferre@i3s.unice.fr; Laetitia Laversa, Université Côte d'Azur, CNRS, I3S, France, laetitia.laversa@univ-cotedazur.fr; Etienne Lozes, Université Côte d'Azur, CNRS, I3S, France, etienne.lozes@univ-cotedazur.fr.

2475-1421/2023/1-ART55
https://doi.org/10.1145/3571248

state machine that can push and pop messages from FIFO queues. Despite its simplicity, most of decision problems concerning this model are undecidable [Brand and Zafiropulo 1983]. For this reason, several model-checking tools, such as SPIN [Holzmann 2004], assume that communication buffers are bounded in order to keep a finite set of configurations. To overcome this limitation, several bounded model-checking techniques for finite state machines have been proposed, including universal and existential buffer boundedness [Genest et al. 2004], bounded context-switch [Torre et al. 2009], or k-synchronizability [Bouajjani et al. 2018], as well as some approaches based on over-approximation [Botbol et al. 2017; Heußner et al. 2009]. One problem of interest, in the case of bounded model-checking techniques, is the completeness of the analysis, i.e., whether the system behavior is completely captured by the bounded semantics. Recently, Bollig *et al.* [Bollig et al. 2021a] proposed a general framework that helps to develop new bounded model-checking techniques for which the completeness problem is ensured to be decidable. While this framework is parametric in the bounded model-checking techniques under consideration, it is quite rigid in the communication model that is assumed among all participants.

In this paper, we show how to further generalize this framework to handle several models of communications. To do so, we first clarify and classify some of these communication models. On the 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* [Lamport 1978], for the correctness of Lamport's distributed mutual exclusion algorithm (see also [van Renesse 1993] for more examples). 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. Such communication models have been considered in more recent works (for instance in [Basu and Bultan 2016]) and have caused some confusion, specifically regarding the difference between causal ordering and mailbox [Bouajjani et al. 2018; Di Giusto et al. 2020].

The classification and axiomatization of communication models for large-scale distributed systems received great attention in the late 90s [Charron-Bost et al. 1996], while the local-scale communication models have only started to be investigated quite recently by Chevrou *et al.* [Chevrou et al. 2016], focusing on a *sequential* view of the behaviors of message-passing applications (to be detailed below). At the same time, several works [Bouajjani et al. 2018; Kragl et al. 2018; Lange and Yoshida 2019; von Gleissenthall et al. 2019] recently addressed the verification of asynchronous message-passing applications by reduction to their synchronous semantics (see also [Lipton 1975] for a seminal work on these questions). These results strongly rely on the ability to safely approximate an asynchronous communication model with 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, interleaving-based, hierarchy established by Chevrou *et al.* [Chevrou et al. 2016], where a communication model is represented by a class of sequential executions. We revisit this hierarchy taking a "non-sequential" point of view: we consider only the direct causality between messages, which leads to a partial order point of view. We define a communication model as a class of *Message Sequence Charts* (MSCs in the following). MSCs are a graphical representation of computations of distributed systems, and they are a simplified version of the ITU recommendation [ITU-T 2011]. 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; black 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

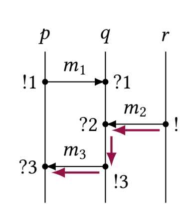


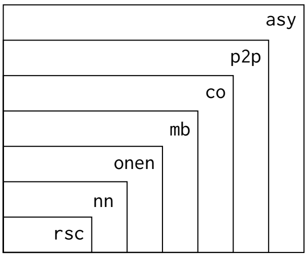
Fig. 1. An MSC.

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 *concurrent*, meaning that the 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 its 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 establishes a causal relation between events. Fig. 1 shows an example of causal path (the red arrows) between the events !2 and ?3.

In this work we interpret communication models as classes of MSCs. This partial order 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 the partial order view and the happens-before relation. It is also more accurate: for instance, as we show in Section 5, some inclusions between communication models are missed by the sequential hierarchy. Such inclusions are interesting to know; for instance, it can be useful to know that if a system is safe when running on mailbox communication, it will also be safe when running on causally ordered communication, but that the converse does not hold.

Our contributions are the following:

- We review 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. Notice that the `asy` (also known as bag) model, `co`, `p2p`, and `rsc` are well-established standards. They have been heavily considered in theoretical aspects of distributed computing and, as already mentioned, they are required to establish the correctness of several distributed algorithms. They are also prominent in applications, because most of them are simple to implement, with the exception of causal ordering. `mb` is a standard choice of communication; it is native in Erlang, but more generally concurrent programs based on the "actor model" use it (e.g., it is a common design pattern used in Go programming). Moreover, it is a cheap "implementation" of causal ordering (while being more restrictive than causal ordering), so it is a natural option if some guarantees enforced by causal ordering are desired, but the full flexibility of causal ordering is not needed. FIFO `1−n` captures, among others, the "job stealing" design pattern for parallelization and finally, FIFO `n−n` captures systems where all participants communicate among them through a "global bus".
- From these definitions, we deduce a new 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 hierarchy, where FIFO `1−n` and mailbox are incomparable. The comparison between the FIFO `1−n` and mailbox classes is non-trivial in our partial order setting, and it motivates the introduction of several alternative characterizations of these communication models.
- We show that all the communication models can be axiomatized in monadic second order logic (MSO) over MSCs. Interestingly, communication models for large-scale distributed systems are quite easy to axiomatize while those for local-scale systems are much more



(a) The hierarchy of MSC classes.

	Weakly sync	Weakly k-sync	∃k bounded	∀k bounded
asy	unbounded STW	✓	✓	✓
p2p	✗ [1]	✓ [1]	✓ [1]	✓ [1]
co	✗	✓	✓	✓
mb	✓ [1]	✓ [1]	✓ [1]	✓ [1]
onen	✓	✓	✓	✓
nn	✓	✓	✓	✓

(b) (Un)decidability results for the synchronizability problems, [1] indicates that the result was shown in [Bollig et al. 2021a].

Fig. 2. Main contributions

involved. Indeed they are easy to define by means of an operational semantics involving FIFO queues, but the axiomatization is rather subtle for mb 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 paper 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, more classical definition for some of these communication models. The goal is to show the relation between the sequential view of Chevrou *et al.* and the partial order one we adopt. Section 4 characterizes the classes of MSCs via MSO logic. In Section 5, we compare all the communication models and show our main result: a strict 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 paper in correspondence to specific notions. A version of this paper with some additional material and all the proofs is available at [Di Giusto et al. 2022].

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 the paper. 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 $send(p, q, m)$; it is executed by process p and sends message m to process q. The corresponding *receive action* executed by q is $rec(p, q, m)$. We write $Send(p, q, \_)$ to denote the set $\{send(p, q, m) \mid m \in \mathbb{M}\}$, and $Rec(p, q, \_)$ for the set $\{rec(p, q, m) \mid m \in \mathbb{M}\}$. Similarly, for $p \in \mathbb{P}$, we set $Send(p, \_, \_) = \{send(p, q, m) \mid q \in \mathbb{P} \text{ and } m \in \mathbb{M}\}$, etc. Moreover, $\Sigma_p = Send(p, \_, \_) \cup$

$Rec(\_, p, \_)$ 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 $send(p, q, m_i)$ (resp. $rec(p, q, m_i)$).

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 [Chevrou et al. 2016] and [Basu and Bultan 2016]). It is also referred to as NON-FIFO (cfr. [Charron-Bost et al. 1996]). 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 is in Fig. 3a; 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 two processes p, q and a message m such that $\lambda(e) = send(p, q, m)$ and $\lambda(f) = rec(p, q, m)$.

(2b) for all $f \in \mathcal{E}$, with $\lambda(f) = rec(p, q, m)$, there is exactly one $e \in \mathcal{E}$ such that $e \lhd f$.

(2c) for all $e \in \mathcal{E}$ such that $\lambda(e) = send(p, q, m)$, there is at most one $f \in \mathcal{E}$ such that $e \lhd f$.

(3) The *happens-before* relation[1] $\leq_{hb}$, defined by $(\rightarrow \cup \lhd)^*$, is a partial order on $\mathcal{E}$.

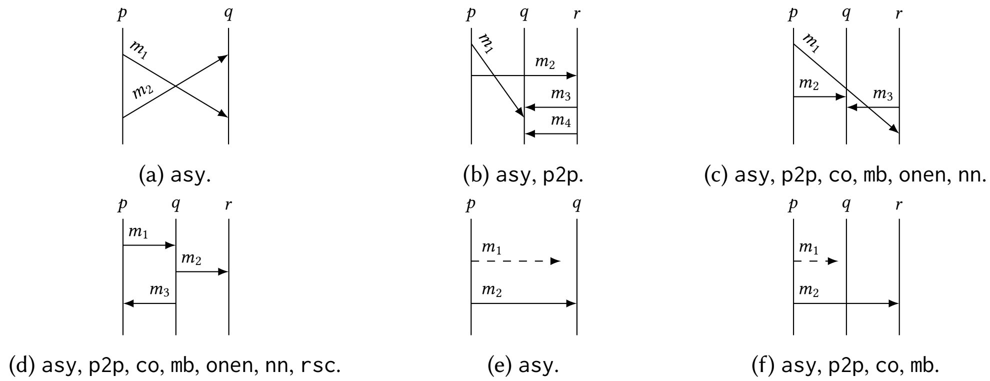
If, for two events e and f, we have that $e \leq_{hb} f$, we say that there is a *causal path* between e and f. Note that the same message m may occur repeatedly on a given MSC, hence the λ labelling function. In most of our examples, we avoid repeating twice a same message, hence events and actions 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 $SendEv(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a send action}\}$, $RecEv(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a receive action}\}$, $Matched(M) = \{e \in \mathcal{E} \mid \text{there is } f \in \mathcal{E} \text{ such that } e \lhd f\}$, and $Unm(M) = \{e \in \mathcal{E} \mid \lambda(e) \text{ is a send action and there is no } f \in \mathcal{E} \text{ such that } e \lhd f\}$.

Example 2.2. 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 $!2 \leq_{hb} ?3$. The set of actions is $\Sigma = \{send(p, q, m_1), send(r, q, m_2), send(q, p, m_3), rec(p, q, m_1), rec(r, q, m_2), rec(q, p, m_3)\}$, or, using the lightweight notation, $\Sigma = \{!1, !2, !3, ?1, ?2, ?3\}$.

Intuitively, a linearization represents a possible scheduling of the events of the distributed system. 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 $\leq_{hb} \subseteq \rightsquigarrow$. In other words, a linearization of M represents a possible way to

[1] This relation was introduced in [Lamport 1978], and is also referred to as the *happened before* relation, or sometimes *causal relation* or *causality relation*, e.g. in [Bouajjani et al. 2018; Charron-Bost et al. 1996] .



(a) asy. (b) asy, p2p. (c) asy, p2p, co, mb, onen, nn.

(d) asy, p2p, co, mb, onen, nn, rsc. (e) asy. (f) asy, p2p, co, mb.

Fig. 3. Examples of MSCs for various communication models.

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.

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* $M_1 \cdot M_2$ 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, $\rightarrow \,=\, \rightarrow_1 \cup \rightarrow_2 \cup R$, where R contains, for all $p \in \mathbb{P}$ such that $(\mathcal{E}_1)_p$ and $(\mathcal{E}_2)_p$ are non-empty, the pair (e_1, e_2), where e_1 is the p-maximal event of M_1 and e_2 is the p-minimal event of M_2. 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. This is usually implemented by processes pairwise connected with FIFO channels. Alternative names are FIFO 1–1 [Chevrou et al. 2016] or simply FIFO [Babaoğlu and Marzullo 1993; Charron-Bost et al. 1996; Tel 2000]. 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 2.3 (p2p*-MSCs).* An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a p2p-MSC if, for any two send events s and s' such that $\lambda(s) \in \mathit{Send}(p, q, \_)$, $\lambda(s') \in \mathit{Send}(p, q, \_)$, and $s \rightarrow^+ s'$

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

Note that 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 one in Fig. 3f is p2p as the two messages are not 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 there exists a causal path from m_1 to m_2, then m_1 must be received before m_2. Causal ordering was introduced by Lamport in [Lamport 1978] with the name "happened before" order. Implementations were proposed in [Kshemkalyani and Singhal 1998; Peterson et al. 1989; Schiper et al. 1989]. 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. 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.

Definition 2.4 (co-*MSC*). An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is *causally ordered* if, for any two send events s and s', such that $\lambda(s) \in Send(\_, q, \_)$, $\lambda(s') \in Send(\_, q, \_)$, and $s \leq_{hb} s'$

- either $s, s' \in Matched(M)$ and $r \rightarrow^* r'$, with r and r' receive events such that $s \lhd r$ and $s' \lhd r'$.
- or $s' \in Unm(M)$.

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 the same process, regardless of the sender, must be received in the same order as they are sent. 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 [Chevrou et al. 2016]. 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 [Cristian and Fetzer 1999] or a global agreement on the order of events [Défago et al. 2004; Raynal 2010]. 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).

Definition 2.5 (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 $\lambda(s) \in Send(\_, q, \_)$, $\lambda(s') \in Send(\_, q, \_)$, and $s \rightsquigarrow s'$

- either $s, s' \in Matched(M)$ and $r \rightsquigarrow r'$, where $s \lhd r$ and $s' \lhd r'$,
- or $s' \in Unm(M)$.

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 all 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 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 is sent before m_2 by the same process, but we cannot find a linearization in which they are received in the same order (here, the reason is that ?2 $\leq_{hb}$?1). 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 2.6 (*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 $\lambda(s) \in Send(p, \_, \_)$, $\lambda(s') \in Send(p, \_, \_)$, and $s \rightarrow^{+} s'$ (which implies $s \rightsquigarrow s'$)

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

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 2.6. 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. 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 [Basu and Bultan 2016] where it is called many-to-many (denoted *-*). However, as underlined in [Chevrou et al. 2016], 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 2.7 (*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 $s, s' \in Matched(M)$ and $r \rightsquigarrow r'$, with r and r' receive events such that $s \lhd r$ and $s' \lhd r'$,
- or $s' \in Unm(M)$.

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 [Charron-Bost et al. 1996], 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 2.8 (*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 $\mathsf{MSC}_{\mathsf{asy}}$ (resp. $\mathsf{MSC}_{\mathsf{p2p}}$, $\mathsf{MSC}_{\mathsf{co}}$, $\mathsf{MSC}_{\mathsf{mb}}$, $\mathsf{MSC}_{\mathsf{onen}}$, $\mathsf{MSC}_{\mathsf{nn}}$, $\mathsf{MSC}_{\mathsf{rsc}}$) 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

We have defined several communication models as classes of MSCs. To compare to Chevrou *et al.* sequential hierarchy of communication models [Chevrou et al. 2016], we provide alternative definitions of these communication models based on executions. We only consider p2p, mb, FIFO 1−n and FIFO n−n, we refer to [Chevrou et al. 2016] for clarifying how the asynchronous, rsc, and co communication models may also be defined as sets of executions and fit in this hierarchy.

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 $\mathfrak{buf} : \mathbb{P} \times \mathbb{P} \to \mathfrak{Q}$ 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 $p, q \in \mathbb{P}$.

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 $a = send(p, q, m)$, then $w'_i = w_i \cdot m$ and $w'_j = w_j$ for all $j \in \mathfrak{Q} \setminus \{i\}$, where $i = \mathfrak{buf}(p, q)$.
- if $a = rec(p, q, m)$, then $w_i = m \cdot w'_i$ and $w'_j = w_j$ 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. e is p2p (resp. mb, onen, nn) if there exists a p2p queuing network (resp. mb, onen, nn) whose set of executions contains e.

Example 3.1. The execution

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

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

Example 3.2. The execution

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

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 [Chevrou et al. 2016] for examples illustrating each class of the hierarchy.

To conclude this brief discussion on queuing networks and executions, we clarify how executions are linked to MSCs classes. Indeed a linearization $\rightsquigarrow$ of an MSC defines a total order on its events, and therefore can be interpreted as 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 for p2p the claim is stronger as for a p2p-MSC M, *all* of its linearizations are p2p executions. This is not the case for the other communication models.

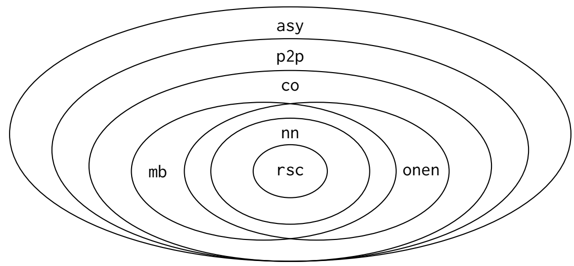


Fig. 4. Hierarchy of communication models based on sets of executions (taken from [Chevrou et al. 2016])

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 an MSO logic formula φ_{com} that captures exactly the class $\mathsf{MSC}_{\text{com}}$ of all com-MSCs. 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 $\varphi ::= \text{true} \mid x \rightarrow y \mid x \lhd y \mid \lambda(x) = a \mid x = y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \varphi \mid \neg\varphi$, 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, $L(\varphi)$ denotes the set of asynchronous MSCs that satisfy φ. The formula true describes the whole set of asynchronous MSCs, i.e., $L(\text{true}) = \mathsf{MSC}_{\text{asy}}$. 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 the formula in [Di Giusto et al. 2022]). We will therefore allow formulas of the form $x \rightarrow^+ y$, $x \rightarrow^* y$ or $x \leq_{hb} y$.

The communication models whose definitions are stated as the existence of a linearization enjoying some properties (mb, FIFO 1−n and FIFO n−n) 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 that are closer to the logic and show their equivalence to those given in Section 2. These alternative definitions will also be heavily used in the following sections. The idea is to characterize the MSCs in terms of the acyclicity of a binary relation that is MSO definable.

Peer-to-Peer MSCs. The MSO formula that defines $\mathsf{MSC}_{\text{p2p}}$ (i.e., the set of p2p-MSCs) directly follows from Definition 2.3:

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

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 $s \rightarrow^+ s'$ and either (i) their receptions happen in the reverse order, or (ii) s is unmatched and s' is matched.

Causally Ordered MSCs. As for p2p, the MSO-definability of $\mathsf{MSC}_{\mathsf{co}}$ follows from Definition 2.4:

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

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. $r' \rightarrow^+ r$, or (ii) s is unmatched and s' is matched.

Mailbox MSCs. For the mailbox communication model, Definition 2.5 cannot be immediately 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. This definition is shown to be equivalent to Definition 2.5 in [Di Giusto et al. 2022].

Definition 4.2 (mb alternative). Let an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be fixed, and let $\sqsubset_{\mathsf{mb}} \subseteq \mathcal{E} \times \mathcal{E}$ be defined as $s \sqsubset_{\mathsf{mb}} s'$ 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 $r_1 \rightarrow^+ r_2$.

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

The $\sqsubset_{\mathsf{mb}}$ 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 $\preceq_{\mathsf{mb}}$ is a partial order, it means that it is possible to find a linearization $\rightsquigarrow$, such that $\rightsquigarrow \subseteq \preceq_{\mathsf{mb}}$. It is easy to see that such a linearization is exactly what we called a mb-linearization in Definition 2.5. The MSO-definability of $\mathsf{MSC}_{\mathsf{mb}}$ follows from Definition 4.2; in particular, note that $\preceq_{\mathsf{mb}}$ is reflexive and transitive by definition, thus we just have to check acyclicity: $\varphi_{\mathsf{mb}} = \neg\exists x.\ x \prec_{\mathsf{mb}} x$ where $x \prec_{\mathsf{mb}} y$ is obtained as the MSO-definable transitive closure of the union of the MSO-definable relations $\rightarrow$, $\lhd$, and $\sqsubset_{\mathsf{mb}}$, where $x \sqsubset_{\mathsf{mb}} y$ may be defined as:

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

FIFO 1−n MSCs. As for the mailbox communication model, we give an alternative definition of onen-MSC; the equivalence with Definition 2.6 is shown in [Di Giusto et al. 2022].

Definition 4.3 (onen alternative). For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, let $\sqsubset_{\mathsf{1n}} \subseteq \mathcal{E} \times \mathcal{E}$ be defined as $e_1 \sqsubset_{\mathsf{1n}} e_2$ 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 $\preceq_{\mathsf{1n}} = (\rightarrow \cup \lhd \cup \sqsubset_{\mathsf{1n}})^*$. M is a onen-*MSC* if $\preceq_{\mathsf{1n}}$ is a partial order.

The $\sqsubset_{\mathsf{1n}}$ relation ensures that messages sent by a process are sent and received in an order that is suitable for the onen communication. Since $\preceq_{\mathsf{1n}}$ is a partial order, it is possible to find a linearization $\rightsquigarrow$ such that $\rightsquigarrow \subseteq \preceq_{\mathsf{1n}}$. It is not difficult to see that such a linearization is exactly what we called a onen-linearization in Definition 2.6. The existence of a MSO formula that defines $\mathsf{MSC}_{\mathsf{onen}}$ follows from Definition 4.3 and the MSO definability of $\sqsubset_{\mathsf{1n}}$:

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

FIFO n−n **MSCs.** This case is the most involved. As before we give an alternative definition that introduces an acyclic relation but equivalence to Definition 2.7 does not follow easily as in previous cases. It requires the introduction of an algorithm that finds the FIFO n−n-linearization and whose correct termination guarantees the acyclicity of the binary relation.

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

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

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

The full proof of the equivalence of Definitions 2.7 and 4.4 can be found in [Di Giusto et al. 2022]. Here we show only the more subtle part. The implication Definition 4.4 $\Rightarrow$ Definition 2.7 follows from the fact that the order of receive events imposes an order on sends and the fact that a nn-linearization is also a mb and onen-linearization.

Proposition 4.5. *Let M be an MSC. If $\sqsubset_{\mathsf{nn}}$ is cyclic, then M is not a* nn-*MSC.*

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 4.5, shows that Definitions 2.7 and 4.4 are equivalent.

Example 4.6. 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,

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.

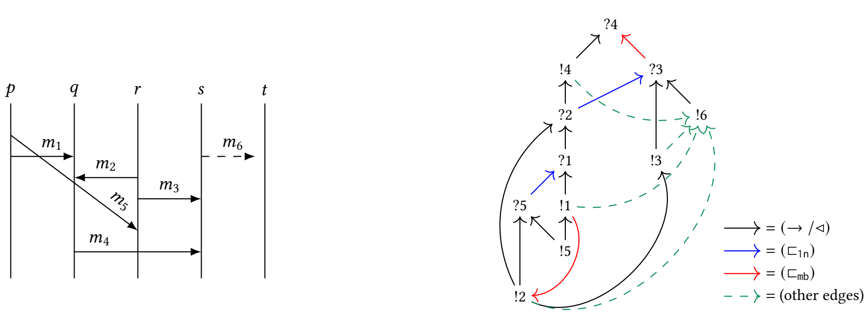


Fig. 5. An MSC and its EDG. In the EDG, only meaningful edges are shown.

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 4.7. *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 send event is added to the linearization only if all the matched send events were already added. □

Proposition 4.8. *Given an MSC M, Algorithm 1 terminates correctly if* $\sqsubset_{\mathsf{nn}}$ *is acyclic.*

The proof proceeds by induction on the number of events added to the linearization and relies on the fact that since $\sqsubset_{\mathsf{nn}}$ is acyclic then the EDG of the MSC is a DAG (see [Di Giusto et al. 2022]).

Finally, we showed the missing implication Definition 2.7 ⇒ Definition 4.4 and completed the proof of the equivalence of these two definitions. Based on Definition 4.4, we can now write the MSO formula for nn-MSCs as $\varphi_{\mathsf{nn}} = \neg\exists x. x \sqsubset^{+}_{\mathsf{nn}} x$, where we can define $x \sqsubset_{\mathsf{nn}} y$ as:

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

and ψ_3, ψ_4 can be specified as:

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

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

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

Realizable with Synchronous Communication MSCs. Following the characterization given in [Charron-Bost et al. 1996, Theorem 4.4], we provide an alternative definition of rsc-MSC that is closer to MSO logic. We first recall the concept of *crown*.

Definition 4.9 (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 4.10 (*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:

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

where $\propto$ is defined as

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

5 HIERARCHY OF CLASSES OF MSCS

In this section we show that the classes of MSCs for all the seven communication models form the hierarchy shown in Fig. 6. Here we just give intuitive explanations for the easy cases and formal proofs for the others. Proofs for all cases can be found in [Di Giusto et al. 2022].

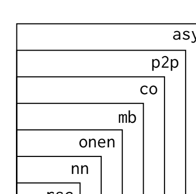
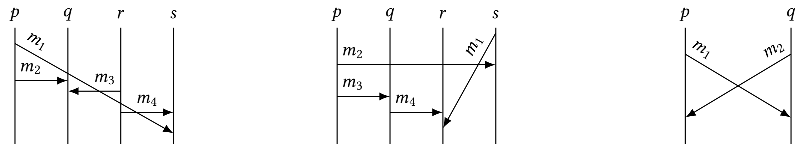


Fig. 6. MSC classes.

Notice that Fig. 4 only talks about single executions; it tells us that there might be an execution that is both mb and FIFO 1−n, but also an execution that is mb but not FIFO 1−n, and vice versa. Consider for instance Fig. 3c, the linearization/execution !1!2!3?1?2?3 is both mb and FIFO 1−n, !1!2!3?2?1?3 is mb but not FIFO 1−n, !1 !3 !2 ?1 ?2 ?3 is FIFO 1−n but not mb. On the other hand, Fig. 6 tells us that, given a onen-MSC, it is always also a mb-MSC; hence, if we are able to find a FIFO 1−n linearization for an MSC, then we can be sure that a mb linearization exists for that MSC. This means that the computation described by a FIFO 1−n MSC is always realizable using the mb communication model.



(a) co, not mb. (b) onen, not nn. (c) nn, not rsc.

Fig. 7. Examples of MSCs for various communication models.

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

Proposition 5.1. *Every* mb*-MSC is a* co*-MSC.*

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 $\lambda(s) \in Send(\_, q, \_)$, $\lambda(s') \in Send(\_, q, \_)$ 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) $s' \in Unm(M)$, or (ii) $s, s' \in Matched(M)$, $s \lhd r$, $s' \lhd r'$ 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 $!4 \sqsubset_{\mathsf{mb}} !1$ and $!2 \sqsubset_{\mathsf{mb}} !3$, which lead to a cyclic dependency, e.g. $!1 \rightarrow !2 \sqsubset_{\mathsf{mb}} !3 \rightarrow !4 \sqsubset_{\mathsf{mb}} !1$. This example and Proposition 5.1 prove that $\mathsf{MSC}_{\mathsf{mb}} \subset \mathsf{MSC}_{\mathsf{co}}$.

In the FIFO n−n communication model, any two messages must be received in the same order as they are sent. It is then easy to observe that each nn-MSC is a onen-MSC, because each nn-linearization is also a onen-linearization. Moreover, Fig. 7b shows an example of MSC that is FIFO 1−n but not FIFO n−n, hence we have that $\mathsf{MSC}_{\mathsf{nn}} \subset \mathsf{MSC}_{\mathsf{onen}}$; 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 nn-linearization, but it is possible to find a onen-linearization, such as $!1\ !2\ ?2\ !3\ ?3\ !4\ ?4\ ?1$. In the rsc model, every send event is immediately followed by its corresponding receive event. rsc is then a special case of FIFO n−n communication, and every rsc-MSC is a nn-MSC because a rsc-linearization is always also a nn-linearization. Besides, Fig. 7c shows an example of MSC that is FIFO n−n but not rsc, therefore $\mathsf{MSC}_{\mathsf{rsc}} \subset \mathsf{MSC}_{\mathsf{nn}}$.

Relation between onen-MSCs and mb-MSCs. Finally, we consider the relation between onen-MSCs and mb-MSCs that is not as straightforward as those seen so far. We start by only considering MSCs without unmatched messages.

Proposition 5.2. *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 $e \prec_{\mathsf{mb}} e$, for some event e. We can always explicitly write a cycle $e \prec_{\mathsf{mb}} e$ only using $\sqsubset_{\mathsf{mb}}$ and $<_{hb}$. For instance, there might be a cycle $e \prec_{\mathsf{mb}} e$ because we have that $e \sqsubset_{\mathsf{mb}} f <_{hb} g \sqsubset_{\mathsf{mb}} h \sqsubset_{\mathsf{mb}} i <_{hb} e$. Consider any two adjacent events s_1 and s_2 in the cycle ξ, where ξ has been written using only $\sqsubset_{\mathsf{mb}}$ 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) $s_1 \sqsubset_{\mathsf{mb}} s_2$. We know, by definition of $\sqsubset_{\mathsf{mb}}$, 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 hypothesis, ξ has to contain at least one $\sqsubset_{\mathsf{mb}}$. 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 $r_1 \sqsubset_{\mathsf{1n}} r_k$, according to the FIFO $1-\mathsf{n}$ semantics. In particular, note that we have $r_1 \sqsubset_{\mathsf{1n}} r_k \leq_{hb} r_2$.

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

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

The opposite direction is also true and the proof (see [Di Giusto et al. 2022]) uses the same technique to prove that a cycle for $\preceq_{\mathsf{1n}}$ always implies a cycle for $\preceq_{\mathsf{mb}}$.

Proposition 5.3. *Every* mb*-MSC without unmatched messages is a* onen*-MSC.*

Interestingly enough, Proposition 5.2 and 5.3 show that the classes of mb-MSCs and onen-MSCs coincide if we do not allow unmatched messages. This changes when we add unmatched messages into the mix. However, Proposition 5.2 still holds.

Proposition 5.4. *Every* onen*-MSC is a* mb*-MSC.*

Proof. Let M be an asynchronous MSC. The proof proceeds as for Proposition 5.2, but unmatched messages introduce some additional cases. Consider any two adjacent events s_1 and s_2 in a cycle ξ for $\prec_{\mathsf{mb}}$, where ξ has been written using only $\sqsubset_{\mathsf{mb}}$ and $<_{hb}$, and we never have two consecutive $<_{hb}$. These are some additional cases:

(3) $u_1 \sqsubset_{\mathsf{mb}} s_2$, where u_1 is the send event of an unmatched message. This case never happens because of how $\sqsubset_{\mathsf{mb}}$ 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 $u_1 \leq_{hb} u_2 \sqsubset_{\mathsf{mb}} s_3$. However, $u_2 \sqsubset_{\mathsf{mb}} s_3$ 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 $u_1 \rightarrow^+ s_k \lhd r_k \leq_{hb} s_2 \lhd r_2$[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 m_1 is unmatched, we should have $s_k \sqsubset_{\mathsf{1n}} u_1$, according to the FIFO 1−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 onen-MSC.

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

We showed that cases (3) and (4) can never happen, whereas (5) and (6) imply that M is not FIFO 1−n. If we combine them with the cases described in Proposition 5.2 we have the full proof. ◻

The MSC in Fig. 3f shows a simple example of an MSC with unmatched messages that is mb but not onen. This, along with Proposition 5.4, effectively shows that $\mathsf{MSC}_{\mathsf{onen}} \subset \mathsf{MSC}_{\mathsf{mb}}$.

6 APPLICATION: SYNCHRONIZABILITY AND BOUNDED MODEL-CHECKING

In this section, we show how the MSO characterization 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 [Di Giusto et al. 2022] for a formal definition of these notions). We write $L_{\mathsf{asy}}(\mathcal{S})$ to denote the set of asynchronous behaviors of $\mathcal{S}$, and we write $L_{\mathrm{com}}(\mathcal{S})$ to denote the restriction of L_{asy} to a specific communication model com, i.e., $L_{\mathrm{com}}(\mathcal{S}) = L_{\mathsf{asy}}(\mathcal{S}) \cap \mathsf{MSC}_{\mathrm{com}}$.

In general, even simple verification problems, e.g., control-state reachability, are undecidable for communicating systems [Brand and Zafiropulo 1983], under all communication models (except rsc, which we will not consider anymore from now on). They may become decidable if we consider only a certain class of behaviors. This motivates the following definition of generic *bounded model-checking* problem. Let C be a class of MSCs, the C-bounded model-checking problem for a communication model com $\in \{\mathsf{asy}, \mathsf{p2p}, \mathsf{co}, \mathsf{mb}, \mathsf{onen}, \mathsf{nn}\}$ is: given a system $\mathcal{S}$ and a MSO specification φ, decide whether $L_{\mathrm{com}}(\mathcal{S}) \cap C \subseteq L(\varphi)$. Here, we consider only classes C of MSCs that describe behaviors that are as close as possible to synchronous ones. 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-approximation, i.e., whether $L_{\mathrm{com}}(\mathcal{S}) \subseteq C$, will be referred to as the "synchronizability problem".

Bollig *et al.* [Bollig et al. 2021a] introduced a general framework that allows us to derive decidability results for the bounded model-checking and synchronizability problems for various classes of MSCs C. Here, we have managed to make this framework parametric in the communication model. To this aim, we require that the communication model, combined with the bounding class C, enforces a bounded treewidth of the MSCs, which is not always the case. Moreover 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.

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

Special Treewidth and Bounded Model-Checking. *Special treewidth* (STW) is a graph measure that indicates how close a graph is to a tree. An MSC is a graph where the nodes are the events and the edges are represented by the $\rightarrow$ and the $\lhd$ relations. Similarly to what has been done by Bollig *et al.* in [Bollig et al. 2021a], but adapted to our generic framework, we adopt a game-based definition for special treewidth: 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 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 $k \in \mathbb{N}$, 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.

The special treewidth of an MSC is the least k such that the associated game is k-winning for Eve (see for instance [Bollig and Gastin 2019]). 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.

Courcelle's theorem implies that the following problem 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[3].

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

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 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 C, i.e., whether $L_{\mathrm{com}}(\mathcal{S}) \subseteq C$.

Definition 6.2. Let a communication model com and a class C of MSCs be fixed. The (com, C)-synchronizability problem is defined as follows: given a system $\mathcal{S}$, decide whether $L_{\mathrm{com}}(\mathcal{S}) \subseteq C$.

In [Bollig et al. 2021a] the authors show that, for com $=$ $\mathsf{p2p}$ and com $=$ mb, the (com, C)-synchronizability problem is decidable for several classes C. We generalize their result to other communication models under a general assumption on the bounding class C.

Theorem 6.3. *For any* com $\in \{\mathsf{asy}, \mathsf{p2p}, \mathsf{co}, \mathsf{mb}, \mathsf{onen}, \mathsf{nn}\}$ *and for all class of MSCs* C, *if* C *is STW-bounded and MSO-definable, then the* (com, C)*-synchronizability problem is decidable.*

The proof of Theorem 6.3 echoes the proof of [Bollig et al. 2021b, Theorem 11], with the main technical argument being the existence of a "borderline violation" (see [Bollig et al. 2021b, Lemma 9]). However, the existence of a borderline violation is more subtle to establish, because $\mathsf{MSC}_{\mathsf{onen}}$ and

[3]cfr. proof in [Di Giusto et al. 2022]

(a) A nn-MSC M. (b) A prefix of M.

Fig. 8. A nn-MSC with a prefix that is neither a onen-MSC nor a nn-MSC.

	Weakly sync	Weakly k-sync	∃k bounded	∀k bounded
asy	unbounded STW	✓	✓	✓
p2p	✗ [1]	✓ [1]	✓ [1]	✓ [1]
co	✗	✓	✓	✓
mb	✓ [1]	✓ [1]	✓ [1]	✓ [1]
onen	✓	✓	✓	✓
nn	✓	✓	✓	✓

Fig. 9. Table summarising the (un)decidability results for the synchronizability problems (each combination of a communication model com and a class C of MSCs is a different synchronizability problem). The symbol ✗ stands for undecidability and unbounded special treewidth of $\mathsf{MSC}_{\mathsf{com}} \cap C$, whereas ✓ stands for decidability and bounded STW of $\mathsf{MSC}_{\mathsf{com}} \cap C$. [1] indicates that the result was shown by Bollig *et al.* [Bollig et al. 2021a]. Unbounded STW stands for unbounded STW of $\mathsf{MSC}_{\mathsf{com}} \cap C$ (but not necessarily undecidability).

$\mathsf{MSC}_{\mathsf{nn}}$ are not prefixed-closed (see Fig. 8). A way to solve this technical issue is to consider a more strict notion of prefix. All details of the proof of Theorem 6.3 can be found in [Di Giusto et al. 2022].

In the remainder, we investigate which combinations of com and C fit the hypotheses of this theorem. We review the classes of weakly synchronous and weakly k-synchronous inspired by [Bouajjani et al. 2018], and the classes of existentially k-bounded and universally k-bounded MSCs [Genest et al. 2004]. Fig. 9 summarizes the decidability results of the (com, C)-synchronizability problem for each combination of com and C we will consider.

Weakly Synchronous MSCs. We start by recalling the definition of the class of weakly synchronous MSCs as introduced in [Bollig et al. 2021a]. 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 6.4 (Exchange). Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC. We say that M is an *exchange* if $SendEv(M)$ 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 6.5 (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.

In [Bollig et al. 2021a] it is shown that, for the class of weakly synchronous MSCs, the synchronizability problem is undecidable for p2p, but decidable for mb. 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 [Bollig et al. 2021b, Theorem 20] for the p2p case (cfr. [Di Giusto et al. 2022]).

Proposition 6.6. *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 6.7. *Let* com $\in \{\mathsf{onen}, \mathsf{nn}\}$. *The following problem is decidable: given a communicating system $\mathcal{S}$, is every MSC in $L_{\mathsf{com}}(\mathcal{S})$ weakly synchronous?*

Proof. We will consider com = onen; the proof for com = nn is similar. We would like to know if every MSC in $L_{\mathsf{onen}}(\mathcal{S})$ is in the class of weakly synchronous MSCs. Since every MSC in $L_{\mathsf{onen}}(\mathcal{S})$ is a onen-MSC, we can equivalently restrict the problem to the class of weakly synchronous MSCs that are also onen-MSCs. Let C be the class of onen weakly synchronous MSCs; we show that C is MSO-definable and STW-bounded, which implies the decidability of *SP* for Theorem 6.3. The class of weakly synchronous MSCs was shown to be MSO-definable in [Bollig et al. 2021a]; 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 $\mathsf{MSC}_{\mathsf{onen}}$ 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 [Bollig et al. 2021a], and since $\mathsf{MSC}_{\mathsf{onen}} \subset \mathsf{MSC}_{\mathsf{mb}}$, we also have that the class of mb weakly synchronous MSCs has a bounded special treewidth. □

Weakly k-Synchronous MSCs. We consider now weakly k-synchronous MSCs ([Bollig et al. 2021a]), which are the weakly synchronous MSCs such that the number of messages sent per exchange is at most k.

Definition 6.8 (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 6.9 (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 6.10. MSC M_2 in Fig. 10 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 [Bollig et al. 2021a], and these results still hold even for our definition of MSC. A direct application of Theorem 6.3 shows that, for weakly k-synchronous MSCs, *SP* is decidable for all communication models.

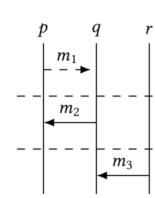


Fig. 10. MSC M_2

Proposition 6.11. *Let* com $\in \{\mathsf{asy}, \mathsf{p2p}, \mathsf{co}, \mathsf{mb}, \mathsf{onen}, \mathsf{nn}\}$. *The following problem is decidable: given a communicating system $\mathcal{S}$, is every MSC in $L_{\mathsf{com}}(\mathcal{S})$ weakly k-synchronous?*

Proof. The class C of weakly k-synchronous MSCs is MSO-definable and STW-bounded, therefore the result follows from Theorem 6.3. □

Existentially Bounded MSCs. We move now to existentially k-bounded MSCs, first introduced by Lohrey and Muschol [Lohrey and Muscholl 2002], that form a relevant class of MSC for extending the Büchi-Elgot-Trakhthenbrot theorem from words to MSCs [Genest et al. 2007, 2004]. 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.[4] 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 6.12. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}$ and $k \in \mathbb{N}$. A linearization $\rightsquigarrow$ of M is called *k-bounded* if, for all $e \in \mathit{SendEv}(M)$, with $\lambda(e) = \mathit{send}(p, q, m)$, we have

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

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

Definition 6.13 (Existentially bounded MSC). Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{asy}}$ 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.

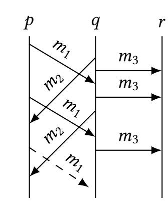


Fig. 11. MSC M_3

Example 6.14. MSC M_3 in Fig. 11 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 6.15. 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 6.16. An MSC M is *causally ordered existentially k-bounded* (co-$\exists k$-bounded) if it is a causally ordered MSC and it is also existentially k-bounded.

When moving to the other communication models, the definitions are not as straightforward. Indeed when defining mb, FIFO 1−n and FIFO n−n, we require the existence of a linearization, which represents a sequence of events that can be executed by a mb, resp. FIFO 1−n and FIFO n−n system. Hence, in order to define $\exists k$-bounded MSCs we should require that there exists a k-bounded linearization that is also a mb-linearization (resp. FIFO 1−n and FIFO n−n), not just any linearization.

Definition 6.17. An MSC M is mb *existentially k-bounded* (mb-$\exists k$-bounded) if it has a k-bounded mb-linearization.

Definition 6.18. An MSC M is onen *existentially k-bounded* (onen-$\exists k$-bounded) if it has a k-bounded onen-linearization.

Definition 6.19. 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 6.7 for weakly synchronous MSCs.

[4]This may look surprising in our general context to count messages in transit in that way, but it can be seen 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 whatever 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 [Lohrey and Muscholl 2002], we introduce a binary relation $\longmapsto_k$ ($\rightsquigarrow_b$ in their work) associated with a given bound k and an MSC M. Let $k \geq 1$ and M be a fixed MSC. We have $r \longmapsto_k s$ if, for some $i \geq 1$ and some channel (p,q)[5]:

(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.

Example 6.20. Consider MSC M_4 in Fig 12. 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 [Lohrey and Muscholl 2002] 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):

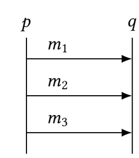


Fig. 12. MSC M_4.

- 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 6.21. *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 $\hookrightarrow_k \not\subseteq \rightsquigarrow$, i.e. there are two events r and s such that $r \hookrightarrow_k s$ and $s \rightsquigarrow r$. By definition of $\hookrightarrow_k$, there are k send events in a channel (p,q) that are executed before s, and whose respective receive events happens after r. If s is executed before r in the linearization, there will be $k + 1$ messages in channel (i.e., $\rightsquigarrow$ is not 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:

- 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.

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

- 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 $r \hookrightarrow_k s$.

□

According to Proposition 6.21, we can write the MSO formula the defines $\exists k$-bounded MSCs as

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

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

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

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

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

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 [Lohrey and Muscholl 2002] 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:

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

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 $\leq_{\mathsf{mb}}$, i.e., $\leq_{\mathsf{mb}} \subseteq \rightsquigarrow$. 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 $(\leq_{\mathsf{mb}} \cup \longmapsto_k) \subseteq \rightsquigarrow$. Such a linerization exists only if $\leq_{\mathsf{mb}} \cup \longmapsto_k$ is acyclic. If $\leq_{\mathsf{mb}} \cup \longmapsto_k$ 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 similar. Summing up:

Proposition 6.22. *An MSC M is* mb-$\exists k$*-bounded iff the relation $\leq_{\mathsf{mb}} \cup \longmapsto_k$ is acyclic.*
An MSC M is onen-$\exists k$-bounded iff the relation $\leq_{\mathsf{1n}} \cup \longmapsto_k$ is acyclic.
An MSC M is nn-$\exists k$-bounded iff the relation $\sqsubset_{\mathsf{nn}} \cup \longmapsto_k$ is acyclic.

The MSO-definability of all the variants of ∃k-bounded MSCs directly follows from Proposition 6.22, since all of these relations were shown to be MSO-definable (Section 4).

Special treewidth. In [Bollig and Gastin 2019, Lemma 5.37] it was shown that the special treewidth of existentially k-bounded MSCs is bounded by $k\,|\mathbb{P}|^2$, for $k \geq 1$. Actually, STW-boundedness was shown for the more general class of Concurrent Behaviours with Matching (CBM), but the result is still valid since $\mathrm{MSC}_{\mathsf{asy}} \subset$ 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 [Lohrey and Muscholl 2002].

Definition 6.23 (Universally bounded MSC). Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{asy}}$ and $k \in \mathbb{N}$. M is *universally k-bounded* ($\forall k$-bounded) if all of its linearizations are k-bounded.

Definition 6.24. 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 6.25. An MSC M is *causally ordered 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 6.26. 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 6.27. 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. Next, we investigate the MSO-definability of all the variants of universally k-bounded MSCs that we discussed. In [Lohrey and Muscholl 2002], 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 of 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 6.28. *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, then by definition 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 $\hookrightarrow_k \not\subseteq \leq_{hb}$, i.e., there are two events r and s such that $r \hookrightarrow_k s$ and $r \not\leq_{hb} s$. If $r \not\leq_{hb} s$, 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[6]. 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. □

It follows that p2p-$\forall k$-bounded and co-$\forall k$-bounded MSCs are MSO-definable by definition, since p2p-MSCs, co-MSCs, and universally k-bounded MSCs are all MSO-definable. We already showed that $\longmapsto_k$ is MSO-definable when considering p2p communication. The characterization for the other communication models is similar to that given in [Lohrey and Muscholl 2002], but it uses the proper relation for each communication model.

[6]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.

Proposition 6.29. *An MSC M is* $\mathtt{mb}$-$\forall k$*-bounded if and only if* $\longmapsto_k \subseteq \preceq_{\mathtt{mb}}$.
An MSC M is onen-$\forall k$-bounded if and only if $\longmapsto_k \subseteq \preceq_{\mathtt{1n}}$.
An MSC M is nn-$\forall k$-bounded if and only if $\longmapsto_k \subseteq \sqsubset_{\mathtt{nn}}$.

Proof. We only show it for the $\mathtt{mb}$ 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 $\longmapsto_k \subseteq \preceq_{\mathtt{mb}}$. For every mailbox linearization $\rightsquigarrow$ of M we have that $\preceq_{\mathtt{mb}} \subseteq \rightsquigarrow$. This implies $\longmapsto_k \subseteq \rightsquigarrow$, that is to say every mailbox linearization is k-bounded.

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

Using Proposition 6.29, we can now easily write the MSO formulas that define these variants of universally k-bounded MSCs.

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

Special treewidth. All the variants of universally k-bounded MSCs that we presented have a bounded special treewidth. This directly follows from the STW-boundedness 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. These communication models either come from the early days of distributed systems, or are idealized models of communicating systems with queues of messages (spanning from systems on chip to micro-services linked with "buses", or simply concurrent programs with FIFO queues in shared memory). We drew the hierarchy of these communication models and characterized each of them with MSO logic. We showed that all the models fit in a single framework that is used to show the 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 [Mattern and Fünfrocken 1994], which we expect to sit somewhere between mailbox and causally ordered within the hierarchy that we presented. Moreover, as shown by Fig. 9, the decidability of the synchronizability problem for weakly synchronous MSCs and fully asynchronous communication is not entailed by our techniques, and could be further investigated.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their careful reading and detailed remarks that helped to improve the final version of this paper.

REFERENCES

Özalp Babaoğlu and Keith Marzullo. 1993. *Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms.* ACM Press/Addison-Wesley Publishing Co., USA, 55–96.

Samik Basu and Tevfik Bultan. 2016. On deciding synchronizability for asynchronously communicating systems. *Theor. Comput. Sci.* 656 (2016), 60–75. https://doi.org/10.1016/j.tcs.2016.09.023

Benedikt Bollig, Cinzia Di Giusto, Alain Finkel, Laetitia Laversa, Étienne Lozes, and Amrita Suresh. 2021a. A Unifying Framework for Deciding Synchronizability. In *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021 (LIPIcs, Vol. 203)*, Serge Haddad and Daniele Varacca (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Virtual Conference, 14:1–14:18. https://doi.org/10.4230/LIPIcs.CONCUR.2021.14

Benedikt Bollig, Cinzia Di Giusto, Alain Finkel, Laetitia Laversa, Étienne Lozes, and Amrita Suresh. 2021b. *A Unifying Framework for Deciding Synchronizability (extended version).* Technical Report. HAL. available at https://hal.archives-ouvertes.fr/hal-03278370/document.

Benedikt Bollig and Paul Gastin. 2019. Non-Sequential Theory of Distributed Systems. *CoRR* abs/1904.06942 (2019), 1–82. arXiv:1904.06942 http://arxiv.org/abs/1904.06942

Vincent Botbol, Emmanuel Chailloux, and Tristan Le Gall. 2017. Static Analysis of Communicating Processes Using Symbolic Transducers. In *Verification, Model Checking, and Abstract Interpretation - 18th International Conference, VMCAI 2017* (January 15-17, 2017). Springer, Paris, France, 73–90. https://doi.org/10.1007/978-3-319-52234-0_5

Ahmed Bouajjani, Constantin Enea, Kailiang Ji, and Shaz Qadeer. 2018. On the Completeness of Verifying Message Passing Programs Under Bounded Asynchrony. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, July 14-17, 2018, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 10982)*, Hana Chockler and Georg Weissenbacher (Eds.). Springer, Oxford, UK, 372–391. https://doi.org/10.1007/978-3-319-96142-2_23

Daniel Brand and Pitro Zafiropulo. 1983. On Communicating Finite-State Machines. *J. ACM* 30, 2 (1983), 323–342. https://doi.org/10.1145/322374.322380

Bernadette Charron-Bost, Friedemann Mattern, and Gerard Tel. 1996. Synchronous, Asynchronous, and Causally Ordered Communication. *Distributed Comput.* 9, 4 (1996), 173–191. https://doi.org/10.1007/s004460050018

Florent Chevrou, Aurélie Hurault, and Philippe Quéinnec. 2016. On the diversity of asynchronous communication. *Formal Aspects Comput.* 28, 5 (2016), 847–879. https://doi.org/10.1007/s00165-016-0379-x

Flaviu Cristian and Christof Fetzer. 1999. The Timed Asynchronous Distributed System Model. *IEEE Trans. Parallel Distributed Syst.* 10, 6 (1999), 642–657. https://doi.org/10.1109/71.774912

Xavier Défago, André Schiper, and Péter Urbán. 2004. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv.* 36, 4 (2004), 372–421. https://doi.org/10.1145/1041680.1041682

Cinzia Di Giusto, Davide Ferré, Laetitia Laversa, and Etienne Lozes. 2022. *A partial order view of message-passing communication models.* Research Report. Université Nice Côte d'Azur, CNRS, I3S, France. https://hal.archives-ouvertes.fr/hal-03823473

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

Blaise Genest, Dietrich Kuske, and Anca Muscholl. 2007. On Communicating Automata with Bounded Channels. *Fundam. Informaticae* 80, 1-3 (2007), 147–167. http://content.iospress.com/articles/fundamenta-informaticae/fi80-1-3-09

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

Alexander Heußner, Tristan Le Gall, and Grégoire Sutre. 2009. Extrapolation-Based Path Invariants for Abstraction Refinement of Fifo Systems. In *Model Checking Software, 16th International SPIN Workshop, June 26-28, 2009. Proceedings.* Springer, Grenoble, France, 107–124. https://doi.org/10.1007/978-3-642-02652-2_11

Gerard J. Holzmann. 2004. *The SPIN Model Checker - primer and reference manual.* Addison-Wesley. I–XII, 1–596 pages.

ITU-T. 2011. *Recommendation ITU-T Z.120: Message Sequence Chart (MSC).* Technical Report. International Telecommunication Union, Geneva.

Bernhard Kragl, Shaz Qadeer, and Thomas A. Henzinger. 2018. Synchronizing the Asynchronous. In *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018 (LIPIcs, Vol. 118)*, Sven Schewe and Lijun Zhang (Eds.). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Beijing, China, 21:1–21:17. https://doi.org/10.4230/LIPIcs.CONCUR.2018.21

Ajay D. Kshemkalyani and Mukesh Singhal. 1998. Necessary and Sufficient Conditions on Information for Causal Message Ordering and Their Optimal Implementation. *Distributed Comput.* 11, 2 (1998), 91–111. https://doi.org/10.1007/

s004460050044

Leslie Lamport. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM* 21, 7 (1978), 558–565. https://doi.org/10.1145/359545.359563

Julien Lange and Nobuko Yoshida. 2019. Verifying Asynchronous Interactions via Communicating Session Automata. In *Computer Aided Verification - 31st International Conference, CAV 2019, July 15-18, 2019, Proceedings, Part I*. Springer, New York City, NY, USA, 97–117. https://doi.org/10.1007/978-3-030-25540-4_6

Richard J. Lipton. 1975. Reduction: A Method of Proving Properties of Parallel Programs. *Commun. ACM* 18, 12 (1975), 717–721. https://doi.org/10.1145/361227.361234

Markus Lohrey and Anca Muscholl. 2002. Bounded MSC Communication. In *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002, April 8-12, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2303)*, Mogens Nielsen and Uffe Engberg (Eds.). Springer, Grenoble, France, 295–309. https://doi.org/10.1007/3-540-45931-6_21

Friedemann Mattern and Stefan Fünfrocken. 1994. A Non-Blocking Lightweight Implementation of Causal Order Message Delivery. In *Theory and Practice in Distributed Systems, International Workshop, September 5-9, 1994, Selected Papers (Lecture Notes in Computer Science, Vol. 938)*, Kenneth P. Birman, Friedemann Mattern, and André Schiper (Eds.). Springer, Dagstuhl Castle, Germany, 197–213. https://doi.org/10.1007/3-540-60042-6_14

Larry L. Peterson, Nick C. Buchholz, and Richard D. Schlichting. 1989. Preserving and Using Context Information in Interprocess Communication. *ACM Trans. Comput. Syst.* 7, 3 (1989), 217–246. https://doi.org/10.1145/65000.65001

Michel Raynal. 2010. *Communication and Agreement Abstractions for Fault-Tolerant Asynchronous Distributed Systems*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00236ED1V01Y201004DCT002

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

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

Salvatore La Torre, P. Madhusudan, and Gennaro Parlato. 2009. Reducing Context-Bounded Concurrent Reachability to Sequential Reachability. In *Computer Aided Verification, 21st International Conference, CAV 2009, June 26 - July 2, 2009. Proceedings*. Springer, Grenoble, France, 477–492. https://doi.org/10.1007/978-3-642-02658-4_36

Robbert van Renesse. 1993. Causal Controversy at Le Mont St.-Michel. *ACM SIGOPS Oper. Syst. Rev.* 27, 2 (1993), 44–53. https://doi.org/10.1145/155848.155857

Klaus von Gleissenthall, Rami Gökhan Kici, Alexander Bakst, Deian Stefan, and Ranjit Jhala. 2019. Pretend synchrony: synchronous verification of asynchronous distributed programs. *PACMPL* 3, POPL (2019), 59:1–59:30. https://doi.org/10.1145/3290372

Received 2022-07-07; accepted 2022-11-07