Stage M2

Davide Ferre'

March 20, 2022

1 Useful stuff

1.1 Message Sequence Charts

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

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

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

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

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

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

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

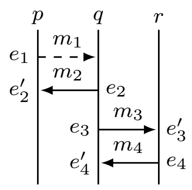

Figure 1: MSC M_1

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

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

We let $\preceq_M = (\rightarrow \cup \lhd \cup \sqsubset_M)^*$. Note that $\leq_M \subseteq \preceq_M$. We call $M \in \mathsf{MSC}$ 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 [4], we also say that a mailbox MSC satisfies *causal delivery.* The set of mailbox MSCs $M \in \mathsf{MSC}$ is denoted by $\mathsf{MSC}_{\mathsf{mb}}$.

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

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

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

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

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

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

1.2 Communicating Systems

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

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

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

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

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

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

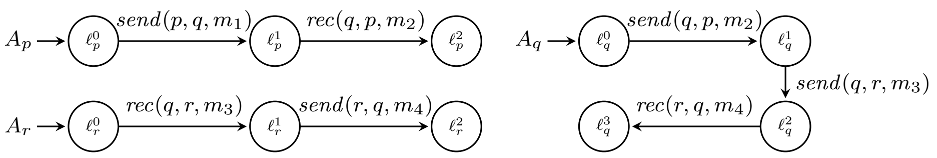

Figure 2: System $\mathcal{S}_1$

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

Note that, following [4, 8], we do not consider final states or final configurations, as our purpose is to reason about all possible traces that can be *generated* by $\mathcal{S}$. The next lemma is obvious for the p2p semantics and follows from Lemma 1.1 for the mailbox semantics.

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

Example 1.3. Fig. 2 depicts $\mathcal{S}_1 = (A_p, A_q, A_r)$ such that MSC M_1 in Fig. 1 belongs to $L_{\mathsf{p2p}}(\mathcal{S}_1)$ and to $L_{\mathsf{mb}}(\mathcal{S}_1)$. There is a unique run ρ of $\mathcal{S}_1$ on M_1. We can see that $(e_3', e_4) \in \rightarrow$ and $\mathit{target}(\rho(e_3')) = \mathit{source}(\rho(e_4)) = \ell_r^1$, $(e_2, e_2') \in \lhd_{M_1}$, and $\mathit{msg}(\rho(e_2)) = \mathit{msg}(\rho(e_2')) = m_2$.

1.3 Conflict Graph

We now recall the notion of a conflict graph associated to an MSC defined in [4]. 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 [4] 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 [8], but we equivalently express it directly in terms of MSCs.

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

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

1.4 Logic and Special Tree-Width

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

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

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

Special Tree-Width. *Special tree-width* [6], 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 [12] and, respectively, [7, 2] 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 [3].

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

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

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

Fact 1.3 ([3])**.** *The special tree-width of an MSC is the least k such that the associated game is k-winning for Eve.*

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

1.5 Model Checking

In general, even simple verification problems, such as control-state reachability, are undecidable for communicating systems [5]. However, they are decidable when we restrict to behaviors of bounded special tree-width, which motivates the following definition of a generic **bounded model-checking problem** for com $\in \{\mathsf{p2p}, \mathsf{mb}\}$:
Input: Two finite sets $\mathbb{P}$ and $\mathbb{M}$, a communicating system $\mathcal{S}$, an MSO sentence φ, and $k \in \mathbb{N}$ (given in unary).
Question: Do we have $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{k\text{-}\mathsf{stw}} \subseteq L(\varphi)$?

Fact 1.4 ([3])**.** *The bounded model-checking problem for* com $=$ $\mathsf{p2p}$ *is decidable. When the formulas φ are from LCPDL, then the problem is solvable in exponential time.*

Note that [3] does not employ the LCPDL modality jump, but it can be integrated easily. Using φ_{mb} or Φ_{mb}, we obtain the corresponding result for mailbox systems as a corollary:

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

1.6 Synchronizability

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

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

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

	Peer-to-Peer	Mailbox
Weakly synchronous	Undecidable [Thm. 1.10]	EXPTIME [Thm. 1.9]
Weakly k-synchronous	Decidable [4, 8] and [Thm. 1.12]	
Strongly k-synchronous	—	Decidable [Thm. ??]
Existentially k-p2p-bounded	Decidable [10, Prop. 5.5]	
Existentially k-mailbox-bounded	—	Decidable [Prop. ??]

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

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

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

The result follows from the following lemma. Note that a similar property was shown in [10, Proposition 5.4] for the specific class of existentially k-bounded MSCs.

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

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

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

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

1.7 Application to Concrete Classes of Synchronizability

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

1.8 A New General Class: Weakly Synchronous MSCs

We first introduce the class of weakly synchronous MSCs. This is a generalization of synchronous MSCs studied earlier, in [4, 8], which we shall discuss later. We say an MSC is weakly synchronous if it is breakable into *exchanges* where an exchange is an MSC that allows one to schedule all sends before all receives. Let us define this formally:

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

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

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

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

An easy adaptation of a characterization from [8] yields the following result for weakly synchronous MSCs:

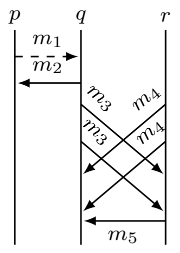

Figure 3: MSC M_2

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

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

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

Moreover, under the mailbox semantics, we can show:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Informally, the system works as follows:

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

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

- Prover1 is

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

- Prover2 is

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

- Verifier1 is

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

- Verifier2 is

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

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

1.9 Weakly k-Synchronous MSCs

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

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

Let us now recall the definition from [4, 8], but (equivalently) expressed directly in terms of MSCs rather than via *executions*. It differs from the weakly synchronous MSCs in that here, we insist on constraining the number of messages sent per exchange to be at most k.

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

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

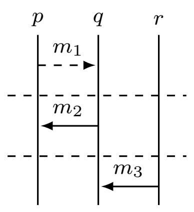

Figure 4: MSC M_3

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

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

Theorem 1.11 ([8]). *Let M be an MSC. Then, M is weakly k-synchronous iff every SCC in its conflict graph $\mathsf{CG}(M)$ is of size at most k and no RS edge occurs on any cyclic path.*

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

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

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

Theorem 1.12 ([4, 8]). *For $\mathsf{com} \in \{\mathsf{p2p}, \mathsf{mb}\}$, the following problem is decidable: Given finite sets $\mathbb{P}$ and $\mathbb{M}$, a communicating system $\mathcal{S}$, and $k \in \mathbb{N}$, is every MSC in $L_{\mathsf{com}}(\mathcal{S})$ weakly k-synchronous?*

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

Remark 1.1. The set of weakly k-synchronous MSCs is not directly expressible in LCPDL (the reason is that LCPDL does not have a built-in counting mechanism). However, its *complement* is expressible in the extension of LCPDL with existentially quantified propositions (we need $k+1$ of them). The model-checking problem for this kind of property is still in EXPTIME and, therefore, so is the problem from Theorem 1.12 when k is given in unary. It is very likely that our approach can also be used to infer the PSPACE upper bound from [4] by showing bounded *path width* and using finite word automata instead of tree automata. Finally, note that the problem to decide whether there exists an integer $k \in \mathbb{N}$ such that all MSCs in $L_{\mathsf{com}}(\mathcal{S})$ are weakly k-synchronous has recently been studied in [11] and requires different techniques.

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

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

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.

2 My stuff

D: Consider also unmatched messages in causal order communication (causal delivery using the terminology from Laetitia's thesis).

2.1 Semantics of causal ordering

Definition 2.1. Given a system $\mathcal{S} = (Loc_p, \delta_p, \ell_p^0)_{p\in\mathbb{P}}$ with n processes, a *configuration* is a tuple $(\overrightarrow{\ell}, \overrightarrow{\mathrm{Buf}}, \overrightarrow{\mathrm{Vec}})$, where $\overrightarrow{\ell} = (\ell_p)_{p\in\mathbb{P}}$ represents the global state of the system, $\overrightarrow{\mathrm{Buf}} = (b_p)_{p\in\mathbb{P}}$ is a vector of buffers, with each $b_p \in \mathbb{M}^*$ representing the content of the buffer of process p, and $\overrightarrow{\mathrm{Vec}} = (v_p)_{p\in\mathbb{P}}$ is a vector of Mattern-Fidge logical clocks, where each $v_p = (time_i)_{i\in\mathbb{P}}$ represents the content of the logical vector clock of process p.

2.2 Message Sequence Charts

Causally ordered MSCs An MSC is causally ordered if all the messages sent to the same process are received in an order which is consistent with the causal ordering of the corresponding send events. More formally, for an MSC $M = (\mathcal{E}, \to, \lhd, \lambda)$ we define an additional binary relation $\blacktriangleleft_M \subseteq \mathcal{E}\times\mathcal{E}$ that represents a constraint under the causal ordering semantics. In particular, given two receive events f_1 and f_2, we have that $f_1 \blacktriangleleft_M f_2$ if both the following hold:

- $\lambda(f_1) \in Rec(\_, q, \_)$, $\lambda(f_2) \in Rec(\_, q, \_)$
- $e_1 \lhd f_1$ and $e_2 \lhd f_2$ for some $e_1, e_2 \in \mathcal{E}$, such that $e_1 \leq_M e_2$.

We let $\lessdot_M = (\to \cup \lhd \cup \blacktriangleleft_M)^*$. Note that $\leq_M \subseteq \lessdot_M$. We call $M \in \mathsf{MSC}$ a *causally ordered (CO) MSC* if $\lessdot_M$ is a partial order. The set of causally ordered MSCs $M \in \mathsf{MSC}$ is denoted by $\mathsf{MSC}_{\mathsf{co}}$.

In literature, several possible implementations of causal ordering can be found. For instance, the algorithm described in [14] makes use of the logical vector clocks introduced by Mattern-Fidge [9, 13] to enforce causal ordering.

2.3 Communicating Systems

Lemma 2.1. *Every prefix of a causally ordered MSC is a causally ordered MSC.*

Proof. Let $M = (\mathcal{E}, \to, \lhd, \lambda) \in \mathsf{MSC}_{\mathsf{co}}$ and let $M_0 = (\mathcal{E}_0, \to_0, \lhd_0, \lambda_0)$ be a prefix of M. By contradiction, suppose that M_0 is not a causally ordered MSC. Then, there are distinct $e, f \in \mathcal{E}_0$ such that $e \lessdot_{M_0} f$ and $f \lessdot_{M_0} e$, with $\lessdot_{M_0} = (\to_0 \cup \lhd_0 \cup \blacktriangleleft_{M_0})^*$. As $\mathcal{E}_0 \subseteq \mathcal{E}$, we have that $\to_0 \subseteq \to$, $\lhd_0 \subseteq \lhd$, and $\blacktriangleleft_{M_0} \subseteq \blacktriangleleft_M$. Finally, we have that $\lessdot_{M_0} \subseteq \lessdot_M$ and M cannot be a causally ordered MSC, which is a contradiction. □

Lemma 1.2 can be easily extendend to com = co.

Lemma 2.2. *For all* com $\in \{\mathsf{p2p}, \mathsf{mb}, \mathsf{co}\}$, *$L_{\mathrm{com}}(\mathcal{S})$ is prefix-closed: $Pref(L_{\mathrm{com}}(\mathcal{S})) \subseteq L_{\mathrm{com}}(\mathcal{S})$.*

Proof. Follows from Lemma 2.1. □

2.4 Model Checking

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

Proof. The set of causally ordered MSCs can be defined using the MSO formula

$$\varphi_{\mathsf{co}} = \neg\exists x.\exists y. \left(\bigvee_{\substack{q\in\mathbb{P} \\ a,b\in Send(\_,q,\_)}} \lambda(x) = a \ \wedge\ \lambda(y) = b \ \wedge\ x \leq y \ \wedge\ \exists x'.\exists y'. \begin{pmatrix} x \lhd x' & \wedge \\ y \lhd y' & \wedge \\ y' \to^+ x' & \end{pmatrix} \right)$$

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

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

Theorem 2.3. *The bounded model-checking problem for* com = co *is decidable.*

Table 2: Summary of the decidability of the synchronizability problem in various classes

	P2P	Causal ordering	Mailbox
Weakly synchronous	Undecidable [Thm. 1.10]	Undecidable [Thm. 2.6]	EXPTIME [Thm. 1.9]
Weakly k-synchronous	Decidable [Thm. 2.7]		

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

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

The latter is decidable due to Fact 1.4. □

2.5 Synchronizability

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

Lemma 2.4. *Let $\mathcal{S}$ be a communicating system,* com $\in \{\mathsf{p2p}, \mathsf{mb}, \mathsf{co}\}$, $k \in \mathbb{N}$, *and* $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-}\mathsf{stw}}$. *Then,* $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$ *iff* $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-}\mathsf{stw}} \subseteq \mathcal{C}$.

Theorem 1.8 can also be extended to com $=$ co.

Theorem 2.5. *Fix finite sets $\mathbb{P}$ and $\mathbb{M}$. Suppose* com $\in \{\mathsf{p2p}, \mathsf{mb}, \mathsf{co}\}$ *and let $\mathcal{C} \subseteq \mathsf{MSC}$ be an MSO-definable and STW-bounded class (over $\mathbb{P}$ and $\mathbb{M}$). The following problem is decidable: Given a communicating system $\mathcal{S}$, do we have $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$?*

Proof. Same as the proof for Theorem 1.8, but using Lemma 2.4 in place of Lemma 1.6, and Theorem 2.3 in place of Theorem 1.5. □

2.5.1 Weakly synchronous causally ordered MSCs

Corollary 1.8.1 can be extended to com $=$ co.

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

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

Theorem 2.6. *The following problem is undecidable: Given finite sets $\mathbb{P}$ and $\mathbb{M}$ as well as a communicating system $\mathcal{S}$, is every MSC in $L_{\mathsf{co}}(\mathcal{S})$ weakly synchronous?*

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

Proposition 2.3. The set of weakly synchronous causally ordered MSCs has unbounded special tree-width.

Proof. Suppose that the set of weakly synchronous causally ordered MSCs is STW-bounded. By Proposition 2.2 and Theorem 2.5, we have that the syncronicity problem for the class of weakly synchronous causally ordered MSCs would be decidable. This is a contradiction, since Theorem 2.6 states that this problem is undecidable. □

[1]Notice that there is no channel between P1 and P2, and we only have unidirectional communication channels from provers to verifiers.

2.5.2 Weakly k-synchronous causally ordered MSCs

Proposition 2.4. The set of weakly k-synchronous causally ordered MSCs is MSO-definable.

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

Theorem 1.8 can be easily extended to com = co.

Theorem 2.7. *For* com $\in \{\mathsf{p2p}, \mathsf{mb}, \mathsf{co}\}$, *the following problem is decidable: Given finite sets* $\mathbb{P}$ *and* $\mathbb{M}$, *a communicating system* $\mathcal{S}$, *and* $k \in \mathbb{N}$, *is every MSC in* $L_{\text{com}}(\mathcal{S})$ *weakly* k*-synchronous?*

Proof. By Proposition 2.4 and Proposition 1.4 we have that the class of causally ordered k-synchronous MSCs is MSO-definable and STW-bounded[2]. Theorem 2.5 ends the proof. □

References

[1] C. Aiswarya and Paul Gastin. “Reasoning About Distributed Systems: WYSIWYG (Invited Talk)”. In: *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*. Ed. by Venkatesh Raman and S. P. Suresh. Vol. 29. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014, pp. 11–30. DOI: 10.4230/LIPIcs.FSTTCS.2014.11. URL: https://doi.org/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*. Vol. 8837. Lecture Notes in Computer Science. Springer, 2014, pp. 1–17.

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

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

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

[6] Bruno Courcelle. “Special tree-width and the verification of monadic second-order graph properties”. In: *FSTTCS*. Vol. 8. LIPIcs. 2010, pp. 13–29.

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

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

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

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

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

[11] 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. To appear. Springer, 2021.

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

[13] Friedemann Mattern. “Virtual time and global states of distributed systems.” In: *Proc. Workshop on Parallel and Distributed Algorithms,* North-Holland / Elsevier, 1989, pp. 215–226.

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