Synchronizability of Communicating Finite State Machines is not Decidable

Alain Finkel, Etienne Lozes

Synchronizability of Communicating Finite State Machines is not Decidable*

Alain Finkel[1] and Etienne Lozes[2]

1 **LSV, ENS Cachan, CNRS, Cachan, France**
`finkel@lsv.fr`
2 **LSV, ENS Cachan, CNRS, Cachan, France**
`lozes@lsv.fr`

Abstract

A system of communicating finite state machines is *synchronizable* [1, 4] 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 [1, 4, 3, 2] for either mailboxes ($*$-1) or peer-to-peer (1-1) 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.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases verification, distributed system, asynchronous communications, choreographies

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.122

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.

Slack elasticity of a distributed system with asynchronous communications is the property that the "observable behaviour" of the system is the same whatever the slack of communications is. There are actually as many notions of slack elasticity as there are notions of observable behaviours (and of distributed systems). Slack elasticity has been studied in various contexts: for hardware design [16], with the goal of ensuring that some code transformations are semantic-preserving, for parallel programming in MPI [18, 19], for ensuring the absence of deadlocks and other bugs, or more recently for web services and choreographies [1, 4, 2], for verifying various properties, among which choreography realizability [3].

* This is a proceedings version. The full version is [11], `https://arxiv.org/abs/1702.07213`.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 122; pp. 122:1–122:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This paper focuses on *synchronizability* [1], a special form of slack elasticity that was defined by Basu and Bultan for analyzing choreographies. Synchronizability is the slack elasticity of the send trace semantics of the system: a system of communicating finite state machines is synchronizable if any asynchronous trace can be mimicked by a synchronous one that contains the same send actions in the same order. Synchronizability was claimed decidable first for mailbox communications [4], where each peer stores all incoming messages in a unique mailbox in a FIFO fashion. Later, the decidability claim was extended to peer-to-peer communications [2], where there is a FIFO queue for every pair of peers. Synchronizability seemed to contrast with many other properties of systems of communicating finite state machines (including deadlock-freedom, absence of orphan messages, boundedness, etc) that are undecidable for systems of just two machines [6]. The proof relied on the claim that synchronizability would be the same as 1-synchronizability, which states that any 1-bounded trace can be mimicked by a synchronous trace.

In this paper, we show that the two claims are actually false: 1-synchronizability does not imply synchronizability, and at least for peer-to-peer communications, synchronizability is undecidable. We also show that the two claims hold, however, if we restrict to systems where the communication topology is an oriented, unidirectional ring, in particular the topology of a system with two peers only. While proving that 1-synchronizability implies synchronizability for ring topologies we also show that 1-synchronizability implies the absence of unspecified receptions and orphan messages, and that the reachability set is channel-recognizable.

Outline. The paper 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 discussions and open problems about other communication models, in particular the mailbox communication model that was the first and the most studied model in previous works on synchronizability. Due to space constraints, several proofs are omitted and can be found in a companion long version [11].

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 [5] representable) reachability sets are known to enjoy a decidable reachability problem [17]. Heussner *et al* developed a CEGAR approach based on regular model-checking [13]. Classifications of communication topologies according to the decidability of the reachability problems are known for FIFO, FIFO+lossy, and FIFO+bag communications [8, 9]. In [15, 14], 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 [10], and were shown to enforce various desirable properties. Existentially-bounded systems are systems of communicating finite state machines that were studied in a language-theoretic perspective: in [12], in particular, correspondences have been established among message sequence charts languages defined on the one hand by (universally/existentially bounded) systems of communicating machines and on the other hand by monadic second order logic over partial orders and automata Whether a system of communicating machines is existentially bounded, respectively existentially k-bounded for a fixed k, is undecidable in the general case, but it is unknown whether it remains undecidable for systems that are non-blocking.

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 $\mathsf{src}(a) \neq \mathsf{dst}(a) \in \{1, \ldots, p\}$. 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 $M = \{a_1^{i_1\to j_1}, a_2^{i_2\to j_2}, \ldots\}$ instead of $\Sigma_M = \ldots$, or $w \in M^*$ instead of $w \in \Sigma_M^*$. The communication topology associated to M is the graph G_M with vertices $\{1, \ldots, p\}$ 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 $\{(i, j) \mid i + 1 = j \bmod p\}$.

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 $\{\tau \in \mathsf{Act}_M^* \mid \text{there is } \upsilon \in S \text{ such that } \tau \leq_{\mathsf{pref}} \upsilon\}$. For a M-trace τ 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 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

1. τ, υ are FIFO, and
2. 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 $i \neq j$.

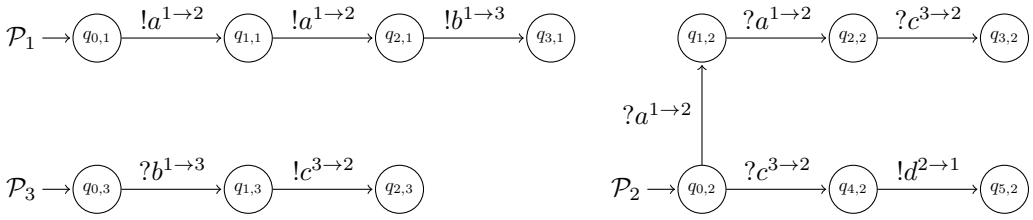


Figure 1 System of Example 1 and Theorem 3.

Let $\gamma = (q_1, \dots, q_p, w_{1,2}, \dots, w_{p-1,p})$, $\gamma' = (q'_1, \dots, q'_p, w'_{1,2}, \dots, w'_{p-1,p})$ 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 $\xrightarrow{\lambda_1}_{\mathcal{S}} \xrightarrow{\lambda_2}_{\mathcal{S}} \dots \xrightarrow{\lambda_n}_{\mathcal{S}}$. 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 $\gamma_0 = (q_{0,1}, \dots, q_{0,p}, \epsilon, \dots, \epsilon)$. 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

1. it is a FIFO trace, and
2. 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 $\mathsf{Traces}_\omega(\mathcal{S})$ for $\bigcup_{k\geq 0} \mathsf{Traces}_k(\mathcal{S})$.

▶ **Example 1.** 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. Then

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

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 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 $\tau_1 \overset{\mathcal{S}}{\sim} \tau_2$.

Synchronizability. Following [4], 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** (Synchronizability [1, 4])**.** 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 3.** *There is a system $\mathcal{S}$ that is 1-synchronizable, but not synchronizable.*

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

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

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

This example contradicts Theorem 4 in [2], 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 [2]. The next section closes this question.

▶ Remark. 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 4.** *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

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

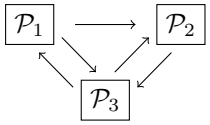
$$!t_{1,1} \cdot !t_{1,2} \cdots !t_{1,n} \cdot !\$ \cdot ?t_{1,1} \cdot !t_{2,1} \cdot ?t_{1,2} \cdot !t_{2,2} \cdots !t_{2,n} \cdot ?\$ \cdot !\$ \cdot ?t_{2,1} \cdot !t_{3,1} \cdots$$

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:

1. there is $n \geq 1$ such that $\mathcal{T}$ admits a n-tiling
2. 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 $\Delta_1 = \{(q, !a^{1\to2}, q') \mid$

[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.

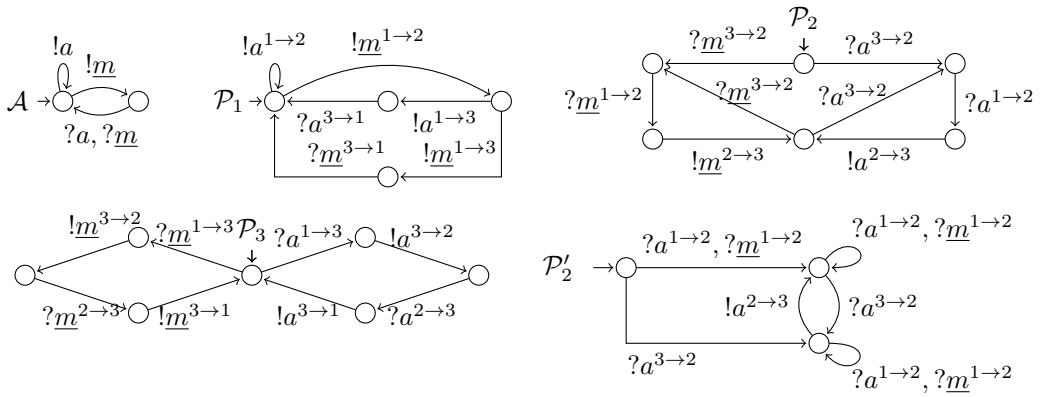


Figure 2 The FIFO automaton $\mathcal{A}$ of Example 5 and its associated systems $\mathcal{S}_\mathcal{A} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ and $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3 \rangle$. The sink state $q_\perp$ and the transitions $q \xrightarrow{?\underline{m}^{3\to2}} q_\perp$ are omitted in the representation of $\mathcal{P}'_2$.

$(q, !a, q') \in \Delta_\mathcal{A}\} \cup \{(q, !a^{1\to3}, q_\delta), (q_\delta, ?a^{3\to1}, q') \mid \delta = (q, ?a, q') \in \Delta_\mathcal{A}\}$. 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).

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

▶ **Example 5.** 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.

▶ **Lemma 6.** *For all $k \geq 0$,* $\mathsf{Traces}_k(\mathcal{S}_\mathcal{A}) = \downarrow \{h(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{A})\} \cup \downarrow \{h(\tau) \cdot !?a^{1\to3} \cdot !?a^{3\to2} \mid \tau \in \mathsf{Traces}_k(\mathcal{A}), (q_0, \epsilon) \xrightarrow{\tau} (q, w), (q, ?a, q') \in \Delta\}$.

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, $\mathcal{J}_0(\mathcal{S}_A) = \{\epsilon\}$ 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 $\underline{m}^{2\to3}$. Note that, by Lemma 6, $\mathcal{S}_\mathcal{A}$

has a send trace that contains $\underline{m}^{2\to3}$ 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.

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

▶ **Example 7.** For $\Sigma = \{a, \underline{m}\}$, and $\mathcal{A}$ as in Example 5, $\mathcal{P}'_2$ 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 $a^{1\to2} \neq \underline{m}^{1\to2}$, $\mathcal{P}'_2$ 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 $!\underline{m}^{1\to3}$ does not occur in τ. 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 $!a^{1\to3}$ 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 5, $L^{\underline{m}}(\mathcal{A}) =\downarrow \left(!a^* \cdot !\underline{m} \cdot ?a\right)^* \cdot !a^* \cdot !\underline{m} \cdot ?\underline{m}$. Let $h' : \mathsf{Act}^*_\Sigma \to \mathsf{Act}^*_M$ be the morphism defined by $h'(!a) =!?a^{1\to2}$ 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 $h'(?\underline{m}) =!?\underline{m}^{1\to3} \cdot !?\underline{m}^{3\to2}$.

▶ **Lemma 8.** $\mathsf{Traces}_0(\mathcal{S}'_{\mathcal{A},\underline{m}}) =\downarrow \{h'(\tau) \mid \tau \in L^{\underline{m}}(\mathcal{A})\}$.

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 $h''(!a^{1\to2}) =!?a^{1\to2}$, $h''(?a^{1\to2}) = \epsilon$, 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 $?a^{1\to2}$, and every state of $\mathcal{P}'_2$ (except the initial one) has a self loop $?a^{1\to2}$. Therefore,

▶ **Lemma 9.** *$\mathcal{S}'_{\mathcal{A},\underline{m}}$ 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 10.** *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 6, there is $\upsilon \in \mathcal{I}_k(\mathcal{S}_\mathcal{A})$ such that $\underline{m}^{2\to3}$ occurs in υ (take $\upsilon = \mathsf{send}(h(\tau \cdot ?\underline{m}))$). By Lemma 6, $\upsilon \notin \mathcal{I}_0(\mathcal{S}_\mathcal{A}) = \emptyset$, and by Lemma 8, $\upsilon \notin \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. 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 $\mathsf{Traces}_k(\mathcal{A}\backslash ?\underline{m}) = \{\tau \in \mathsf{Traces}_k(\mathcal{A}) \mid ?\underline{m} \text{ does not occur in } \tau\}$, 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 6, it holds that $\mathsf{Traces}_k(\mathcal{S}_{\mathcal{A}}) =$

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

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

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

and therefore, by Lemma 8, $\mathcal{I}_k(\mathcal{S}_{\mathcal{A}}) \subseteq \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. 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 9 $\mathcal{I}_k(\mathcal{S}'_{\mathcal{A},\underline{m}}) = \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$, 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 11.** *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 10, $\mathcal{S}''_{\mathcal{A},\underline{m}}$ is non synchronizable iff there is a trace τ such that $\tau \cdot ?\underline{m} \in \mathsf{Traces}_\omega(\mathcal{A})$. By Lemma 4, this is an undecidable problem. ◀

4 The case of oriented rings

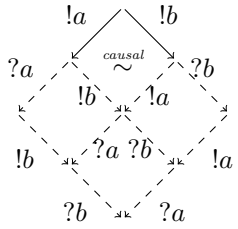
In the previous section we established the undecidability of synchronizability for systems with (at least) three peers. In this section, we show that this result is tight, in the sense that synchronizability is decidable if G_M is an oriented ring, in particular if the system involves two peers only. This relies on the fact that 1-synchronizability implies synchronizability for such systems. This property is highly non-trivial, and below we only sketch the main steps of the proof, identifying when the hypothesis on the ring topology becomes necessary.

The starting point is a confluence property on arbitrary 1-synchronizable systems.

▶ **Lemma 12.** *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 υ_1, υ_2 are any two of the six different shuffles of $!a \cdot ?a$ with $!b \cdot ?b$, then $\tau \cdot \upsilon_1 \in \mathsf{Traces}_\omega(\mathcal{S})$, $\tau \cdot \upsilon_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ and $\tau \cdot \upsilon_1 \overset{\mathcal{S}}{\sim} \tau \cdot \upsilon_2$.



▶ Remark. 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 $\mathsf{dst}(a) \neq \mathsf{src}(b)$.

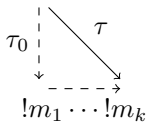
Lemma 12 then generalizes to arbitrary sequences of send actions with rather technical arguments.

▶ **Lemma 13.** *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 $\tau \cdot \upsilon_1 \overset{\mathcal{S}}{\sim} \tau \cdot \upsilon_2$.

For the rest of the proof, the hypothesis on the communication topology being an oriented ring becomes necessary. We follow the rough idea in [4], also used for half-duplex systems [7], and show a trace normalization property.

▶ **Definition 14** (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 15** (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 $\tau \overset{\mathcal{S}}{\sim} \mathsf{norm}(\tau)$.*



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

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

By Lemma 13, $\upsilon_1, \upsilon_2 \in \mathsf{Traces}_\omega(\mathcal{S})$ 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

$$\begin{aligned} \upsilon'_1 = &\quad \tau'_0 \cdot !a \cdot ?a \cdot !b_1 \cdots !b_n \\ \upsilon'_2 = &\quad \tau'_0 \cdot !a \cdot !b_1 \cdots !b_n \cdot ?a \end{aligned}$$

lead to two configurations 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

$$\begin{aligned} \upsilon''_1 = &\quad \tau'_0 \cdot !a \cdot ?a \cdot !b_1 \cdots !b_n \cdot !a_2 \cdots !a_n \\ \upsilon''_2 = &\quad \tau'_0 \cdot !a \cdot !b_1 \cdots !b_n \cdot ?a \cdot !a_2 \cdots !a_n \end{aligned}$$

belong to $\mathsf{Traces}_\omega(\mathcal{S})$ and $\upsilon''_1 \overset{\mathcal{S}}{\sim} \upsilon''_2$. Consider first υ''_1: this is $\mathsf{norm}(\tau)$ as defined above, therefore $\mathsf{norm}(\tau) \in \mathsf{Traces}_\omega(\mathcal{S})$, and $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \upsilon''_2$. Consider now υ''_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 $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \tau$. ◀

As a consequence of Lemma 15, 1-synchronizability implies several interesting properties on the reachability set.

▶ **Definition 16** (Channel-recognizable reachability set [17, 7])**.** 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 $\{q_1 \cdots q_p \cdot w_1 \cdots w_p \mid \gamma_0 \xrightarrow{\tau} (q_1, \ldots, q_p, w_1, \ldots, w_p), \tau \in \mathsf{Traces}_\omega(\mathcal{S})\}$. $\mathsf{Reach}(\mathcal{S})$ is *channel-recognizable* (or QDD representable [5]) if it is a recognizable (and rational) language.

▶ **Theorem 17.** *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 $m_1, \ldots m_n \in M$ such that $\gamma \xrightarrow{?m_1 \cdots ?m_n} \gamma'$.*

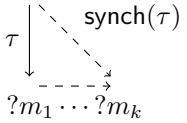
In particular, $\mathcal{S}$ neither has orphan messages nor unspecified receptions [7].

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 15, $\mathsf{Reach}(\mathcal{S}) = \bigcup\{\mathsf{Reach}^!(\gamma) \mid \gamma \in S\}$, 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 15, $\gamma_0 \xrightarrow{\tau_0 \cdot !m_1 \cdots !m_r} \gamma$ 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 13, $\tau_0 \cdot \tau_1 \cdot \overline{\tau_1} \in \mathsf{Traces}_\omega(\mathcal{S})$ (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'.

◀

▶ **Theorem 18.** *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.*



In order to prove Theorem 18, we prove by induction on the length of τ that $\tau \cdot ?m_1 \cdots ?m_k \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau)$ for some messages $m_1, \ldots, m_k$, where $\mathsf{synch}(\tau)$ denotes the unique synchronous M-trace such that $\mathsf{send}(\mathsf{synch}(\tau)) = \mathsf{send}(\tau)$.

▶ **Theorem 19.** *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 Extensions

We considered the notions of synchronizability and language synchronizability introduced by Basu and Bultan [2] and we showed that both are not decidable for systems with peer-to-peer FIFO communications, called (1-1) type systems in [2]. 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.

Non FIFO communications. In [2], 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 [2], but the same kind of argument would also hold for lossy communications. However, our Example 1 is a counter-example for Lemma 3 in [2], *i.e.* the notion of language 1-synchronizability for bag communications defined in [2] 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.

Non peer-to-peer communications. The other communication models considered in [2] keep the FIFO queue model, but differ in the way 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 [1, 4]. Our Example 1 is not easy to adapt for this communication model. We therefore design a completely different counter-example.

▶ **Example 20.** Consider the system of communicating machines depicted in Fig. 3. Assume that the machines communicate via mailboxes, like in [1, 4], 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 3. Therefore $\mathcal{S}$ is language 1-synchronizable but not language synchronizable, which contradicts Theorem 1 in [1], Theorem 2 in [3], and Theorem 2 in [2]. It can be noticed that it does not contradict Theorem 1 in [4], but it contradicts the Lemma 1 of the same paper, which is used to prove Theorem 1.

Many problems are therefore left for future work: the (un)decidability of synchronizability for the mailbox semantics, the largest class of communication topologies for which 1-synchronizability implies synchronizability (either for the peer-to-peer semantics or for the mailbox one), the study of language synchronizability, etc. Our intention in this work was limited to the identification of some of these problems, and maybe to explain why the errors in [1, 4, 2] were missed by so many reviewers.

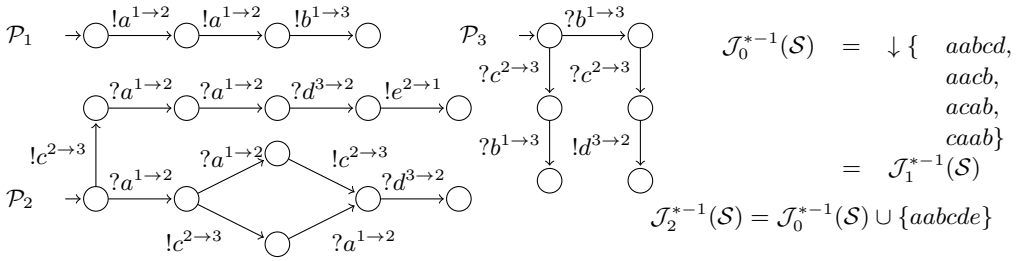


Figure 3 Language 1-synchronizability does not imply language synchronizability for 1-$*$ (mailbox) communications *à la* [1, 4].

Acknowledgement. We would like to thank the anonymous reviewers of our paper for relevant suggestions of improvements, and for an accurate reading of our proofs.

References

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

2 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.

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

4 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.

5 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.

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

7 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.

8 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.

9 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.

10 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.

11 Alain Finkel and Etienne Lozes. Synchronizability of communicating finite state machines is not decidable. Technical report, arXiv, 2017. URL: https://arxiv.org/abs/1702.07213.

12 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.

13 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.

14 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.

15 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.

16 Rajit Manohar and Alain J. Martin. Slack elasticity in concurrent computing. In *Procs. of Math. of Prog. Construction (MPC'98)*, pages 272–285, 1998. doi:10.1007/BFb0054295.

17 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.

18 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.

19 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. doi:10.1007/978-3-642-15646-5_16.