# Demystifying asynchronous communication and its variants*

Subtitles†

ANONYMOUS AUTHOR(S)

Text of abstract ....

Additional Key Words and Phrases: keyword1, keyword2, keyword3

**ACM Reference Format:**
Anonymous Author(s). 2018. Demystifying asynchronous communication and its variants: Subtitles. *Proc. ACM Program. Lang.* 1, CONF, Article 1 (January 2018), 30 pages.

## 1 INTRODUCTION

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

## 2 PRELIMINARIES/BASICS

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

## 3 ASYNCHRONOUS COMMUNICATION MODELS OVERVIEW

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

## 4 ASYNCHRONOUS COMMUNICATION MODELS OPERATIONAL SEMANTICS

- TODO...

## 5 ASYNCHRONOUS COMMUNICATION MODELS AS CLASSES OF MSCS, MSO-DEFINABILITY

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

## 6 EQUIVALENCE OF THE TWO DEFINITIONS

- TODO...

## 7 HIERARCHY OF ASYNCHRONOUS CLASSES OF MSCS

...

*Title note
†Subtitle note

2018. 2475-1421/2018/1-ART1 $15.00
https://doi.org/

## 8 AN APPLICATION: SPECIAL TREEWIDTH AND DECIDABILITY OF THE SYNCHRONIZABILITY PROBLEM

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

## 9 CONCLUSION

# 10 PRELIMINARIES

## 10.1 Message Sequence Charts

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

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

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

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

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

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

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

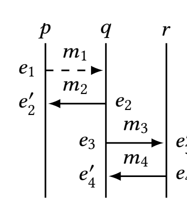


Fig. 1. MSC $M_1$

*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 $\lambda(e_1) \in Send(\_,q,\_)$, $\lambda(e_2) \in Send(\_,q,\_)$, and one of the following holds:

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

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

**Example 10.2.** MSC $M_1$ is a mailbox MSC. Indeed, even though the order $\rightsquigarrow$ defined in Example 10.1 does not respect all mailbox constraints, particularly the fact that $e_4 \sqsubset_{M_1} e_1$, there is a total order $\rightsquigarrow \supseteq \leq_{M_1}$ 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 $\leq_M \subseteq \rightsquigarrow$. That is, every mailbox linearization is a p2p linearization, but the converse is not necessarily true (Example 10.2). 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 $\lambda'$ is the restriction of $\mathcal{E}$ to $E$, is called a *prefix* of $M$. In particular, the empty MSC is a prefix of $M$. We denote the set of prefixes of $M$ by $\mathit{Pref}(M)$. This is extended to sets $L \subseteq \mathsf{MSC}$ as expected, letting $\mathit{Pref}(L) = \bigcup_{M \in L} \mathit{Pref}(M)$.

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

Proof. Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda) \in \mathsf{MSC}_{n-1}$ 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 $e \leq_{M_0} f \leq_{M_0} e$ with $\leq_{M_0} = (\rightarrow_0 \cup \lhd_0 \cup \sqsubset_{M_0})^*$. 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, $\leq_{M_0} \subseteq \leq_M$ and $M$ is not a mailbox MSC, which is a contradiction. □

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

## 10.2 Communicating Systems

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

**Definition 10.1.** A *communicating system* over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $\mathcal{S} = (A_p)_{p \in \mathbb{P}}$. For each $p \in \mathbb{P}$, $A_p = (\mathit{Loc}_p, \delta_p, \ell_p^0)$ is a finite transition system where $\mathit{Loc}_p$ 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

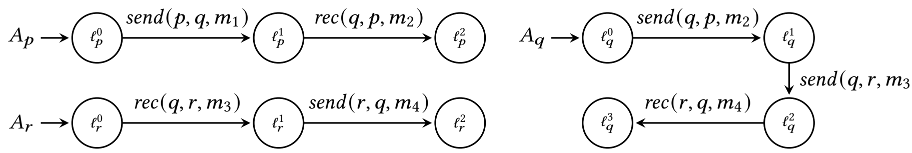


Fig. 2. System $\mathcal{S}_1$

essentially equivalent, but they have different advantages depending on the context. We refer to [Aiswarya and Gastin 2014] for a thorough discussion.

Let $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ be an MSC. A *run* of $\mathcal{S}$ on $M$ is a mapping $\rho : \mathcal{E} \rightarrow \bigcup_{p \in \mathbb{P}} \delta_p$ that assigns to every event $e$ the transition $\rho(e)$ that is executed at $e$. Thus, we require that (i) for all $e \in \mathcal{E}$, we have $action(\rho(e)) = \lambda(e)$, (ii) for all $(e, f) \in \rightarrow$, $target(\rho(e)) = source(\rho(f))$, (iii) for all $(e, f) \in \lhd$, $msg(\rho(e)) = msg(\rho(f))$, 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 $source(\rho(e)) = \ell_p^0$.

Letting run $\mathcal{S}$ directly on MSCs is actually very convenient. This allows us to associate with $\mathcal{S}$ its p2p language and mailbox language in one go. The *p2p language* of $\mathcal{S}$ is $L_{\mathsf{p2p}}(\mathcal{S}) = \{M \in \mathsf{MSC}_{\mathsf{p2p}} \mid$ there is a run of $\mathcal{S}$ on $M\}$. The *mailbox language* of $\mathcal{S}$ is $L_{\mathsf{mb}}(\mathcal{S}) = \{M \in \mathsf{MSC}_{n-1} \mid$ there is a run of $\mathcal{S}$ on $M\}$.

Note that, following [Bouajjani et al. 2018; Di Giusto et al. 2020], we do not consider final states or final configurations, as our purpose is to reason about all possible traces that can be *generated* by $\mathcal{S}$. The next lemma is obvious for the p2p semantics and follows from Lemma 10.1 for the mailbox semantics.

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

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

### 10.3 Conflict Graph

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

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

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

## 10.4 Logic and Special Tree-Width

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

Given a sentence $\varphi$, i.e., a formula without free variables, we let $L(\varphi)$ denote the set of (p2p) MSCs that satisfy $\varphi$. 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 $L(\varphi_{\mathsf{mb}}) = \mathsf{MSC}_{n-1}$. 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 :

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

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

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

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

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

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

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

## 11 (3) ASYNCHRONOUS COMMUNICATION MODELS OVERVIEW

In synchronous communication, send and receive events are essentially viewed as a single entity, i.e. a receive event always happens simultaneously with its corresponding send event. The whole idea behind (fully) asynchronous communication is to decouple send and receive events, so that a receive event can happen indefinitely after its corresponding send event. However, by introducing some additional contraints on asynchronous communication, we can obtain new communication models that sit somewhere between synchronous and fully asynchronous communication. These kind of communication models are often used interchangably in literature and generally referred to as "asynchronous", without providing a clear definition. In this section, we will present 7 different asynchronous communication models, which were already introduced in [Chevrou et al. 2016] (even though they do not consider unmatched messages). A major difference is that in [Chevrou et al. 2016] these communication models are addressed from a linearizations standpoint, whereas we are interested in MSCs. Recall that a single MSC can have several possible linearizations. The work in [Chevrou et al. 2016] describes the properties that a single linearization must satisfy in order to be realizable by a system that uses a given communication model. On the other hand, we are interested in understanding if a given MSC describes a computation that can be realized by a system that uses some communication model $CM$. In other words, given a MSC we want to know if it has at least one linearization that respects the constraints imposed by $CM$. If that is the case, the MSC represents a behaviour that can be exhibited by a system that uses $CM$ as a communication model. These are two fundamentally dissimilar problems; at the end of this section we provide an example to clarify the difference. In our work, we are going to formally characterize the classes of MSCs which represent valid computations for all of these 7 asynchronous communication model. We also show how these classes form a well-defined hierarchy, which does not correspond entirely to that found in [Chevrou et al. 2016].

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

### 11.1 Fully asynchronous

In the fully asychronous communication model (or simply asynchronous) messages can be received at any time once they have been sent. In asynchronous communication, send events are non-blocking, i.e. the sender of a message does not have to wait for it to be delivered to the recipient, in order to resume normal operations. Fig. 3 shows a computation that can be executed by a system that uses asynchronous communication; indeed, even if $m_1$ is sent before $m_2$, $q$ does not have to receive $m_1$ first. For convenience, we will refer to a system that uses asynchronous communication as an asynchronous system. In a similar way, an MSC such that in Fig. 3 will be referred to as an asynchronous MSC, since it represents a computation that is valid for an asynchronous system. The same jargon will also be used for all the other communication models. We will call $\mathrm{MSC}_{\mathrm{asy}}$ the set of all asynchronous MSCs.

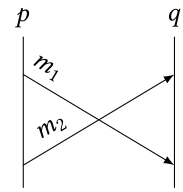


Fig. 3. An asynchronous MSC.

### 11.2 FIFO 1−1 (p2p)

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

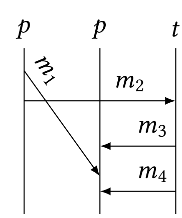


Fig. 4. A p2p MSC.

### 11.3 Causally ordered

In the causally ordered communication model, messages are delivered to a process according to the causality of their emissions. In other words, if there are two messages $m_1$ and $m_2$ with the same recipient, such that $m_1$ is causally sent before $m_2$ (i.e. there exists a causal path from the first send to the second one), then $m_1$ must be received before $m_2$. Fig. 5a, which is identical to Fig.4, shows an example of non-causally ordered MSC; there is a causal path between the sending of $m_1$ and $m_3$ (highlighted with red arrows), hence $m_1$ should be received before $m_3$ according to the causally ordered communication model, which is not the case. On the other hand, the second MSC shown in Fig. 5 is causally ordered; note that the only two messages with the same recipient are $m_2$ and $m_3$, but there is no causal path between their respective send events (i.e. the causally ordered communication model does not introduce any new constraint that must be satisfied). Let $\mathsf{MSC}_{\mathrm{co}}$ be the set of causally ordered MSCs.

[1]Please note that our definition of Communicating Finite-State Machine is different from the classical one. FIFO channels are replaced by bag channels, which do not ensure any specific order on the delivery of messages.


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

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

Fig. 5. Two examples of MSCs.

### 11.4 FIFO $n-1$ (mailbox)

In the FIFO $n-1$ communicating model, any two messages sent to a process $q$ must be received in the same order as they have been sent (according to absolute time). Note that these two messages might be sent by different processes and the two send events might be concurrent (i.e. there is no causal path between them). In other words, if a process $q$ receives $m_1$ before $m_2$, then $m_1$ must have been sent before $m_2$ in absolute time. Essentially, the FIFO $n-1$ coordinates all the senders of a single receiver. A high-level implementation of the mailbox communication model could consist in a single incoming FIFO channel for each process, which is shared by all the other processes. A send event would consist in pushing the message on the shared FIFO channel. The MSC shown in Fig. 5a is not a mailbox MSC; $m_1$ and $m_3$ have the same recipient, but they are not received in the same order as they are sent. The MSC in Fig. 5b is mailbox; indeed, we are able to find a linearization that respects the mailbox constraints, such as !1 !2 !3 ?2 ?3 ?1 (note that $m_2$ is both sent and received before $m_3$). Such a linearization will be referred to as a *mailbox linearization*. At this stage, the difference between the class of causally ordered MSCs and the class of mailbox MSCs might not be clear. We will clarify later how all these classes of MSCs are related to each other. Let $\mathrm{MSC}_{n-1}$ be the set of mailbox MSCs.

### 11.5 FIFO $1-n$

The FIFO $1-n$ communicating model is the dual of FIFO $n-1$, it coordinates a sender with all the receivers. Any two messages sent by a process $p$ must be received in the same order (in absolute time) as they have been sent. Note that these two messages might be received by different processes and the two receive events might be concurrent (i.e. there is no causal path between them). In other words, if a process $p$ sends $m_1$ before $m_2$, then $m_1$ must be received before $m_2$ in absolute time. A high-level implementation of the FIFO $1-n$ communication model could consist in a single outgoing FIFO channel for each process $P_i$, which is shared by all the other processes. A send event would consist in pushing the message on the outgoing FIFO channel. The MSC shown in Fig. 5a is not a FIFO $1-n$ MSC; $m_1$ and $m_2$ are sent in this order by the same process, but they are received in the opposite order (note that there is a causal path between the reception of $m_2$ and the reception of $m_1$, so ?2 happens before ?1 in every linearization of this MSC). Fig. 5b shows an example of FIFO $1-n$ MSC; $m_1$ is sent before $m_2$ by the same process, and we are able to find a linearization where $m_1$ is received before $m_2$, such as !1 !2 !3 ?1 ?2 ?3. Such a linearization will be referred to as a *$1-n$ linearization*. Let $\mathrm{MSC}_{1-n}$ be the set of $1-n$ MSCs.

### 11.6 FIFO $n$–$n$

In the FIFO $n$–$n$ communicating model, messages are globally ordered and delivered according to the their emission order. Any two messages must be received in the same order as they have been sent, in absolute time. Note that these two messages might be received by different processes and the two receive events might be concurrent (i.e. there is no causal path between them). In other words, if a message $m_1$ is sent before $m_2$ in absolute time, then $m_1$ must be received before $m_2$ in absolute time. The FIFO $n$–$n$ coordinates all the senders with all the receivers. A high-level implementation of the FIFO 1–$n$ communication model could consist in a single FIFO channel shared by all processes. The MSC shown in Fig. 5a is clearly not a FIFO $n$–$n$ MSC; if we consider messages $m_1$ and $m_2$ we have that, in every linearization, !1 happens before !2 and ?2 happens before ?1. This violates the constraints imposed by the FIFO $n$–$n$ communication model. The MSC in Fig. 5b is $n$–$n$ because we are able to find a linearization that satisfies the $n$–$n$ communication model, e.g. !1 !2 !3 ?1 ?2 ?3. Such a linearization will be referred to as an *$n$–$n$ linearization*. Let $\mathrm{MSC}_{n-n}$ be the set of $n$–$n$ MSCs.

### 11.7 Realizable with Synchronous Communication (RSC)

The RSC communication model imposes that a send event is always immediately followed by its corresponding receive event. In an execution of a system that uses the RSC communication model it is impossible to find an event that is executed between a send and its corresponding receive. An asynchronous distributed system that implements the RSC communication model effectively behaves as a synchronous system. None of the MSCs shown in Fig. 5 is a RSC MSC; indeed, for both of them it is impossible to find a linearization where each send event is immediately followed by the corresponding receive event. The MSC shown in Fig. 6 is an example of RSC MSC; we can easily find a linearization that respects the constraints of the RSC communication model, such as !1 ?1 !2 ?2 !3 ?3. Such a linearization will be referred to as an RSC *linearization*. Let $\mathrm{MSC}_{\mathrm{RSC}}$ be the set of RSC MSCs.

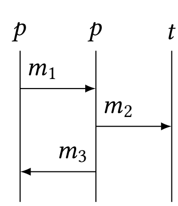


Fig. 6. A RSC MSC.

### 11.8 Hierarchy of MSC classes

We find of particular interest to study the relation between the classes of MSCs for all of these communication models. For instance, the MSC shown in Fig. 5a is both asynchronous and FIFO 1–1, in the sense that we are able to find systems using those communication models that can produce the behaviour described by the MSC. Is it always the case than a FIFO 1–1 MSC is also an asynchronous MSC? What about the other communication models? In Section **??** we prove that the classes of MSCs for all these communication models form a very neat hierarchy, which is graphically shown in Fig. 7.

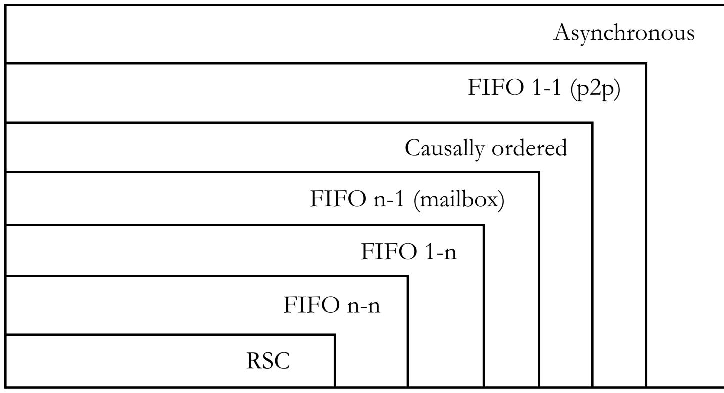


Fig. 7. The hierarchy of MSC classes.

## 12 (5) ASYNCHRONOUS COMMUNICATION MODELS AS CLASSES OF MSCS, MSO-DEFINABILITY

In Section 11 we gave a high-level description of 7 communication models and we talked about the corresponding classes of MSCs. Here, we formally define those classes and show that they are all MSO-definable, i.e. there is a Monadic Second Order Logic formula that defines each of them.

### 12.1 Definitions

We start with asynchronous MSCs, which represent valid computations for asynchronous systems. This is the most general definition of MSC, and it will serve as a basis on which the other communication models will build on, by adding some additional constraints.

**Definition 12.1** (Asynchronous MSC)**.** An *asynchronous MSC* (or simply MSC) over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, where $\mathcal{E}$ is a finite (possibly empty) set of *events* and $\lambda : \mathcal{E} \rightarrow \Sigma$ is a labeling function that associates an action to each event. For $p \in \mathbb{P}$, let $\mathcal{E}_p = \{e \in \mathcal{E} \mid \lambda(e) \in \Sigma_p\}$ be the set of events that are executed by $p$. We require that $\rightarrow$ (the *process relation*) is the disjoint union $\bigcup_{p\in\mathbb{P}} \rightarrow_p$ of relations $\rightarrow_p \subseteq \mathcal{E}_p \times \mathcal{E}_p$ such that $\rightarrow_p$ is the direct successor relation of a total order on $\mathcal{E}_p$. For an event $e \in \mathcal{E}$, a set of actions $A \subseteq \Sigma$, and a relation $R \subseteq \mathcal{E} \times \mathcal{E}$, let $\#_A(R, e) = |\{f \in \mathcal{E} \mid (f, e) \in R \text{ and } \lambda(f) \in A\}|$. We require that $\lhd \subseteq \mathcal{E} \times \mathcal{E}$ (the *message relation*) satisfies the following:

(1) for every pair $(e, f) \in \lhd$, there is a send action $send(p, q, m) \in \Sigma$ such that $\lambda(e) = send(p, q, m)$, $\lambda(f) = rec(p, q, m)$.
(2) for all $f \in \mathcal{E}$ such that $\lambda(f)$ is a receive action, there is exactly one $e \in \mathcal{E}$ such that $e \lhd f$.

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

D: Show example of causal path.

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

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

D: Provide example of linearization.

The class of asynchronous MSCs is the biggest and most general one out of the seven. All the others classes of MSCs will be obtained by adding some constraints to asynchronous MSCs, according to the specific communication model. As in the asynchronous case, we say that $M$ is a FIFO 1−1 MSC if there is a FIFO 1−1 system that can produce the behaviour shown by $M$. We give here the formal definition of FIFO 1−1 MSC, which also considers unmatched messages.

**Definition 12.2** (FIFO 1−1 MSCs)**.** A FIFO 1−1 MSC is an asynchronous MSC where we require that, for every pair $(e, f) \in \lhd$, such that $\lambda(e) = send(p, q, m)$, $\lambda(f) = rec(p, q, m)$, we have $\#_{Send(p,q,\_)}(\rightarrow^+, e) = \#_{Rec(p,q,\_)}(\rightarrow^+, f)$.

The additional constraint satisfied by FIFO 1−1 MSCs ensures that messages sent from any fixed process $p$ to another fixed process $q$ are always received in the same order as they are sent, i.e. when $q$ receives a message from $p$, it must have already received all the messages that were previously sent to him by $p$. Note that, for each pair $(p, q)$ of processes, we cannot have an unmatched message $m_1$ (sent by $p$) followed by a matched message $m_2$ (sent by $p$). In order to receive $m_2$, $q$ must have already received $m_1$, according to the definition of FIFO 1−1 MSC; because of the FIFO policy, $m_1$ is blocking the reception of $m_2$. By definition, every FIFO 1−1 MSC is an asynchronous MSC. Indeed, it is always possible to find an asynchronous system that can realize a computation described by a FIFO 1−1 MSC. In other word, the possible behaviours, i.e. MSCs, generated by a system $S$ that uses FIFO 1−1 communication are a subset of all the behaviours that $S$ would be able to generate using asynchronous communication.

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

D: Provide example of FIFO 1−1 and non-FIFO 1−1 MSCs.

**Definition 12.3** (Causally ordered MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is *causally ordered* if, for any two send events $s$ and $s'$, such that $\lambda(s) = Send(\_, q, \_)$, $\lambda(s') = Send(\_, q, \_)$, and $s \leq_M s'$, we have either:

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

D: Provide example of causally ordered and non-causally ordered MSCs.

Moving on to FIFO $n$−1 communication, we say that $M$ is a FIFO $n$−1 MSC if there is a FIFO $n$−1 system that can realize the computation described by $M$.

**Definition 12.4** (FIFO $n$−1 MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *FIFO $n$−1 MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events $s$ and $s'$, such that $\lambda(s) = Send(\_, q, \_)$, $\lambda(s') = Send(\_, q, \_)$, and $s \rightsquigarrow s'$, we have either:

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

Such a linearization will be referred to as a *FIFO $n{-}1$ linearization*, and we will sometimes use the symbol $\overset{\text{n-1}}{\rightsquigarrow}$ to denote one. Note that the definition of FIFO $n{-}1$ MSC is based on the existence of a linearization that satisfies some properties. The same kind of "existential" definition will be used for the remaining communication models. In practice, to claim that an MSC is FIFO $n{-}1$ we just need to find a single valid FIFO $n{-}1$ linearization, regardless of all the others; that linearization is a total order on the events that can be executed by a FIFO $n{-}1$ system. As for unmatched messages, note that we cannot have two messages $m_1$ and $m_2$, addressed to the same process but possibly sent by different processes, such that $m_1$ is unmatched and $m_2$ is matched; $m_2$ can only be received after $m_1$, and this is consistent with the high-level definition of the FIFO $n{-}1$ communication model that we gave in Section 11.

D: Provide example of FIFO $n{-}1$ and non-FIFO $n{-}1$ MSCs.

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

**Definition 12.5** (FIFO $1{-}n$ MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *$1{-}n$ MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events $s$ and $s'$, such that $\lambda(s) = Send(p, \_, \_)$, $\lambda(s') = Send(p, \_, \_)$, and $s \rightarrow^+ s'$ (which implies $s \rightsquigarrow s'$), we have either:

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

Such a linearization will be referred to as a *FIFO $1{-}n$ linearization*, and we will sometimes use the symbol $\overset{\text{1-n}}{\rightsquigarrow}$ to denote one. Note that the definition is very similar to the FIFO $n{-}1$ case, but here $s$ and $s'$ are two send events executed by the same process, and not addressed to the same process. In a FIFO $1{-}n$ MSC we cannot have two messages $m_1$ and $m_2$, sent by the same process, such that $m_1$ is unmatched and $m_2$ is matched; indeed, according to the FIFO $1{-}n$ communication model, $m_1$ must be received before $m_2$.

D: Provide example of FIFO $1{-}n$ and non-FIFO $1{-}n$ MSCs.

**Definition 12.6** (FIFO $n{-}n$ MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *$n{-}n$ MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events $s$ and $s'$, such that $s \rightsquigarrow s'$, we have either:

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

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

**Definition 12.7** (RSC MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *RSC MSC* if it has no unmatched send events and there is a linearization $\rightsquigarrow$ where any matched send event is immediately followed by its respective receive event.

### 12.2 Alternative definitions

In this section, we will provide some alternative equivalent definitions of FIFO $n-1$ MSC, FIFO $1-n$ MSC, FIFO $n-n$ MSC, and RSC MSC. These definitions will be useful to prove the MSO-definability of these classes of MSCs.

**Definition 12.8** (FIFO $n-1$ alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, we define an additional binary relation that represents a constraint under the FIFO $n-1$ semantics, which ensures that messages received by a process are sent in the same order as they are received. Let $\sqsubset_M \subseteq \mathcal{E} \times \mathcal{E}$ be defined as $s \sqsubset_M s'$ if there is $q \in \mathbb{P}$ such that $\lambda(s) \in Send(\_, q, \_)$, $\lambda(s') \in Send(\_, q, \_)$, and one of the following holds:

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

We let $\preceq_M = (\rightarrow \cup \lhd \cup \sqsubset_M)^*$. Note that $\leq_M \subseteq \preceq_M$. We call $M \in \mathsf{MSC}_{\mathsf{asy}}$ a *FIFO $n-1$ MSC* if $\preceq_M$ is a partial order. The $\sqsubset_M$ relation ensures that send events addressed to the same process are executed in an order that is suitable for the FIFO $n-1$ communication. Note that if $\preceq_M$ is a partial order, it means that it is possible to find a linearization $\rightsquigarrow$, i.e. a total order on the events, such that $\rightsquigarrow \subseteq \preceq_M$. It is not difficult to see that such a linearization is exactly what we called a FIFO $n-1$ linearization in Definition 12.4. The two definition of FIFO $n-1$ MSC that we gave are equivalent.

D: Proof of equivalence of 2 definitions?

**Definition 12.9** (FIFO $1-n$ alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, we define an additional binary relation that represents a constraint under the $1-n$ semantics, which ensures that messages sent from the same process are received in the same order. Let $\blacktriangleleft_M \subseteq \mathcal{E} \times \mathcal{E}$ be defined as $e_1 \blacktriangleleft_M e_2$ if there are two events $e_1$ and $e_2$, and $p \in \mathbb{P}$ such that either:

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

We let $\lessdot_M = (\rightarrow \cup \lhd \cup \blacktriangleleft_M)^*$. Note that $\leq_M \subseteq \lessdot_M$. We call $M \in \mathsf{MSC}_{\mathsf{asy}}$ a *$1-n$ MSC* if $\lessdot_M$ is a partial order. The $\blacktriangleleft_M$ relation ensures that messages sent by a process are sent and received in an order that is suitable for the FIFO $1-n$ communication. Note that if $\lessdot_M$ is a partial order, it is possible to find a linearization $\rightsquigarrow$, such that $\rightsquigarrow \subseteq \lessdot_M$. It is not difficult to see that such a linearization is exactly what we called a FIFO $1-n$ linearization in Definition 12.5. The two definition of FIFO $1-n$ MSC that we gave are equivalent.

D: Proof of equivalence of 2 definitions?

**Definition 12.10** (FIFO $n-n$ alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, let $\ltimes_M = (\rightarrow \cup \lhd \cup \sqsubset_M \cup \blacktriangleleft_M)^*$. We define an additional binary relation $\bowtie_M \subseteq \mathcal{E} \times \mathcal{E}$, such that for two events $e_1$ and $e_2$ we have $e_1 \bowtie_M e_2$ if one of the following holds:

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

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

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

It is not trivial to see that Definition 12.10 and Definition 12.6 are equivalent. To show that, we need some preliminary results and definitions.

**Proposition 12.1.** Let $M$ be an MSC. Given two matched send events $s_1$ and $s_2$, and their respective receive events $r_1$ and $r_2$, $r_1 \bowtie_M r_2 \implies s_1 \bowtie_M s_2$.

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

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

□

**Proposition 12.2.** Let $M$ be an MSC. If $\bowtie_M$ is cyclic, then $M$ is not $n{-}n$.

Proof. According to Definition 12.6, an MSC is FIFO $n{-}n$ if it has at least one FIFO $n{-}n$ linearization. Note that, because of how it is defined, any FIFO $n{-}n$ linearization is always both a FIFO $n{-}1$ and a FIFO $1{-}n$ linearization. It follows that the cyclicity of $\ltimes_M$ (not $\bowtie_M$) implies that $M$ is not $n{-}n$, because it means that we are not even able to find a linearization that is both FIFO $n{-}1$ and $1{-}n$. Moreover, since in a $n{-}n$ linearization the order in which messages are sent matches the order in which they are received, and unmatched send events can be executed only after matched send events, a $n{-}n$ MSC always has to satisfy the constraints imposed by the $\bowtie_M$ relation. If $\bowtie_M$ is cyclic, then for sure there is no $n{-}n$ linearization for $M$.

D: This proof does not convince me... maybe rewrite it better

□

Let the *Event Dependency Graph* (EDG) of a $n{-}n$ MSC $M$ be a graph that has events as nodes and an edge between two events $e_1$ and $e_2$ if $e_1 \bowtie_M e_2$. We now present an algorithm that, given the EDG of an $n{-}n$ MSC $M$, computes a $n{-}n$ linearization of $M$. We then show that, if $\bowtie_M$ is acyclic (i.e. it is a partial order), this algorithm always terminates correctly. This, along with Proposition 12.2, effectively shows that Definition 12.6 and Definition 12.10 are equivalent.

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

(1) If there is a matched send event $s$ with in-degree 0 in the EDG, add $s$ to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 2.
(2) If there are no matched send events in the EDG and there is an unmatched send event $s$ with in-degree 0 in the EDG, add $s$ to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 3.
(3) If there is a receive event $r$ with in-degree 0 in the EDG, such that $r$ is the receive event of the first message whose sent event was already added to the linearization, add $r$ to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 4.
(4) Throw an error and terminate.

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

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

**Proposition 12.3.** If the above algorithm returns a linearization for an MSC $M$, it is a $n$–$n$ linearization.

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

**Proposition 12.4.** Given an MSC $M$, the above algorithm always terminates correctly if $\bowtie_M$ is acyclic.

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

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

in-degree 0 if there are still matched send events in the EDG. Let $s$ be one of those matched send events and let $u$ be an unmatched send. Because of rule 4 in the definition of $\bowtie_M$, we have that $s \bowtie_M u$, which implies that $u$ cannot have in-degree 0 if $s$ is still in the EDG.

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

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

We have now effectively proved that Definition 12.10 of FIFO $n{-}n$ MSC is equivalent to Definition 12.6.

Following the characterization given in [Charron-Bost et al. 1996, Theorem 4.4], we also give an alternative but equivalent definition of RSC MSC.

**Definition 12.11.** Let $M$ be an MSC. A crown of size $k$ in $M$ is a sequence $\langle (s_i, r_i),\ i \in \{1, \ldots, k\} \rangle$ of pairs of corresponding send and receive events such that

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

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

**Definition 12.12** (RSC alternative). An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *RSC MSC* if and only if it does not contain any crown.

## 12.3 Monadic Second-Order Logic

The set of MSO formulas over (asynchronous) MSCs (over $\mathbb{P}$ and $\mathbb{M}$) is given by the grammar $\varphi ::= \mathit{true} \mid x \rightarrow y \mid x \lhd y \mid \lambda(x) = a \mid x = y \mid x \in X \mid \exists x.\varphi \mid \exists X.\varphi \mid \varphi \vee \varphi \mid \neg\varphi$, where $a \in \Sigma$, $x$ and $y$ are first-order variables, interpreted as events of an MSC, and $X$ is a second-order variable, interpreted as a set of events. We assume that we have an infinite supply of variables, and we use common abbreviations such as $\wedge$, $\Rightarrow$, $\forall$, etc. The satisfaction relation is defined in the standard way and self-explanatory. For example, the formula $\neg\exists x.(\bigvee_{a \in Send(\_,\_,\_)} \lambda(x) = a \ \wedge\ \neg \mathit{matched}(x))$ 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 $\varphi$, i.e., a formula without free variables, we let $L(\varphi)$ denote the set of MSCs that satisfy $\varphi$. Since we have defined the set of MSO formulas over asynchronous MSCs, the formula $\varphi_{\text{asy}} = \mathit{true}$ clearly describes the set of asynchronous MSCs, i.e. $L(\varphi_{\text{asy}}) = \text{MSC}_{\text{asy}}$. It is worth mentioning that the (reflexive) transitive closure of a binary relation defined by an MSO formula[2] with free variables $x$ and $y$, such as $x \to y$, is MSO-definable so that the logic can freely use formulas of the form $x \to^+ y$, $x \to^* y$ or $x \leq y$ (where $\leq$ is interpreted as $\leq_M$ for the given MSC $M$).

*FIFO* 1−1 *MSCs.* The set of FIFO 1−1 MSCs is MSO-definable as

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

where $\psi_1$ and $\psi_2$ are

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

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

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

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

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

where $\psi_1$ and $\psi_2$ are the same formulas used for p2p.

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

*FIFO* $n$−1 *MSCs.* Given an MSC $M$, it is a FIFO $n$−1 MSC if and only if it satisfies the MSO formula

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

This formula closely follows Definition 12.8. The set $\mathrm{MSC}_{n-1}$ of FIFO $n$−1 MSCs is therefore MSO-definable as $\mathrm{MSC}_{n-1} = L(\varphi_{\mathrm{mb}})$.

*FIFO* 1−$n$ *MSCs.* Following Definition 12.9, an MSC $M$ is a 1−$n$ MSC if and only if it satisfies the MSO formula

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

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

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

[2] See Section ?? for details.

The MSO formula for $x \blacktriangleleft_M y$ closely follows Definition 12.9. The set $\mathrm{MSC}_{1-n}$ of $1-n$ MSCs is therefore MSO-definable as $\mathrm{MSC}_{1-n} = L(\varphi_{1-n})$.

*FIFO $n-n$ MSCs.* Following Definition 12.10, an MSC $M$ is a $n-n$ MSC if and only if it satisfies the MSO formula

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

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

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

where $\psi_3$ and $\psi_4$ are defined as

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

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

The MSO formula for $x \ltimes_M y$ closely follows Definition 12.10. The set $\mathrm{MSC}_{n-n}$ of $n-n$ MSCs is therefore MSO-definable as $\mathrm{MSC}_{n-n} = L(\varphi_{n-n})$.

RSC *MSCs.* Following Definition 12.12, an MSC $M$ is a RSC MSC if and only if it satisfies the MSO formula

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

where $\propto$ is defined as

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

## REFERENCES


C. Aiswarya and Paul Gastin. 2014. Reasoning About Distributed Systems: WYSIWYG (Invited Talk). In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India (LIPIcs, Vol. 29)*, Venkatesh Raman and S. P. Suresh (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 11–30. https://doi.org/10.4230/LIPIcs.FSTTCS.2014.11

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

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

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

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

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

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

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

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

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

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

### 12.4 Definitions

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

**Definition 12.13** (Asynchronous MSC)**.** An *asynchronous MSC* (or simply MSC) over $\mathbb{P}$ and $\mathbb{M}$ is a tuple $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, where $\mathcal{E}$ is a finite (possibly empty) set of *events* and $\lambda : \mathcal{E} \rightarrow \Sigma$ is a labeling function that associates an action to each event. For $p \in \mathbb{P}$, let $\mathcal{E}_p = \{e \in \mathcal{E} \mid \lambda(e) \in \Sigma_p\}$ be the set of events that are executed by $p$. We require that $\rightarrow$ (the *process relation*) is the disjoint union $\bigcup_{p \in \mathbb{P}} \rightarrow_p$ of relations $\rightarrow_p \subseteq \mathcal{E}_p \times \mathcal{E}_p$ such that $\rightarrow_p$ is the direct successor relation of a total order on $\mathcal{E}_p$. For an event $e \in \mathcal{E}$, a set of actions $A \subseteq \Sigma$, and a relation $R \subseteq \mathcal{E} \times \mathcal{E}$, let $\#_A(R, e) = |\{f \in \mathcal{E} \mid (f, e) \in R \text{ and } \lambda(f) \in A\}|$. We require that $\lhd \subseteq \mathcal{E} \times \mathcal{E}$ (the *message relation*) satisfies the following:

(1) for every pair $(e, f) \in \lhd$, there is a send action $send(p, q, m) \in \Sigma$ such that $\lambda(e) = send(p, q, m)$, $\lambda(f) = rec(p, q, m)$.
(2) for all $f \in \mathcal{E}$ such that $\lambda(f)$ is a receive action, there is exactly one $e \in \mathcal{E}$ such that $e \lhd f$.

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

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

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

D: Provide example of linearization.

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

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

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

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

**Definition 12.14** (Peer-to-peer MSCs). A *p2p MSC* (or simply *MSC*) is an asynchronous MSC where we require that, for every pair $(e, f) \in \lhd$, such that $\lambda(e) = send(p, q, m)$, $\lambda(f) = rec(p, q, m)$, we have $\#_{Send(p,q,\_)}(\rightarrow^+, e) = \#_{Rec(p,q,\_)}(\rightarrow^+, f)$.

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

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

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

**Definition 12.15** (Causally ordered MSC). An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is *causally ordered* if, for any two send events $s$ and $s'$, such that $\lambda(s) = Send(\_, q, \_)$, $\lambda(s') = Send(\_, q, \_)$, and $s \leq_M s'$, we have either:

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

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

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

D: Provide example of MSC which is not mailbox

**Definition 12.16** (Mailbox MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *mailbox MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events $s$ and $s'$, such that $\lambda(s) = \mathit{Send}(\_, q, \_)$, $\lambda(s') = \mathit{Send}(\_, q, \_)$, and $s \rightsquigarrow s'$, we have either:

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

Such a linearization will be referred to as a *mailbox linearization*, and the symbol $\overset{\text{n-1}}{\rightsquigarrow}$ will be used to denote one. Let $\mathsf{MSC}_{n-1}$ denote the set of all the mailbox MSCs over two given sets $\mathbb{P}$ and $\mathbb{M}$. By definition, every mailbox MSC is a p2p MSC. Conversely, Example **??** shows a p2p MSC which is not a mailbox MSC. It follows that $\mathsf{MSC}_{n-1} \subset \mathsf{MSC}_{\mathrm{p2p}}$. We show here that each mailbox MSC is also a causally ordered MSC.

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

**Proposition 12.5.** Every mailbox MSC is a causally ordered MSC.

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

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

**Definition 12.17** ($1{-}n$ MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a $1{-}n$ *MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events $s$ and $s'$, such that $\lambda(s) = \mathit{Send}(p, \_, \_)$, $\lambda(s') = \mathit{Send}(p, \_, \_)$, and $s \rightarrow^+ s'$ (which implies $s \rightsquigarrow s'$), we have either:

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

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

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

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

**Proposition 12.6.** Every $1{-}n$ MSC is a causally ordered MSC.

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

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

(2) $s \in \mathit{Unm}(M)$ and $s' \in \mathit{Matched}(M)$.

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

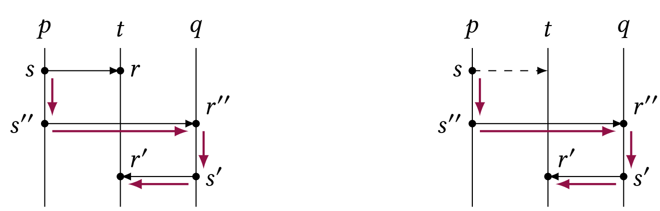


Fig. 8. Two examples of $1\!-\!n$ MSCs.

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

**Definition 12.18** ($1\!-\!n$ alternative)**.** For an MSC $M = (\mathcal{E}, \to, \lhd, \lambda)$, we define an additional binary relation that represents a constraint under the $1\!-\!n$ semantics, which ensures that messages sent from the same process are received in the same order. Let $\blacktriangleleft_M \subseteq \mathcal{E} \times \mathcal{E}$ be defined as $e_1 \blacktriangleleft_M e_2$ if there are two events $e_1$ and $e_2$, and $p \in \mathbb{P}$ such that either:

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

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

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

**Proposition 12.7.** Every $1\!-\!n$ MSC without unmatched messages is a mailbox MSC.

Proof. We show that the contrapositive is true, i.e. if an MSC is not mailbox (and it does not have unmatched messages), it is also not $1-n$. Suppose $M$ is an asynchronous MSC, but not mailbox. There must be a cycle $\xi$ such that $e \preceq e$, for some event $e$. Recall that $\preceq = (\rightarrow \cup \lhd \cup \sqsubset)^*$ and $\leq = (\rightarrow \cup \lhd)^*$. We can always explicitely write a cycle $e \preceq e$ only using $\sqsubset$ and $\leq$. For instance, there might be a cycle $e \preceq e$ because we have that $e \sqsubset f \leq g \sqsubset h \sqsubset i \leq e$. Consider any two adiacent events $s_1$ and $s_2$ in the cycle $\xi$, where $\xi$ has been written using only $\sqsubset$ and $\leq$, and we never have two consecutive $\leq$[3]. We have two cases:

(1) $s_1 \sqsubset s_2$. We know, by definition of $\sqsubset$, that $s_1$ and $s_2$ must be two send events and that $r_1 \rightarrow^+ r_2$, where $r_1$ and $r_2$ are the receive events that match with $s_1$ and $s_2$, respectively (we are not considering unmatched messages by hypothesis).
(2) $s_1 \leq s_2$. Since $M$ is asynchronous by hyphotesis, $\xi$ has to contain at least one $\sqsubset$[4]; recall that we also wrote $\xi$ in such a way that we do not have two consecutive $\leq$. It is not difficult to see that $s_1$ and $s_2$ have to be send events, since they belong to $\xi$. We have two cases:
   (a) $r_1$ is in the causal path, i.e. $s_1 \lhd r_1 \leq s_2$. In particular, note that $r_1 \leq r_2$.
   (b) $r_1$ is not in the causal path, hence there must be a message $m_k$ sent by the same process that sent $s_1$, such that $s_1 \rightarrow^+ s_k \lhd r_k \leq s_2 \lhd r_2$, where $s_k$ and $r_k$ are the send and receive events associated with $m_k$, respectively. Since messages $m_1$ and $m_k$ are sent by the same process and $s_1 \rightarrow^+ s_k$, we should have $r_1 \blacktriangleleft r_k$, according to the $1-n$ semantics. In particular, note the we have $r_1 \blacktriangleleft r_k \leq r_2$.

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

Notice that, for either cases, a relation between two send events $s_1$ and $s_2$ (i.e. $s_1 \sqsubset s_2$ or $s_1 \leq s_2$) always implies a relation between the respective receive events $r_1$ and $r_2$, according to the $1-n$ semantics. It follows that $\xi$, which is a cycle for the $\preceq$ relation, always implies a cycle for the $\lessdot$ relation[5], as shown by the following example. Let $M$ be a non-mailbox MSC, and suppose we have a cycle $s_1 \sqsubset s_2 \sqsubset s_3 \leq s_4 \sqsubset s_5 \leq s_1$. $s_1 \sqsubset s_2$ falls into case (1), so it implies $r_1 \rightarrow^+ r_2$. The same goes for $s_2 \sqsubset r_3$, which implies $r_2 \rightarrow^+ r_3$. $s_3 \leq s_4$ falls into case (2), and implies that $r_3 \lessdot r_4$. $s_4 \sqsubset s_5$ falls into case (1) and it implies $r_4 \rightarrow^+ r_5$. $s_5 \leq s_1$ falls into case (2) and implies that $r_5 \lessdot r_1$. Putting all these implications together, we have that $r_1 \rightarrow^+ r_2 \rightarrow^+ r_3 \lessdot r_4 \rightarrow^+ r_5 \lessdot r_1$, which is a cycle for $\lessdot$. Note that, given any cycle for $\preceq$, we are always able to apply this technique to obtain a cycle for $\lessdot$. □

Proposition **??** remains true even if we consider unmatched messages.

**Proposition 12.8.** Every $1-n$ MSC is a mailbox MSC.

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

(3) $u_1 \sqsubset s_2$, where $u_1$ is the send event of an unmatched message. This case never happens because of how $\sqsubset$ is defined.
(4) $u_1 \leq u_2$, where $u_1$ and $u_2$ are both send events of unmatched messages. Since both $u_1$ and $u_2$ are part of the cycle $\xi$, there must be an event $s_3$ such that $u_1 \leq u_2 \sqsubset s_3$. However, $u_2 \sqsubset s_3$ falls into case (3), which can never happen.

[3]This is always possible, since $a \leq b \leq c$ is written as $a \leq c$.
[4]If that was not the case, $\preceq$ would also be cyclic and $M$ would not be an asynchronous MSC.
[5]If $\lessdot$ is cyclic, $M$ is not a $1-n$ MSC.

(5) $u_1 \leq s_2$, where $u_1$ is the send event of an unmatched message and $s_2$ is the send event of a matched message. Since we have a causal path between $u_1$ and $s_2$, there has to be a message $m_k$, sent by the same process that sent $m_1$, such that $u_1 \rightarrow^+ s_k \lhd r_k \leq s_2 \lhd r_2$[6], where $s_k$ and $r_k$ are the send and receive events associated with $m_k$, respectively. Since messages $m_1$ and $m_k$ are sent by the same process and $m_1$ is unmatched, we should have $s_k \blacktriangleleft u_1$, according to the $1{-}n$ semantics, but $u_1 \rightarrow^+ s_k$. It follows that if $\xi$ contains $u_1 \leq s_2$, we can immediately conclude that $M$ is not a $1{-}n$ MSC.

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

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

**Definition 12.19** ($n{-}n$ MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *$n{-}n$ MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events $s$ and $s'$, such that $s \rightsquigarrow s'$, we have either:

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

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

**Definition 12.20** ($n{-}n$ alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, let $\ltimes_M = (\rightarrow \cup \lhd \cup \sqsubset_M \cup \blacktriangleleft_M)^*$. We define an additional binary relation $\bowtie_M \subseteq \mathcal{E} \times \mathcal{E}$, such that for two events $e_1$ and $e_2$ we have $e_1 \bowtie_M e_2$ if one of the following holds:

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

Note that $\leq_M \subseteq \ltimes_M$, $\lessdot_M \subseteq \ltimes_M$, and $\ltimes_M \subseteq \bowtie_M$. We call $M \in \mathsf{MSC}_{\mathrm{asy}}$ a *$n{-}n$ MSC* if $\bowtie_M$ is acyclic.

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

It is not trivial to see that Definition 12.10 is equivalent to Definition 12.6. To show that, we need some preliminary results and definitions.

**Proposition 12.9.** Let $M$ be an MSC. Given two matched send events $s_1$ and $s_2$, and their respective receive events $r_1$ and $r_2$, $r_1 \bowtie_M r_2 \implies s_1 \bowtie_M s_2$.

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

- $r_1 \ltimes_M r_2$. Two cases: either (i) $s_1 \ltimes_M s_2$, or (ii) $s_1 \not\ltimes_M s_2$. The first case clearly implies $s_1 \bowtie_M s_2$, for rule 1 in the definition of $\bowtie_M$. The second too, because of rule 3.

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

- $r_1 \not\ltimes_M r_2$, but $r_1 \bowtie_M r_2$. This is only possible if rule 2 in the definition of $\bowtie_M$ was used, which implies $s_1 \ltimes_M s_2$ and, for rule 1, $s_1 \bowtie_M s_2$.

□

**Proposition 12.10.** Let $M$ be an MSC. If $\bowtie_M$ is cyclic, then $M$ is not $n{-}n$.

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

Let the *Event Dependency Graph* (EDG) of a $n{-}n$ MSC $M$ be a graph that has events as nodes and an edge between two events $e_1$ and $e_2$ if $e_1 \bowtie_M e_2$. We now present an algorithm that, given the EDG of an $n{-}n$ MSC $M$, computes a $n{-}n$ linearization of $M$. We then show that, if $\bowtie_M$ is acyclic (i.e. it is a partial order), this algorithm always terminates correctly. This, along with Proposition 12.2, effectively shows that Definition 12.6 and Definition 12.10 are equivalent.

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

(1) If there is a matched send event $s$ with in-degree 0 in the EDG, add $s$ to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 2.
(2) If there are no matched send events in the EDG and there is an unmatched send event $s$ with in-degree 0 in the EDG, add $s$ to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 3.
(3) If there is a receive event $r$ with in-degree 0 in the EDG, such that $r$ is the receive event of the first message whose sent event was already added to the linearization, add $r$ to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 4.
(4) Throw an error and terminate.
(5) If all the events of $M$ were added to the linearization, return the linearization and terminate. Otherwise, go back to step 1.

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

**Proposition 12.11.** If the above algorithm returns a linearization for an MSC $M$, it is a $n{-}n$ linearization.

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

**Proposition 12.12.** Given an MSC $M$, the above algorithm always terminates correctly if $\bowtie_M$ is acyclic.

Proof. We want to prove that, if $\bowtie_M$ is acyclic, step 4 of the algorithm is never executed, i.e. it terminates correctly. Note that the acyclicity of $\bowtie_M$ implies that the EDG of $M$ is a DAG. Moreover, at every step of the algorithm we remove nodes and edges from the EDG, so it still remains a DAG.

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

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

have added $s$ to the linearization before $s_e$. Because we got to a contradiction, the hypothesis that $r$ has in-degree $> 0$ must be false, and we can indeed apply step 3.

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

**Definition 12.21** (RSC MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *RSC MSC* if it has no unmatched send events and there is a linearization $\rightsquigarrow$ where any matched send event is immediately followed by its respective receive event.

Following the characterization given in [Charron-Bost et al. 1996, Theorem 4.4], we also give an alternative but equivalent definition of RSC MSC.

**Definition 12.22.** Let $M$ be an MSC. A crown of size $k$ in $M$ is a sequence $\langle (s_i, r_i),\ i \in \{1, \ldots, k\} \rangle$ of pairs of corresponding send and receive events such that

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

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

**Definition 12.23** (RSC alternative)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a *RSC MSC* if and only if it does not contain any crown.

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

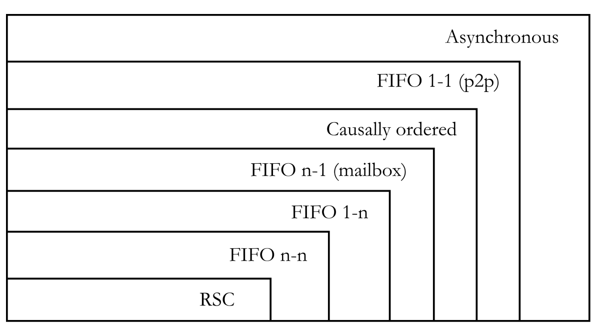


Fig. 9. The hierarchy of MSC classes.

## 12.5 Monadic Second-Order Logic

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

Given a sentence $\varphi$, i.e., a formula without free variables, we let $L(\varphi)$ denote the set of MSCs that satisfy $\varphi$. Since we have defined the set of MSO formulas over asynchronous MSCs, the formula $\varphi_{\mathrm{asy}} = true$ clearly describes the set of asynchronous MSCs, i.e. $L(\varphi_{\mathrm{asy}}) = \mathsf{MSC}_{\mathrm{asy}}$. It is worth mentioning that the (reflexive) transitive closure of a binary relation defined by an MSO formula[7] 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 $x \rightarrow^+ y$, $x \rightarrow^* y$ or $x \leq y$ (where $\leq$ is interpreted as $\leq_M$ for the given MSC $M$).

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

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

where $\psi_1$ and $\psi_2$ are

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

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

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

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

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

where $\psi_1$ and $\psi_2$ are the same formulas used for p2p.

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

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

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

The set $\text{MSC}_{n-1}$ of mailbox MSCs is therefore MSO-definable as $\text{MSC}_{n-1} = L(\varphi_{\text{mb}})$.

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

*1−n MSCs.* Following Definition 12.9, an MSC $M$ is a 1−n MSC if and only if it satisfies the MSO formula

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

[7]See Section ?? for details.

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

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

The MSO formula for $x \blacktriangleleft_M y$ closely follows Definition 12.9. The set $\mathrm{MSC}_{1-n}$ of $1-n$ MSCs is therefore MSO-definable as $\mathrm{MSC}_{1-n} = L(\varphi_{1-n})$.

*$n-n$ MSCs.* Following Definition 12.10, an MSC $M$ is a $n-n$ MSC if and only if it satisfies the MSO formula

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

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

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

where $\psi_3$ and $\psi_4$ are defined as

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

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

The MSO formula for $x \ltimes_M y$ closely follows Definition 12.10. The set $\mathrm{MSC}_{n-n}$ of $n-n$ MSCs is therefore MSO-definable as $\mathrm{MSC}_{n-n} = L(\varphi_{n-n})$.

RSC *MSCs.* Following Definition 12.12, an MSC $M$ is a RSC MSC if and only if it satisfies the MSO formula

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

where $\propto$ is defined as

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