Logical Methods in Computer Science
Volume 19, Issue 4, 2023, pp. 33:1–33:28
https://lmcs.episciences.org/

Submitted Aug. 20, 2018
Published Dec. 20, 2023

# SYNCHRONIZABILITY OF COMMUNICATING FINITE STATE MACHINES IS NOT DECIDABLE

ALAIN FINKEL [a] AND ETIENNE LOZES [b]

[a] Université Paris-Saclay, CNRS, ENS Paris-Saclay, Institut Universitaire de France, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France
*URL*: `https://ens-paris-saclay.fr/alain-finkel/`
*e-mail address*: alain.finkel@ens-paris-saclay.fr

[b] Université Côte d'Azur, CNRS, I3S, France
*e-mail address*: elozes@i3s.unice.fr

Abstract. A system of communicating finite state machines is *synchronizable* if its send trace semantics, i.e. the set of sequences of sendings it can perform, is the same when its communications are FIFO asynchronous and when they are just rendez-vous synchronizations. This property was claimed to be decidable in several conference and journal papers for either mailboxes or peer-to-peer communications, thanks to a form of small model property. In this paper, we show that this small model property does not hold neither for mailbox communications, nor for peer-to-peer communications, therefore the decidability of synchronizability becomes an open question. We close this question for peer-to-peer communications, and we show that synchronizability is actually undecidable. We show that synchronizability is decidable if the topology of communications is an oriented ring. We also show that, in this case, synchronizability implies the absence of unspecified receptions, and the channel-recognizability of the reachability set.

## 1. Introduction

Asynchronous distributed systems are error prone not only because they are difficult to program, but also because they are difficult to execute in a reproducible way. The slack of communications, measured by the number of messages that can be buffered in a same communication channel, is not always under the control of the programmer, and even when it is, it may be delicate to choose the right size of the communication buffers.

The synchronizability of a system of communicating machines is a property introduced by Basu and Bultan [FBS05, BB11] that formalizes the idea of a distributed system that is "slack elastic", in the sense that its behaviour is the same whatever the size of the buffers, and in particular it is enough to detect bugs by considering executions with buffers of size one [BBO12b, BB16]. Synchronizability can also be used for checking other properties like choreography realizability [BBO12a].

*Key words and phrases:* verification, distributed system, asynchronous communications, choreographies.
* Extended version of an article published in ICALP'17 proceedings.

For instance, consider the machines

$$P \;= !a \cdot !b \qquad \text{and} \qquad Q \;= ?a \cdot ?b$$

where $P$ may send two messages $a$ and $b$ in sequence to $Q$, and $Q$ is ready to receive them. These two machines form a synchronous system $P|Q$: the asynchronous trace $!a \cdot !b \cdot ?a \cdot ?b$ is "equivalent" to the synchronous trace $!a \cdot ?a \cdot !b \cdot ?b$. Two traces are considered "equivalent" by Basu and Bultan if they present the same sequence of send actions, i.e. that they are identical after erasing all receive actions. This is the case for the above example, as both traces result in $!a \cdot !b$ after erasing the receive actions. A system is language synchronizable if all of its traces are equivalent to a synchronous trace. An additional requirement is that two "equivalent" traces lead to the same configuration; when it is the case, the system is called synchronizable. For instance, taking

$$P \;= !a \cdot ?b \qquad \text{and} \qquad Q \;= ?a \cdot !b + !b \cdot ?a$$

$P|Q$ is language synchronizable but it is not synchronizable, because the asynchronous trace $!a \cdot !b \cdot ?a \cdot ?b$ does not lead to the same configuration as the synchronous trace $!a \cdot ?a \cdot !b \cdot ?b$.

For systems with more than two machines, there are at least two distinct reasonable semantics of a system of communicating machines with FIFO queues: either each message sent from $P$ to $Q$ is stored in a queue which is specific to the pair $(P, Q)$, which we will call the "peer-to-peer" semantics, or all messages sent to $Q$ from several other peers are mixed toghether in a queue that is specific to $Q$, which we will call the "mailbox" semantics.

Basu and Bultan claimed that synchronizability is decidable, first for the mailbox semantics [BBO12b], and later for other semantics, including the peer-to-peer one [BB16]. Their main argument was a small model property, stating that if all 1-bounded traces are equivalent to synchronous traces then the system is synchronizable.

This paper corrects some of these claims and discuss some related questions.

- We establish the undecidability of synchronizability for systems of 3-CFSMs with the peer-to-peer semantics (Theorem 3.14). This shows that the claim on the decidability of synchronizability for the peer-to-peer semantics [BB16] is actually wrong.
- We provide counter-examples to the small model property for systems of 3-CFSMs both for the peer-to-peer semantics (Example 2.2) and the mailbox semantics (Example 5.1) which illustrate that the claims in [BBO12b, BB16] are not proved correctly. The fact that the small model property is false for the peer-to-peer semantics is a consequence of the previous result but this is not a consequence for the mailbox semantics.
- We prove that the small model property is true for systems of 2-CFSMs and more generally for systems of communicating machines on an oriented ring both under the mailbox and peer-to-peer semantics (actually both are the same in that case) and therefore synchronizability is decidable for oriented rings (Theorem 4.16).
- We show that the reachability set of synchronizable systems is channel-recognizable (Theorem 4.14), ie the set of reachable configurations is regular.
- Finally, we show that the counter-examples we gave invalidate other claims, in particular a result used for checking *stability* [ASY16, AS18].

**Outline.** The paper first focuses on the peer-to-peer communication model. Section 2 introduces all notions of communicating finite state machines and synchronizability. In Section 3, we show that synchronizability is undecidable. Section 4 shows the decidability of synchronizability on ring topologies. Section 5 concludes with various discussions, including counter-examples about the mailbox semantics.

**Related Work.** The analysis of systems of communicating finite state machines has always been a very active topic of research. Systems with channel-recognizable (aka QDD [BG99] representable) reachability sets are known to enjoy a decidable reachability problem [Pac87]. Heussner *et al* developed a CEGAR approach based on regular model-checking [HGS12]. Classifications of communication topologies according to the decidability of the reachability problems are known for FIFO, FIFO+lossy, and FIFO+bag communications [CS08, CHS14]. In [LMP08, HLMS12], the bounded context-switch reachability problem for communicating machines extended with local stacks modeling recursive function calls is shown decidable under various assumptions. Session types dialects have been introduced for systems of communicating finite state machines [DY12], and were shown to enforce various desirable properties.

Several notions similar to the one of synchronizability have also been studied in different context. *Slack elasticity* seems to be the most general name given to a the property that a given distributed system with asynchronous communications "behaves the same" whatever the slack of communications is. This property has been studied in hardware design [MM98], with the goal of ensuring that some code transformations are semantic-preserving, in high performance computing, for ensuring the absence of deadlocks and other bugs in MPI programs [Sie05, VVGK10], but also for communicating finite state machines, like in this work, with a slightly different way of comparing the behaviours of the system at different buffer bounds. Genest *et al* introduced the notion of existentially bounded systems of communicating finite state machines, that is defined on top of Mazurkiewicz traces, aka message sequence charts in the context of communicating finite state machines [GKM06]. Finally, a notion similar to the one of existentially bounded systems has been recently introduced and christened "$k$-synchronous systems" [BEJQ18]. Existential boundedness, $k$-synchronous systems, and synchronizability are further compared in Section 5.5.

## 2. Preliminaries

### 2.1. **Messages and topologies.**

A *message set* $M$ is a tuple $\langle \Sigma_M, p, \mathsf{src}, \mathsf{dst}\rangle$ where $\Sigma_M$ is a finite set of letters (more often called messages), $p \geq 1$ and $\mathsf{src}, \mathsf{dst}$ are functions that associate to every letter $a \in \Sigma$ naturals $\mathsf{src}(a) \neq \mathsf{dst}(a) \in \{1, \dots, p\}$. We often write $a^{i \to j}$ for a message $a$ such that $\mathsf{src}(a) = i$ and $\mathsf{dst}(a) = j$; we often identify $M$ and $\Sigma_M$ and write for instance $M = \{a_1^{i_1 \to j_1}, a_2^{i_2 \to j_2}, \dots\}$ instead of $\Sigma_M = \dots$, or $w \in M^*$ instead of $w \in \Sigma_M^*$. The communication topology associated to $M$ is the graph $G_M$ with vertices $\{1, \dots, p\}$ and with an edge from $i$ to $j$ if there is a message $a \in \Sigma_M$ such that $\mathsf{src}(a) = i$ and $\mathsf{dst}(a) = j$. $G_M$ is an *oriented ring* if the set of edges of $G_M$ is $\{(i,j) \mid i + 1 = j \bmod p\}$.

### 2.2. **Traces.**

An *action* $\lambda$ over $M$ is either a send action $!a$ or a receive action $?a$, with $a \in \Sigma_M$. The peer $\mathsf{peer}(\lambda)$ of action $\lambda$ is defined as $\mathsf{peer}(!a) = \mathsf{src}(a)$ and $\mathsf{peer}(?a) = \mathsf{dst}(a)$. We write $\mathsf{Act}_{i,M}$ for the set of actions of peer $i$ and $\mathsf{Act}_M$ for the set of all actions over $M$. An *$M$-trace* $\tau$ is a finite (possibly empty) sequence of actions. We write $\mathsf{Act}_M^*$ for the set of $M$-traces, $\epsilon$ for the empty $M$-trace, and $\tau_1 \cdot \tau_2$ for the concatenation of two $M$-traces. We sometimes write $!?a$ for $!a \cdot ?a$. An $M$-trace $\tau$ is a prefix of $\upsilon$, $\tau \leq_{\mathsf{pref}} \upsilon$

if there is $\theta$ such that $\upsilon = \tau \cdot \theta$. The prefix closure $\downarrow S$ of a set of $M$-traces $S$ is the set $\{\tau \in \mathsf{Act}_M^* \mid \text{there is } \upsilon \in S \text{ such that } \tau \leq_{\mathsf{pref}} \upsilon\}$.

For an $M$-trace $\tau$ and peer ids $i, j \in \{1, \ldots, p\}$ we write

- $\mathsf{send}(\tau)$ (resp. $\mathsf{recv}(\tau)$) for the sequence of messages sent (resp. received) during $\tau$, *i.e.* $\mathsf{send}(!a) = a$, $\mathsf{send}(?a) = \epsilon$, and $\mathsf{send}(\tau_1 \cdot \tau_2) = \mathsf{send}(\tau_1) \cdot \mathsf{send}(\tau_2)$ (resp. $\mathsf{recv}(!a) = \epsilon$, $\mathsf{recv}(?a) = a$, and $\mathsf{recv}(\tau_1 \cdot \tau_2) = \mathsf{recv}(\tau_1) \cdot \mathsf{recv}(\tau_2)$).
- $\mathsf{onPeer}_i(\tau)$ for the $M$-trace of actions $\lambda$ in $\tau$ such that $\mathsf{peer}(\lambda) = i$.
- $\mathsf{onChannel}_{i \to j}(\tau)$ for the $M$-trace of actions $\lambda$ in $\tau$ such that $\lambda \in \{!a, ?a\}$ for some $a \in M$ with $\mathsf{src}(a) = i$ and $\mathsf{dst}(a) = j$.
- $\mathsf{buffer}_{i \to j}(\tau)$ for the word $w \in M^*$, if it exists, such that $\mathsf{send}(\mathsf{onChannel}_{i \to j}(\tau)) = \mathsf{recv}(\mathsf{onChannel}_{i \to j}(\tau)) \cdot w$.

**Example 2.1.** Consider $M = \langle \{a, b\}, 2, \mathsf{src}, \mathsf{dst} \rangle$ with $\mathsf{src}(a) = \mathsf{dst}(b) = 1$ and $\mathsf{src}(b) = \mathsf{dst}(a) = 2$, and let $\tau = !a?b$. Then $\mathsf{send}(\tau) = a$, $\mathsf{onPeer}_1(\tau) = \tau$, and $\mathsf{buffer}_{1 \to 2}(\tau) = a$.

An $M$-trace $\tau$ is *FIFO* (resp. a *$k$-bounded FIFO*, for $k \geq 1$) if for all $i, j \in \{1, \ldots, p\}$, for all prefixes $\tau'$ of $\tau$, $\mathsf{buffer}_{i \to j}(\tau')$ is defined (resp. defined and of length at most $k$); in other words, $\tau$ is FIFO if for every prefix $\tau'$ of $\tau$, for all $i \neq j$, the sequence of messages received from $i$ by $j$ in $\tau'$ is a prefix of the sequence of message sent from $i$ to $j$ in $\tau'$. Intuitively, an $M$-trace is *FIFO* if it is an execution of a machine that manipulates FIFO queues, with one queue per pair of peers.

An $M$-trace is *synchronous* if it is of the form $!?a_1 \cdot !?a_2 \cdots !?a_\ell$ for some $\ell \geq 0$ and $a_1, \ldots, a_\ell \in M$. In particular, a synchronous $M$-trace is a 1-bounded FIFO $M$-trace (but the converse is false). An $M$-trace $\tau$ is *stable* if $\mathsf{buffer}_{i \to j}(\tau) = \epsilon$ for all $i \neq j \in \{1, \ldots, p\}$.

Two $M$-traces $\tau, \upsilon$ are *causal-equivalent* $\tau \overset{\text{causal}}{\sim} \upsilon$ if

- $\tau, \upsilon$ are FIFO, and
- for all $i \in \{1, \ldots, p\}$, $\mathsf{onPeer}_i(\tau) = \mathsf{onPeer}_i(\upsilon)$.

Intuitively, $\tau \overset{\text{causal}}{\sim} \upsilon$ if $\tau$ is obtained from $\upsilon$ by iteratively commuting adjacent actions that are not from the same peer and do not form a "matching send/receive pair" (this is why $\tau, \upsilon$ are deemed to be FIFO). The relation $\overset{\text{causal}}{\sim}$ is a congruence with respect to concatenation.

## 2.3. Peers, systems, configurations.

A system (of communicating machines) over a message set $M$ is a tuple $\mathcal{S} = \langle \mathcal{P}_1, \ldots, \mathcal{P}_p \rangle$ where for all $i \in \{1, \ldots, p\}$, the peer $\mathcal{P}_i$ is a finite state automaton $\langle Q_i, q_{0,i}, \Delta_i \rangle$ over the alphabet $\mathsf{Act}_{i,M}$ and with (implicitly) $Q_i$ as the set of accepting states. We write $L(\mathcal{P}_i)$ for the set of $M$-traces that label a path in $\mathcal{P}_i$ starting at the initial state $q_{0,i}$.

Let the system $\mathcal{S}$ be fixed. A *configuration* $\gamma$ of $\mathcal{S}$ is a tuple $(q_1, \ldots, q_p, w_{1,2}, \ldots, w_{p-1,p})$ where $q_i$ is a state of $\mathcal{P}_i$ and for all $i \neq j$, $w_{i,j} \in M^*$ is the content of channel $i \to j$. A configuration is *stable* if $w_{i,j} = \epsilon$ for all $i, j \in \{1, \ldots, p\}$ with $i \neq j$.

Let $\gamma = (q_1, \ldots, q_p, w_{1,2}, \ldots, w_{p-1,p})$, $\gamma' = (q'_1, \ldots, q'_p, w'_{1,2}, \ldots, w'_{p-1,p})$ and $m \in M$ with $\mathsf{src}(m) = i$ and $\mathsf{dst}(m) = j$. We write $\gamma \xrightarrow{!m}_{\mathcal{S}} \gamma'$ (resp. $\gamma \xrightarrow{?m}_{\mathcal{S}} \gamma'$) if $(q_i, !m, q'_i) \in \Delta_i$ (resp. $(q_j, ?m, q'_j) \in \Delta_j$), $w'_{i,j} = w_{i,j} \cdot m$ (resp. $w_{i,j} = m \cdot w'_{i,j}$) and for all $k, \ell$ with $k \neq i$ (resp. with $k \neq j$), $q_k = q'_k$ and $w'_{k,\ell} = w_{k,\ell}$ (resp. $w'_{\ell,k} = w_{\ell,k}$). If $\tau = \lambda_1 \cdot \lambda_2 \cdots \lambda_n$, we write $\xrightarrow{\tau}_{\mathcal{S}}$ for $\xrightarrow{\lambda_1}_{\mathcal{S}} \xrightarrow{\lambda_2}_{\mathcal{S}} \ldots \xrightarrow{\lambda_n}_{\mathcal{S}}$. We often write $\xrightarrow{\tau}$ instead of $\xrightarrow{\tau}_{\mathcal{S}}$ when $\mathcal{S}$ is clear from the context. The

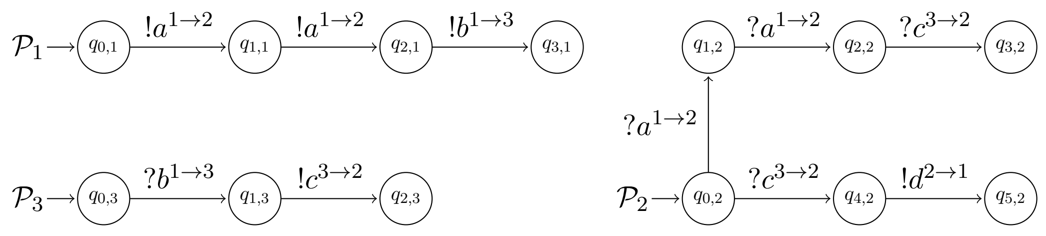


Figure 1: System of Example 2.2 and Theorem 2.4.

*initial configuration* of $\mathcal{S}$ is the stable configuration $\gamma_0 = (q_{0,1}, \dots, q_{0,p}, \epsilon, \dots, \epsilon)$. An $M$-trace $\tau$ is a trace of system $\mathcal{S}$ if there is $\gamma$ such that $\gamma_0 \xrightarrow{\tau} \gamma$. Equivalently, $\tau$ is a trace of $\mathcal{S}$ if

- it is a FIFO trace, and
- for all $i \in \{1, \dots, p\}$, $\mathsf{onPeer}_i(\tau) \in L(\mathcal{P}_i)$.

For $k \geq 1$, we write $\mathsf{Traces}_k(\mathcal{S})$ for the set of $k$-bounded traces of $\mathcal{S}$, $\mathsf{Traces}_0(\mathcal{S})$ for the set of synchronous traces of $\mathcal{S}$, and $\mathsf{Traces}(\mathcal{S})$ for $\bigcup_{k\geq0} \mathsf{Traces}_k(\mathcal{S})$.

**Example 2.2.** Consider the message set $M = \{a^{1\to2}, b^{1\to3}, c^{3\to2}, d^{2\to1}\}$ and the system $\mathcal{S} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ where $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ are as depicted in Fig. 1.

$$\begin{aligned} L(\mathcal{P}_1) = &\ \downarrow \{!a^{1\to2} \cdot !a^{1\to2} \cdot !b^{1\to3}\} \\ L(\mathcal{P}_2) = &\ \downarrow \{?a^{1\to2} \cdot ?a^{1\to2} \cdot ?c^{3\to2}\ ,\ ?c^{3\to2} \cdot !d^{2\to1}\} \\ L(\mathcal{P}_3) = &\ \downarrow \{?b^{1\to3} \cdot !c^{3\to2}\}. \end{aligned}$$

An example of a stable trace is $!a^{1\to2} \cdot !a^{1\to2} \cdot !?b^{1\to3} \cdot !c^{3\to2} \cdot ?a^{1\to2} \cdot ?a^{1\to2} \cdot ?c^{3\to2}$. Let $\tau = !a^{1\to2} \cdot !a^{1\to2} \cdot !?b^{1\to3} \cdot !?c^{3\to2} \cdot !d^{2\to1}$. Then $\tau \in \mathsf{Traces}_2(\mathcal{S})$ is a 2-bounded trace of the system $\mathcal{S}$, and $\gamma_0 \xrightarrow{\tau} (q_{3,1}, q_{5,2}, q_{2,3}, a^{1\to2}a^{1\to2}, \epsilon, d^{2\to1}, \epsilon, \epsilon, \epsilon)$.

Two traces $\tau_1, \tau_2$ are $\mathcal{S}$-*equivalent*, denoted with $\tau_1 \overset{\mathcal{S}}{\sim} \tau_2$, if $\tau_1, \tau_2 \in \mathsf{Traces}(\mathcal{S})$ and there is $\gamma$ such that $\gamma_0 \xrightarrow{\tau_i} \gamma$ for both $i = 1, 2$. It follows from the definition of $\overset{\text{causal}}{\sim}$ that if $\tau_1 \overset{\text{causal}}{\sim} \tau_2$ and $\tau_1, \tau_2 \in \mathsf{Traces}(\mathcal{S})$, then $\tau_1 \overset{\mathcal{S}}{\sim} \tau_2$.

## 2.4. **Synchronizability.**

Following [BBO12b], we define the observable behaviour of a system as its set of send traces enriched with their final configurations when they are stable. Formally, for any $k \geq 0$, we write $\mathcal{J}_k(\mathcal{S})$ and $\mathcal{I}_k(\mathcal{S})$ for the sets

$$\begin{aligned} \mathcal{J}_k(\mathcal{S}) = &\ \{\mathsf{send}(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{S})\} \\ \mathcal{I}_k(\mathcal{S}) = &\ \mathcal{J}_k(\mathcal{S}) \cup \{(\mathsf{send}(\tau), \gamma) \mid \gamma_0 \xrightarrow{\tau} \gamma, \gamma \text{ stable}, \tau \in \mathsf{Traces}_k(\mathcal{S})\}. \end{aligned}$$

Synchronizability is then defined as the slack elasticity of these observable behaviours.

**Definition 2.3** (Synchronizability [BB11, BBO12b])**.** A system $\mathcal{S}$ is *synchronizable* if $\mathcal{I}_0(\mathcal{S}) = \mathcal{I}(\mathcal{S})$. $\mathcal{S}$ is called *language synchronizable* if $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}(\mathcal{S})$.

For convenience, we also introduce a notion of $k$-synchronizability: for $k \geq 1$, a system $\mathcal{S}$ is *$k$-synchronizable* if $\mathcal{I}_0(\mathcal{S}) = \mathcal{I}_k(\mathcal{S})$, and *language* $k$-synchronizable if $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_k(\mathcal{S})$. A system is therefore (language) synchronizable if and only if it is (language) $k$-synchronizable for all $k \geq 1$.

**Theorem 2.4.** *There is a system $\mathcal{S}$ that is 1-synchronizable, but not synchronizable.*

*Proof.* Consider again the system $\mathcal{S}$ of Example 2.2. Let $\gamma_{ijk} := (q_{i,1}, q_{j,2}, q_{k,3}, \epsilon, \ldots, \epsilon)$. If the buffers are 1-bounded, $\mathcal{P}_1$ must wait that the first $a$ message has been received before sending the $b$ message. Therefore

$$\begin{array}{rcl} \mathcal{J}_0(\mathcal{S}) & = & \downarrow\{a^{1\to2} \cdot a^{1\to2} \cdot b^{1\to3} \cdot c^{3\to2}\} \\ \mathcal{J}_1(\mathcal{S}) & = & \mathcal{J}_0(\mathcal{S}) \end{array}$$

On the other hands, if the buffers can host two transiting messages, it becomes possible for $\mathcal{P}_1$ to send $b$ before the first $a$ is received by $\mathcal{P}_2$, so it becomes possible for $\mathcal{P}_3$ to receive $b$ and send $c$, and finally $\mathcal{P}_2$ may decide to receive $c$ before receiving any $a$ message. Consequently,

$$\begin{array}{rcl} \mathcal{J}_2(\mathcal{S}) & = & \downarrow\{a^{1\to2} \cdot a^{1\to2} \cdot b^{1\to3} \cdot c^{3\to2} \cdot d^{2\to1}\} \\ \mathcal{I}_k(\mathcal{S}) & = & \mathcal{J}_k(\mathcal{S}) \cup \mathsf{Stab} \qquad \text{for all } k \geq 0 \end{array}$$

where $\mathsf{Stab} = \{(\epsilon, \gamma_0), (a^{1\to2}, \gamma_{110}), (a^{1\to2} \cdot a^{1\to2}, \gamma_{220}), (a^{1\to2} \cdot a^{1\to2} \cdot b^{1\to3}, \gamma_{321}), (a^{1\to2} \cdot a^{1\to2} \cdot b^{1\to3} \cdot c^{3\to2}, \gamma_{332})\}$. □

This example contradicts[1] Theorem 4 in [BB16], which stated that $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_1(\mathcal{S})$ implies $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}(\mathcal{S})$. This also shows that the decidability of synchronizability for peer-to-peer communications is open despite the claim in [BB16]. The next section closes this question.

**Remark 2.5.** In Section 5, we give a counter-example that addresses communications with mailboxes, *i.e.* the first communication model considered in all works about synchronizability, and we list several other published theorems that our counter-example contradicts.

## 3. Undecidability of Synchronizability

In this section, we show the undecidability of synchronizability for systems with *at least three* peers. Although the reachability problem is undecidable for two peers, we cannot establish the undecidability of the synchronizability of a system with two peers. The reasons for that are twofolds.

First, synchronizability only deals with messages that are sent and received, which is orthogonal to reachability. We therefore rely on the undecidability of the message reception problem: given a FIFO automaton $\mathcal{A}$ (i.e. an automaton that can both enqueue and dequeue messages in a unique channel) and a special message $\underline{m}$, decide whether there exists a trace of $\mathcal{A}$ that contains $?\underline{m}$.

Second, synchronizability constrains a lot the communications. In particular, when an automaton must be in a mixed state (ready to send and receive), it imposes some commutativity of the two actions (see next section), and as a consequence, a synchronizable system with two peers cannot simulate a FIFO automaton. A third peer is necessary to get rid of all the constraints imposed by synchronizability.

To sum up, we reduce the message reception problem on a FIFO automaton $\mathcal{A}$ to the synchronizability of a system with three peers: we construct a system $\mathcal{S}''_{\mathcal{A},m}$ such that the synchronizability of $\mathcal{S}''_{\mathcal{A},m}$ is equivalent to the non-reception of the special message $\underline{m}$ in $\mathcal{A}$.

[1]see also the discussion in Section 5.2

3.1. **An Undecidable Problem on FIFO automata.** A *FIFO automaton* is a finite state automaton $\mathcal{A} = \langle Q, \mathsf{Act}_\Sigma, \Delta, q_0\rangle$ over an alphabet of the form $\mathsf{Act}_\Sigma$ for some finite set of letters $\Sigma$ with all states being accepting states. A FIFO automaton can be thought as a system with only one peer, with the difference that, according to our definition of systems, a peer can only send messages to peers different from itself, whereas a FIFO automaton enqueues and dequeues letters in a unique FIFO queue, and thus, in a sense, "communicates with itself". All notions we introduced for systems are obviously extended to FIFO automata. In particular, a configuration of $\mathcal{A}$ is a tuple $\gamma = (q, w) \in Q \times \Sigma^*$, it is stable if $w = \epsilon$, and the transition relation $\gamma \xrightarrow{\tau} \gamma'$ is defined exactly the same way as for systems.Let us now state the problem that we will consider

**Definition 3.1** (Message reception problem)**.** The *message reception problem* is the following decision problem

**Input:** a FIFO automaton $\mathcal{A} = \langle Q, \mathsf{Act}_\Sigma, \Delta, q_0\rangle$ and a distinguished message $\underline{m} \in \Sigma$.
**Question:** is there a trace $\tau$ such that $\tau \cdot ?m \in \mathsf{Traces}(\mathcal{A})$ ?

**Remark 3.2.** A similar but different problem to the message reception problem is the *executable reception problem* in [BZ83] which consists to decide for a given control-state $q$ and a message $m$ such that $q \xrightarrow{?m}$, whether there exists a reachable configuration $(q, mw)$ with $w \in \Sigma^*$ : this problem is proved undecidable for systems of 2-CFSMs in the (non available) associated technical report [BZ81]. The proof reduces the halting problem to the executable reception problem by using the two unidirectional FIFO channels to simulate the tape of a Turing machine. We present another proof technique (using the undecidability of the existence of a tiling [LP98]) for another model, the FIFO automata model and another property. Then we will simulate any FIFO automaton $\mathcal{A}$ by an associated particular system of 3-CFSMs $\mathcal{S}_\mathcal{A}$ that will have the property to be synchronizable iff a message $m$ never appears in a trace of $\mathcal{A}$. We dont try to simulate $\mathcal{A}$ by a system of 2-CFSMs since for systems of 2-CFSMs, synchronizability is decidable and message reception problem is undecidable.

**Lemma 3.3.** *The message reception problem is undecidable.*

*Proof.* We consider the problem of the existence of a finite, but arbitrarily large, tiling for a given set of Wang tiles and a pair of initial and final tiles. More precisely, consider a tuple $\mathcal{T} = \langle T, t_0, t_F, H, V\rangle$ where

- $T$ is a finite set of tiles,
- $t_0, t_F \in T$ are respectively the initial tile and the final tile, and
- $H, V \subseteq T \times T$ are horizontal and vertical compatibility relations.

Without loss of generality, we assume that there is a "padding tile" $\square$ such that $(t, \square) \in H \cap V$ for all $t \in T$. For a natural $n \geq 1$, a $n$-tiling is a function $f : \mathbb{N} \times \{1, \ldots, n\} \to T$ such that

- $f(0, 1) = t_0$,
- there are $(i_F, j_F) \in \mathbb{N} \times \{1, \ldots, n\}$ such that $f(i_F, j_F) = t_F$,
- $(f(i, j), f(i, j + 1)) \in H$ for all $(i, j) \in \mathbb{N} \times \{1, \ldots, n - 1\}$, and
- $(f(i, j), f(i + 1, j)) \in V$ for all $(i, j) \in \mathbb{N} \times \{1, \ldots, n\}$.

The problem of deciding, given a tuple $\mathcal{T} = \langle T, t_0, t_F, H, V\rangle$, whether there is some $n \geq 1$ for which there exists a $n$-tiling, is undecidable [LP98], intuitively because it is equivalent to the halting problem for a Turing machine working with a half-infinite ribbon.

In the remainder, we explain how this tiling problem can be reduced to the message reception problem. Intuitively, we construct a FIFO automaton that outputs the first row of the tiling, storing it into the queue, and guessing at the same time the width $n$ of the tiling. Then, for all next row $i+1$, the automaton outputs the row tile after tile, popping a tile of row $i$ in the queue in between so as to check that each tile of row $i+1$ vertically coincides with the corresponding tile of row $i$.

More precisely, let $\mathcal{T} = \langle T, t_0, t_F, H, V\rangle$ be fixed. We define the FIFO automaton $\mathcal{A}_{\mathcal{T}} = \langle Q, \Sigma, \Delta, q_0\rangle$ with $Q = \{q_{t,0}, q_{\downarrow = t}, q_{\leftarrow = t}, q_{\leftarrow = t, \downarrow = t'} \mid t \in T, t' \in T \cup \{\$\}\} \cup \{q_0, q_1\}$, $\Sigma = T \cup \{\$\}$, and $\Delta \subseteq Q \times \mathsf{Act}_\Sigma \times Q$, with

$$\begin{array}{rcl}\Delta & = & \{(q_0, !t_0, q_{t_0,0})\} \cup \{(q_{t,0}, !t', q_{t',0}) \mid (t,t') \in H\} \cup \{(q_{t,0}, !\$, q_1) \mid t \in T\} \\ & \cup & \{(q_1, ?t, q_{\downarrow = t}) \mid t \in T)\} \cup \{(q_{\downarrow = t}, !t', q_{\leftarrow = t'}) \mid (t,t') \in V\} \\ & \cup & \{(q_{\leftarrow = t}, ?t', q_{\leftarrow = t, \downarrow = t'}) \mid t \in T, t' \in T \cup \{\$\}\} \\ & \cup & \{(q_{\leftarrow = t, \downarrow = t'}, !t'', q_{\leftarrow = t''}) \mid (t,t'') \in H \text{ and } (t',t'') \in V\} \\ & \cup & \{(q_{\leftarrow = t, \downarrow = \$}, !\$, q_1) \mid t \in T\}\end{array}$$

Note that the automaton moves to state $q_{\downarrow = t}$ after it has popped the first tile $t$ of row $i$ (it needs to remember it), then moves to state $q_{\leftarrow = t'}$ after it has decided to put a tile $t'$ on row $i+1$ above tile $t$ (it only needs to remember $t'$), then moves to state $q_{\leftarrow = t', \downarrow = t''}$ after it has popped the second tile $t''$ of row $i$, and so on. Therefore, any execution of $\mathcal{A}_{\mathcal{T}}$ is of the form

$$!t_{1,1} \cdot !t_{1,2} \cdots !t_{1,n} \cdot !\$ \cdot ?t_{1,1} \cdot !t_{2,1} \cdot ?t_{1,2} \cdot !t_{2,2} \cdots !t_{2,n} \cdot ?\$ \cdot !\$ \cdot ?t_{2,1} \cdot !t_{3,1} \cdots$$

where $t_{1,1} = t_0$, $(t_{i,j}, t_{i+1,j}) \in V$ and $(t_{i,j}, t_{i,j+1}) \in H$. The following two are thus equivalent:

- there is $n \geq 1$ such that $\mathcal{T}$ admits a $n$-tiling
- there is a trace $\tau \in \mathsf{Traces}(\mathcal{A})$ that contains $?t_F$. □

Note that, from this result, we can easily deduce the undecidability of the reachability problem for a system consisting of two machines, a sender and a receiver, a FIFO channel between them, and an extra channel with rendez-vous synchronization. Indeed, such a system may simulate a FIFO automaton: the sender does exactly the same as the FIFO automaton, except that for reception it uses a rendez-vous synchronization to ask the receiver for performing a reception, and waits for an acknowledgment that this reception has indeed been done. In the following, we will exploit this idea, although not with two machines and a rendez-vous channel, but with three machines and FIFO channels only.

3.2. **A System that Simulates a FIFO Automaton.** We are now going to define a communicating system that simulates a FIFO automaton $\mathcal{A}$. This system, that we will write $\mathcal{S}_{\mathcal{A}}$, will later be completed so as to reduce the message reception problem to synchronizability. The system $\mathcal{S}_{\mathcal{A}}$ consists of three peers $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ that are connected through the following topology.

$$\begin{tikzcd} \boxed{\mathcal{P}_1} \arrow[rr] \arrow[dr, shift left] & & \boxed{\mathcal{P}_2} \arrow[dl, shift left] \\ & \boxed{\mathcal{P}_3} \arrow[ul, shift left] \arrow[ur, shift left] & \end{tikzcd}$$

Intuitively, we want $\mathcal{P}_1$ to mimick $\mathcal{A}$'s decisions and the channel $1 \to 2$ to mimick $\mathcal{A}$'s queue. When $\mathcal{A}$ would enqueue a letter $a$ , peer 1 sends $a^{1\to 2}$ to peer 2, and when $\mathcal{A}$ would dequeue a letter $a$, peer $\mathcal{P}_1$ sends to peer $\mathcal{P}_2$ via peer $\mathcal{P}_3$ the order to dequeue $a$, and waits for the acknowledgement that the order has been correcly executed. So the only role of $\mathcal{P}_3$ is to enable a second channel between $\mathcal{P}_1$ and $\mathcal{P}_2$ for "rendez-vous communications".

Let us now define these peers and $\mathcal{S}_\mathcal{A}$ more formaly. Let $\mathcal{A} = \langle Q_\mathcal{A}, \mathsf{Act}_\Sigma, \Delta_\mathcal{A}, q_0\rangle$ a FIFO automaton be fixed. Let $M$ be such that all messages of $\Sigma$ can be exchanged among all peers in all directions but $2 \to 1$, *i.e.*

$$M = \{a^{1\to2}, a^{1\to3}, a^{2\to3}, a^{3\to1}, a^{3\to2} \mid a \in \Sigma\}$$

Peer $\mathcal{P}_1$ is obtained by replacing every $!a$ transition of $\mathcal{A}$ with a $!a^{1\to2}$ transition, and every $?a$ transition with the sequence of transitions $!a^{1\to3}?a^{3\to1}$. Formally, $\mathcal{P}_1 = \langle Q_1, q_{0,1}, \Delta_1\rangle$ is defined by $Q_1 = Q_\mathcal{A} \uplus \{q_\delta \mid \delta \in \Delta_\mathcal{A}\}$ and

$$\begin{array}{rcl}\Delta_1 & = & \{(q, !a^{1\to2}, q') \mid (q, !a, q') \in \Delta_\mathcal{A}\} \\ & \cup & \{(q, !a^{1\to3}, q_\delta), (q_\delta, ?a^{3\to1}, q') \mid \delta = (q, ?a, q') \in \Delta_\mathcal{A}\}.\end{array}$$

The *modus operandi* of $\mathcal{P}_2$ and $\mathcal{P}_3$ is rather simple: $\mathcal{P}_3$ propagates all messages it receives, and $\mathcal{P}_2$ executes all orders it receives and sends back an acknowledgement when this is done. So both $\mathcal{P}_2$ and $\mathcal{P}_3$ operate with an "infinite loop". For some technical reason that will be later explained, we need to make sure that $\mathcal{P}_2$ never comes back to its initial state. To do so, we simply "unroll the loop" once in $\mathcal{P}_2$. Formally, let $\mathcal{P}_2 = \langle Q_2, q_{0,2}, \Delta_2\rangle$ and $\mathcal{P}_3 = \langle Q_3, q_{0,3}, \Delta_3\rangle$ be defined by

$$\begin{array}{l} Q_2 = \{q_{0,2}, q_{1,2}\} \cup \{q_{a,1}, q_{a,2} \mid a \in \Sigma\} \qquad Q_3 = \{q_{0,3}\} \cup \{q_{a,1}, q_{a,2}, q_{a,3} \mid a \in \Sigma\} \\ \Delta_2 = \{(q_{0,2}, ?a^{3\to2}, q_{a,1}), (q_{1,2}, ?a^{3\to2}, q_{a,1}), (q_{a,1}, ?a^{1\to2}, q_{a,2}), (q_{a,2}, !a^{2\to3}, q_{1,2}) \mid a \in \Sigma\} \\ \Delta_3 = \{(q_{0,3}, ?a^{1\to3}, q_{a,1}), (q_{a,1}, !a^{3\to2}, q_{a,2}), (q_{a,2}, ?a^{2\to3}, q_{a,3}), (q_{a,3}, !a^{3\to1}, q_{0,3}) \mid a \in \Sigma\}\end{array}$$

**Example 3.4.** Consider $\Sigma = \{a, \underline{m}\}$ and the FIFO automaton $\mathcal{A} = \langle\{q_0, q_1\}, \mathsf{Act}_\Sigma, \Delta, q_0\rangle$ with transition relation $\Delta_\mathcal{A} = \{(q_0, !a, q_0), (q_0, !\underline{m}, q_1), (q_1, ?a, q_0), (q_1, ?\underline{m}, q_0)\}$. Then $\mathcal{A}$ and the peers $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ are depicted in Fig. 2.

Let $\mathcal{S}_\mathcal{A} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\rangle$. There is a tight correspondence between the $k$-bounded traces of $\mathcal{A}$, for $k \geq 1$, and the $k$-bounded traces of $\mathcal{S}_\mathcal{A}$: every trace $\tau \in \mathsf{Traces}_k(\mathcal{A})$ induces the trace $h(\tau) \in \mathsf{Traces}_k(\mathcal{S}_\mathcal{A})$ where $h : \mathsf{Act}_\Sigma \to \mathsf{Act}_M^*$ is the homomorphism from the traces of $\mathcal{A}$ to the traces of $\mathcal{S}_\mathcal{A}$ defined by $h(!a) = !a^{1\to2}$ and $h(?a) = !?a^{1\to3} \cdot !?a^{3\to2} \cdot ?a^{1\to2} \cdot !?a^{2\to3} \cdot !?a^{3\to1}$. The converse is not true: there are traces of $\mathcal{S}_\mathcal{A}$ that are not prefixes of a trace $h(\tau)$ for some $\tau \in \mathsf{Traces}_k(\mathcal{A})$. This happens when $\mathcal{P}_1$ sends an order to dequeue $a^{1\to3}$ that correspond to a transition $?a$ that $\mathcal{A}$ cannot execute. In that case, the system blocks when $\mathcal{P}_2$ has to execute the order. To sum up, we obtain the following result.

**Lemma 3.5.** *For all $k \geq 0$,*

$$\begin{array}{rl}\mathsf{Traces}_k(\mathcal{S}_\mathcal{A}) = & \downarrow\{h(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{A})\} \\ & \cup \downarrow\{h(\tau) \cdot !?a^{1\to3} \cdot !?a^{3\to2} \mid \exists q, b, w, q' \ s.t. \\ & \qquad \tau \in \mathsf{Traces}_k(\mathcal{A}), (q_0, \epsilon) \xrightarrow{\tau} (q, bw), (q, ?a, q') \in \Delta \mathit{and}\ b \neq a\}.\end{array}$$

3.3. **A Synchronizable System.** Let us fix a special message $\underline{m} \in \Sigma$. In this section, we define a system $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3\rangle$ where $\mathcal{P}_1$ and $\mathcal{P}_3$ are like in the system $\mathcal{S}_\mathcal{A}$, but $\mathcal{P}'_2$ is a new peer. This system will later be combined with $\mathcal{S}_\mathcal{A}$ so as to form the whole system that will be used in the reduction of the message reception problem to the synchronizability problem. The main purpose of $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3\rangle$ is to be a synchronizable system that will "make synchronizable" all traces of $\mathcal{S}_\mathcal{A}$ except the ones that contain $m^{2\to3}$, which are the only ones we want to care about in the reduction. Our outline for this section is therefore the following: (1) define $\mathcal{S}'_{\mathcal{A},\underline{m}}$, (2) compute its synchronous traces, (3) show that they "make

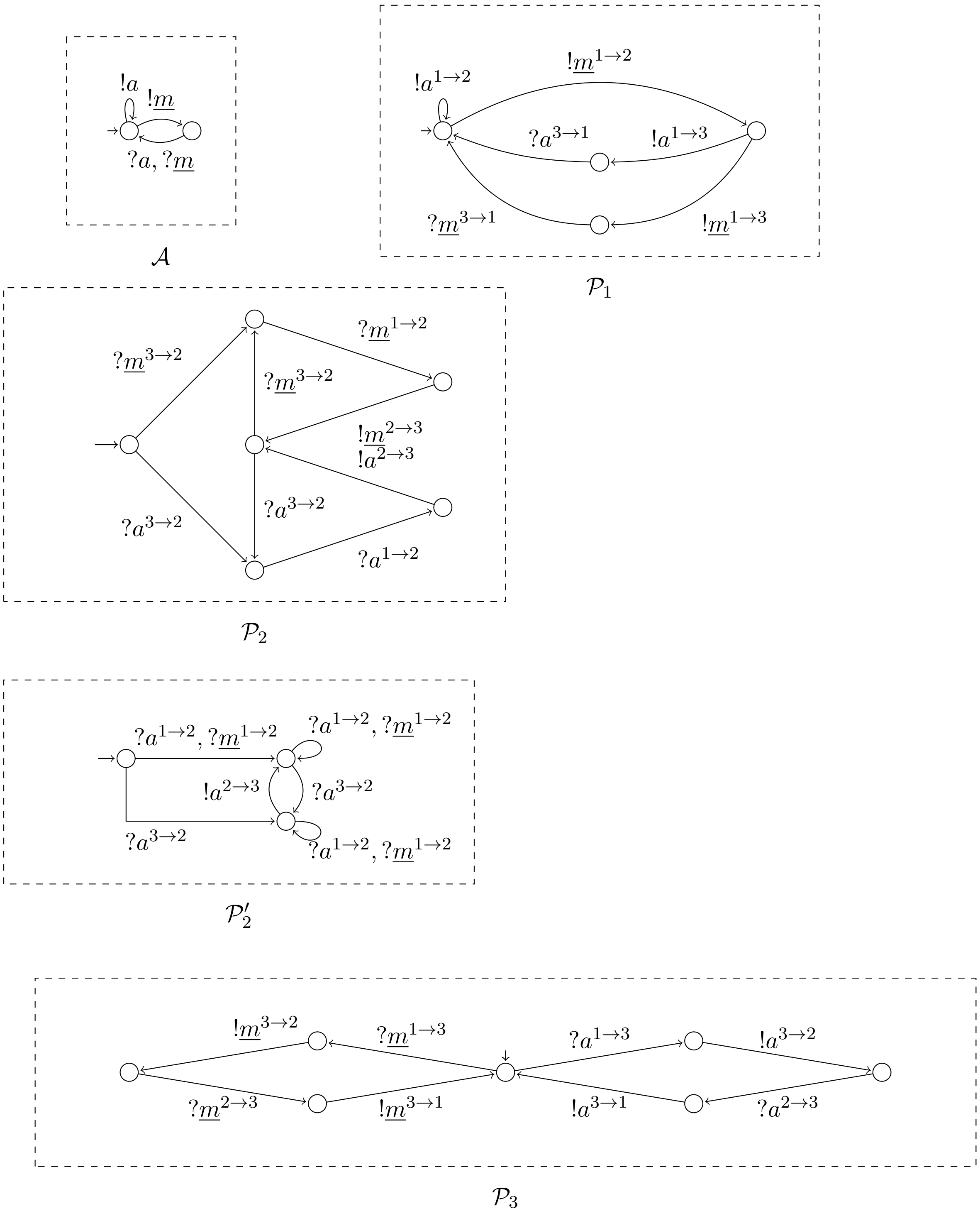

$!a$
$!\underline{m}$
$?a, ?\underline{m}$
$\mathcal{A}$
$!\underline{m}^{1\to 2}$
$!a^{1\to 2}$
$?a^{3\to 1}$
$!a^{1\to 3}$
$?\underline{m}^{3\to 1}$
$!\underline{m}^{1\to 3}$
$\mathcal{P}_1$
$?\underline{m}^{1\to 2}$
$?\underline{m}^{3\to 2}$
$?\underline{m}^{3\to 2}$
$!\underline{m}^{2\to 3}$
$!a^{2\to 3}$
$?a^{3\to 2}$
$?a^{3\to 2}$
$?a^{1\to 2}$
$\mathcal{P}_2$
$?a^{1\to 2}, ?\underline{m}^{1\to 2}$
$?a^{1\to 2}, ?\underline{m}^{1\to 2}$
$!a^{2\to 3}$
$?a^{3\to 2}$
$?a^{3\to 2}$
$?a^{1\to 2}, ?\underline{m}^{1\to 2}$
$\mathcal{P}'_2$
$!\underline{m}^{3\to 2}$
$?\underline{m}^{1\to 3}$
$?a^{1\to 3}$
$!a^{3\to 2}$
$?\underline{m}^{2\to 3}$
$!\underline{m}^{3\to 1}$
$!a^{3\to 1}$
$?a^{2\to 3}$
$\mathcal{P}_3$

Figure 2: The FIFO automaton $\mathcal{A}$ of Example 3.4 and its associated systems $\mathcal{S}_{\mathcal{A}} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ and $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3 \rangle$. The sink state $q_\perp$ and the transitions $q \xrightarrow{?\underline{m}^{3\to 2}} q_\perp$ are omitted in the representation of $\mathcal{P}'_2$.

synchronizable" the asynchronous traces of $\mathcal{S}_\mathcal{A}$ where $!\underline{m}^{2\to3}$ does not occur, and (4) show that it is a synchronizable system.

Let us start with the definition of $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3\rangle$. Intuitively, the new peer $\mathcal{P}'_2$ will always be able to receive any message from peer $\mathcal{P}_1$, in particular at the time the message is sent. Moreover, like $\mathcal{P}_2$, $\mathcal{P}'_2$ will also be able to receive orders to dequeue from peer $\mathcal{P}_3$, but instead of executing the order before sending an acknowledgement, it will ignore the order as follows. If $\mathcal{P}'_2$ receives the order to dequeue a message $a^{1\to2} \neq \underline{m}^{1\to2}$, $\mathcal{P}'_2$ acknowledges $\mathcal{P}_3$ but does not dequeue $a$ in the $1 \to 2$ queue. If the order was to dequeue $\underline{m}$, $\mathcal{P}'_2$ blocks in the sink state $q_\perp$ and does not send an acknowledgement to $\mathcal{P}_3$. As for $\mathcal{P}_2$, we "unroll the loop" so as to make sure that it not possible to come back to the initial state of $\mathcal{P}'_2$. Formally, $\mathcal{P}'_2$ is defined as follows.

$$\begin{array}{rcl}
Q'_2 & = & \{q_{0,2}, q'_{0,2}\} \cup \{q'_{a,1} \mid a \in \Sigma, a \neq \underline{m},\} \cup \{q_\perp\} \\
\Delta'_2 & = & \{(q_{0,2}, ?a^{1\to2}, q'_{0,2}), (q, ?a^{1\to2}, q) \mid a \in \Sigma, q \neq q_{0,2}\} \\
& \cup & \{(q_{0,2}, ?a^{3\to2}, q'_{a,1}), (q'_{0,2}, ?a^{3\to2}, q'_{a,1}), (q'_{a,1}, !a^{2\to3}, q'_{0,2}), \mid a \in \Sigma, a \neq \underline{m}\} \\
& \cup & \{(q, ?\underline{m}^{3\to2}, q_\perp) \mid q \in Q'_2\}
\end{array}$$

**Example 3.6.** For $\Sigma = \{a, \underline{m}\}$, and $\mathcal{A}$ as in Example 3.4, $\mathcal{P}'_2$ is depicted in Fig. 2 (omitting the transitions to the sink state $q_\perp$).

Let us now compute the set of all synchronous traces of $\mathcal{S}'_{\mathcal{A},\underline{m}}$. Observe first that the system $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3\rangle$ contains many synchronous traces: when $\mathcal{P}_1$ sends a message $a^{1\to2}$, it can always do it synchronously, because $\mathcal{P}'_2$ is always ready to receive it. When $\mathcal{P}_1$ sends an order for dequeuing, the transmission of this order to $\mathcal{P}'_2$ through $\mathcal{P}_3$ can be synchronous. If this order is not the order to dequeue $\underline{m}^{1\to2}$, then $\mathcal{P}'_2$ sends the acknowledgment to $\mathcal{P}_1$ through $\mathcal{P}_3$, which can also happen synchronously. Note in particular that, unlike in $\mathcal{S}_\mathcal{A}$, peer 1 does not block forever after it has sent an order $a^{1\to3}$ in a configuration where the first message to be dequeued in channel $1 \to 2$ is not $a$, because $\mathcal{P}'_2$ now acknowledges any order (except for $\underline{m}$). Therefore any trace $\tau$ labeling a path in automaton $\mathcal{P}_1$ can be lifted to a synchronous trace $\tau' \in \mathsf{Traces}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$ provided $!\underline{m}^{1\to3}$ does not occur in $\tau$. However, if $\mathcal{P}_1$ takes a $!\underline{m}^{1\to3}$ transition, it gets blocked for ever waiting for $\underline{m}^{3\to1}$. Therefore, if $!m^{1\to3}$ occurs in a synchronous trace $\tau$ of $\mathcal{S}'_{\mathcal{A},\underline{m}}$, it must be in the last four actions, and this trace leads to a deadlock configuration in which both 1 and 3 wait for an acknowledgement and 2 is in the sink state.

Let us now formalize further these observations. Let $L^{\underline{m}}(\mathcal{A})$ be the set of traces $\tau$ recognized by $\mathcal{A}$ as a finite state automaton (over the alphabet $\mathsf{Act}_\Sigma$) such that either $?\underline{m}$ does not occur in $\tau$, or it occurs only once and it is the last action of $\tau$.

**Example 3.7.** With $\mathcal{A}$ as in Example 3.4, $L^{\underline{m}}(\mathcal{A}) = \downarrow\Big((!a^* \cdot !\underline{m} \cdot ?a)^* \cdot !a^* \cdot !\underline{m} \cdot ?\underline{m}\Big)$.

The next lemma formalizes the observations we did about how synchronous traces of $\mathcal{S}'_{\mathcal{A},\underline{m}}$ correspond, up to an homomorphism, to $L^{\underline{m}}(\mathcal{A})$, and gives the desired computation of the synchronous traces of $\mathcal{S}'_{\mathcal{A},\underline{m}}$.

**Lemma 3.8.** $\mathsf{Traces}_0(\mathcal{S}'_{\mathcal{A},\underline{m}}) = \downarrow\{h'(\tau) \mid \tau \in L^{\underline{m}}(\mathcal{A})\}$, *where* $h' : \mathsf{Act}^*_\Sigma \to \mathsf{Act}^*_M$ *is the morphism defined by*

- $h'(!a) = !?a^{1\to2}$ *for all* $a \in \Sigma$,
- $h'(?a) = !?a^{1\to3} \cdot !?a^{3\to2} \cdot !?a^{2\to3} \cdot !?a^{3\to1}$ *for all* $a \neq \underline{m}$, *and*
- $h'(?\underline{m}) = !?\underline{m}^{1\to3} \cdot !?\underline{m}^{3\to2}$.

As a consequence, we get the following result, which will be later used to "make synchronizable" all traces of $\mathcal{S}_{\mathcal{A}}$ that do not contain $!m^{2\to 3}$.

**Lemma 3.9.** *For all trace $\tau \in \mathsf{Traces}(\mathcal{S}_{\mathcal{A}})$ such that $!\underline{m}^{2\to 3} \notin \tau$, there is a synchronous trace $\tau' \in \mathsf{Traces}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$ such that $\mathsf{send}(\tau) = \mathsf{send}(\tau')$.*

*Proof.* Let $\tau \in \mathsf{Traces}(\mathcal{S}_{\mathcal{A}})$ such that $!\underline{m}^{2\to 3} \notin \tau$ be fixed. By Lemma 3.5, there is $\tau_0 \in \mathsf{Traces}(\mathcal{A})$ such that $\tau = h(\tau_0)$. By definition of $h$, $?\underline{m}$ does not occur in $\tau_0$. Let $\tau' = h'(\tau_0)$. By Lemma 3.8, $\tau' \in \mathsf{Traces}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$, and by definition of $h$ and $h'$, and the fact that $?\underline{m}$ does not occur in $\tau_0$, $\mathsf{send}(\tau) = \mathsf{send}(\tau')$. □

Let us finally establish the synchronizability of $\mathcal{S}'_{\mathcal{A},\underline{m}}$. We consider some arbitrary asynchronous trace $\tau \in \mathsf{Traces}(\mathcal{S}'_{\mathcal{A},\underline{m}})$ that we need to be equivalent, up to receive actions, to a synchronous trace. Let us reason message by message on $\tau$, by case analysis on the channel of the message.

- If $\mathcal{P}_1$ sends a message $a^{1\to 2}$ to $\mathcal{P}'_2$, it is always possible to make sure that $\mathcal{P}'_2$ receives it immediately. Indeed, there are two cases: if $a^{1\to 2}$ was not received in $\tau$, adding $?a^{1\to 2}$ right after $!a^{1\to 2}$ in $\tau$ yields a valid trace in $\mathsf{Traces}(\mathcal{S}'_{\mathcal{A},\underline{m}})$, because the transitions $?a^{1\to 2}$ in $\mathcal{P}'_2$ do not modify the control state; similary, if $a^{1\to 2}$ was received in $\tau$ but not immediately after $!a^{1\to 2}$, it is possible to move $?a^{1\to 2}$ immediately after $!a^{1\to 2}$ in $\tau$ while keeepin a valid trace in $\mathsf{Traces}(\mathcal{S}'_{\mathcal{A},\underline{m}})$, again because the transitions $?a^{1\to 2}$ in $\mathcal{P}'_2$ do not modify the control state. In the remainder, we therefore assume that all $!a^{1\to 2}$ in $\tau$ are immediately followed by $?a^{1\to 2}$.
- If $\mathcal{P}_1$ sends a message to $\mathcal{P}_3$, it is always possible to make sure that $\mathcal{P}_3$ receives it immediately. Indeed, it is always the case that whenever $\mathcal{P}_1$ sends a message to $\mathcal{P}_3$, $\mathcal{P}_3$ is in its initial state, otherwise $\mathcal{P}_1$ would be waiting for an acknowledgment from $\mathcal{P}_3$, and won't be able to send a message to $\mathcal{P}_3$.
- For the same reason, if $\mathcal{P}'_2$ sends a message $a^{2\to 3}$ to $\mathcal{P}_3$, it must be the case that $\mathcal{P}_3$ is blocked waiting for this message, and we can either move $?a^{2\to 3}$ right after $!a^{2\to 3}$ or insert it in $\tau$ if it was not there.
- For the same reason, if $\mathcal{P}_3$ sends a message to $\mathcal{P}_1$, it is always possible to make sure that $\mathcal{P}_1$ receives it immediately.
- Finally, let us consider the case of $\mathcal{P}_3$ sending a message $a^{3\to 2}$ to $\mathcal{P}'_2$. It must be the case that $\mathcal{P}'_2$ is either in its initial state $q_{0,2}$ or in the similar receiving state $q_{0,2'}$ at the moment of the sending. Indeed, if $\mathcal{P}'_2$ was in a state $q'_{a,1}$, $\mathcal{P}_3$ would be blocked waiting for an acknowledgment from $\mathcal{P}'_2$, so it would not have been able to send a message to $\mathcal{P}'_2$. So $\mathcal{P}'_2$ is either in its initial state $q_{0,2}$ or in the similar receiving state $q_{0,2'}$ at the moment of the sending $!a^{3\to 2}$. With the same reasoning, it also holds that the buffer $3 \to 2$ was empty before the sending of $a^{3\to 2}$. Since there are no send transitions from $q_{0,2}$ and $q_{0,2'}$, and since we assumed above that all $?a^{1\to 2}$ immediately follow their matching send in $\tau$, the only possible first action of $\mathcal{P}'_2$ in $\tau$ after $!a^{3\to 2}$ is $?a^{3\to 2}$. If this action exists in $\tau$, we can move it right after the sending of $\mathcal{P}_3$ up to causal equivalence. If $?a^{2\to 3}$ does not happen in $\tau$ after this $!a^{3\to 2}$, it means that no further action of $\mathcal{P}'_2$ occurs in $\tau$ after $!a^{3\to 2}$. So we can insert $?a^{3\to 2}$ in $\tau$ right after $!a^{3\to 2}$ while keeping a valid trace in $\mathsf{Traces}(\mathcal{S}'_{\mathcal{A},\underline{m}})$.

In order to sum up what we showed with this case analysis, let us introduce the homomorphism $h'' : \mathsf{Act}_M^* \to \mathsf{Act}_M^*$ such that

- $h''(!a^{1\to 2}) = !?a^{1\to 2}$,

- $h''(?a^{1\to 2}) = \epsilon$, and
- $h''(\lambda) = \lambda$ otherwise.

For any given $\tau \in \mathsf{Traces}(\mathcal{S}'_{\mathcal{A},\underline{m}})$, our case analysis shows that $h''(\tau) \in \mathsf{Traces}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. It is also easy to observe that $\mathsf{send}(\tau) = \mathsf{send}(h''(\tau))$. As a consequence, we get the desired result.

**Lemma 3.10.** *$\mathcal{S}'_{\mathcal{A},\underline{m}}$ is synchronizable.*

3.4. **Reducing the Message Reception Problem to Synchronizability.** We are now close to reach our initial goal, namely to reduce the message reception problem to synchronizability. Let us consider the system $\mathcal{S}''_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}_2 \cup \mathcal{P}'_2, \mathcal{P}_3 \rangle$, where $\mathcal{P}_2 \cup \mathcal{P}'_2 = \langle Q_2 \cup Q'_2, q_{02}, \Delta_2 \cup \Delta'_2 \rangle$ is obtained by merging the initial state $q_{0,2}$ of $\mathcal{P}_2$ and $\mathcal{P}'_2$.

It is now time to explain why we defined $\mathcal{P}_2$ and $\mathcal{P}'_2$ so that it is not possible to come back to the initial state $q_{0,2}$. While doing so, we make sure that any trace of $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is either a trace of $\mathcal{S}_{\mathcal{A}}$ or a trace of $\mathcal{S}'_{\mathcal{A},\underline{m}}$.

$$\mathsf{Traces}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathsf{Traces}_k(\mathcal{S}_{\mathcal{A}}) \cup \mathsf{Traces}_k(\mathcal{S}'_{\mathcal{A},\underline{m}}),$$

In particular,

$$\mathcal{J}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{J}_k(\mathcal{S}_{\mathcal{A}}) \cup \mathcal{J}_k(\mathcal{S}'_{\mathcal{A},\underline{m}}), \qquad \text{and} \qquad \mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_k(\mathcal{S}_{\mathcal{A}}) \cup \mathcal{I}_k(\mathcal{S}'_{\mathcal{A},\underline{m}}).$$

The next lemma establishes the soundness of the reduction of message reception to language synchronizability. The reduction to synchronizability will be later treated.

**Lemma 3.11.** *$\mathcal{S}''_{\mathcal{A},\underline{m}}$ is not language synchronizable iff there is a trace $\tau \in \mathsf{Traces}(\mathcal{A})$ such that $?\underline{m}$ occurs in $\tau$.*

*Proof.* ($\Rightarrow$) Assume that $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is not language synchronizable and let us show that there is a trace $\tau \in \mathsf{Traces}(\mathcal{A})$ such that $?\underline{m}$ occurs in $\tau$.

Let us first observe that if $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is not synchronizable, then $\mathcal{J}(\mathcal{S}''_{\mathcal{A},\underline{m}}) \setminus \mathcal{J}_0(\mathcal{S}''_{\mathcal{A},\underline{m}}) \neq \emptyset$. Since by Lemma 3.10 $\mathcal{S}'_{\mathcal{A},\underline{m}}$ is synchronizable, $\mathcal{J}(\mathcal{S}'_{\mathcal{A},\underline{m}}) \setminus \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}}) = \emptyset$. So $\mathcal{J}(\mathcal{S}_{\mathcal{A}}) \setminus \mathcal{J}_0(\mathcal{S}''_{\mathcal{A},\underline{m}}) \neq \emptyset$, and in particular

$$\mathcal{J}(\mathcal{S}_{\mathcal{A}}) \setminus \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}}) \neq \emptyset.$$

Let $\tau_1 \in \mathsf{Traces}(\mathcal{S}_{\mathcal{A}})$ be such that $\mathsf{send}(\tau_1) \notin \mathcal{J}_0(\mathcal{S}_{\mathcal{A},\underline{m}})$. Then $!\underline{m}^{2\to 3}$ occurs in $\tau_1$. Indeed, if it was not the case, then by Lemma 3.9, we would have $\mathsf{send}(\tau_1) \in \mathcal{J}_0(\mathcal{S}_{\mathcal{A},\underline{m}})$. Now, since $!\underline{m}^{2\to 3}$ occurs in $\tau_1$, by Lemma 3.5, there is a trace $\tau \in \mathsf{Traces}(\mathcal{A})$ such that $h(\tau) = \tau_1$, and by definition of $h$, $?\underline{m}$ occurs in $\tau$.

($\Leftarrow$) Assume that there is a trace $\tau \in \mathsf{Traces}(\mathcal{A})$ such that $?\underline{m}$ occurs in $\tau$, and let us show that $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is not language synchronizable. By Lemma 3.5, $h(\tau) \in \mathsf{Traces}(\mathcal{S}_{\mathcal{A}})$. In order to show that $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is not synchronizable (resp. not language synchronizable), let us show that $\mathsf{send}(h(\tau)) \in \mathcal{J}(\mathcal{S}''_{\mathcal{A},\underline{m}})$ and $\mathsf{send}(h(\tau)) \notin \mathcal{J}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$.

- $\mathsf{send}(h(\tau)) \in \mathcal{J}(\mathcal{S}''_{\mathcal{A},\underline{m}})$. Indeed, $\mathsf{send}(h(\tau)) \in \mathcal{J}(\mathcal{S}_{A})$ because $h(\tau) \in \mathsf{Traces}(\mathcal{S}_{\mathcal{A}})$, and $\mathcal{J}(\mathcal{S}_{\mathcal{A}}) \subseteq \mathcal{J}(\mathcal{S}''_{\mathcal{A},\underline{m}})$.
- Let us show that $\mathsf{send}(h(\tau)) \notin \mathcal{J}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$. Since $\mathcal{J}_0(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{J}_0(\mathcal{S}_{\mathcal{A}}) \cup \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$, we need to show that $\mathsf{send}(h(\tau)) \notin \mathcal{J}_0(\mathcal{S}_{\mathcal{A}})$ and $\mathsf{send}(h(\tau)) \notin \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$.

– $\mathsf{send}(h(\tau)) \notin \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. Indeed, since $?\underline{m}$ occurs in $\tau$, $!\underline{m}^{2\to 3}$ occurs in $h(\tau)$, but by definition, $\mathcal{P}'_2$, contains no $!\underline{m}^{2\to 3}$ transition, so a trace that contains $!\underline{m}^{2\to 3}$ can't be trace of $\mathcal{S}''_{\mathcal{A},\underline{m}}$, therefore $\mathsf{send}(h(\tau)) \notin \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$.
– $\mathsf{send}(h(\tau)) \notin \mathcal{J}_0(\mathcal{S}_{\mathcal{A}})$. Let us assume by absurd that there is a synchronous trace $\tau' \in \mathsf{Traces}_0(\mathcal{S}_{\mathcal{A}})$ such that $\mathsf{send}(\tau') = \mathsf{send}(h(\tau))$. By Lemma 3.5, there is $\tau_0 \in \mathsf{Traces}_0(\mathcal{A})$ such that either $\tau'$ is a prefix of $h(\tau_0)$, or $\tau'$ is a prefix of $h(\tau_0) \cdot !?a^{1\to 3} \cdot !?a^{3\to 2}$ for some $a \in \Sigma$. Since $\tau'$ is a synchronous trace, and by definition of $h$, $\tau_0$ must only contain receptions, and since $\tau_0$ corresponds to a trace in $\mathcal{A}$, $\tau_0$ is the empty trace. So $\tau'$ is a prefix of $!?a^{1\to 3} \cdot !?a^{3\to 2}$ for some $a \in \Sigma$, and $!a^{2\to 3}$ does not occur in $\tau'$. This contradicts $\mathsf{send}(\tau') = \mathsf{send}(h(\tau))$ and the fact that $!a^{2\to 3}$ does occur in $h(\tau)$.

□

Let us now establish the soundness of the reduction of message reception to synchronizability, instead of language synchronizability. It is slightly more involved due to the possible existence of stable traces of $\mathcal{S}_{\mathcal{A}}$ that are not “catched” by a stable synchronous trace of $\mathcal{SA}$. This is actually only a minor problem, and we will actually fix it with the following extra hypothesis on the FIFO automaton $\mathcal{A}$.

**Definition 3.12.** A FIFO automaton $\mathcal{A}$ is *good for reduction* if the only stable trace of $\mathcal{A}$ is the empty trace.

Note that the FIFO automaton $\mathcal{A}$ that we defined in the proof of the undecidability of the message reception problem is good for reduction: indeed, after the first row of the tiling has been queued, the automaton always queues a new tile right after it has dequeued a tile, or queues the marker of the end of the row (\$) right after it dequeues it. So the buffer always contains either at least one tile or the \$ marker, except in the initial configuration.

**Lemma 3.13.** *Assume that $\mathcal{A}$ is good for reduction. Then $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is not synchronizable iff there is a trace $\tau \in \mathsf{Traces}(\mathcal{A})$ such that $?\underline{m}$ occurs in $\tau$.*

*Proof.* Let us show that, under the hypothesis that $\mathcal{A}$ is good for reduction, $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is synchronizable if and only if $\mathcal{S}'_{\mathcal{A},\underline{m}}$ is language synchronizable, which, by Lemma 3.11 will entail what we need to prove.

($\Rightarrow$) Let us assume that $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is synchronizable and let us show that $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is language synchronizable. This implication actually holds for any system $\mathcal{S}$. Indeed, if $\mathcal{S}$ is synchronizable, then $\mathcal{I}(\mathcal{S}) \setminus \mathcal{I}_0(\mathcal{S}) = \emptyset$. Since $\mathcal{J}(\mathcal{S}) \subseteq \mathcal{I}(\mathcal{S})$, we have in particular

$$\mathcal{J}(\mathcal{S}) \setminus \mathcal{I}_0(\mathcal{S}) = \emptyset.$$

By definition, $\mathcal{I}_0(\mathcal{S}) = \mathcal{J}_0(\mathcal{S}) \cup \mathsf{Stab}$, where $\mathsf{Stab}$ is a set of pairs $(\mathsf{send}(\tau), \gamma)$; such pairs do not belong to $\mathcal{J}(\mathcal{S})$, so

$$\mathcal{J}(\mathcal{S}) \setminus \mathcal{I}_0(\mathcal{S}) = \mathcal{J}(\mathcal{S}) \setminus \mathcal{J}_0(\mathcal{S}).$$

As a consequence, $\mathcal{J}(\mathcal{S}) \setminus \mathcal{J}_0(\mathcal{S}) = \emptyset$, and since $\mathcal{J}_0(\mathcal{S}) \subseteq \mathcal{J}(\mathcal{S})$, we finally get

$$\mathcal{J}(\mathcal{S}) = \mathcal{I}_0(\mathcal{S}).$$

($\Leftarrow$) Let us assume that $\mathcal{J}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{J}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$, and let us show that $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$. The inclusion $\mathcal{I}_0(\mathcal{S}''_{\mathcal{A},\underline{m}}) \subseteq \mathcal{I}(\mathcal{S}''_{\mathcal{A},\underline{m}})$ holds for any system. Let us therefore show that $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) \subseteq \mathcal{I}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$. Since $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_k(\mathcal{S}_{\mathcal{A}}) \cup \mathcal{I}_k(\mathcal{S}'_{\mathcal{A},\underline{m}})$ for all $k \geq 0$, we have to show that $\mathcal{I}(\mathcal{S}_{\mathcal{A}}) \subseteq \mathcal{I}_0(\mathcal{S}_{\mathcal{A}}) \cup \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$ and $\mathcal{I}(\mathcal{S}'_{\mathcal{A},\underline{m}}) \subseteq \mathcal{I}_0(\mathcal{S}_{\mathcal{A}}) \cup \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$.

• $\mathcal{I}(\mathcal{S}'_{\mathcal{A},\underline{m}}) \subseteq \mathcal{I}_0(\mathcal{S}_{\mathcal{A}}) \cup \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$ follows from Lemma 3.10.

- Let us show that $\mathcal{I}(\mathcal{S}_\mathcal{A}) \subseteq \mathcal{I}_0(\mathcal{S}_\mathcal{A}) \cup \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. Since $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is language synchronizable by hypothesis, we have in particular that $\mathcal{J}(\mathcal{A}) \subseteq \mathcal{J}_0(\mathcal{S}_\mathcal{A}) \cup \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. So we only need to prove that for all stable trace $\tau$ of $\mathcal{S}_\mathcal{A}$, there is a stable synchronous trace $\tau'$ of $\mathcal{S}''_{\mathcal{A},\underline{m}}$ leading to the same configuration and such that $\mathsf{send}(\tau) = \mathsf{send}(\tau')$. We will actually show that the only stable traces of $\mathcal{S}_\mathcal{A}$ are synchronous, and therefore we can even take $\tau' = \tau$.
  Let $\tau \in \mathsf{Traces}(\mathcal{S}_\mathcal{A})$ a stable trace be fixed, and let us show that $\tau$ is synchronous. By Lemma 3.5, there is a trace $\tau_0 \in \mathsf{Traces}(\mathcal{A})$ and a message $a \in \Sigma$ such that either $\tau = h(\tau_0)$, or $\tau = h(\tau_0) \cdot !?a^{1\to3}$, or $\tau = h(\tau_0) \cdot !?a^{1\to3} \cdot !?a^{3\to2}$. By definition of $h$, if $\tau$ is stable, then $\tau_0$ is stable too. Since $\mathcal{A}$ is good for reduction, $\tau_0$ must be the empty trace. So either $\tau$ is the empty trace, or $\tau = !?a^{1\to3}$, or $\tau = !?a^{1\to3} \cdot !?a^{3\to2}$. In all cases, $\tau$ is a synchronous trace, which ends the proof.

□

**Theorem 3.14.** *Synchronizability (resp. language synchronizability) is undecidable.*

*Proof.* Let a FIFO automaton $\mathcal{A}$ that is good for reduction over the message alphabet $\Sigma$ and let a message $\underline{m} \in \Sigma$ be fixed. By Lemma 3.11, $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is (language) synchronizable iff $\mathcal{J}_k(\mathcal{S}_\mathcal{A}) \setminus \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}}) = \emptyset$. By Lemma 3.13, $\mathcal{J}_k(\mathcal{S}_\mathcal{A}) \setminus \mathcal{J}_0(\mathcal{S}'_{\mathcal{A},\underline{m}}) = \emptyset$ iff there is no trace $\tau$ such that $\tau \cdot ?\underline{m} \in \mathsf{Traces}(\mathcal{A})$. By Lemma 3.3, this is an undecidable problem. □

## 4. The case of oriented rings

In the previous section we established the undecidability of synchronizability for systems with (at least) three peers. In this section, we show that this result is tight, in the sense that synchronizability is decidable if $G_M$ is an oriented ring, in particular if the system involves two peers only. This relies on the fact that 1-synchronizability implies synchronizability for such systems. In order to show this result, we first establish some confluence properties on traces for arbitrary topologies. With the help of this confluence properties, we can state a trace normalization property that is similar to the one that was used in [BBO12b] and for half-duplex systems [CF05]. This trace normalization property implies that 1-synchronizable systems on oriented rings have no unspecified receptions[2], and their reachability set is channel-recognizable. Finally, this trace normalization property leads to a proof that 1-synchronizability implies synchronizability when $G_M$ is an oriented ring.

### 4.1. Confluence properties.

The following confluence property holds for any synchronizable system (see also Fig 3).

**Lemma 4.1** (Weak commutativity). *Let $\mathcal{S}$ be a 1-synchronizable system. Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $a, b \in M$ be such that*
(1) $\tau \cdot !a \in \mathsf{Traces}_1(\mathcal{S})$,
(2) $\tau \cdot !b \in \mathsf{Traces}_1(\mathcal{S})$, *and*
(3) $\mathsf{src}(a) \neq \mathsf{src}(b)$.
*If $v_1, v_2$ are any two of the six different shuffles of $!a \cdot ?a$ with $!b \cdot ?b$, then $\tau \cdot v_1 \in \mathsf{Traces}(\mathcal{S})$, $\tau \cdot v_2 \in \mathsf{Traces}(\mathcal{S})$ and $\tau \cdot v_1 \overset{\mathcal{S}}{\sim} \tau \cdot v_2$.*

[2]An unspecified reception occurs when a process $P$ is in a receiving state, some messages awaits for $P$ receiving them, but they are not the ones that $P$ may dequeue. See [CF05] for formal definitions.

$$\begin{tikzcd}
& & {} \arrow[dl, "!a"'] \arrow[dr, "!b"] & & \\
& {} \arrow[dl, dashed, "?a"'] \arrow[dr, dashed, "!b"] \arrow[rr, phantom, "\overset{\text{causal}}{\sim}"] & & {} \arrow[dl, dashed, "!a"'] \arrow[dr, dashed, "?b"] & \\
{} \arrow[dr, dashed, "!b"'] & & {} \arrow[dl, dashed, "?a"'] \arrow[dr, dashed, "?b"] & & {} \arrow[dl, dashed, "!a"] \\
& {} \arrow[dr, dashed, "?b"'] & & {} \arrow[dl, dashed, "?a"] & \\
& & {} & &
\end{tikzcd}$$

Figure 3: Diagrammatic representation of Lemma 4.1

**Remark 4.2.** This lemma should not be misunderstood as a consequence of causal equivalence. Observe indeed that the square on top of the diagram is the only square that commutes for causal equivalence. The three other squares only commute with respect to $\overset{\mathcal{S}}{\sim}$, and they commute for $\overset{\text{causal}}{\sim}$ only if some extra assumptions on $a$ and $b$ are made. For instance, the left square does commute for $\overset{\text{causal}}{\sim}$ if and only if $\mathsf{dst}(a) \neq \mathsf{src}(b)$.

Before we prove Lemma 4.1, let us first prove the following.

**Lemma 4.3.** *Let $a, b$ be two messages such that $\mathsf{src}(a) \neq \mathsf{src}(b)$. Then for all peers $i$, for all shuffle $v$ of $!a \cdot ?a$ with $!b \cdot ?b$, either $\mathsf{onPeer}_i(v) = \mathsf{onPeer}_i(!?a \cdot !?b)$ or $\mathsf{onPeer}_i(v) = \mathsf{onPeer}_i(!?b \cdot !?a)$.*

*Proof.* Let $i$ and $v$ be fixed. Since $\mathsf{src}(a) \neq \mathsf{src}(b)$, it is not the case that both $!a$ and $!b$ occur in $\mathsf{onPeer}_i(v)$. By symmetry, let us assume that $!b$ does not occur in $\mathsf{onPeer}_i(v)$. We consider two cases:

(1) Let us assume that $?a$ does not occur in $\mathsf{onPeer}_i(v)$. Then $\mathsf{onPeer}_i(v) \in \{!a, ?b, !a \cdot ?b\}$, and in all cases, $\mathsf{onPeer}_i(v) = \mathsf{onPeer}_i(!?a \cdot !?b)$.
(2) Let us assume that $?a$ occurs in $\mathsf{onPeer}_i(v)$. Then $!a$ does not occur in $\mathsf{onPeer}_i(v)$, therefore $\mathsf{onPeer}_i(v)$ contains only receptions, and $\mathsf{onPeer}_i(v) \in \{?a, ?a \cdot ?b, ?b, ?b \cdot ?a\}$. In every case, either $\mathsf{onPeer}_i(v) = \mathsf{onPeer}_i(!?a \cdot !?b)$ or $\mathsf{onPeer}_i(v) = \mathsf{onPeer}_i(!?b \cdot !?a)$. □

Let us prove now Lemma 4.1.

*Proof.* Observe first that, since $\mathsf{src}(a) \neq \mathsf{src}(b)$, $\tau \cdot !a \cdot !b \in \mathsf{Traces}_1(\mathcal{S})$ and $\tau \cdot !b \cdot !a \in \mathsf{Traces}_1(\mathcal{S})$, and since $\mathcal{S}$ is 1-synchronizable, $\tau \cdot !?a \cdot !?b \in \mathsf{Traces}_0(\mathcal{S})$ and $\tau \cdot !?b \cdot !?a \in \mathsf{Traces}_0(\mathcal{S})$. By Lemma 4.3, it follows that for all shuffle $v$ of $!a \cdot ?a$ with $!b \cdot ?b$, $\tau \cdot v \in \mathsf{Traces}_1(\mathcal{S})$. It remains to show that

$$\text{for all two shuffles } v,\, v' \text{ of } !a \cdot ?a \text{ with } !b \cdot ?b, \quad \tau \cdot v \overset{\mathcal{S}}{\sim} \tau \cdot v'. \qquad (P)$$

Let $\tau_{ab} = !a \cdot !b \cdot ?a \cdot ?b$ and $\tau_{ba} = !b \cdot !a \cdot ?a \cdot ?b$, and let $v$ be a shuffle of $!a \cdot ?a$ with $!b \cdot ?b$. Since $\mathcal{S}$ is 1-synchronizable, the stable configuration that $\tau \cdot v$ leads to only depends on the order in which the send actions $!a$ and $!b$ are executed in $v$, i.e. either $\tau \cdot v \overset{\mathcal{S}}{\sim} \tau_{ab}$ or $\tau \cdot v \overset{\mathcal{S}}{\sim} \tau_{ba}$. Moreover, $\tau_{ab} \overset{\text{causal}}{\sim} \tau_{ba}$, hence $(P)$. □

Our aim now is to generalize Lemma 4.1 to arbitrary sequences of send actions (see Lemma 4.9 below and the corresponding diagram). For this, we need to reason by induction on the length of the sequence of send actions. The first step is to establish the following property: a synchronous trace followed by a sequence of send actions can be completed to form a fully synchronous trace.

$$\begin{tikzcd} & {} \arrow[dl, "!?a"'] \arrow[dr, "!?b_1\cdots!?b_n"] & \\ {} \arrow[dr, dashed, "!?b_1\cdots!?b_n"'] & & {} \arrow[dl, dashed, "!?a"] \\ & {} & \end{tikzcd}$$

Figure 4: Diagram of Lemma 4.5

**Lemma 4.4.** *Let $\mathcal{S}$ be a 1-synchronizable system. Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $a_1, \cdots, a_n \in M$ be such that*

(1) $\tau \cdot !a_1 \cdots !a_n \in \mathsf{Traces}_n(\mathcal{S})$
(2) $\mathsf{src}(a_i) = \mathsf{src}(a_j)$ *for all* $i, j \in \{1, \dots, n\}$.

*Then* $\tau \cdot !?a_1 \cdots !?a_n \in \mathsf{Traces}_0(\mathcal{S})$.

*Proof.* By induction on $n$. Let $a_1, \dots, a_{n+1}$ be fixed, and let $\tau_n = \tau \cdot !?a_1 \cdots !?a_n$. By induction hypothesis, $\tau_n \in \mathsf{Traces}_0(\mathcal{S})$. Let $\tau'_{n+1} = \tau_n \cdot !a_{n+1}$. Then

- $\mathsf{onPeer}_i(\tau'_{n+1}) = \mathsf{onPeer}_i(\tau_n)$ for all $i \neq \mathsf{src}(a_{n+1})$, and $\tau_n \in \mathsf{Traces}(\mathcal{S})$
- for $i = \mathsf{src}(a_{n+1})$, $\mathsf{onPeer}_i(\tau'_{n+1}) = \mathsf{onPeer}_i(\tau \cdot !a_1 \cdots !a_{n+1})$ and $\tau \cdot !a_1 \cdots !a_n \in \mathsf{Traces}(\mathcal{S})$
- $\tau'_{n+1}$ is 1-bounded FIFO

therefore $\tau'_{n+1} \in \mathsf{Traces}_1(\mathcal{S})$.

By 1-synchronizability, it follows that $\tau'_{n+1} \cdot ?a_{n+1} \in \mathsf{Traces}_0(\mathcal{S})$. □

The second step is a confluence property that allows to commute a synchronization on one message and a sequence of synchronizations on other messages with different senders (see also the diagram on Figure 4).

**Lemma 4.5.** *Let $\mathcal{S}$ be a 1-synchronizable system. Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $a, b_1, \dots, b_n \in M$ be such that*

(1) $\tau \cdot !?a \in \mathsf{Traces}_0(\mathcal{S})$
(2) $\tau \cdot !?b_1 \cdots !?b_n \in \mathsf{Traces}_0(\mathcal{S})$
(3) $\mathsf{src}(a) \neq \mathsf{src}(b_i)$ *for all* $i \in \{1, \dots, n\}$.

*Then the following holds*

- $\tau \cdot !?a \cdot !?b_1 \cdots !?b_n \in \mathsf{Traces}_0(\mathcal{S})$,
- $\tau \cdot !?b_1 \cdots !?b_n \cdot !?a \in \mathsf{Traces}_0(\mathcal{S})$, *and*
- $\tau \cdot !?a \cdot !?b_1 \cdots !?b_n \overset{\mathcal{S}}{\sim} \tau \cdot !?b_1 \cdots !?b_n \cdot !?a$.

*Proof.* By induction on $n$. Let $a, b_1 \dots, b_{n+1}$ be fixed, let $\tau_n = \tau \cdot !?b_1 \cdots !?b_n$. By induction hypothesis, $\tau_n \cdot !?a \in \mathsf{Traces}_0(\mathcal{S})$, and by hypothesis $\tau_n \cdot !?b_{n+1} \in \mathsf{Traces}_0(\mathcal{S})$. By Lemma 4.1, $\tau_n \cdot !?a \cdot !?b_{n+1} \in \mathsf{Traces}_0(\mathcal{S})$, $\tau_n \cdot !?b_{n+1} \cdot !?a \in \mathsf{Traces}_0(\mathcal{S})$, and

$$\tau_n \cdot !?a \cdot !?b_{n+1} \overset{\mathcal{S}}{\sim} \tau_n \cdot !?b_{n+1} \cdot !?a.$$

On the other hand, by induction hypothesis, $\tau_n \cdot !?a \overset{\mathcal{S}}{\sim} \tau \cdot !?a \cdot !?b_1 \cdots !?b_n$, and by right congruence of $\overset{\mathcal{S}}{\sim}$

$$\tau_n \cdot !?a \cdot !?b_{n+1} \overset{\mathcal{S}}{\sim} \tau \cdot !?a \cdot !?b_1 \cdots !?b_{n+1}$$

$$\begin{tikzcd}
& {} \arrow[dl, "!?a_1\cdots!?a_n"'] \arrow[dr, "!?b_1\cdots!?b_m"] \arrow[dd, dashed, "!?c_1\cdots!?c_{n+m}"] & \\
{} \arrow[dr, dashed, "!?b_1\cdots!?b_m"'] & & {} \arrow[dl, dashed, "!?a_1\cdots!?a_n"] \\
& {} &
\end{tikzcd}$$

Figure 5: Diagram of Lemma 4.6

By transitivity of $\overset{\mathcal{S}}{\sim}$, we can relate the two right members of the above identities, *i.e.*

$$\tau_n \cdot !?b_{n+1} \cdot !?a \overset{\mathcal{S}}{\sim} \tau \cdot !?a \cdot !?b_1 \cdots !?b_{n+1}$$

which shows the claim. □

The next lemma expresses the following, rather technical property: considering two sequences of synchronizations that are orthogonal (with different senders), it is possible to combine them in a single synchronous trace by shuffling the synchronization in any order. Diagramatically, it expresses that all paths that result from a shuffle inside the diamond lead to the same configuration (see Figure 5).

**Lemma 4.6.** *Let $\mathcal{S}$ be a 1-synchronizable system. Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $a_1, \ldots, a_n, b_1, \ldots, b_m \in M$ be such that*

(1) $\tau \cdot !?a_1 \cdots !?a_n \in \mathsf{Traces}_0(\mathcal{S})$
(2) $\tau \cdot !?b_1 \cdots !?b_m \in \mathsf{Traces}_0(\mathcal{S})$
(3) $\mathsf{src}(a_i) \neq \mathsf{src}(b_j)$ *for all* $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$

*Then for all shuffle $c_1 \ldots c_{m+n}$ of $a_1 \cdots a_n$ with $b_1 \cdots b_m$,*

- $\tau \cdot !?c_1 \cdots !?c_{n+m} \in \mathsf{Traces}_0(\mathcal{S})$, *and*
- $\tau \cdot !?a_1 \cdots !?a_n \cdot !?b_1 \cdots !?b_m \overset{\mathcal{S}}{\sim} \tau \cdot !?c_1 \cdots !?c_m$.

*Proof.* By induction on $n + m$. Let $a_1, \ldots, a_n, b_1 \ldots, b_m$ be fixed, and let $c_1 \cdots c_{n+m}$ be a shuffle of $a_1 \cdots a_n$ with $b_1 \cdots b_m$.

- Assume that $c_1 = a_1$. Let $\tau' = \tau \cdot !?a_1$. By Lemma 4.5, $\tau' \cdot !?b_1 \cdots !?b_m \in \mathsf{Traces}_0(\mathcal{S})$, and by hypothesis $\tau' \cdot !?a_2 \cdots !?a_n \in \mathsf{Traces}_0(\mathcal{S})$, so we can use the induction hypothesis with $(a'_1, \ldots, a'_{n-1}) = (a_2, \ldots, a_n)$. We get $\tau' \cdot !?c_2 \cdots !?c_n \in \mathsf{Traces}_0(\mathcal{S})$, and

$$\tau' \cdot !?c_2 \cdots !?c_n \overset{\mathcal{S}}{\sim} \tau' \cdot !?a_2 \cdots !?a_n \cdot !?b_1 \cdots !?b_m$$

  which shows the claim.
- Assume that $c_1 = b_1$. Then by the same arguments,

$$\tau \cdot !?c_1 \cdots !?c_n \overset{\mathcal{S}}{\sim} \tau \cdot !?b_1 \cdots !?b_m \cdot !?a_1 \cdots !?a_n$$

  Since this holds for all shuffle $c_1, \ldots, c_{n+m}$, this also holds for $c_1 = a_1, \ldots, c_n = a_n, c_{n+1} = b_1, \cdots, c_{n+m} = b_m$, which shows the claim. □

The next lemma generalizes Lemma 4.4: a sequence of send following a synchronous trace can be completed in a synchronous trace, regardless whether these sends are from the same sender or not.

**Lemma 4.7.** *Let $\mathcal{S}$ be a 1-synchronizable system. Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $m_1, \cdots, m_n \in M$ be such that $\tau \cdot !m_1 \cdots !m_n \in \mathsf{Traces}_n(\mathcal{S})$ Then $\tau \cdot !?m_1 \cdots !?m_n \in \mathsf{Traces}_0(\mathcal{S})$.*

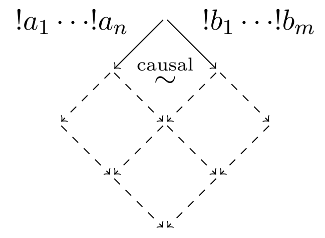


Figure 6: Diagrammatic representation of Lemma 4.9

*Proof.* By induction on $n$. Let $m_1, \ldots, m_n$ be fixed with $n \geq 1$. There are two subsequences $a_1, \ldots, a_r$ and $b_1, \ldots, b_m$ such that

- $\mathsf{src}(a_\ell) = \mathsf{src}(m_1)$ for all $\ell \in \{1, \ldots, r\}$,
- $\mathsf{src}(b_\ell) \neq \mathsf{src}(m_1)$ for all $\ell \in \{1, \ldots, m\}$,
- $m_1 \cdots m_n$ is a shuffle of $a_1 \cdots a_r$ with $b_1 \cdots b_m$

By hypothesis, $\tau \cdot !a_1 \cdots !a_r \in \mathsf{Traces}(\mathcal{S})$ and $\tau \cdot !b_1 \cdots !b_m \in \mathsf{Traces}(\mathcal{S})$. By Lemma 4.4, $\tau \cdot !?a_1 \cdots !?a_r \in \mathsf{Traces}_0(\mathcal{S})$, and by induction hypothesis $\tau \cdot !?b_1 \cdots !?b_m \in \mathsf{Traces}_0(\mathcal{S})$, and finally by Lemma 4.6 $\tau \cdot !?m_1 \cdots !?m_n \in \mathsf{Traces}_0(\mathcal{S})$. □

The next lemma, the last one before the main lemma we aim at, is a purely combinatorial property that does not have anything to do with synchronizability. It says that if a trace is a shuffle of two synchronous traces, and if it projected on a given machine, then this projection looks like the projection of a synchronous trace that is a shuffle of the original messages.

**Lemma 4.8.** *Let $a_1, \ldots, a_n, b_1, \cdots, b_m \in M$, and let $\tau$ be a shuffle of $!?a_1 \cdots !?a_n$ with $!?b_1 \cdots !?b_m$. Then for all $i \in \{1, \ldots, p\}$ there is a shuffle $c_1 \cdots c_{n+m}$ of $a_1 \cdots a_n$ with $b_1 \cdots b_m$ such that* $\mathsf{onPeer}_i(\tau) = \mathsf{onPeer}_i(!?c_1 \cdots !?c_{n+m})$.

*Proof.* Let us fix $\tau = \lambda_1 \cdots \lambda_{2(n+m)}$ a shuffle $!?a_1 \cdots !?a_n$ with $!?b_1 \cdots !?b_m$. Consider the trace $\tau' = \tau'_1 \cdots \tau'_{2(n+m)}$ where, for all $k \in \{1, \ldots, 2(n+m)\}$, $\tau'_k$ is defined as follows:

- if there are $m, j$ such that $\lambda_k = !m^{i \to j}$ or $\lambda_k = ?m^{j \to i}$, then $\tau'_k = !?m$
- otherwise, if $\lambda_k = !m_k$, then $\tau_i = !?m_k$, else $\lambda_i = ?m_k$ and $\tau'_k = \epsilon$

Then by construction $\tau'$ is of the form $!?c_1 \cdots !?c_{n+m}$ with $c_1, \ldots, c_{n+m}$ a shuffle of $a_1, \ldots, a_n$ with $b_1, \ldots, b_m$. Moreover, $\mathsf{onPeer}_i(\tau') = \mathsf{onPeer}_i(\tau)$, which ends the proof. □

We are now ready to generalise Lemma 4.1 to an arbitrary long sequence of send actions, which is the main property we wanted to establish with this long serie of lemmas.

**Lemma 4.9** (Strong commutativity)**.** *Let $\mathcal{S}$ be a 1-synchronizable system. Let $a_1, \ldots, a_n, b_1, \ldots b_m \in M$ and $\tau \in \mathsf{Traces}_0(\mathcal{S})$ be such that*

(1) $\tau \cdot !a_1 \cdots !a_n \in \mathsf{Traces}_n(\mathcal{S})$,
(2) $\tau \cdot !b_1 \cdots !b_m \in \mathsf{Traces}_m(\mathcal{S})$, *and*
(3) $\mathsf{src}(a_i) \neq \mathsf{src}(b_j)$ *for all* $i \in \{1, \ldots, n\}$ *and* $j \in \{1, \ldots, m\}$.

*Then for any two different shuffles $\upsilon_1, \upsilon_2$ of $!?a_1 \cdot !?a_2 \cdots !?a_n$ with $!?b_1 \cdot !?b_2 \cdots !?b_m$, it holds that $\tau \cdot \upsilon_1 \in \mathsf{Traces}(\mathcal{S})$ , $\tau \cdot \upsilon_2 \in \mathsf{Traces}(\mathcal{S})$ and $\tau \cdot \upsilon_1 \overset{\mathcal{S}}{\sim} \tau \cdot \upsilon_2$.*

*Proof.* Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $a_1, \ldots, a_n, b_1, \ldots, b_m$, be fixed. Let $\upsilon$ be a shuffle of $!?a_1 \cdots !?a_n$ with $!?b_1 \cdots !?b_m$. We want to show that $\tau \cdot \upsilon \in \mathsf{Traces}(\mathcal{S})$. Clearly, $\tau \cdot \upsilon \in$

$\mathsf{Traces}(\mathcal{S})$ is a FIFO trace. Therefore, it is enough to find for all $i \in \{1, \ldots, p\}$ a trace $\tau_i$ such that

$$\tau_i \in \mathsf{Traces}(\mathcal{S}) \qquad \text{and} \qquad \mathsf{onPeer}_i(\tau \cdot \upsilon) = \mathsf{onPeer}_i(\tau_i). \tag{4.1}$$

Let $i \in \{1, \ldots, p\}$ be fixed, and let us construct $\tau_i$ that validates (4.1). By hypothesis

$$\tau \cdot !a_1 \cdots !a_n \in \mathsf{Traces}(\mathcal{S}) \text{ and } \tau \cdot !b_1 \cdots !b_n \in \mathsf{Traces}(\mathcal{S})$$

therefore, by Lemma 4.7,

$$\tau \cdot !?a_1 \cdots !?a_n \in \mathsf{Traces}_0(\mathcal{S}) \text{ and } \tau \cdot !?b_1 \cdots !?b_n \in \mathsf{Traces}_0(\mathcal{S}). \tag{4.2}$$

On the other hand, by Lemma 4.8, there is a shuffle $c_1 \ldots c_{n+m}$ of $a_1 \cdots a_n$ with $b_1 \cdots b_m$ such that

$$\mathsf{onPeer}_i(\upsilon) = \mathsf{onPeer}_i(!?c_1 \cdots !?c_{n+m}) \tag{4.3}$$

Let $\tau_i = \tau \cdot !?c_1 \cdots !?c_{n+m}$. By Lemma 4.6 and (4.2), $\tau_i \in \mathsf{Traces}_0(\mathcal{S})$, and by (4.3), the second part of (4.1) holds. □

## 4.2. Trace normalization.

In this section and the next one, it will be necessary to assume that the communication topology is an oriented ring.

**Definition 4.10** (Normalized trace)**.** A $M$-trace $\tau$ is *normalized* if there is a synchronous $M$-trace $\tau_0$, $n \geq 0$, and messages $a_1, \ldots, a_n$ such that $\tau = \tau_0 \cdot !a_1 \cdots !a_n$.

**Lemma 4.11** (Trace Normalization)**.** *Assume $M$ is such that the communication topology $G_M$ is an oriented ring. Let $\mathcal{S} = \langle \mathcal{P}_1, \ldots, \mathcal{P}_p \rangle$ be a 1-synchronizable $M$-system. For all $\tau \in \mathsf{Traces}(\mathcal{S})$, there is a normalized trace $\mathsf{norm}(\tau) \in \mathsf{Traces}(\mathcal{S})$ such that $\tau \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau)$.*

$$\begin{tikzcd} {} \arrow[d, dashed, "\tau_0"'] \arrow[dr, "\tau"] & \\ {} \arrow[r, dashed, "!m_1 \cdots !m_k"'] & {} \end{tikzcd}$$

*Proof.* By induction on $\tau$. Let $\tau = \tau' \cdot \lambda$, be fixed. Let us assume by induction hypothesis that there is a normalized trace $\mathsf{norm}(\tau') \in \mathsf{Traces}(\mathcal{S})$ such that $\tau' \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau')$. Let us reason by case analysis on the last action $\lambda$ of $\tau$. The easy case is when $\lambda$ is a send action: then, $\mathsf{norm}(\tau') \cdot \lambda$ is a normalized trace, and $\mathsf{norm}(\tau') \cdot \lambda \overset{\mathcal{S}}{\sim} \tau' \cdot \lambda$ by right congruence of $\overset{\mathcal{S}}{\sim}$. The difficult case is when $\lambda$ is $?a$ for some $a \in M$. Let $i = \mathsf{src}(a)$, $j = \mathsf{dst}(a)$, *i.e.* $i + 1 = j \bmod p$. By the definitions of a normal trace and $\overset{\text{causal}}{\sim}$, there are $\tau'_0 \in \mathsf{Traces}_0(\mathcal{S})$, $a_1, \ldots, a_n, b_1, \ldots, b_m \in M$ such that

$$\mathsf{norm}(\tau') \overset{\text{causal}}{\sim} \tau'_0 \cdot !a_1 \cdots !a_n \cdot !b_1 \cdots !b_m$$

with $\mathsf{src}(a_k) = i$ for all $k \in \{1, \ldots, n\}$ and $\mathsf{src}(b_k) \neq i$ for all $k \in \{1, \ldots, m\}$. Since $G_M$ is an oriented ring, $\mathsf{dst}(a_1) = j$, therefore $a_1 = a$ (because by hypothesis $j$ may receive $a$ in the configuration that $\mathsf{norm}(\tau')$ leads to). Let $\mathsf{norm}(\tau) = \tau'_0 \cdot !a \cdot ?a \cdot !b_1 \cdots !b_m \cdot !a_2 \cdots !a_n$ and let us show that $\mathsf{norm}(\tau) \in \mathsf{Traces}(\mathcal{S})$ and $\tau \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau)$.

Since $\mathsf{norm}(\tau') \in \mathsf{Traces}(\mathcal{S})$, we have in particular that $\tau'_0 \cdot !a \in \mathsf{Traces}_1(\mathcal{S})$ and $\tau'_0 \cdot !b_1 \cdots !b_n \in \mathsf{Traces}(\mathcal{S})$. Consider the two traces

$$\begin{aligned} \upsilon_1 = &\ \tau'_0 \cdot !a \cdot ?a \cdot !b_1 \cdots !b_n \cdot ?b_1 \cdots ?b_n \\ \upsilon_2 = &\ \tau'_0 \cdot !a \cdot !b_1 \cdots !b_n \cdot ?a \cdot ?b_1 \cdots ?b_n. \end{aligned}$$

By Lemma 4.9, $\upsilon_1, \upsilon_2 \in \mathsf{Traces}(\mathcal{S})$ and both lead to the same configuration, and in particular to the same control state $q$ for peer $j$. The actions $?b_1$, $?b_2, \ldots ?b_n$ are not executed by peer $j$ (because $\mathsf{src}(m) \neq i$ implies $\mathsf{dst}(m) \neq j$ on an oriented ring), so the two traces

$$\begin{aligned} \upsilon_1' = &\quad \tau_0' \cdot !a \cdot ?a \cdot !b_1 \cdots !b_n \\ \upsilon_2' = &\quad \tau_0' \cdot !a \cdot !b_1 \cdots !b_n \cdot ?a \end{aligned}$$

lead to two configurations $\gamma_1', \gamma_2'$ with the same control state $q$ for peer $j$ as in the configuration reached after $\upsilon_1$ or $\upsilon_2$. On the other hand, for all $k \neq j$, $\mathsf{onPeer}_k(\upsilon_1') = \mathsf{onPeer}_k(\upsilon_2')$, therefore $\upsilon_1' \overset{\mathcal{S}}{\sim} \upsilon_2'$. Since $\tau_0' \cdot !a \cdot !a_2 \cdots !a_n \in \mathsf{Traces}_n(\mathcal{S})$, and $\mathsf{onPeer}_i(\tau_0' \cdot !a) = \mathsf{onPeer}_i(\upsilon_1') = \mathsf{onPeer}_i(\upsilon_2')$, the two traces

$$\begin{aligned} \upsilon_1'' = &\quad \tau_0' \cdot !a \cdot ?a \cdot !b_1 \cdots !b_n \cdot !a_2 \cdots !a_n \\ \upsilon_2'' = &\quad \tau_0' \cdot !a \cdot !b_1 \cdots !b_n \cdot ?a \cdot !a_2 \cdots !a_n \end{aligned}$$

belong to $\mathsf{Traces}(\mathcal{S})$ and $\upsilon_1'' \overset{\mathcal{S}}{\sim} \upsilon_2''$. Consider first $\upsilon_1''$: this is $\mathsf{norm}(\tau)$ as defined above, therefore $\mathsf{norm}(\tau) \in \mathsf{Traces}(\mathcal{S})$, and $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \upsilon_2''$. Consider now $\upsilon_2''$. By definition, $\upsilon_2'' \overset{\text{causal}}{\sim} \mathsf{norm}(\tau') \cdot ?a$. By hypothesis, $\mathsf{norm}(\tau') \overset{\mathcal{S}}{\sim} \tau'$, therefore $\mathsf{norm}(\tau') \cdot ?a \overset{\text{causal}}{\sim} \tau$. To sum up, $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \upsilon_2'' \overset{\text{causal}}{\sim} \mathsf{norm}(\tau') \cdot ?a \overset{\mathcal{S}}{\sim} \tau$, therefore $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \tau$. □

Note how we used the hypothesis that the communication topology is an oriented ring in the proof of Lemma 4.11. As a hint that this trace normalization does not hold if a peer can send to two different peers, consider the following example:

**Example 4.12.** Let $\mathcal{P}_1 = !a \cdot !b$, $\mathcal{P}_2 = ?a$, $\mathcal{P}_3 = ?b$, in other words, $\mathcal{P}_1$ sends a message to both $\mathcal{P}_2$ and $\mathcal{P}_3$, and they can receive this message. The system is obviously 1-synchronizable. But the only trace $\overset{\mathcal{S}}{\sim}$-equivalent to $\tau = !a \cdot !b \cdot ?b$ is $\tau$ itself, which is not a normalized trace.

## 4.3. **Reachability set.**

As a consequence of Lemma 4.11, 1-synchronizability implies several interesting properties on the reachability set.

**Definition 4.13** (Channel-recognizable reachability set [Pac87, CF05]). Let $\mathcal{S} = \langle \mathcal{P}_1, \ldots, \mathcal{P}_p \rangle$ with $\mathcal{P}_i = \langle Q_i, \Delta_i, q_{0,i} \rangle$. The (coding of the) *reachability set* of $\mathcal{S}$ is the language $\mathsf{Reach}(\mathcal{S})$ over the alphabet $(M \cup \bigcup_{i=1}^{p} Q_i)^*$ defined as $\{q_1 \cdots q_p \cdot w_1 \cdots w_p \mid \gamma_0 \xrightarrow{\tau} (q_1, \ldots, q_p, w_1, \ldots, w_p), \tau \in \mathsf{Traces}(\mathcal{S})\}$. $\mathsf{Reach}(\mathcal{S})$ is *channel recognizable* (or QDD representable [BG99]) if it is a recognizable (and rational) language.

**Theorem 4.14.** *Let $M$ be a message set such that $G_M$ is an oriented ring, and let $\mathcal{S}$ be a $M$-system that is 1-synchronizable. Then*
1. *the reachability set of $\mathcal{S}$ is channel recognizable,*
2. *for all $\tau \in \mathsf{Traces}(\mathcal{S})$, for all $\gamma_0 \xrightarrow{\tau} \gamma$, there is a stable configuration $\gamma'$, $n \geq 0$ and $m_1, \ldots m_n \in M$ such that $\gamma \xrightarrow{?m_1 \cdots ?m_n} \gamma'$.*

*In particular, $\mathcal{S}$ does not have unspecified receptions.*

*Proof.*
1. Let $S$ be the set of stable configurations $\gamma$ such that $\gamma_0 \xrightarrow{\tau} \gamma$ for some $\tau \in \mathsf{Traces}_0(\mathcal{S})$; $S$ is finite and effective. By Lemma 4.11, $\mathsf{Reach}(\mathcal{S}) = \bigcup \{\mathsf{Reach}^!(\gamma) \mid \gamma \in S\}$, where $\mathsf{Reach}^!(\gamma) = \{q_1 \cdots q_p \cdot w_1 \cdots w_p \mid \gamma \xrightarrow{!a_1 \cdots !a_n} (q_1, \ldots, q_p, w_1, \ldots, w_p), n \geq 0, a_1, \ldots a_n \in M\}$ is an effective rational language.

(2) Assume $\gamma_0 \xrightarrow{\tau} \gamma$. By Lemma 4.11, $\gamma_0 \xrightarrow{\tau_0 \cdot !m_1 \cdots !m_r} \gamma$ for some $\tau_0 \in \mathsf{Traces}_0(\mathcal{S})$. Then $\tau_0 \cdot !m_1 \cdots !m_r \overset{\text{causal}}{\sim} \tau_0 \cdot \tau_1$ where $\tau_1 := !a_1 \cdots !a_n \cdot b_1 \cdots b_m$ for some $a_1, \ldots, a_n, b_1, b_m$ such that $\mathsf{src}(a_i) \neq \mathsf{src}(b_j)$ for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. By Lemma 4.9, $\tau_0 \cdot \tau_1 \cdot \overline{\tau_1} \in \mathsf{Traces}(\mathcal{S})$ (where $\overline{\tau_1} = ?a_1 \cdots ?a_n \cdot ?b_1 \cdots ?b_m$), and therefore $\gamma_0 \xrightarrow{\tau_0 \cdot \tau_1} \gamma \xrightarrow{\overline{\tau_1}} \gamma'$ for some stable configuration $\gamma'$. □

## 4.4. 1-synchronizability implies synchronizability.

**Theorem 4.15.** *Let $M$ be a message set such that $G_M$ is an oriented ring. For any $M$-system $\mathcal{S}$, $\mathcal{S}$ is* 1- *synchronizable if and only if it is synchronizable.*

*Proof.* We only need to show that 1-synchronizability implies synchronizability. Let us assume that $\mathcal{S}$ is 1-synchronizable. Let $\mathsf{synch}(\tau)$ denote the unique synchronous $M$-trace such that $\mathsf{send}(\mathsf{synch}(\tau)) = \mathsf{send}(\tau)$. We prove by induction on $\tau$ the following property (which implies in particular that $\mathcal{S}$ is synchronizable):

for all $\tau \in \mathsf{Traces}(\mathcal{S})$, there are $m_1, \ldots, m_k \in M$ such that

$(C1)$ $\mathsf{synch}(\tau) \in \mathsf{Traces}_0(\mathcal{S})$
$(C2)$ $\tau \cdot ?m_1 \cdots ?m_k \in \mathsf{Traces}(\mathcal{S})$, and
$(C3)$ $\tau \cdot ?m_1 \cdots ?m_k \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau)$.

$$\begin{tikzcd}
\cdot \arrow[d, "\tau"'] \arrow[dr, dashed, "\mathsf{synch}(\tau)"] & \\
\cdot \arrow[r, dashed, "?m_1 \cdots ?m_k"'] & \cdot
\end{tikzcd}$$

Let $\tau = \tau' \cdot \lambda$ be fixed and assume that there are $m'_1, \ldots, m'_k \in M$ such that $\tau' \cdot ?m'_1 \cdots ?m'_k \in \mathsf{Traces}(\mathcal{S})$, $\mathsf{synch}(\tau') \in \mathsf{Traces}_0(\mathcal{S})$, and $\tau' \cdot ?m'_1 \cdots ?m'_k \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau')$. Let us show that $(C1)$, $(C2)$, and $(C3)$ hold for $\tau$. We reason by case analysis on the last action $\lambda$ of $\tau$.

- Assume $\lambda = ?a$. Then $\mathsf{synch}(\tau) = \mathsf{synch}(\tau') \in \mathsf{Traces}_0(\mathcal{S})$, which proves $(C1)$. Let $i = \mathsf{dst}(a)$. Since peer $i$ only receives on one channel, there are $m_1, \ldots, m_{k-1}$ such that
$$\tau' \cdot ?m'_1 \cdot ?m'_k \overset{\text{causal}}{\sim} \tau' \cdot ?a \cdot ?m_1 \cdot ?m_{k-1}.$$
Since $\tau' \cdot ?m'_1 \cdot ?m'_k \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau)$ by induction hypothesis, $(C2)$ and $(C3)$ hold.
- Assume $\lambda = !a$. By Lemma 4.11, there is $\mathsf{norm}(\tau') = \tau_0 \cdot !m''_1 \cdots !m''_k$ with $\tau_0 \in \mathsf{Traces}_0(\mathcal{S})$ such that $\tau' \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau')$. Since $\tau' \cdot ?m'_1 \cdots ?m'_k$ leads to a stable configuration, $m''_1, \ldots, m''_k$ is a permutation of $m'_1, \ldots, m'_k$ that do not swap messages of a same channel. Since $G_M$ is an oriented ring, $\mathsf{norm}(\tau') \overset{\text{causal}}{\sim} \tau_0 \cdot !m'_1 \cdots !m'_k$. Since $\tau' \cdot !a \in \mathsf{Traces}(\mathcal{S})$, it holds that $\tau_0 \cdot !m_1 \cdots !m_k \cdot !a \in \mathsf{Traces}(\mathcal{S})$, which implies by Lemma 4.9 that the two traces
$$\begin{aligned} \upsilon_1 = &\quad \tau_0 \cdot !m'_1 \cdots !m'_k \cdots ?m_1 \cdots ?m'_k \cdot !a \cdot ?a \\ \upsilon_2 = &\quad \tau_0 \cdot !m'_1 \cdots !m'_k \cdot !a \cdots ?m'_1 \cdots ?m'_k \cdot ?a \end{aligned}$$
belong to $\mathsf{Traces}(\mathcal{S})$ and verify $\upsilon_1 \overset{\mathcal{S}}{\sim} \upsilon_2$. Consider first $\upsilon_1$, and let $\upsilon'_1 = \tau_0 \cdot !m'_1 \cdots !m'_k \cdot ?m'_1 \cdots ?m'_k$. Since $\mathsf{norm}(\tau') \overset{\text{causal}}{\sim} \tau_0 \cdot !m'_1 \cdots !m'_k \overset{\mathcal{S}}{\sim} \tau'$ and $\tau' \cdot ?m_1 \cdots ?m_k \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau')$, it holds that $\upsilon'_1 \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau')$. Therefore, $\mathsf{synch}(\tau') \cdot !a \cdot ?a = \mathsf{synch}(\tau)$ belongs to $\mathsf{Traces}(\mathcal{S})$, which shows $(C1)$, and $\mathsf{synch}(\tau) \overset{\mathcal{S}}{\sim} \upsilon_1$. Consider now $\upsilon_2$, and let $\upsilon'_2 = \tau_0 \cdot !m'_1 \cdots !m'_k \cdot !a \overset{\text{causal}}{\sim} \mathsf{norm}(\tau') \cdot !a$. Then $\upsilon'_2 \overset{\mathcal{S}}{\sim} \tau' \cdot !a = \tau$, therefore $\tau \cdot ?m'_1 \cdots ?m'_k \cdot ?a \in \mathsf{Traces}(\mathcal{S})$, which shows $(C2)$, and $\tau \cdot ?m'_1 \cdots ?m'_k \cdot ?a \overset{\mathcal{S}}{\sim} \upsilon_2$. Since $\upsilon_2 \overset{\mathcal{S}}{\sim} \upsilon_1 \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau)$, this shows $(C3)$. □

**Theorem 4.16.** *Let $M$ be a message set such that $G_M$ is an oriented ring. The problem of deciding whether a given $M$-system is synchronizable is decidable.*

Since a system with two machines is a particular case of a system that is an oriented ring, we deduce from the above result that synchronizability is decidable in that particular case.

**Theorem 4.17.** *Synchronizability is decidable for systems of 2-CFSMs.*

## 5. Extensions and Related Works

### 5.1. Synchronizability for other communication models.

We considered the notions of synchronizability and language synchronizability introduced by Basu and Bultan [BB16] and we showed that both are not decidable for systems with peer-to-peer FIFO communications, called (1-1) type systems in [BB16]. In the same work, Basu and Bultan considered the question of the decidability of language synchronizability for other communication models. All the results we presented so far do not have any immediate consequences on their claims for these communication models. Therefore, we briefly discuss now what we can say about the decidability of language synchronizability for the other communication models that have been considered.

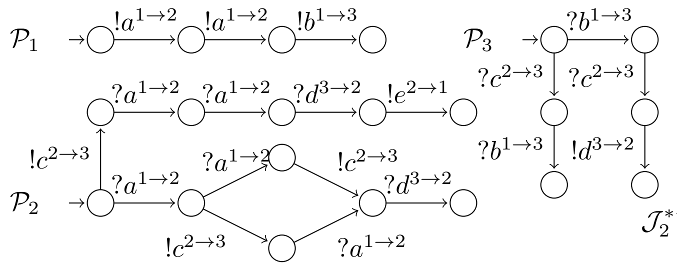
#### 5.1.1. *Bags.*

In [BB16], language synchronizability is studied for systems where peers communicate through bags instead of queues, thus allowing to reorder messages. Language synchronizability is decidable for bag communications: $\mathsf{Traces}^{bag}(\mathcal{S})$ is the trace language of a Petri net, $T_0(\mathcal{S}) = \{\tau \in \mathsf{Act}_M^* \mid \mathsf{send}(\tau) \in \mathcal{J}_0^{bag}(\mathcal{S})\}$ is an effective regular language, $\mathcal{S}$ is language synchronizable iff $\mathsf{Traces}^{bag}(\mathcal{S}) \subseteq T_0(\mathcal{S})$, and whether the trace language of a Petri is included in a given regular language reduces to the coverability problem. Lossy communications where not considered in [BB16], but the same kind of argument would also hold for lossy communications. However, our Example 2.2 is a counter-example for Lemma 3 in [BB16], *i.e.* the notion of language 1-synchronizability for bag communications defined in [BB16] does not imply language synchronizability. The question whether (language) synchronizability can be decided more efficiently than by reduction to the coverability problem for Petri nets is open.

#### 5.1.2. *Mailboxes.*

The other communication models considered in [BB16] keep the FIFO queue model, but differ in the way how queues are distributed among peers. The $*$-1 (mailbox) model assumes a queue per receiver. This model is the first model that was considered for (language) synchronizability [BB11, BBO12b]. Our Example 2.2 is not easy to adapt for this communication model. We therefore design a completely different counter-example.

**Example 5.1.** Consider the system of communicating machines depicted in Fig. 7. Assume that the machines communicate via mailboxes, like in [BB11, BBO12b], i.e. all messages that are sent to peer $i$ wait in a same FIFO queue $Q_i$, and let $\mathcal{J}_k^{*-1}(\mathcal{S})$ denote the $k$-bounded send traces of $\mathcal{S}$ within this model of communications. Then $\mathcal{J}_0^{*-1}(\mathcal{S}) = \mathcal{J}_1^{*-1}(\mathcal{S}) \neq \mathcal{J}_2^{*-1}(\mathcal{S})$, as depicted in Figure 7. Therefore $\mathcal{S}$ is language 1-synchronizable but not language synchronizable, which contradicts Theorem 1 in [BB11], Theorem 2 in [BBO12a], and



$$\mathcal{J}_0^{*-1}(\mathcal{S}) = \downarrow\{\ aabcd,\ aacb,\ acab,\ caab\} = \mathcal{J}_1^{*-1}(\mathcal{S})$$

$$\mathcal{J}_2^{*-1}(\mathcal{S}) = \mathcal{J}_0^{*-1}(\mathcal{S}) \cup \{aabcde\}$$

Figure 7: Language 1-synchronizability does not imply language synchronizability for 1-$*$ (mailbox) communications *à la* [BB11, BBO12b].

Theorem 2 in [BB16]. It can be noticed that it does not contradict Theorem 1 in [BBO12b], but it contradicts the Lemma 1 of the same paper, which is used to prove Theorem 1.

5.2. **Analysis of the original mistake.** We analyse the original mistake looking at the proof of Theorem 1 in [BB11]. The proof attempt is by absurd: the authors assume a sequence of send actions $m_1 \dots m_n$ that exists in $\mathcal{I}(S)$ but not in $\mathcal{I}_1(S)$. There exists a prefix $m_1 \dots m_l$ in $\mathcal{I}_1(S)$ such that $m_1 \dots m_{l+1} \notin \mathcal{I}_1(S)$. So there are two traces $\tau \in \mathsf{Traces}(S)$ and $\tau' \in \mathsf{Traces}_1(S)$ with $\mathsf{send}(\tau) = m_1 \dots m_{l+1}$ and $\mathsf{send}(\tau') = m_1 \dots m_l$. The authors claim that the only reason why $\tau'$ cannot be extended (in $\mathsf{Traces}_1(S)$) to a trace that ends with $!m_{l+1}$ is because the buffer where $m_{l+1}$ should go is full. But they miss another explanation: it could simply be that the configuration after $\tau'$ has control states from which it is not possible to take a transition labeled with a $!m_{l+1}$, even after a few receptions. This configuration has a priori nothing in common with the configuration reached in $\tau$ right before $!m_{l+1}$.

5.3. **Realizability of choreographies.** Let us recall that a choreography $C$ is a finite automaton describing the exchange of messages between processes. A transition $(q, m^{i\to j}, q')$ in $C$ is interpreted as follows: process $P_i$, in state $q$, sends message $m$ to process $P_j$ and moves to state $q'$; and in the same way, process $P_j$, in state $q$, receives message $m$ from process $P_i$ and moves into state $q'$. The communication has to be specified and can be done by rendez-vous, bags, fifo channels ; the topology of communications could be peer-to-peer or with mailboxes. From a choreography $C$, one may construct the system $S_C$ of communicating processes $P_i$ such that each process $P_i$ is the (natural) projection of $C$ ; then $C$ coincides with the synchronous composition of the peer-to-peer system of $P_i$ (Proposition 4 in [SAAB20]). But choreography-defined peer-to-peer systems form a *strict subclass* of peer-to-peer systems.

Since the word *realizability* is used with different meanings, for example in [BBO12a] and in [SAAB20], we distinguish here two notions of realizability. A choreography $C$ is said *mailbox-realizable* (resp. *peer-to-peer-realizable*) if the system $S_C$ with respect to the mailbox semantics (resp. with respect to the peer-to-peer semantics) is synchronizable.

Basu, Bultan and Ouederni considered the question of the decidability of the mailbox-realizability of choreographies [BBO12a]. Assuming (from a previous paper from Basu and Bultan [BB11]) that $\mathcal{I}_0(S) = \mathcal{I}_1(S)$ implies $\mathcal{I}_0(S) = \mathcal{I}(S)$, they established the decidability of the mailbox-realizability of choreographies. Our counter-example shows that this decidability proof is not correct hence the decidability of the mailbox-realizability is, to the best of our knowledge, still an open problem. They did not studied the peer-to-peer-realizability problem.

Very recenly, Schewe *et al* [SAAB20] considered the peer-to-per-realizability problem and proposed a proof of decidability noticing that all our counter-examples are not choreography-defined peer-to-peer systems. They did not studied the mailbox-realizability problem.

5.4. **Branching synchronizability and stability.** Branching synchronizability is defined in [OSB13] and Theorem 1 says that a system $S$ of processes communicating through fifo channels and mailboxes is branching synchronizable iff its associated synchronous system $S_{rdv}$ is branching equivalent (i.e. bisimilar) to $S$ in which all channels are bounded by 1. It is immediate to deduce from Theorem 1 that branching synchronizability is decidable but this is false. The proof of Theorem 1 is not given in [OSB13] and it is said that it is on the web page of the first author, Ouederni; we did not found the complete paper on her web pages. Stability [ASY16] seems to be another name for branching synchronizability. More precisely, let $\mathsf{LTS}^!_k(\mathcal{S})$ denote the labeled transition system restricted to $k$-bounded configurations, where receive actions are considered as internal actions ($\tau$ transitions in CCS dialect). A system $\mathcal{S}$ is $k$-stable if $\mathsf{LTS}^!(\mathcal{S}) \overset{\text{branch}}{\sim} \mathsf{LTS}^!_k(\mathcal{S})$, where $\overset{\text{branch}}{\sim}$ denotes the branching bisimulation. In particular, a system that is 0-stable is synchronizable. Theorem 1 in [ASY16] claims that the following implication would hold for any $k \geq 1$: if $\mathsf{LTS}^!_k(\mathcal{S}) \overset{\text{branch}}{\sim} \mathsf{LTS}^!_{k+1}(\mathcal{S})$, then $\mathsf{LTS}^!_{k+1}(\mathcal{S}) \overset{\text{branch}}{\sim} \mathsf{LTS}^!_{k+2}(\mathcal{S})$. Our example 2.2 is a counter-example to this implication for $k = 0$, and it could be generalized to a counter-example for other values of $k$ by changing the number of consecutive $a$ messages that are sent by the first peer (and, symmetrically, received by the second peer). Therefore the claim of Theorem 1 in [ASY16] is not correct.

In [AS18], the authors consider the $\mathsf{LTS}^{!?}_k(\mathcal{S})$ (note the "?") associated with a given system: this LTS is the "standard" one that keeps the receive actions as being "observable". A new notion, also called stability is defined accordingly: a system (strongly) $k$-stable if $\mathsf{LTS}^{!?}(\mathcal{S}) \overset{\text{branch}}{\sim} \mathsf{LTS}^{!?}_k(\mathcal{S})$, and (strongly) stable if it is strongly $k$-stable for some $k$. It is not difficult to observe that a system is strongly $k$-stable if and only if all its traces are $k$-bounded: indeed, if all traces are $k$-bounded, $\mathsf{LTS}^{!?}(\mathcal{S}) = \mathsf{LTS}^{!?}_k(\mathcal{S})$, and if not, there is a trace with $k + 1$ unmatched send actions in $\mathsf{LTS}^{!?}(\mathcal{S})$, therefore $\mathsf{LTS}^{!?}(\mathcal{S})$ is not trace equivalent to $\mathsf{LTS}^{!?}_k(\mathcal{S})$. All results of [AS18] are therefore trivially correct.

5.5. **Existentially bounded systems.** Existentially bounded systems have been introduced by Genest, Kuske and Muscholl [GKM06]. A system $\mathcal{S}$ is existentially $k$-bounded, $k \geq 1$, if for all trace $\tau \in \mathsf{Traces}(\mathcal{S})$, there is a trace $\tau' \in \mathsf{Traces}_k(\mathcal{S})$ such that $\tau \overset{\text{causal}}{\sim} \tau'$. Unlike synchronizability, existential boundedness takes into account the receive actions, but bases on a more relaxed notion of trace (also called message sequence chart, MSC for short).

Existential boundedness and synchronizability are incomparable. For instance, a system with two peers $P_1$ and $P_2$, defined (in CCS notation) as $P_1 = !a$ and $P_2 = 0$ (idle), is existentially 1-bounded, but not synchronizable. Conversely, there are synchronous systems that are not existentially 1-bounded: consider $P = !a.!a || ?b.?b$ (i.e. all shuffles of the two), and $Q = ?a.?a || !b!b$, and assume that $P, Q$ are represented as (single-threaded) communicating automata. Then this system is synchronous, but the trace $!a!a!b!b?a?a?b?b$ is not causally equivalent to a 1-bounded trace.

Although Genest *et al* did not explicitly defined it, one could consider existentially 0-bounded systems. This is a quite restricted notion, but it would imply synchronizability and would generalize half-duplex systems.

Genest *et al* showed that for any given $k \geq 1$, it is decidable whether a given system $\mathcal{S}$ of communicating machines with peer-to-peer communications is existentially $k$-bounded (Proposition 5.5, [GKM10]). Note that what we call a system is what Genest *et al* called a deadlock-free system, since we do not have any notion of accepting states.

5.6. **Communication layers.** Finally, following the work of Lipton on reduction [Lip75], there has been recently a lot of interest on the verification of FIFO systems on the idea of grouping communications in closed rounds [CCM09, KQH18], in particular to abstract a round of communications as a single operation. In [BEJQ18], the authors define the notion of $k$-synchronous systems: a system $\mathcal{S}$ of machines communicating with mailboxes is $k$-synchronous if for all $\tau \in \mathsf{Traces}_k(\mathcal{S})$, there are $\tau_1, \dots, \tau_n$ such that

- $\tau \overset{\text{causal}}{\sim} \tau_1 \cdots \tau_n$,
- for all $i = 1, \dots, n$ $\tau_i$ contains at mots $k$ send actions, and
- every message received in $\tau_i$ has been sent in $\tau_i$

The classes of $k$-synchronous systems, of existentially $k$ bounded systems, and the one of synchronizable systems are incomparable, although they share very similar ideas.

## 6. Conclusion and Perspectives

We established the undecidability of synchronizability for communicating finite state machines communicating with peer-to-peer channels. We also proposed a counter-example for an argument of the proofs that synchronizability is decidable for mailbox communications. Finally, we showed the decidability of synchronizability for systems organized on an oriented ring.

Although we identified some problems and fixed them, our work leaves open a bunch of questions. The first one is the decidability of synchronizability for the mailboxes semantics - we only found a counter example to the proof of Basu and Bultan, but we did not show that it is undecidable. Another question is the decidability of the LTL/CTL model checking for synchronizable systems, either on traces, or on sequences of configurations. We also left open the exact complexity of synchronizability for oriented rings. We believe these questions are rather technical and sometimes very challenging.

**Acknowledgements.** We thank the anonymous reviewers of the Conference ICALP 2017 and of the Journal LMCS who produced detailed and usefull reviews that helped and motivated us to produce a better version of this paper. In particular, we added many explanations of the technical proofs, we gave more understandable details and we created Section 5 devoted to explain mistakes in previous decidability proofs and results. We also thank Shrisha Rao for his carrefull reading of the paper.

## References

[AS18] Lakhdar Akroun and Gwen Salaün. Automated verification of automata communicating via FIFO and bag buffers. *Formal Methods in System Design*, 52(3):260–276, 2018.

[ASY16] Lakhdar Akroun, Gwen Salaün, and Lina Ye. Automated analysis of asynchronously communicating systems. In *SPIN'16*, pages 1–18, 2016.

[BB11] Samik Basu and Tevfik Bultan. Choreography conformance via synchronizability. In *Procs. of WWW 2011*, pages 795–804, 2011. `doi:10.1145/1963405.1963516`.

[BB16] Samik Basu and Tevfik Bultan. On deciding synchronizability for asynchronously communicating systems. *Theor. Comput. Sci.*, 656:60–75, 2016. `doi:10.1016/j.tcs.2016.09.023`.

[BBO12a] Samik Basu, Tevfik Bultan, and Meriem Ouederni. Deciding choreography realizability. In *Procs. of POPL'12*, pages 191–202, 2012. `doi:10.1145/2103656.2103680`.

[BBO12b] Samik Basu, Tevfik Bultan, and Meriem Ouederni. Synchronizability for verification of asynchronously communicating systems. In *Procs. of VMCAI 2012*, 2012. `doi:10.1007/978-3-642-27940-9_5`.

[BEJQ18] Ahmed Bouajjani, Constantin Enea, Kailiang Ji, and Shaz Qadeer. On the completeness of verifying message passing programs under bounded asynchrony. In *CAV 2018*, pages 372–391, 2018.

[BG99] Bernard Boigelot and Patrice Godefroid. Symbolic verification of communication protocols with infinite state spaces using qdds. *Formal Methods in System Design*, 14(3):237–255, 1999. `doi:10.1023/A:1008719024240`.

[BZ81] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. Technical Report 1053, Tech. Rep. RZ, IBM Zurich Research Lab., Ruschlikon, Switzerland, January 1981.

[BZ83] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, April 1983. `doi:10.1145/322374.322380`.

[CCM09] Mouna Chaouch-Saad, Bernadette Charron-Bost, and Stephan Merz. A reduction theorem for the verification of round-based distributed algorithms. In Olivier Bournez and Igor Potapov, editors, *Reachability Problems, 3rd International Workshop, RP 2009, Palaiseau, France, September 23-25, 2009. Proceedings*, volume 5797 of *Lecture Notes in Computer Science*, pages 93–106. Springer, 2009. `doi:10.1007/978-3-642-04420-5\_10`.

[CF05] Gerald Cécé and Alain Finkel. Verification of programs with half-duplex communication. *Inf. Comput.*, 202(2):166–190, 2005. `doi:10.1016/j.ic.2005.05.006`.

[CHS14] Lorenzo Clemente, Frédéric Herbreteau, and Grégoire Sutre. Decidable topologies for communicating automata with FIFO and bag channels. In *Procs. of CONCUR 2014*, pages 281–296, 2014. `doi:10.1007/978-3-662-44584-6_20`.

[CS08] Pierre Chambart and Philippe Schnoebelen. Mixing lossy and perfect fifo channels. In *Procs. of CONCUR 2008*, pages 340–355, 2008. `doi:10.1007/978-3-540-85361-9_28`.

[DY12] Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty session types meet communicating automata. In *Procs. of ESOP 2012*, pages 194–213, 2012. `doi:10.1007/978-3-642-28869-2_10`.

[FBS05] Xiang Fu, Tevfik Bultan, and Jianwen Su. Synchronizability of conversations among web services. *IEEE Trans. Software Eng.*, 31(12):1042–1055, 2005. `doi:10.1109/TSE.2005.141`.

[GKM06] Blaise Genest, Dietrich Kuske, and Anca Muscholl. A kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6):920–956, 2006. `doi:10.1016/j.ic.2006.01.005`.

[GKM10] Blaise Genest, Dietrich Kuske, and Anca Muscholl. On communicating automata with bounded channels. *Fundamenta Informaticae*, 2010.

[HGS12] Alexander Heußner, Tristan Le Gall, and Grégoire Sutre. Mcscm: A general framework for the verification of communicating machines. In *Procs. of TACAS 2012*, pages 478–484, 2012. `doi:10.1007/978-3-642-28756-5_34`.

[HLMS12] Alexander Heußner, Jérôme Leroux, Anca Muscholl, and Grégoire Sutre. Reachability analysis of communicating pushdown systems. *Logical Methods in Computer Science*, 8(3), 2012. `doi:10.2168/LMCS-8(3:23)2012`.

[KQH18] Bernhard Kragl, Shaz Qadeer, and Thomas A. Henzinger. Synchronizing the asynchronous. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China*, volume 118 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. `doi:10.4230/LIPIcs.CONCUR.2018.21`.

[Lip75] Richard J. Lipton. Reduction: A method of proving properties of parallel programs. *Commun. ACM*, 18(12):717–721, 1975. doi:10.1145/361227.361234.

[LMP08] Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. Context-bounded analysis of concurrent queue systems. In *Procs. of TACAS 2008*, pages 299–314, 2008. doi:10.1007/978-3-540-78800-3_21.

[LP98] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the theory of computation, 2nd Edition.* Prentice Hall, 1998.

[MM98] Rajit Manohar and Alain J. Martin. Slack elasticity in concurrent computing. *Mathematics of Program Construction*, pages 272–285, 1998. doi:10.1007/BFb0054295.

[OSB13] Meriem Ouederni, Gwen Salaün, and Tevfik Bultan. Compatibility checking for asynchronously communicating software. In José Luiz Fiadeiro, Zhiming Liu, and Jinyun Xue, editors, *Formal Aspects of Component Software - 10th International Symposium, FACS 2013, Nanchang, China, October 27-29, 2013, Revised Selected Papers*, volume 8348 of *Lecture Notes in Computer Science*, pages 310–328. Springer, 2013. doi:10.1007/978-3-319-07602-7_19.

[Pac87] Jan Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. of Protocol Specification, Testing, and Verification, VII*, 1987.

[SAAB20] Klaus-Dieter Schewe, Yamine Aït-Ameur, and Sarah Benyagoub. Realisability of choreographies. In *International Symposium on Foundations of Information and Knowledge Systems*, pages 263–280. Springer, 2020.

[Sie05] Stephen F. Siegel. Efficient verification of halting properties for MPI programs with wildcard receives. In *Procs. of VMCAI 2005*, pages 413–429, 2005. doi:10.1007/978-3-540-30579-8_27.

[VVGK10] Sarvani Vakkalanka, Anh Vo, Ganesh Gopalakrishnan, and Robert M. Kirby. Precise dynamic analysis for slack elasticity: Adding buffering without adding bugs. In Rainer Keller, Edgar Gabriel, Michael Resch, and Jack Dongarra, editors, *Recent Advances in the Message Passing Interface*, pages 152–159, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-15646-5_16.