Synchronizability of Communicating Finite State Machines is not Decidable

Alain Finkel, Etienne Lozes

SYNCHRONIZABILITY OF COMMUNICATING FINITE STATE MACHINES IS NOT DECIDABLE

ALAIN FINKEL AND ETIENNE LOZES

LSV, ENS Paris-Saclay, CNRS, Université Paris-Saclay, France
e-mail address: finkel@lsv.fr

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 orphan messages, 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 [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.

Preprint submitted to
Logical Methods in Computer Science

More precisely, a system is called synchronizable if every trace is equivalent to a synchronous trace when considering send actions only. For instance, the two machines defined (using a notation *à la* CCS) as follows

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

is synchronous, because the asynchronous trace $!a!b?a?b$ is equivalent to the synchronous trace $!a?a!b?b$ if we consider all receptions as "invisible actions". However, if $P = !a!b!c$ and Q remains the same, then the system $P|Q$ is not synchronizable, because no synchronous trace is equivalent to $!a!b!c$.

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 synchronisability 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 provide counter-examples to the small model property both for the peer-to-peer semantics (Example 2.1) and the mailbox semantics (Example 5.1) which illustrate that the claims in [BBO12b, BB16] are not proved correctly.
- we show that the claim on the decidability of synchronisability for the peer-to-peer semantics is actually wrong, and establish the undecidability of synchronizability (Theorem 3.8).
- we show that the small model property holds for systems of communicating machines on an oriented ring (Theorem 4.15), 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 also show that the reachability set of such 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 synchronisability 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 5.3.

2. Preliminaries

Messages and topologies. A *message set* M is a tuple $\langle \Sigma_M, p, \mathsf{src}, \mathsf{dst}\rangle$ where Σ_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 Σ_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 $\{(i, j) \mid i + 1 = j \mod p\}$.

Traces. An *action* λ 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 λ 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 Act_M for the set of all actions over M. A *M-trace* τ is a finite (possibly empty) sequence of actions. We write Act_M^* for the set of M-traces, ϵ for the empty M-trace, and $\tau_1 \cdot \tau_2$ for the concatenation of two M-traces. We sometimes write $!?a$ for $!a\cdot?a$. A M-trace τ is a prefix of υ, $\tau \leq_{\mathsf{pref}} \upsilon$ if there is θ such that $\upsilon = \tau\cdot\theta$. The prefix closure $\downarrow S$ of a set of M-traces S is the set $\{\tau \in \mathsf{Act}_M^* \mid$ there is $\upsilon \in S$ such that $\tau \leq_{\mathsf{pref}} \upsilon\}$. For a M-trace τ and peer ids $i, j \in \{1, \dots, p\}$ we write

- $\mathsf{send}(\tau)$ (resp. $\mathsf{recv}(\tau)$) for the sequence of messages sent (resp. received) during τ, *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 λ in τ such that $\mathsf{peer}(\lambda) = i$.
- $\mathsf{onChannel}_{i\to j}(\tau)$ for the M-trace of actions λ in τ 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 $\mathsf{send}(\mathsf{onChannel}_{i\to j}(\tau)) = \mathsf{recv}(\mathsf{onChannel}_{i\to j}(\tau)) \cdot w$.

A M-trace τ is *FIFO* (resp. a *k-bounded FIFO*, for $k \geq 1$) if for all $i,j \in \{1,\ldots,p\}$, for all prefixes τ' of τ, $\mathsf{buffer}_{i\to j}(\tau')$ is defined (resp. defined and of length at most k). A M-trace is *synchronous* if it is of the form $!?a_1 \cdot !?a_2 \cdots !?a_k$ for some $k \geq 0$ and $a_1,\ldots,a_k \in M$. In particular, a synchronous M-trace is a 1-bounded FIFO M-trace (but the converse is false). A M-trace τ is *stable* if $\mathsf{buffer}_{i\to j}(\tau) = \epsilon$ for all $i \neq j \in \{1,\ldots,p\}$.

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

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

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

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

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

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 $\mathsf{Traces}_\omega(\mathcal{S})$ for $\bigcup_{k\geq 0}\mathsf{Traces}_k(\mathcal{S})$.

Example 2.1. Consider the message set $M = \{a^{1\to 2}, b^{1\to 3}, c^{3\to 2}, d^{2\to 1}\}$ 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.Then

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

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

Two traces τ_1, τ_2 are *$\mathcal{S}$-equivalent*, $\tau_1 \overset{\mathcal{S}}{\sim} \tau_2$, if $\tau_1,\tau_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ and there is γ 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 $\tau_1,\tau_2 \in \mathsf{Traces}_\omega(\mathcal{S})$, then $\tau_1 \overset{\mathcal{S}}{\sim} \tau_2$.

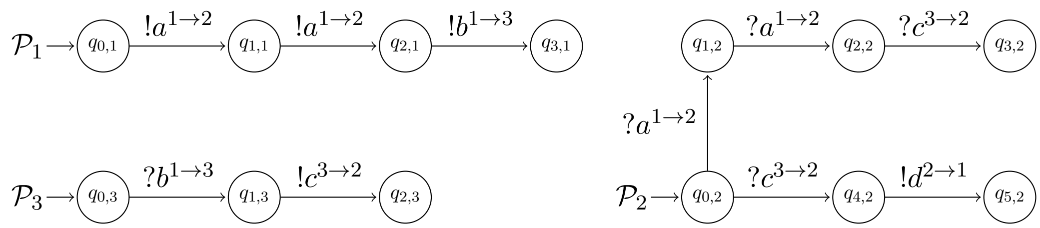


Figure 1: System of Example 2.1 and Theorem 2.3.

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

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

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

Definition 2.2 (Synchronizability [BB11, BBO12b])**.** A system $\mathcal{S}$ is *synchronizable* if $\mathcal{I}_0(\mathcal{S}) = \mathcal{I}_\omega(\mathcal{S})$. $\mathcal{S}$ is called *language synchronizable* if $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_\omega(\mathcal{S})$.

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

Proof. Consider again the system $\mathcal{S}$ of Example 2.1. Let $\gamma_{ijk} := (q_{i,1}, q_{j,2}, q_{k,3}, \epsilon, \ldots, \epsilon)$. Then

$$\begin{aligned} \mathcal{J}_0(\mathcal{S}) &= \downarrow \{a^{1\to 2} \cdot a^{1\to 2} \cdot b^{1\to 3} \cdot c^{3\to 2}\} \\ \mathcal{J}_1(\mathcal{S}) &= \mathcal{J}_0(\mathcal{S}) \\ \mathcal{J}_2(\mathcal{S}) &= \downarrow \{a^{1\to 2} \cdot a^{1\to 2} \cdot b^{1\to 3} \cdot c^{3\to 2} \cdot d^{2\to 1}\} \\ \mathcal{I}_k(\mathcal{S}) &= \mathcal{J}_k(\mathcal{S}) \cup \mathsf{Stab} \qquad \text{for all } k \geq 0 \end{aligned}$$

where $\mathsf{Stab} = \{(\epsilon, \gamma_0), (a^{1\to 2}, \gamma_{101}), (a^{1\to 2} \cdot a^{1\to 2}, \gamma_{202}), (a^{1\to 2} \cdot a^{1\to 2} \cdot b^{1\to 3}, \gamma_{312}), (a^{1\to 2} \cdot a^{1\to 2} \cdot b^{1\to 3} \cdot c^{3\to 2}, \gamma_{323})\}$. $\square$

This example contradicts Theorem 4 in [BB16], which stated that $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_1(\mathcal{S})$ implies $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_\omega(\mathcal{S})$. 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 2.4. 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. The key idea is to reduce a decision problem on a FIFO automaton $\mathcal{A}$, i.e. an automaton that can both enqueue and dequeue messages in a unique channel, to the synchronizability of a system $\mathcal{S}_{\mathcal{A}}$. The reduction is quite delicate, because synchronizability constrains a lot the way $\mathcal{S}_{\mathcal{A}}$ can be defined (a hint for that being that $\mathcal{S}_{\mathcal{A}}$ *must* involve three peers). It is also delicate to reduce from a classical decision problem on FIFO automata like *e.g.* the reachability of a control state, and we first establish the undecidability of a well-suited decision problem on FIFO automata, roughly the reception of a message m with some extra constraints. We can then 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 m in $\mathcal{A}$.

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 Act_Σ for some finite set of letters Σ 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. For technical reasons, we consider two mild restrictions on FIFO automata:

($R1$)**:** for all $\gamma_0 \xrightarrow{\tau} (q, w)$, either $\tau = \epsilon$ or $w \neq \epsilon$ (in other words, all reachable configurations are unstable, except the initial one);

($R2$)**:** for all $(q_0, \lambda, q) \in \Delta$, $\lambda = !a$ for some $a \in \Sigma$ (in other words, there is no receive action labeling a transition from the initial state).

Lemma 3.1. *The following decision problem is undecidable.*
Input: *a FIFO automaton $\mathcal{A}$ that satisfies ($R1$) and ($R2$), and a message m.*
Question: *is there a M-trace τ such that $\tau \cdot ?m \in \mathsf{Traces}_\omega(\mathcal{A})$?*

Proof. This kind of result is often considered folklore, but it seems it could be informative to detail a possible construction. We reduce from the existence of a finite tiling given a set of tiles and a pair of initial and final tiles. Intuitively, we construct a FIFO automaton that outputs the first row of the tiling, storing it into the queue, and 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. 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 initial and final tiles, 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

- $f(0, 0) = t_0$,
- 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.[1] 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

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

Therefore, any execution of $\mathcal{A}_{\mathcal{T}}$ is of the form

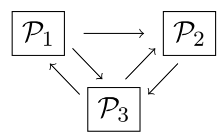
$$!t_{1,1} \cdot !t_{1,2} \cdots !t_{1,n} \cdot !\$ \cdot ?t_{1,1} \cdot !t_{2,1} \cdot ?t_{1,2} \cdot !t_{2,2} \cdots !t_{2,n} \cdot ?\$ \cdot !\$ \cdot ?t_{2,1} \cdot !t_{3,1} \cdots$$

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 $\tau \in \mathsf{Traces}_\omega(\mathcal{A})$ that contains $?t_F$. □

Let us now fix a FIFO automaton $\mathcal{A} = \langle Q_{\mathcal{A}}, \mathsf{Act}_\Sigma, \Delta_{\mathcal{A}}, q_0\rangle$ that satisfies $(R1)$ and $(R2)$. Let $M = M_1 \cup M_2 \cup M_3$ be such that all messages of Σ can be exchanged among all peers in all directions but $2 \to 1$, *i.e.*

$$\begin{array}{rcl}
M_1 & = & \{a^{1\to2}, a^{1\to3}, a^{3\to1} \mid a \in \Sigma\} \\
M_2 & = & \{a^{3\to2}, a^{1\to2}, a^{2\to3} \mid a \in \Sigma\} \\
M_3 & = & \{a^{1\to3}, a^{3\to1}, a^{3\to2}, a^{2\to3} \mid a \in \Sigma\}
\end{array}$$



Intuitively, we want $\mathcal{P}_1$ to mimick $\mathcal{A}$'s decisions and the channel $1 \to 2$ to mimick $\mathcal{A}$'s queue as follows. When $\mathcal{A}$ would enqueue a letter a , peer 1 sends $a^{1\to2}$ to peer 2, and when $\mathcal{A}$ would dequeue a letter a, peer 1 sends to peer 2 via peer 3 the order to dequeue a, and waits for the acknowledgement that the order has been correcly executed. Formally, let $\mathcal{P}_1 = \langle Q_1, q_{0,1}, \Delta_1\rangle$ be defined by $Q_1 = Q_{\mathcal{A}} \uplus \{q_\delta \mid \delta \in \Delta_{\mathcal{A}}\}$ and $\Delta_1 = \{(q, !a^{1\to2}, q') \mid (q, !a, q') \in \Delta_{\mathcal{A}}\} \cup \{(q, !a^{1\to3}, q_\delta), (q_\delta, ?a^{3\to1}, q') \mid \delta = (q, ?a, q') \in \Delta_{\mathcal{A}}\}$. The roles of peers 2 and 3 is then rather simple: peer 3 propagates all messages it receives, and peer 2 executes all orders it receives and sends back an acknowledgement when this is done. 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 as we just informally described, with a slight complication about the initial state of $\mathcal{P}_2$ (this is motivated by technical reasons that will become clear soon).

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

Example 3.2. 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

[1] Note that, due to the presence of the padding tile, this problem is equivalent to the problem of the existence of a finite rectangular tiling that contains t_0 at the beginning of the first row and t_F anywhere in the rectangle, which in turn is equivalent to the termination of a Turing machine.

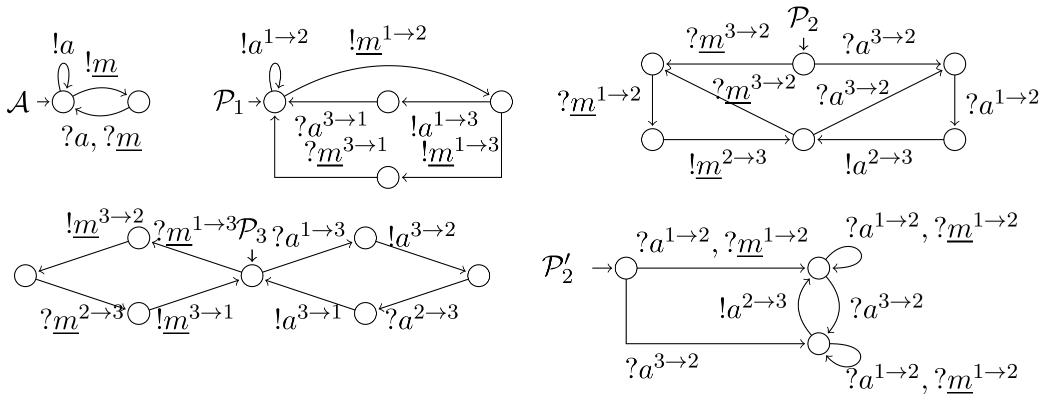


Figure 2: The FIFO automaton $\mathcal{A}$ of Example 3.2 and its associated systems $\mathcal{S}_\mathcal{A} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ and $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3 \rangle$. The sink state $q_\perp$ and the transitions $q \xrightarrow{?\underline{m}^{3\to 2}} q_\perp$ are omitted in the representation of $\mathcal{P}'_2$.

$h(\tau) \in \mathsf{Traces}_k(\mathcal{S}_\mathcal{A})$ 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 $h(!a) = !a^{1\to 2}$ and $h(?a) = !?a^{1\to 3} \cdot !?a^{3\to 2} \cdot ?a^{1\to 2} \cdot !?a^{2\to 3} \cdot !?a^{3\to 1}$. 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 $a^{1\to 3}$ 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.

Lemma 3.3. *For all $k \geq 0$,*

$$\begin{aligned}\mathsf{Traces}_k(\mathcal{S}_\mathcal{A}) = &\downarrow \{h(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{A})\} \\ &\cup \downarrow \{h(\tau) \cdot !?a^{1\to 3} \cdot !?a^{3\to 2} \mid \tau \in \mathsf{Traces}_k(\mathcal{A}), (q_0, \epsilon) \xrightarrow{\tau} (q, w), (q, ?a, q') \in \Delta\}.\end{aligned}$$

Since $\mathcal{A}$ satisfies $(R1)$, all stable configurations that are reachable in $\mathcal{S}_\mathcal{A}$ are reachable by a synchronous trace, and since it satisfies $(R2)$, the only reachable stable configuration is the initial configuration. Moreover, $\mathcal{J}_0(\mathcal{S}_\mathcal{A}) = \{\epsilon\}$ and $\mathcal{J}_k(\mathcal{S}_\mathcal{A}) \neq \{\epsilon\}$ for $k \geq 1$ (provided $\mathcal{A}$ sends at least one message). As a consequence, $\mathcal{S}_\mathcal{A}$ is not synchronizable.

Let us fix now a special message $\underline{m} \in \Sigma$. We would like to turn $\mathcal{S}_\mathcal{A}$ into a system that is synchronizable, except for the send traces that contain $\underline{m}^{2\to 3}$. Note that, by Lemma 3.3, $\mathcal{S}_\mathcal{A}$ has a send trace that contains $\underline{m}^{2\to 3}$ if and only if there are traces of $\mathcal{A}$ that contain $?\underline{m}$. Roughly, we need to introduce new behaviours for the peer 2 that will "flood" the system with many synchronous traces. Let $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3 \rangle$ be the system $\mathcal{S}_\mathcal{A}$ in which the peer $\mathcal{P}_2$ is replaced with the peer $\mathcal{P}'_2 = \langle Q'_2, q_{0,2}, \Delta'_2 \rangle$ defined as follows.

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

Example 3.4. For $\Sigma = \{a, \underline{m}\}$, and $\mathcal{A}$ as in Example 3.2, $\mathcal{P}'_2$ is depicted in Fig. 2 (omitting the transitions to the sink state $q_\perp$).

Intuitively, $\mathcal{P}'_2$ can always receive any message from peer $\mathcal{P}_1$. Like $\mathcal{P}_2$, it can also receive orders to dequeue from peer $\mathcal{P}_3$, but instead of executing the order before sending an acknowledgement, it ignores the order as follows. If $\mathcal{P}_2'$ receives the order to dequeue a message $a^{1\to 2} \neq \underline{m}^{1\to 2}$, $\mathcal{P}_2'$ acknowledges $\mathcal{P}_3$ but does not dequeue in the $1 \to 2$ queue. If the order was to dequeue $\underline{m}$, $\mathcal{P}_2'$ blocks in the sink state $q_\perp$. The system $\mathcal{S}_\mathcal{A}' = \langle \mathcal{P}_1, \mathcal{P}_2', \mathcal{P}_3 \rangle$ contains many synchronous traces: any M-trace $\tau \in L(\mathcal{P}_1)$ 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 $!\underline{m}^{1\to 3}$ does not occur in τ. However, if $\mathcal{P}_1$ takes a $!\underline{m}^{1\to 3}$ transition, it gets blocked for ever waiting for $\underline{m}^{3\to 1}$. Therefore, if $!a^{1\to 3}$ occurs in a synchronous trace τ 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 $L^{\underline{m}}(\mathcal{A})$ be the set of traces τ recognized by $\mathcal{A}$ as a finite state automaton (over the alphabet Act_Σ) such that either $?\underline{m}$ does not occur in τ, or it occurs only once and it is the last action of τ. For instance, with $\mathcal{A}$ as in Example 3.2, $L^{\underline{m}}(\mathcal{A}) = \downarrow \left(!a^* \cdot !\underline{m} \cdot ?a\right)^* \cdot !a^* \cdot !\underline{m} \cdot ?\underline{m}$. Let $h' : \mathsf{Act}_\Sigma^* \to \mathsf{Act}_M^*$ be the morphism defined by $h'(!a) = !?a^{1\to 2}$ for all $a \in \Sigma$, $h'(?a) = !?a^{1\to 3} \cdot !?a^{3\to 2} \cdot !?a^{2\to 3} \cdot !?a^{3\to 1}$ for all $a \neq \underline{m}$, and $h'(?\underline{m}) = !?\underline{m}^{1\to 3} \cdot !?\underline{m}^{3\to 2}$.

Lemma 3.5. $\mathsf{Traces}_0(\mathcal{S}_{\mathcal{A},\underline{m}}') = \downarrow \{h'(\tau) \mid \tau \in L^{\underline{m}}(\mathcal{A})\}$.

Let us now consider an arbitrary trace $\tau \in \mathsf{Traces}_\omega(\mathcal{S}_{\mathcal{A},\underline{m}}')$. Let $h'' : \mathsf{Act}_M^* \to \mathsf{Act}_M^*$ be such that $h''(!a^{1\to 2}) = !?a^{1\to 2}$, $h''(?a^{1\to 2}) = \epsilon$, and $h''(\lambda) = \lambda$ otherwise. Then $h''(\tau) \in \mathsf{Traces}_0(\mathcal{S}_{\mathcal{A},\underline{m}}')$ and $\tau \overset{\mathcal{S}}{\sim} h''(\tau)$ for $\mathcal{S} = \mathcal{S}_{\mathcal{A},\underline{m}}'$. Indeed, τ and $h''(\tau)$ are the same up to insertions and deletions of receive actions $?a^{1\to 2}$, and every state of $\mathcal{P}_2'$ (except the initial one) has a self loop $?a^{1\to 2}$. Therefore,

Lemma 3.6. *$\mathcal{S}_{\mathcal{A},\underline{m}}'$ is synchronizable.*

Let us now 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 $\mathcal{P}_2'$. Note that $\mathcal{I}_k(\mathcal{S}_{\mathcal{A},\underline{m}}'') = \mathcal{I}_k(\mathcal{S}_\mathcal{A}) \cup \mathcal{I}_k(\mathcal{S}_{\mathcal{A},\underline{m}}')$, because $q_{0,2}$ has no incoming edge in $\mathcal{P}_2 \cup \mathcal{P}_2'$.

Lemma 3.7. *Let $k \geq 1$. The following two are equivalent:*
(1) *there is τ such that $\tau \cdot ?\underline{m} \in \mathsf{Traces}_k(\mathcal{A})$;*
(2) *$\mathcal{I}_k(\mathcal{S}_{\mathcal{A},\underline{m}}'') \neq \mathcal{I}_0(\mathcal{S}_{\mathcal{A},\underline{m}}'')$.*

Proof. Let $k \geq 1$ be fixed.

(1) $\Longrightarrow$ (2)**:** Let τ be such that $\tau \cdot ?\underline{m} \in \mathsf{Traces}_k(\mathcal{A})$. By Lemma 3.3, there is $\upsilon \in \mathcal{I}_k(\mathcal{S}_\mathcal{A})$ such that $\underline{m}^{2\to 3}$ occurs in υ (take $\upsilon = \mathsf{send}(h(\tau \cdot ?\underline{m}))$). By Lemma 3.3, $\upsilon \notin \mathcal{I}_0(\mathcal{S}_\mathcal{A}) = \emptyset$, and by Lemma 3.5, $\upsilon \notin \mathcal{I}_0(\mathcal{S}_{\mathcal{A},\underline{m}}')$. Therefore $\upsilon \in \mathcal{I}_k(\mathcal{S}_{\mathcal{A},\underline{m}}'') \setminus \mathcal{I}_0(\mathcal{S}_{\mathcal{A},\underline{m}}'')$.

(2) $\Longrightarrow$ (1)**:** By contraposite. Let $\mathsf{Traces}_k(\mathcal{A} \backslash ?\underline{m}) = \{\tau \in \mathsf{Traces}_k(\mathcal{A}) \mid ?\underline{m}$ does not occur in $\tau\}$, and let us assume $\neg(1)$, *i.e.* $\mathsf{Traces}_k(\mathcal{A} \backslash ?\underline{m}) = \mathsf{Traces}_k(\mathcal{A})$. Let us show that $\mathcal{I}_k(\mathcal{S}_{\mathcal{A},\underline{m}}'') = \mathcal{I}_0(\mathcal{S}_{\mathcal{A},\underline{m}}'')$. From the assumption $\neg(1)$ and Lemma 3.3, it holds that $\mathsf{Traces}_k(\mathcal{S}_\mathcal{A}) =$

$$\begin{aligned} & \downarrow \{h(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{A} \backslash ?\underline{m})\} \\ \cup \ & \downarrow \{h(\tau) \cdot !?a^{1\to 3} \cdot !?a^{3\to 2} \mid \tau \in \mathsf{Traces}_k(\mathcal{A} \backslash ?\underline{m}), (q_0, \epsilon) \xrightarrow{\tau} (q, w), (q, ?a, q') \in \Delta\}. \end{aligned}$$

By $\mathsf{send}(h(\tau)) = \mathsf{send}(h'(\tau))$ and $\mathsf{Traces}_k(\mathcal{A} \backslash ?\underline{m}) \subseteq L^{\underline{m}}(\mathcal{A})$, we get that

$$\mathcal{I}_k(\mathcal{S}_\mathcal{A}) \quad \subseteq \quad \downarrow \{\mathsf{send}(h'(\tau)) \mid \tau \in L^{\underline{m}}(\mathcal{A})\}$$

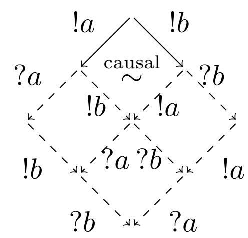


Figure 3: Diagrammatic representation of Lemma 4.1

and therefore, by Lemma 3.5, $\mathcal{I}_k(\mathcal{S}_\mathcal{A}) \subseteq \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. Since $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_k(\mathcal{S}_\mathcal{A}) \cup \mathcal{I}_k(\mathcal{S}'_{\mathcal{A},\underline{m}})$ and since by Lemma 3.6 $\mathcal{I}_k(\mathcal{S}'_{\mathcal{A},\underline{m}}) = \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$, we get that $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) \subseteq \mathcal{I}_0(\mathcal{S}''_{\mathcal{A}\underline{m}})$, and thus $\mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) = \mathcal{I}_0(\mathcal{S}''_{\mathcal{A}\underline{m}})$. □

Theorem 3.8. *Synchronizability (resp. language synchronizability) is undecidable.*

Proof. Let a FIFO automaton $\mathcal{A}$ satisfying $(R1)$ and $(R2)$ and a message $\underline{m}$ be fixed. By Lemma 3.7, $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is non synchronizable iff there is a trace τ such that $\tau \cdot ?\underline{m} \in \mathsf{Traces}_\omega(\mathcal{A})$. By Lemma 3.1, 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 nor orphan messages, 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 synchronisable system (see also Fig 3).

Lemma 4.1. *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 $\tau \cdot v_1 \in \mathsf{Traces}_\omega(\mathcal{S})$, $\tau \cdot v_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ and $\tau \cdot v_1 \overset{\mathcal{S}}{\sim} \tau \cdot v_2$.

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 υ of $!a \cdot ?a$ with $!b \cdot ?b$, either $\mathsf{onPeer}_i(\upsilon) = \mathsf{onPeer}_i(!?a \cdot !?b)$ or $\mathsf{onPeer}_i(\upsilon) = \mathsf{onPeer}_i(!?b \cdot !?a)$.*

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

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

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 υ of $!a \cdot ?a$ with $!b \cdot ?b$, $\tau \cdot \upsilon \in \mathsf{Traces}_1(\mathcal{S})$. It remains to show that

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

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

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* $i, j \in \{1, \ldots, n\}$.

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

Proof. By induction on n. Let $a_1, \ldots, a_{n+1}$ 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 $\tau_n \in \mathsf{Traces}_\omega(\mathcal{S})$
- 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 $\tau \cdot !a_1 \cdots !a_n \in \mathsf{Traces}_\omega(\mathcal{S})$
- τ'_{n+1} is 1-bounded FIFO

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

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

Lemma 4.5. *Let $\mathcal{S}$ be a 1-synchronizable system. Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $a, b_1, \ldots, b_n \in M$ 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* $i \in \{1, \ldots, n\}$.
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 $a, b_1 \ldots, b_{n+1}$ 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 $\overset{\mathcal{S}}{\sim}$

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

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. □

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. □

We can now generalize Lemma 4.4.

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 $\tau \cdot !?m_1 \cdots !?m_n \in \mathsf{Traces}_0(\mathcal{S})$.*

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, $\tau \cdot !a_1 \cdots !a_r \in \mathsf{Traces}_\omega(\mathcal{S})$ and $\tau \cdot !b_1 \cdots !b_m \in \mathsf{Traces}_\omega(\mathcal{S})$. 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 $\tau \cdot !?m_1 \cdots !?m_n \in \mathsf{Traces}_0(\mathcal{S})$. ☐

Lemma 4.8. *Let $a_1, \ldots, a_n, b_1, \cdots, b_m \in M$, and let τ 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 $k \in \{1, \ldots, 2(n+m)\}$, τ'_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 τ' 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. ☐

Lemma 4.9. *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 υ_1, υ_2 of $!?a_1 \cdot !?a_2 \cdots !?a_n$ with $!?b_1 \cdot !?b_2 \cdots !?b_m$, it holds that $\tau \cdot \upsilon_1 \in \mathsf{Traces}_\omega(\mathcal{S})$, $\tau \cdot \upsilon_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ 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 υ be a shuffle of $!?a_1 \cdots !?a_n$ with $!?b_1 \cdots !?b_m$. We want to show that $\tau \cdot \upsilon \in \mathsf{Traces}_\omega(\mathcal{S})$. Clearly, $\tau \cdot \upsilon \in \mathsf{Traces}_\omega(\mathcal{S})$ is a FIFO trace. Therefore, it is enough to find for all $i \in \{1, \ldots, p\}$ a trace τ_i such that

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

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

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

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 τ is *normalized* if there is a synchronous M-trace τ_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 $\tau \in \mathsf{Traces}_\omega(\mathcal{S})$, there is a normalized trace $\mathsf{norm}(\tau) \in \mathsf{Traces}_\omega(\mathcal{S})$ such that $\tau \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau)$.*

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

As an hint that the trace normalization does not hold if a peer can send to two different peers, consider the following examples:

Example 4.12. Let $\mathcal{P}_1 = !a^{1\to 2} \cdot !b^{1\to 3}$, $\mathcal{P}_2 = ?a^{1\to 2}$, $\mathcal{P}_3 = ?b^{1\to 3}$, and $\tau = !a \cdot !b \cdot ?b$. Then the system is 1-synchronizable, but the only trace $\overset{\mathcal{S}}{\sim}$-equivalent to τ is τ itself, which is not normalized.

Proof. By induction on τ. Let $\tau = \tau' \cdot \lambda$, be fixed. Let us assume by induction hypothesis that there is a normalized trace $\mathsf{norm}(\tau') \in \mathsf{Traces}_\omega(\mathcal{S})$ such that $\tau' \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau')$. Let us reason by case analysis on the last action λ of τ. The easy case is when λ 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 λ 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 $\mathsf{norm}(\tau) \in \mathsf{Traces}_\omega(\mathcal{S})$ and $\tau \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau)$.

Since $\mathsf{norm}(\tau') \in \mathsf{Traces}_\omega(\mathcal{S})$, we have in particular that $\tau_0' \cdot !a \in \mathsf{Traces}_1(\mathcal{S})$ and $\tau_0' \cdot !b_1 \cdots !b_n \in \mathsf{Traces}_\omega(\mathcal{S})$. 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, $\upsilon_1, \upsilon_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ 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

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

lead to two configurations γ_1', γ_2' with the same control state q for peer j as in the configuration reached after υ_1 or υ_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

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

belong to $\mathsf{Traces}_\omega(\mathcal{S})$ and $v_1'' \overset{\mathcal{S}}{\sim} v_2''$. Consider first v_1'': this is $\mathsf{norm}(\tau)$ as defined above, therefore $\mathsf{norm}(\tau) \in \mathsf{Traces}_\omega(\mathcal{S})$, and $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} v_2''$. Consider now v_2''. By definition, $v_2'' \overset{\text{causal}}{\sim} \mathsf{norm}(\tau') \cdot ?a$. By hypothesis, $\mathsf{norm}(\tau') \overset{\mathcal{S}}{\sim} \tau'$, therefore $\mathsf{norm}(\tau') \cdot ?a \overset{\text{causal}}{\sim} \tau$. To sum up, $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} v_2'' \overset{\text{causal}}{\sim} \mathsf{norm}(\tau') \cdot ?a \overset{\mathcal{S}}{\sim} \tau$, therefore $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \tau$. □

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 $\mathcal{S} = \langle \mathcal{P}_1, \dots, \mathcal{P}_p \rangle$ 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 $\{q_1 \cdots q_p \cdot w_1 \cdots w_p \mid \gamma_0 \xrightarrow{\tau} (q_1, \dots, q_p, w_1, \dots, w_p), \tau \in \mathsf{Traces}_\omega(\mathcal{S})\}$. $\mathsf{Reach}(\mathcal{S})$ 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 $\tau \in \mathsf{Traces}_\omega(\mathcal{S})$, for all $\gamma_0 \xrightarrow{\tau} \gamma$, there is a stable configuration γ', $n \geq 0$ and $m_1, \dots m_n \in M$ such that $\gamma \xrightarrow{?m_1 \cdots ?m_n} \gamma'$.*

In particular, $\mathcal{S}$ neither has orphan messages nor unspecified receptions [CF05]*.*

Proof.

(1) Let S be the set of stable configurations γ 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 $\mathsf{Reach}^!(\gamma) = \{q_1 \cdots q_p \cdot w_1 \cdots w_p \mid \gamma \xrightarrow{!a_1 \cdots !a_n} (q_1, \dots, q_p, w_1, \dots, w_p), n \geq 0, a_1, \dots a_n \in M\}$ 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 $a_1, \dots, a_n, b_1, b_m$ such that $\mathsf{src}(a_i) \neq \mathsf{src}(b_j)$ for all $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$. By Lemma 4.9, $\tau_0 \cdot \tau_1 \cdot \overline{\tau_1} \in \mathsf{Traces}_\omega(\mathcal{S})$ (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 γ'. □

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 τ the following property (which implies in particular that $\mathcal{S}$ is synchronizable):

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

$$\begin{array}{ll} (C1) & \mathsf{synch}(\tau) \in \mathsf{Traces}_0(\mathcal{S}) \\ (C2) & \tau \cdot ?m_1 \cdots ?m_k \in \mathsf{Traces}_\omega(\mathcal{S}), \text{ and} \\ (C3) & \tau \cdot ?m_1 \cdots ?m_k \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau). \end{array}$$

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

Let $\tau = \tau' \cdot \lambda$ be fixed and assume that there are $m'_1, \ldots, m'_k \in M$ such that $\tau' \cdot ?m'_1 \cdots ?m'_k \in \mathsf{Traces}_\omega(\mathcal{S})$, $\mathsf{synch}(\tau') \in \mathsf{Traces}_0(\mathcal{S})$, 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 τ. We reason by case analysis on the last action λ of τ.

- 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 $\tau' \cdot !a \in \mathsf{Traces}_\omega(\mathcal{S})$, it holds that $\tau_0 \cdot !m_1 \cdots !m_k \cdot !a \in \mathsf{Traces}_\omega(\mathcal{S})$, which implies by Lemma 4.9 that the two traces

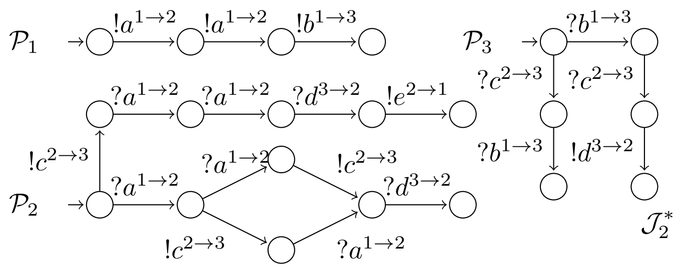
$$\begin{array}{rl} \upsilon_1 = & \tau_0 \cdot !m'_1 \cdots !m'_k \cdots ?m_1 \cdots ?m'_k \cdot !a \cdot ?a \\ \upsilon_2 = & \tau_0 \cdot !m'_1 \cdots !m'_k \cdot !a \cdots ?m'_1 \cdots ?m'_k \cdot ?a \end{array}$$

  belong to $\mathsf{Traces}_\omega(\mathcal{S})$ and verify $\upsilon_1 \overset{\mathcal{S}}{\sim} \upsilon_2$. Consider first υ_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 $\mathsf{Traces}_\omega(\mathcal{S})$, which shows $(C1)$, and $\mathsf{synch}(\tau) \overset{\mathcal{S}}{\sim} \upsilon_1$. Consider now υ_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 $\tau \cdot ?m'_1 \cdots ?m'_k \cdot ?a \in \mathsf{Traces}_\omega(\mathcal{S})$, 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 $(C3)$. $\square$

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.*

5. 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.



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

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

Figure 4: Language 1-synchronizability does not imply language synchronizability for 1-$*$ (mailbox) communications *à la* [BB11, BBO12b].

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: $\mathsf{Traces}_\omega^{bag}(\mathcal{S})$ 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 $\mathsf{Traces}_\omega^{bag}(\mathcal{S}) \subseteq T_0(\mathcal{S})$, 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 2.1 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 2.1 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. 4. 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 4. Therefore $\mathcal{S}$ is language 1-synchronizable but not language synchronizable, which contradicts Theorem 1 in [BB11], Theorem 2 in [BBO12a], and 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. **Stability.** Stability [ASY16] is a notion close to synchronizability that has been studied for mailbox systems (but it could be defined for other communication models). Let $\mathsf{LTS}_k^!(\mathcal{S})$ denote the labeled transition system restricted to k-bounded configurations, where receive actions are considered as internal actions (τ transitions in CCS dialect). In particular, a system is synchronisable if $\mathsf{LTS}_\omega^!(\mathcal{S})$ is weak trace equivalent to $\mathsf{LTS}_0^!(\mathcal{S})$. A system

$\mathcal{S}$ is k-stable if $\mathsf{LTS}^!_\omega(\mathcal{S}) \overset{\text{branch}}{\sim} \mathsf{LTS}^!_k(\mathcal{S})$, 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 2.1 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 $\mathsf{LTS}^{!?}_\omega(\mathcal{S}) \overset{\text{branch}}{\sim} \mathsf{LTS}^{!?}_k(\mathcal{S})$, 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, $\mathsf{LTS}^{!?}_\omega(\mathcal{S}) = \mathsf{LTS}^{!?}_k(\mathcal{S})$, and if not, there is a trace with $k+1$ unmatched send actions in $\mathsf{LTS}^{!?}_\omega(\mathcal{S})$, therefore $\mathsf{LTS}^{!?}_\omega(\mathcal{S})$ is not trace equivalent to $\mathsf{LTS}^{!?}_k(\mathcal{S})$. All results of [AS18] are therefore trivially correct.

5.3. **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 $\tau \in \mathsf{Traces}_\omega(\mathcal{S})$, there is a trace $\tau' \in \mathsf{Traces}_k(\mathcal{S})$ such that $\tau \overset{\text{causal}}{\sim} \tau'$. Unlike synchronisability, 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.

Finally, a notion close to existential boundedness has been recently proved decidable for mailbox communications. 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 $\tau_1, \ldots, \tau_n$ such that

- $\tau \overset{\text{causal}}{\sim} \tau_1 \cdots \tau_n$,
- for all $i = 1, \ldots, n$ τ_i contains at mots k send actions, and
- every send action of τ_i that is unmatched within τ_i remains unmatched for ever, and further sends on this mailbox will be allowed up to a limit of k unmatched sends

Therefore k-synchronous systems are existentially k-bounded. On the other hand, the classes of k-synchronous systems and the one of synchronizable systems are incomparable.

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.

[BB16] Samik Basu and Tevfik Bultan. On deciding synchronizability for asynchronously communicating systems. *Theor. Comput. Sci.*, 656:60–75, 2016.

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

[BBO12b] Samik Basu, Tevfik Bultan, and Meriem Ouederni. Synchronizability for verification of asynchronously communicating systems. In *Procs. of VMCAI 2012*, 2012.

[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.

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

[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.

[CS08] Pierre Chambart and Philippe Schnoebelen. Mixing lossy and perfect fifo channels. In *Procs. of CONCUR 2008*, pages 340–355, 2008.

[DY12] Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty session types meet communicating automata. In *Procs. of ESOP 2012*, pages 194–213, 2012.

[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.

[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.

[HLMS12] Alexander Heußner, Jérôme Leroux, Anca Muscholl, and Grégoire Sutre. Reachability analysis of communicating pushdown systems. *Logical Methods in Computer Science*, 8(3), 2012.

[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.

[MM98] Rajit Manohar and Alain J. Martin. Slack elasticity in concurrent computing. In *Procs. of Math. of Prog. Construction (MPC'98)*, pages 272–285, 1998.

[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.

[Sie05] Stephen F. Siegel. Efficient verification of halting properties for MPI programs with wildcard receives. In *Procs. of VMCAI 2005*, pages 413–429, 2005.

[VVGK10] Sarvani Vakkalanka, Anh Vo, Ganesh Gopalakrishnan, and Robert M. Kirby. Precise dynamic analysis for slack elasticity: Adding buffering without adding bugs. In *Procs. of EuroMPI 2010*, pages 152–159, 2010.