

A Unifying Framework for Deciding Synchronizability

Benedikt Bollig ✉ 

Université Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, France

Cinzia Di Giusto ✉ 

Université Côte d’Azur, CNRS, I3S, France

Alain Finkel ✉ 

Université Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, France

Institut Universitaire de France, Paris, France

Laetitia Laversa ✉ 

Université Côte d’Azur, CNRS, I3S, France

Etienne Lozes ✉ 

Université Côte d’Azur, CNRS, I3S, France

Amrita Suresh ✉ 

Université Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, France

Abstract

Several notions of synchronizability of a message-passing system have been introduced in the literature. Roughly, a system is called synchronizable if every execution can be rescheduled so that it meets certain criteria, e.g., a channel bound. We provide a framework, based on MSO logic and (special) tree-width, that unifies existing definitions, explains their good properties, and allows one to easily derive other, more general definitions and decidability results for synchronizability.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases communicating finite-state machines, message sequence charts, synchronizability, MSO logic, special tree-width

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2021.14

Related Version *Full Version:* <https://hal.archives-ouvertes.fr/hal-03278370>

1 Introduction

Communication systems. The model of concurrent processes communicating asynchronously through FIFO channels is used since the 1960s in applications such as communication protocols [28], hardware design, MPI programs, and more recently for designing and verifying session types [23], web contracts, choreographies, concurrent programs, Erlang, Rust, etc. Since communication systems use FIFO channels, it is well known that all non-trivial properties (e.g., are all channels bounded?) are undecidable [9], essentially because a FIFO channel may simulate the tape of Turing machines and the counters of Minsky machines. However, there are many subclasses of communication systems for which the control-state reachability problem becomes decidable: e.g., synchronizable systems and existentially bounded systems (executions can be reorganized or decomposed into a finite number of sequences in which all channels are bounded), flat FIFO machines [15,17] (the graph of the machine does not contain nested loops), channel-recognizable systems [4], unreliable (lossy, insertion, duplication) FIFO systems [11], input-bounded FIFO machines [5], and half-duplex systems [10].



© Benedikt Bollig, Cinzia Di Giusto, Alain Finkel, Laetitia Laversa, Etienne Lozes, and Amrita Suresh; licensed under Creative Commons License CC-BY 4.0

32nd International Conference on Concurrency Theory (CONCUR 2021).

Editors: Serge Haddad and Daniele Varacca; Article No. 14; pp. 14:1–14:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

On the boundedness problem. We focus on the boundedness problem, which is known to be undecidable. We could limit our analysis to decide whether for a given integer $k \geq 0$, known in advance, the FIFO channels are k -bounded, and this property is generally decidable in PSPACE. Unfortunately, the k -boundedness property is too binding since we could want to design an *unbounded* system that is able, for example, to make unbounded iterations of sending and receiving messages. Hence, to cope with this limitation, one can find variants of the boundedness property that essentially reduce to say that every unbounded execution of a system (i.e., channels are unbounded along the execution) is equivalent (for instance, causally equivalent) to another *bounded* execution.

About synchronizability. To mention some examples, Lohrey and Muscholl introduced *existentially k -bounded* systems [25] (see also [18,19,24]) where all accepting executions leading to a stable (with empty channels) final configuration can be re-ordered into a k -bounded execution. This property is undecidable, even for a given k [18]. A more general definition, still called existentially bounded, is given in [2014] where the considered executions are *not* supposed to be final or stable [22]. In [21,25], the notion of *universally k -bounded* (all possible schedulings of an execution are k -bounded) is also discussed and the authors show that the property is undecidable in general. In 2011, Basu and Bultan introduced *synchronizable* systems [3], for which every execution is equivalent (for the projection on sending messages) to one of the same system but communicating by rendezvous; to avoid ambiguity, we call such systems *send-synchronizable*. In 2018, Bouajjani et al., called a system \mathcal{S} *k -synchronizable* [8] (to avoid confusion we call such systems *weakly k -synchronizable*) if every MSC of \mathcal{S} admits a linearization (which is not necessarily an execution) that can be divided into blocks of at most k messages. After each block, a message is either read, or will never be read. This constraint seems to imply that buffers are bounded to k messages. However, as the linearization need not be an execution, this implies that a weakly k -synchronizable execution, even with the more efficient reschedule, can need unbounded channels to be run by the system.

Communication architecture and variants. A key difference between these works is that they consider different communication architectures. Existentially bounded systems have been studied for p2p (with one queue per pair of processes), whereas k -synchronizability has been studied for mailbox communication, for which each process merges all its incoming messages in a unique queue. The decidability results for k -synchronizability have been extended to p2p communications [14], but it is unknown whether the decidability results for existentially bounded systems extend to mailbox communication. Moreover, variants of those definitions can be obtained depending on if we consider messages that are sent but never read, called unmatched messages. Indeed the challenges that arise in [8] are due to mailbox communication and unmatched messages blocking a channel so that all messages sent afterwards will never be read. To clarify and overcome this issue, we propose *strong k -synchronizability*, a new definition that is suitable for mailbox communication: an execution is called *strongly k -synchronizable* if it can be rescheduled into another k -bounded execution such that there are at most k messages in the channels before emptying them.

Contributions. Our contributions can be summarized as follows:

- In order to unify the notions of synchronizability, we introduce a general framework based on monadic second-order (MSO) logic and (special) tree-width that captures most existing definitions of systems that may work with bounded channels. Moreover, reachability and model checking are shown decidable in this framework.

- We show that existentially bounded systems can be expressed in our framework and, as a consequence, the existentially k -bounded property is decidable by using the generic proof.
- We generalize the existing notion of (weak) k -synchronizability in [8] and we introduce three new classes of synchronizable systems: weakly synchronizable (which are more general than weakly k -synchronizable), strongly synchronizable and strongly k -synchronizable (which are particular cases of weakly synchronizable). We then prove that these properties all fit in our framework and are all shown decidable using the generic proof.
- We then deduce that reachability and model checking are decidable for these classes (only control-state reachability was shown to be decidable for weakly k -synchronizable in [8] and it is clearly also decidable for existentially/universally bounded systems but reachability properties are generally not studied for these classes of systems).
- In order to obtain better complexity results for some classes (strongly and weakly synchronizable systems), we also use the fragment of propositional dynamic logic with loop and converse (LCPDL) instead of MSO logic in our framework.
- We provide a comparison between synchronizable classes both for p2p and mailbox semantics (see Fig. 8 for p2p systems and Fig. 9 for mailbox systems). In particular, we clarify the link between weakly synchronizable and existentially bounded systems for both p2p and mailbox systems, which was left open in [8] and solved only for p2p systems in [23, Theorem 7] where weakly synchronizable systems are shown to be included into existentially bounded ones when considering executions (and not MSCs as in our case).

Outline. Section 2 defines some preliminary notions such as p2p/mailbox message sequence charts (MSCs), and communicating systems. Section 3 presents the unifying MSO framework and two general theorems on k -synchronizability and model checking. In Section 4, we apply the MSO framework to different existing definitions of synchronizability, and we introduce a new decidable one. Section 5 studies the relations between the classes. In Section 6, we conclude with some final remarks. Due to space constraints, some proofs are given in the full version of the paper, available at: <https://hal.archives-ouvertes.fr/hal-03278370>

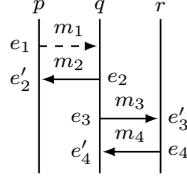
2 Preliminaries

2.1 Message Sequence Charts

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

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

14:4 A Unifying Framework for Deciding Synchronizability



■ **Figure 1** MSC M_1 .

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

Finally, letting $\leq_M = (\rightarrow \cup \triangleleft)^*$, we require that \leq_M is a partial order.

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

► **Example 1.** For a set of processes $\mathbb{P} = \{p, q, r\}$ and a set of messages $\mathbb{M} = \{m_1, m_2, m_3, m_4\}$, $M_1 = (\mathcal{E}, \rightarrow, \triangleleft, \lambda)$ is an MSC where, for example, $e_2 \triangleleft e'_2$ and $e'_3 \rightarrow e_4$. The dashed arrow means that the send event e_1 does not have a matching receive, so $e_1 \in Unm(M_1)$. Moreover, $e_2 \leq_{M_1} e_4$, but $e_1 \not\leq_{M_1} e_4$. We can find a total order $\rightsquigarrow \supseteq \leq_{M_1}$ such that $e_1 \rightsquigarrow e_2 \rightsquigarrow e'_2 \rightsquigarrow e_3 \rightsquigarrow e'_3 \rightsquigarrow e_4 \rightsquigarrow e'_4$. We call \rightsquigarrow a linearization, which is formally defined below.

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

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

We let $\preceq_M = (\rightarrow \cup \triangleleft \cup \sqsubset_M)^*$. Note that $\leq_M \subseteq \preceq_M$. We call $M \in MSC$ a *mailbox MSC* if \preceq_M is a partial order. Intuitively, this means that events can be scheduled in a way that corresponds to the mailbox semantics, i.e., with one incoming channel per process. Following the terminology in [8], we also say that a mailbox MSC satisfies *causal delivery*. The set of mailbox MSCs $M \in MSC$ is denoted by MSC_{mb} .

► **Example 2.** MSC M_1 is a mailbox MSC. Indeed, even though the order \rightsquigarrow defined in Example 1 does not respect all mailbox constraints, particularly the fact that $e_4 \sqsubset_{M_1} e_1$, there is a total order $\rightsquigarrow \supseteq \preceq_{M_1}$ such that $e_2 \rightsquigarrow e_3 \rightsquigarrow e'_3 \rightsquigarrow e_4 \rightsquigarrow e_1 \rightsquigarrow e'_2 \rightsquigarrow e'_4$. We call \rightsquigarrow a mailbox linearization, which is formally defined below.

Linearizations, Prefixes, and Concatenation. Consider $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda) \in MSC$. A *p2p linearization* (or simply *linearization*) of M is a (reflexive) total order $\rightsquigarrow \subseteq \mathcal{E} \times \mathcal{E}$ such that $\leq_M \subseteq \rightsquigarrow$. Similarly, a *mailbox linearization* of M is a total order $\rightsquigarrow \subseteq \mathcal{E} \times \mathcal{E}$ such that

$\preceq_M \subseteq \rightsquigarrow$. That is, every mailbox linearization is a p2p linearization, but the converse is not necessarily true (Example 2). Note that an MSC is a mailbox MSC iff it has at least one mailbox linearization.

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

► **Lemma 3.** *Every prefix of a mailbox MSC is a mailbox MSC.*

Let $M_1 = (\mathcal{E}_1, \rightarrow_1, \triangleleft_1, \lambda_1)$ and $M_2 = (\mathcal{E}_2, \rightarrow_2, \triangleleft_2, \lambda_2)$ be two MSCs. Their *concatenation* $M_1 \cdot M_2 = (\mathcal{E}, \rightarrow, \triangleleft, \lambda)$ is defined if, for all $(p, q) \in \mathbb{C}$, $e_1 \in \text{Unm}(M_1)$, and $e_2 \in \mathcal{E}_2$ such that $\lambda(e_1) \in \text{Send}(p, q, _)$ and $\lambda(e_2) \in \text{Send}(p, q, _)$, we have $e_2 \in \text{Unm}(M_2)$. As expected, \mathcal{E} is the disjoint union of \mathcal{E}_1 and \mathcal{E}_2 , $\triangleleft = \triangleleft_1 \cup \triangleleft_2$, λ is the “union” of λ_1 and λ_2 , and $\rightarrow = \rightarrow_1 \cup \rightarrow_2 \cup R$. Here, R contains, for all $p \in \mathbb{P}$ such that $(\mathcal{E}_1)_p$ and $(\mathcal{E}_2)_p$ are non-empty, the pair (e_1, e_2) where e_1 is the maximal p -event in M_1 and e_2 is the minimal p -event in M_2 . Note that $M_1 \cdot M_2$ is indeed an MSC and that concatenation is associative.

2.2 Communicating Systems

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

► **Definition 4.** *A communicating system over \mathbb{P} and \mathbb{M} is a tuple $\mathcal{S} = (A_p)_{p \in \mathbb{P}}$. For each $p \in \mathbb{P}$, $A_p = (\text{Loc}_p, \delta_p, \ell_p^0)$ is a finite transition system where Loc_p is a finite set of local (control) states, $\delta_p \subseteq \text{Loc}_p \times \Sigma_p \times \text{Loc}_p$ is the transition relation, and $\ell_p^0 \in \text{Loc}_p$ is the initial state.*

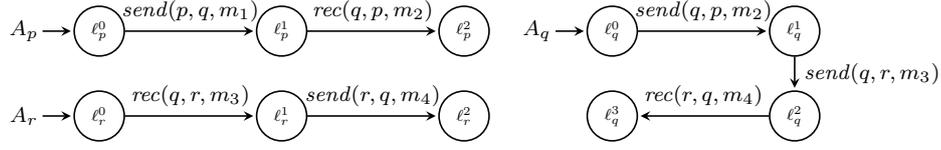
Given $p \in \mathbb{P}$ and a transition $t = (\ell, a, \ell') \in \delta_p$, we let $\text{source}(t) = \ell$, $\text{target}(t) = \ell'$, $\text{action}(t) = a$, and $\text{msg}(t) = m$ if $a \in \text{Send}(_, _, m) \cup \text{Rec}(_, _, m)$.

There are in general two ways to define the semantics of a communicating system. Most often it is defined as a global infinite transition system that keeps track of the various local control states and all (unbounded) channel contents. As, in this paper, our arguments are based on a graph view of MSCs, we will define the language of \mathcal{S} directly as a set of MSCs. These two semantic views are essentially equivalent, but they have different advantages depending on the context. We refer to [1] for a thorough discussion.

Let $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda)$ be an MSC. A *run* of \mathcal{S} on M is a mapping $\rho : \mathcal{E} \rightarrow \bigcup_{p \in \mathbb{P}} \delta_p$ that assigns to every event e the transition $\rho(e)$ that is executed at e . Thus, we require that

- (i) for all $e \in \mathcal{E}$, we have $\text{action}(\rho(e)) = \lambda(e)$,
- (ii) for all $(e, f) \in \rightarrow$, $\text{target}(\rho(e)) = \text{source}(\rho(f))$,
- (iii) for all $(e, f) \in \triangleleft$, $\text{msg}(\rho(e)) = \text{msg}(\rho(f))$, and
- (iv) for all $p \in \mathbb{P}$ and $e \in \mathcal{E}_p$ such that there is no $f \in \mathcal{E}$ with $f \rightarrow e$, we have $\text{source}(\rho(e)) = \ell_p^0$.

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



■ **Figure 2** System \mathcal{S}_1 .

Note that, following [8, 14], we do not consider final states or final configurations, as our purpose is to reason about all possible traces that can be *generated* by \mathcal{S} . The next lemma is obvious for the p2p semantics and follows from Lemma 3 for the mailbox semantics.

► **Lemma 5.** For all $\text{com} \in \{\text{p2p}, \text{mb}\}$, $L_{\text{com}}(\mathcal{S})$ is prefix-closed: $\text{Pref}(L_{\text{com}}(\mathcal{S})) \subseteq L_{\text{com}}(\mathcal{S})$.

► **Example 6.** Fig. 2 depicts $\mathcal{S}_1 = (A_p, A_q, A_r)$ such that MSC M_1 in Fig. 1 belongs to $L_{\text{p2p}}(\mathcal{S}_1)$ and to $L_{\text{mb}}(\mathcal{S}_1)$. There is a unique run ρ of \mathcal{S}_1 on M_1 . We can see that $(e'_3, e_4) \in \rightarrow$ and $\text{target}(\rho(e'_3)) = \text{source}(\rho(e_4)) = \ell_r^1$, $(e_2, e'_2) \in \triangleleft_{M_1}$, and $\text{msg}(\rho(e_2)) = \text{msg}(\rho(e'_2)) = m_2$.

2.3 Conflict Graph

We now recall the notion of a conflict graph associated to an MSC defined in [8]. This graph is used to depict the causal dependencies between message exchanges. Intuitively, we have a dependency whenever two messages have a process in common. For instance, an \xrightarrow{SS} dependency between message exchanges v and v' expresses the fact that v' has been sent after v , by the same process. This notion is of interest because it was seen in [8] that the notion of synchronizability in MSCs (which is studied in this paper) can be graphically characterized by the nature of the associated conflict graph. It is defined in terms of linearizations in [14], but we equivalently express it directly in terms of MSCs.

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

► **Definition 7 (Conflict graph).** The conflict graph $\text{CG}(M)$ of an MSC $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda)$ is the labeled graph $(\text{Nodes}, \text{Edges})$, with $\text{Edges} \subseteq \text{Nodes} \times \{S, R\}^2 \times \text{Nodes}$, defined by $\text{Nodes} = \triangleleft \cup \text{Unm}(M)$ and $\text{Edges} = \{(\mu(e), \tau(e)\tau(f), \mu(f)) \mid (e, f) \in \rightarrow^+\}$. In particular, a node of $\text{CG}(M)$ is either a single unmatched send event or a message pair $(e, e') \in \triangleleft$.

3 Model Checking and Synchronizability

In this section, we survey two classical decision problems for communicating systems. The first problem is the model-checking problem, in which one checks whether a given system satisfies a given specification. A canonical specification language for MSCs is monadic second-order (MSO) logic. However, model checking in full generality is undecidable. A common approach is, therefore, to restrict the behavior of the given system to MSCs of bounded (special) tree-width. Next, we introduce MSO logic and special tree-width.

3.1 Logic and Special Tree-Width

Monadic Second-Order Logic. The set of MSO formulas over MSCs (over \mathbb{P} and \mathbb{M}) is given by the grammar $\varphi ::= x \rightarrow y \mid x \triangleleft y \mid \lambda(x) = a \mid x = y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \varphi \mid \neg\varphi$, where $a \in \Sigma$, x and y are first-order variables, interpreted as events of an MSC, and X is a

$$\begin{array}{ll}
M \models E\sigma \text{ if } \llbracket \sigma \rrbracket_M \neq \emptyset & \llbracket \rightarrow \rrbracket_M := \rightarrow \text{ and } \llbracket \triangleleft \rrbracket_M := \triangleleft \\
\llbracket a \rrbracket_M & := \{e \in \mathcal{E} \mid \lambda(e) = a\} & \llbracket \text{test}(\sigma) \rrbracket_M & := \{(e, e) \mid e \in \llbracket \sigma \rrbracket_M\} \\
\llbracket \langle \pi \rangle \sigma \rrbracket_M & := \{e \in \mathcal{E} \mid \exists f \in \llbracket \sigma \rrbracket_M : (e, f) \in \llbracket \pi \rrbracket_M\} & \llbracket \text{jump} \rrbracket_M & := \mathcal{E} \times \mathcal{E} \\
\llbracket \text{Loop}(\pi) \rrbracket_M & := \{e \in \mathcal{E} \mid (e, e) \in \llbracket \pi \rrbracket_M\} & \llbracket \pi_1 + \pi_2 \rrbracket_M & := \llbracket \pi_1 \rrbracket_M \cup \llbracket \pi_2 \rrbracket_M \\
\llbracket \pi^{-1} \rrbracket_M & := \{(e, f) \in \mathcal{E} \times \mathcal{E} \mid (f, e) \in \llbracket \pi \rrbracket_M\} & \llbracket \pi^* \rrbracket_M & := \bigcup_{n \in \mathbb{N}} \llbracket \pi \rrbracket_M^n \\
\llbracket \pi_1 \cdot \pi_2 \rrbracket_M & := \{(e, f) \in \mathcal{E} \times \mathcal{E} \mid \exists g \in \mathcal{E} : (e, g) \in \llbracket \pi_1 \rrbracket_M \text{ and } (g, f) \in \llbracket \pi_2 \rrbracket_M\}
\end{array}$$

■ **Figure 3** Semantics of LCPDL.

second-order variable, interpreted as a set of events. We assume that we have an infinite supply of variables, and we use common abbreviations such as \wedge , \forall , etc. The satisfaction relation is defined in the standard way and self-explanatory. For example, the formula $\neg \exists x. (\bigvee_{a \in \text{Send}(_, _, _)} \lambda(x) = a \wedge \neg \text{matched}(x))$ with $\text{matched}(x) = \exists y. x \triangleleft y$ says that there are no unmatched send events. It is not satisfied by MSC M_1 of Fig. 1, as message m_1 is not received, but by M_4 from Fig. 6.

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

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

Propositional Dynamic Logic (PDL). For better complexity, we also consider PDL with Loop and Converse, henceforth called LCPDL (cf. [6, 7, 27] for more details). Its syntax is:

$$\begin{array}{ll}
\Phi ::= E\sigma \mid \Phi \vee \Phi \mid \neg \Phi & \text{(sentence)} \\
\sigma ::= a \mid \sigma \vee \sigma \mid \neg \sigma \mid \langle \pi \rangle \sigma \mid \text{Loop}(\pi) & \text{(event formula)} \\
\pi ::= \rightarrow \mid \triangleleft \mid \text{test}(\sigma) \mid \text{jump} \mid \pi + \pi \mid \pi \cdot \pi \mid \pi^* \mid \pi^{-1} & \text{(path formula)}
\end{array}$$

where $a \in \Sigma$. We use the symbol \top to denote a tautology event formula (such as $a \vee \neg a$). We describe the semantics for the logic in Fig. 3 (apart from the obvious cases). A sentence Φ is evaluated wrt. an MSC $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda)$. An event formula σ is evaluated wrt. M and an event $e \in \mathcal{E}$ so that it defines a unary relation $\llbracket \sigma \rrbracket_M \subseteq \mathcal{E}$. Finally, a path formula π is evaluated over two events, and so it defines a binary relation $\llbracket \pi \rrbracket_M \subseteq \mathcal{E} \times \mathcal{E}$. Finally, we let $L(\Phi) = \{M \in \text{MSC} \mid M \models \Phi\}$. Note that every LCPDL-definable property is MSO-definable.

It can be seen below that the mailbox semantics can be readily translated into the LCPDL formula $\Phi_{\text{mb}} = \neg E(\text{Loop}(\langle \triangleleft + \rightarrow + \sqsubset \rangle^+))$ such that $L(\Phi_{\text{mb}}) = \text{MSC}_{\text{mb}}$. Hereby, we let

$$\sqsubset = \triangleleft \cdot \rightarrow^+ \cdot \triangleleft^{-1} + \sum_{\substack{q \in \mathbb{P} \\ a, b \in \text{Send}(_, q, _)}} \text{test}(a) \cdot \triangleleft \cdot \text{jump} \cdot \text{test}(b \wedge \neg \langle \triangleleft \rangle \top).$$

Special Tree-Width. *Special tree-width* [12], is a graph measure that indicates how close a graph is to a tree (we may also use classical *tree-width* instead). This or similar measures are commonly employed in verification. For instance, tree-width and split-width have been used in [26] and, respectively, [2, 13] to reason about graph behaviors generated by pushdown and queue systems. There are several ways to define the special tree-width of an MSC. We adopt the following game-based definition from [7].

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

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

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

► **Fact 8** ([7]). *The special tree-width of an MSC is the least k such that the associated game is k -winning for Eve.*

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

3.2 Model Checking

In general, even simple verification problems, such as control-state reachability, are undecidable for communicating systems [9]. However, they are decidable when we restrict to behaviors of bounded special tree-width, which motivates the following definition of a generic **bounded model-checking problem** for $\text{com} \in \{\text{p2p}, \text{mb}\}$:

Input: Two finite sets \mathbb{P} and \mathbb{M} , a communicating system \mathcal{S} , an MSO sentence φ , and $k \in \mathbb{N}$ (given in unary).

Question: Do we have $L_{\text{com}}(\mathcal{S}) \cap \text{MSC}^{k\text{-stw}} \subseteq L(\varphi)$?

► **Fact 9** ([7]). *The bounded model-checking problem for $\text{com} = \text{p2p}$ is decidable. When the formulas φ are from LCPDL, then the problem is solvable in exponential time.*

Note that [7] does not employ the LCPDL modality jump, but it can be integrated easily. Using φ_{mb} or Φ_{mb} , we obtain the corresponding result for mailbox systems as a corollary:

► **Theorem 10.** *The bounded model-checking problem for $\text{com} = \text{mb}$ is decidable. When the formulas φ are from LCPDL, then the problem is solvable in exponential time.*

3.3 Synchronizability

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

■ **Table 1** Summary of the decidability of the synchronizability problem in various classes.

	PEER-TO-PEER	MAILBOX
Weakly synchronous	Undecidable [Thm. 22]	EXPTIME [Thm. 21]
Weakly k -synchronous	Decidable [8, 14] and [Thm. 29]	
Strongly k -synchronous	—	Decidable [Thm. 35]
Existentially k -p2p-bounded	Decidable [18, Prop. 5.5]	
Existentially k -mailbox-bounded	—	Decidable [Prop. 40]

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

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

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

► **Lemma 11.** *Let \mathcal{S} be a communicating system, $\text{com} \in \{\text{p2p}, \text{mb}\}$, $k \in \mathbb{N}$, and $\mathcal{C} \subseteq \text{MSC}^{k\text{-stw}}$. Then, $L_{\text{com}}(\mathcal{S}) \subseteq \mathcal{C}$ iff $L_{\text{com}}(\mathcal{S}) \cap \text{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$.*

The result follows from the following lemma. Note that a similar property was shown in [18, Proposition 5.4] for the specific class of existentially k -bounded MSCs.

► **Lemma 12.** *Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \text{MSC}^{k\text{-stw}}$. For all $M \in \text{MSC} \setminus \mathcal{C}$, we have $(\text{Pref}(M) \cap \text{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset$.*

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

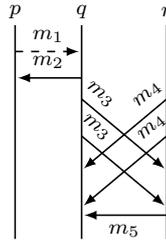
► **Theorem 13.** *Fix finite sets \mathbb{P} and \mathbb{M} . Suppose $\text{com} \in \{\text{p2p}, \text{mb}\}$ and let $\mathcal{C} \subseteq \text{MSC}$ be an MSO-definable and STW-bounded class (over \mathbb{P} and \mathbb{M}). The following problem is decidable: Given a communicating system \mathcal{S} , do we have $L_{\text{com}}(\mathcal{S}) \subseteq \mathcal{C}$?*

Proof. Consider the MSO-formula φ such that $L(\varphi) = \mathcal{C}$, and let $k \in \mathbb{N}$ such that $\mathcal{C} \subseteq \text{MSC}^{k\text{-stw}}$. We have $L_{\text{com}}(\mathcal{S}) \subseteq \mathcal{C} \stackrel{\text{Lemma 11}}{\iff} L_{\text{com}}(\mathcal{S}) \cap \text{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C} \iff L_{\text{com}}(\mathcal{S}) \cap \text{MSC}^{(k+2)\text{-stw}} \subseteq L(\varphi)$. The latter can be solved thanks to Fact 9 and Theorem 10. ◀

► **Remark 14.** Note that, in some cases (cf. Section 4), \mathbb{P} and \mathbb{M} are part of the input and the concrete class \mathcal{C} may be parameterized by a natural number so that it is part of the input, too. Then, we need to be able to compute the MSO formula characterizing the class as well as the bound on the special tree-width.

4 Application to Concrete Classes of Synchronizability

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



■ **Figure 4** MSC M_2 .

4.1 A New General Class: Weakly Synchronous MSCs

We first introduce the class of weakly synchronous MSCs. This is a generalization of synchronous MSCs studied earlier, in [8, 14], which we shall discuss later. We say an MSC is weakly synchronous if it is breakable into *exchanges* where an exchange is an MSC that allows one to schedule all sends before all receives. Let us define this formally:

► **Definition 15** (exchange). *Let $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda)$ be an MSC. We say that M is an exchange if $\text{SendEv}(M)$ is a \leq_M -downward-closed set.*

► **Definition 16** (weakly synchronous). *We say that $M \in \text{MSC}$ is weakly synchronous if it is of the form $M = M_1 \cdot \dots \cdot M_n$ such that every M_i is an exchange.*

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

► **Example 17.** Consider the MSC M_2 in Fig. 4. It is weakly synchronous. Indeed, m_1 , m_2 , and m_5 are independent and can be put alone in an exchange. Repetitions of m_3 and m_4 are interlaced, but they constitute an exchange, as we can do all sends and then all receptions.

An easy adaptation of a characterization from [14] yields the following result for weakly synchronous MSCs:

► **Proposition 18.** *Let M be an MSC. Then, M is weakly synchronous iff no RS edge occurs on any cyclic path in the conflict graph $\text{CG}(M)$.*

It is easily seen that the characterization from Proposition 18 is LCPDL-definable:

► **Corollary 19.** *The sets of weakly synchronous MSCs and weakly synchronous mailbox MSCs are LCPDL-definable. Both formulas have polynomial size.*

Moreover, under the mailbox semantics, we can show:

► **Proposition 20.** *The set of weakly synchronous mailbox MSCs is STW-bounded (in fact, it is included in $\text{MSC}^{4|\mathbb{P}|-\text{stw}}$).*

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

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

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

also places (at most) n pebbles on the last send event of each process and also (at most) n pebbles on the first receive event of each process. Eve erases the (at most) n \rightarrow -edges between the last send event and the first receive event.

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

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

- if e is an unmatched send of process p , Eve places her last pebble on the next send event of p (if it exists), let us call it e' . Then Eve erases the \rightarrow -edge (e, e') , and now e is completely disconnected, so it can be removed and the pebble can be taken back.
- if $e \triangleleft e'$, with e' a receive event of process q , then due to the mailbox semantics e' is the first receive event of q , so it has a pebble placed on it. Eve removes the \triangleleft -edge between e and e' , then using the extra pebble she disconnects e and places a pebble on the \rightarrow -successor of e , then she also disconnects e' and places a pebble on the \rightarrow -successor of e' .

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

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

► **Theorem 21.** *The following problem is decidable in exponential time: Given \mathbb{P} , \mathbb{M} , and a communicating system \mathcal{S} (over \mathbb{P} and \mathbb{M}), is every MSC in $L_{\text{mb}}(\mathcal{S})$ weakly synchronous?*

Proof. According to Corollary 19, we determine the LCPDL formula Φ_{wsmb} such that $L(\Phi_{\text{wsmb}})$ is the set of weakly synchronous mailbox MSCs. Moreover, recall from Proposition 20 that the special tree-width of all weakly synchronous mailbox MSCs is bounded by $4|\mathbb{P}|$. By Lemma 11, $L_{\text{mb}}(\mathcal{S}) \subseteq L(\Phi_{\text{wsmb}})$ iff $L_{\text{mb}}(\mathcal{S}) \cap \text{MSC}^{(4|\mathbb{P}|+2)\text{-stw}} \subseteq L(\Phi_{\text{wsmb}})$. The latter is an instance of the bounded model-checking problem. As the length of Φ_{wsmb} is polynomial in $|\mathbb{P}|$, we obtain that the original problem is decidable in exponential time by Theorem 10. ◀

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

► **Theorem 22.** *The following problem is undecidable: Given finite sets \mathbb{P} and \mathbb{M} as well as a communicating system \mathcal{S} , is every MSC in $L_{\text{p2p}}(\mathcal{S})$ weakly synchronous?*

4.2 Weakly k -Synchronous MSCs

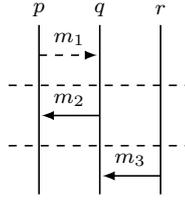
This negative result for the p2p semantics motivates the study of other classes. In fact, our framework captures several classes introduced in the literature.

► **Definition 23** (k -exchange). *Let $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda)$ be an MSC and $k \in \mathbb{N}$. We call M a k -exchange if M is an exchange and $|\text{SendEv}(M)| \leq k$.*

Let us now recall the definition from [8, 14], but (equivalently) expressed directly in terms of MSCs rather than via *executions*. It differs from the weakly synchronous MSCs in that here, we insist on constraining the number of messages sent per exchange to be at most k .

► **Definition 24** (weakly k -synchronous). *Let $k \in \mathbb{N}$. We say that $M \in \text{MSC}$ is weakly k -synchronous if it is of the form $M = M_1 \cdot \dots \cdot M_n$ such that every M_i is a k -exchange.*

► **Example 25.** MSC M_3 in Fig. 5 is weakly 1-synchronous, as it can be decomposed into three 1-exchanges (the decomposition is depicted by the horizontal dashed lines). We remark that $M_3 \in \text{MSC}_{\text{mb}}$. Note that there is a p2p linearization that respects the decomposition.



■ **Figure 5** MSC M_3 .

On the other hand, a mailbox linearization needs to reorganize actions from different MSCs: the sending of m_3 needs to be done before the sending of m_1 . Note that M_1 in Fig. 1 is also weakly 1-synchronous.

► **Proposition 26.** *Let $k \in \mathbb{N}$. The set of weakly k -synchronous p2p (mailbox, respectively) MSCs is effectively MSO-definable.*

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

► **Theorem 27** ([14]). *Let M be an MSC. Then, M is weakly k -synchronous iff every SCC in its conflict graph $\text{CG}(M)$ is of size at most k and no RS edge occurs on any cyclic path.*

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

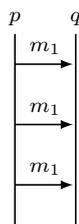
► **Proposition 28.** *Let $k \in \mathbb{N}$. The set of MSCs that are weakly k -synchronous have special tree-width bounded by $2k + |\mathbb{P}|$.*

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

► **Theorem 29** ([8, 14]). *For $\text{com} \in \{\text{p2p}, \text{mb}\}$, the following problem is decidable: Given finite sets \mathbb{P} and \mathbb{M} , a communicating system \mathcal{S} , and $k \in \mathbb{N}$, is every MSC in $L_{\text{com}}(\mathcal{S})$ weakly k -synchronous?*

Proof. We proceed similarly to the proof of Theorem 21. For the given \mathbb{P} , \mathbb{M} , and k , we first determine, using Proposition 26, the MSO formula φ_k such that $L(\varphi_k)$ is the set of weakly k -synchronous p2p/mailbox MSCs. From Proposition 28, we know that the special tree-width of all weakly k -synchronous MSCs is bounded by $2k + |\mathbb{P}|$. By Lemma 11, we have $L_{\text{com}}(\mathcal{S}) \subseteq L(\varphi_k)$ iff $L_{\text{com}}(\mathcal{S}) \cap \text{MSC}^{(2k+|\mathbb{P}|+2)\text{-stw}} \subseteq L(\varphi_k)$. The latter is an instance of the bounded model-checking problem. By Fact 9 and Theorem 10, we obtain decidability. ◀

► **Remark 30.** The set of weakly k -synchronous MSCs is not directly expressible in LCPDL (the reason is that LCPDL does not have a built-in counting mechanism). However, its complement is expressible in the extension of LCPDL with existentially quantified propositions (we need $k + 1$ of them). The model-checking problem for this kind of property is still in EXPTIME and, therefore, so is the problem from Theorem 29 when k is given in unary. It is very likely that our approach can also be used to infer the PSPACE upper bound from [8] by showing bounded path width and using finite word automata instead of tree automata. Finally, note that the problem to decide whether there exists an integer $k \in \mathbb{N}$ such that all MSCs in $L_{\text{com}}(\mathcal{S})$ are weakly k -synchronous has recently been studied in [20] and requires different techniques.



■ **Figure 6** MSC M_4 .

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

► **Definition 31** (modified k -exchange). *Let $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda)$ be an MSC and $k \in \mathbb{N}$. We call M a modified k -exchange if $|\text{SendEv}(M)| \leq k$.*

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

4.3 Strongly k -Synchronous MSCs and Other Classes

Our framework can be applied to a variety of other classes. Here we show how the decidability results can be shown for a variant of the class of weakly k -synchronous MSCs.

► **Definition 32.** *Let $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda) \in \text{MSC}_{\text{mb}}$. We call M strongly k -synchronous if it can be written as $M = M_1 \cdot \dots \cdot M_n$ such that every MSC $M_i = (\mathcal{E}_i, \rightarrow_i, \triangleleft_i, \lambda_i)$ is a k -exchange and, for all $(e, f) \in \sqsubset_M$, there are $1 \leq i \leq j \leq n$ such that $e \in \mathcal{E}_i$ and $f \in \mathcal{E}_j$.*

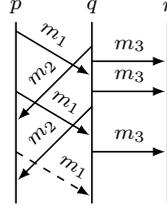
► **Example 33.** MSC $M_4 \in \text{MSC}_{\text{mb}}$ in Fig. 6 is strongly 1-synchronous. Indeed, we can decompose it into 1-exchanges and this decomposition allows for a total order compatible with \sqsubset_{M_4} . Moreover, MSC M_3 in Fig. 5, which is weakly 1-synchronous, is strongly 3-synchronous. Indeed, we need to put the three messages in the same k -exchange to regain our total order. Finally, for all k , MSC M_1 in Fig. 1 is not strongly k -synchronous, as we cannot put all messages in the same k -exchange, where all sends are followed by all receptions. Here, this is not possible as the reception of m_3 has to take place before the sending of m_4 .

► **Proposition 34.** *For all $k \in \mathbb{N}$, the set of strongly k -synchronous mailbox MSCs is MSO-definable and STW-bounded.*

The proof proceeds similarly to what has been shown in the previous cases, but MSO-definability now relies on an *extended* conflict graph. As a corollary, we thus obtain:

► **Theorem 35.** *The following problem is decidable: Given finite sets \mathbb{P} and \mathbb{M} , a communicating system \mathcal{S} , and $k \in \mathbb{N}$, is every MSC in $L_{\text{mb}}(\mathcal{S})$ strongly k -synchronous?*

► **Remark 36.** Only mailbox MSCs are considered for the definition of strongly k -synchronous MSCs for the following reason: A natural p2p analogue of Definition 32 would require from the decomposition that, for all $(e, f) \in \sqsubset_M$, there are indices $1 \leq i \leq j \leq n$ such that $e \in \mathcal{E}_i$ and $f \in \mathcal{E}_j$. But this is always satisfied. So the natural definition of “strongly k -synchronous MSCs” would coincide with weakly k -synchronous MSCs.



■ **Figure 7** MSC M_5 .

Like the variant for the case of weakly synchronous MSCs, we can also generalize strongly k -synchronous MSCs by removing the restriction on the number of messages per exchange:

► **Definition 37.** Let $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda) \in \text{MSC}_{\text{mb}}$. We call M strongly synchronous if it can be written as $M = M_1 \cdot \dots \cdot M_n$ such that every MSC $M_i = (\mathcal{E}_i, \rightarrow_i, \triangleleft_i, \lambda_i)$ is an exchange and, for all $(e, f) \in \sqsubset_M$, there are indices $1 \leq i \leq j \leq n$ such that $e \in \mathcal{E}_i$ and $f \in \mathcal{E}_j$.

Similarly to the constructions for strongly k -synchronous MSCs, we can obtain a graphical characterization where we only look for the absence of RS -edges in a cycle. Hence, this class is also MSO-definable (in fact, even LCPDL-definable) and STW-bounded.

4.4 Existentially k -Bounded MSCs

Now, we turn to existentially k -bounded MSCs [18,19,24]. Synchronizability has been studied for the p2p case in [18], so we only consider the mailbox case here. A linearization \rightsquigarrow of an MSC $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda) \in \text{MSC}$ is called k -mailbox-bounded if, for all $e \in \text{Matched}(M)$, say with $\lambda(e) = \text{send}(p, q, m)$, we have $\#\text{Send}(_, q, _)(\rightsquigarrow, e) - \#\text{Rec}(_, q, _)(\rightsquigarrow, e) \leq k$.

► **Definition 38.** Let $M = (\mathcal{E}, \rightarrow, \triangleleft, \lambda) \in \text{MSC}$ and $k \in \mathbb{N}$. We call M existentially k -mailbox-bounded if it has some mailbox linearization that is k -mailbox-bounded.

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

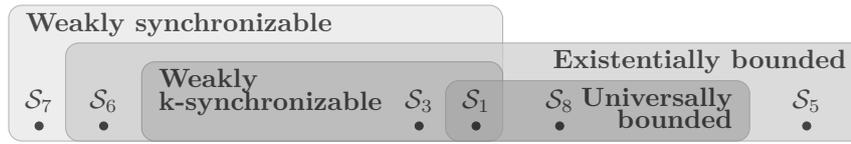
► **Example 39.** MSC M_5 in Fig. 7 is existentially 1-mailbox-bounded, as witnessed by the (informally given) linearization $s(q, p, m_2) \rightsquigarrow s(p, q, m_1) \rightsquigarrow s(q, r, m_3) \rightsquigarrow r(q, r, m_3) \rightsquigarrow r(p, q, m_1) \rightsquigarrow s(p, q, m_1) \rightsquigarrow r(q, p, m_2) \rightsquigarrow s(q, r, m_3) \dots$. Note that M_5 is neither weakly nor strongly synchronous as we cannot divide it into exchanges.

► **Proposition 40.** For all $k \in \mathbb{N}$, the set of existentially k -mailbox-bounded MSCs is MSO-definable and STW-bounded.

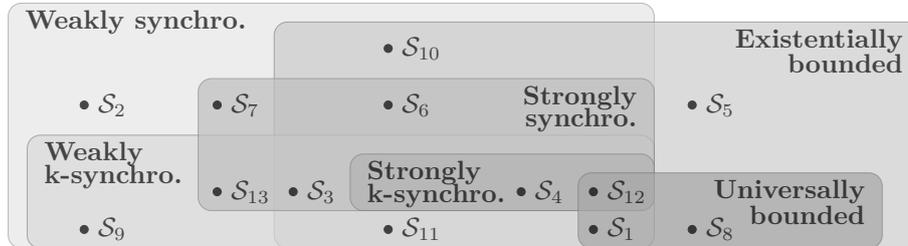
This extension is also valid for the p2p definition of existentially k -bounded MSCs, which were addressed in [18]. Finally, our framework can also be adapted to treat universally bounded systems [21,24].

5 Relations Between Classes

In this section we study how the classes introduced and recalled so far are related to each other. Notably, depending on the semantics (p2p or mailbox), we obtain two different classifications. The results are summed up in Figures 8 and 9. Here, we define existentially k -p2p-bounded MSCs and universally bounded counterparts as expected.



■ **Figure 8** Hierarchy of classes for p2p systems.



■ **Figure 9** Hierarchy of classes for mailbox systems.

To refer to those systems we use the following terminology: a system \mathcal{S} is called weakly synchronizable (resp. strongly synchronizable) if all MSCs M in the respective language are weakly synchronous (resp. strongly synchronous). A system is called weakly k -synchronizable (resp. strongly k -synchronizable, existentially bounded or universally bounded) if all MSCs are weakly k -synchronous (resp. strongly k -synchronous, existentially k -bounded or universally k -bounded). A similar comparison relating existentially bounded systems, weakly k -synchronizable systems, as well as other systems that have not been described here, can also be found in [23] for p2p systems.

We give some results showing the inclusion of certain classes. Recall that strong k -synchronizability is tailored to mailbox systems (cf. also Remark 36) so that, for p2p systems, we only consider the case of weak (k -)synchronizability.

► **Proposition 41.** *Every weakly k -synchronous MSC is existentially k -p2p-bounded. Moreover, every strongly k -synchronous mailbox MSC is existentially k -mailbox-bounded.*

Finally, if a system is weakly synchronizable and universally k -bounded then, there is a k' such that it is also weakly k' -synchronizable. The equivalent property is also valid for strong classes.

► **Proposition 42.** *Every weakly (resp. strongly) synchronizable and universally k -bounded system is weakly (resp. strongly) k' -synchronizable for a k' .*

6 Conclusion and Perspectives

We have presented a unifying framework based on MSO logic and (special) tree-width, that brings together existing definitions, explains their good properties, and allows one to easily derive other, more general definitions and decidability results for synchronizability. Let us notice that the send-synchronizability does not fit in our framework because the question $L_{p2p}(\mathcal{S}) \subseteq \mathcal{C}_0$ would be decidable (by Theorem 13), where \mathcal{C}_0 is the set of send-synchronizable MSCs, but this property is equivalent to checking whether the system \mathcal{S} is send-synchronizable and this last property is undecidable [16].

Many other related questions could be studied in the future. For example, we could think about the hypotheses to add to our general framework to make the problem “does there exist an $k \geq 0$ such that $L_{p2p}(\mathcal{S}) \subseteq \mathcal{C}_k$?” decidable. From very recent work [20], one

knows that the problem “does there exist an $k \geq 0$ such that the system is (weakly/strongly) k -synchronizable?” is decidable; but it remains to be seen if it would be possible to obtain these results by showing that these properties can be expressed in a decidable extension of our framework. Let us remark that the decidability of the question whether there exists an $k \geq 0$ such that $L_{\text{p2p}}(\mathcal{S}) \subseteq \mathcal{C}_k$ allows us to build a bounded model checking strategy by first deciding whether there exists such an $k \geq 0$ and then by testing if $L_{\text{p2p}}(\mathcal{S}) \subseteq \mathcal{C}_k$ for $k = 0, 1, 2, \dots$. One may use this strategy for weakly/strongly synchronizable systems, but not for existentially bounded systems (except for deadlock-free systems) or for deterministic deadlock-free universally bounded systems. In [23], Lange and Yoshida introduced an *asynchronous compatibility* property and it would also be interesting to verify whether this property could be expressed into our framework.

References

- 1 C. Aiswarya and Paul Gastin. Reasoning about distributed systems: WYSIWYG (invited talk). In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPICs*, pages 11–30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPICs.FSTTCS.2014.11.
- 2 C. Aiswarya, Paul Gastin, and K. Narayan Kumar. Verifying communicating multi-pushdown systems via split-width. In *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014*, volume 8837 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2014.
- 3 Samik Basu and Tefik Bultan. Choreography conformance via synchronizability. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 795–804. ACM, 2011. doi:10.1145/1963405.1963516.
- 4 Bernard Boigelot and Patrice Godefroid. Symbolic verification of communication protocols with infinite state spaces using qdds (extended abstract). In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification, 8th International Conference, CAV '96, New Brunswick, NJ, USA, July 31 - August 3, 1996, Proceedings*, volume 1102 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1996. doi:10.1007/3-540-61474-5_53.
- 5 Benedikt Bollig, Alain Finkel, and Amrita Suresh. Bounded reachability problems are decidable in FIFO machines. In Igor Konnov and Laura Kovacs, editors, *Proceedings of the 31st International Conference on Concurrency Theory (CONCUR'20)*, volume 171 of *Leibniz International Proceedings in Informatics*, pages 49:1–49:17, Vienna, Austria, 2020. Leibniz-Zentrum für Informatik. doi:10.4230/LIPICs.CONCUR.2020.49.
- 6 Benedikt Bollig, Marie Fortin, and Paul Gastin. Communicating finite-state machines, first-order logic, and star-free propositional dynamic logic. *J. Comput. Syst. Sci.*, 115:22–53, 2021.
- 7 Benedikt Bollig and Paul Gastin. Non-sequential theory of distributed systems. *CoRR*, abs/1904.06942, 2019. arXiv:1904.06942.
- 8 Ahmed Bouajjani, Constantin Enea, Kailiang Ji, and Shaz Qadeer. On the completeness of verifying message passing programs under bounded asynchrony. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, volume 10982 of *Lecture Notes in Computer Science*, pages 372–391. Springer, 2018. doi:10.1007/978-3-319-96142-2_23.
- 9 Daniel Brand and Pitro Zafropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983. doi:10.1145/322374.322380.

- 10 Gérard Cécé and Alain Finkel. Verification of programs with half-duplex communication. *Information and Computation*, 202(2):166–190, 2005. doi:10.1016/j.ic.2005.05.006.
- 11 Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996. URL: <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/CFP-IC96.ps>.
- 12 Bruno Courcelle. Special tree-width and the verification of monadic second-order graph properties. In *FSTTCS*, volume 8 of *LIPICs*, pages 13–29, 2010. doi:10.4230/LIPICs.FSTTCS.2010.13.
- 13 Aiswarya Cyriac, Paul Gastin, and K. Narayan Kumar. MSO decidability of multi-pushdown systems via split-width. In Maciej Koutny and Irek Ulidowski, editors, *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 547–561. Springer, 2012. doi:10.1007/978-3-642-32940-1_38.
- 14 Cinzia Di Giusto, Laetitia Laversa, and Étienne Lozes. On the k-synchronizability of systems. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Proceedings*, volume 12077 of *Lecture Notes in Computer Science*, pages 157–176. Springer, 2020. doi:10.1007/978-3-030-45231-5_9.
- 15 Javier Esparza, Pierre Ganty, and Rupak Majumdar. A perfect model for bounded verification. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science, LICS '12*, pages 285–294, Washington, DC, USA, 2012. IEEE Computer Society. doi:10.1109/LICS.2012.39.
- 16 Alain Finkel and Étienne Lozes. Synchronizability of communicating finite state machines is not decidable. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPICs*, pages 122:1–122:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
- 17 Alain Finkel and M. Praveen. Verification of Flat FIFO Systems. *Logical Methods in Computer Science*, 20(4), 2020. doi:10.23638/LMCS-16(4:4)2020.
- 18 Blaise Genest, Dietrich Kuske, and Anca Muscholl. On communicating automata with bounded channels. *Fundamenta Informaticae*, 80(1-3):147–167, 2007.
- 19 Blaise Genest, Anca Muscholl, and Dietrich Kuske. A kleene theorem for a class of communicating automata with effective algorithms. In Cristian Calude, Elena Calude, and Michael J. Dinneen, editors, *Developments in Language Theory, 8th International Conference, DLT 2004, Auckland, New Zealand, December 13-17, 2004, Proceedings*, volume 3340 of *Lecture Notes in Computer Science*, pages 30–48. Springer, 2004. doi:10.1007/978-3-540-30550-7_4.
- 20 Cinzia Di Giusto, Laetitia Laversa, and Étienne Lozes. Guessing the buffer bound for k-synchronizability. In *Implementation and Application of Automata - 25th International Conference, CIAA 2021, Proceedings*, *Lecture Notes in Computer Science*. Springer, 2021. To appear.
- 21 Jesper G. Henriksen, Madhavan Mukund, K. Narayan Kumar, Milind Sohoni, and P.S. Thiagarajan. A theory of regular msc languages. *Information and Computation*, 202(1):1–38, 2005.
- 22 Dietrich Kuske and Anca Muscholl. Communicating automata, 2014.
- 23 Julien Lange and Nobuko Yoshida. Verifying asynchronous interactions via communicating session automata. *CoRR*, abs/1901.09606, 2019. arXiv:1901.09606.
- 24 Markus Lohrey and Anca Muscholl. Bounded MSC communication. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 295–309. Springer, 2002. doi:10.1007/3-540-45931-6_21.

14:18 A Unifying Framework for Deciding Synchronizability

- 25 Markus Lohrey and Anca Muscholl. Bounded MSC communication. *Inf. Comput.*, 189(2):160–181, 2004. doi:10.1016/j.ic.2003.10.002.
- 26 P. Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 283–294. ACM, 2011.
- 27 Robert S. Streett. Propositional dynamic logic of looping and converse. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA*, pages 375–383. ACM, 1981.
- 28 Gregor von Bochmann. Communication protocols and error recovery procedures. *Operating Systems Review*, 9(3):45–50, 1975.