Synchronizability of Communicating Finite State Machines is not Decidable

Alain Finkel[1] and Etienne Lozes[1]

1 **LSV, ENS Cachan, CNRS, France**
`{finkel,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]. In this paper, 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.

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

Digital Object Identifier 10.4230/LIPIcs...

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 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 [15], with the goal of ensuring that some code transformations are semantic-preserving, for parallel programming in MPI [17, 18], 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 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 [4, 2], by 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 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 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 about other communication models and open problems. Due to space constraints, several proofs are deferred to the appendix.

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 [16]. Heussner *et al* developed a CEGAR approach based on regular model-checking [12]. 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 [14, 13], 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 [11], 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 $\{\tau \in \mathsf{Act}_M^* \mid$ there is $\upsilon \in S$ 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 $i \neq j$.

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 **1.** it is a FIFO trace, and **2.** 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 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

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

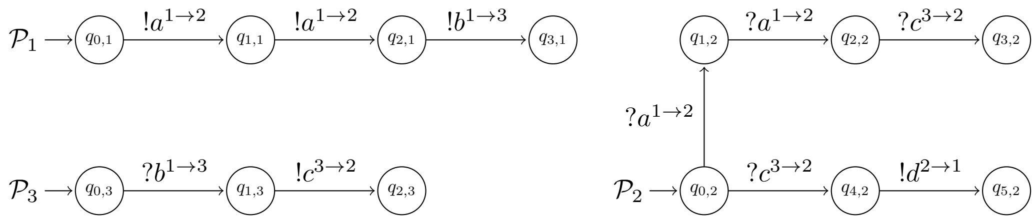


Figure 1 System of Example 1 and Theorem 3.

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 $\mathcal{I}_k(\mathcal{S})$ for the set

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

Synchronizability is then defined as the slack elasticity of this observable behaviour.

▶ **Definition 2** (Synchronizability [4]). A system $\mathcal{S}$ is *synchronizable* if $\mathcal{I}_0(\mathcal{S}) = \mathcal{I}_\omega(\mathcal{S})$.

▶ Remark. In [1, 2] the authors take another definition of behaviour that does not consider stable configurations as observable; formally, instead of $\mathcal{I}_k(\mathcal{S})$, the definition of synchronizability is based on $\mathcal{J}_k(\mathcal{S}) = \{\mathsf{send}(\tau) \mid \tau\in\mathsf{Traces}_k(\mathcal{S})\}$. The fact that we follow [4] rather than [1, 2] will be important in Section 4, but the undecidability result (Theorem 11) also holds for the definition of synchronizability of [1, 2] (and the proof is actually simpler).

For convenience, we also introduce a notion of k-synchronizability: for $k\geq 1$, a system $\mathcal{S}$ is k-*synchronizable* if $\mathcal{I}_0(\mathcal{S}) = \mathcal{I}_k(\mathcal{S})$. A system is therefore synchronizable if and only if it is 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 $\gamma_{ijk} := (q_{i,1}, q_{j,2}, q_{k,3}, \epsilon, \dots, \epsilon)$. Then

$$\begin{array}{rcl}
\mathcal{J}_0(\mathcal{S}) & = & \downarrow\{a^{1\to2}\cdot a^{1\to2}\cdot b^{1\to3}\cdot c^{3\to2}\}\\
\mathcal{J}_1(\mathcal{S}) & = & \mathcal{J}_0(\mathcal{S})\\
\mathcal{J}_2(\mathcal{S}) & = & \downarrow\{a^{1\to2}\cdot a^{1\to2}\cdot b^{1\to3}\cdot c^{3\to2}\cdot d^{2\to1}\}\\
\mathcal{I}_k(\mathcal{S}) & = & \mathcal{J}_k(\mathcal{S})\cup\mathsf{Stab} \qquad \text{for all } k\geq 0
\end{array}$$

where $\mathsf{Stab} = \{(\epsilon,\gamma_0), (a^{1\to2},\gamma_{101}), (a^{1\to2}\cdot a^{1\to2},\gamma_{202}), (a^{1\to2}\cdot a^{1\to2}\cdot b^{1\to3},\gamma_{312}), (a^{1\to2}\cdot a^{1\to2}\cdot b^{1\to3}\cdot c^{3\to2},\gamma_{323})\}$. ◀

This contradicts the claim that 1-synchronizability implies synchronizability in [4], which was the key argument for proving the decidability of synchronizability. As a remark, the claim that $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_1(\mathcal{S})$ implies $\mathcal{J}_0(\mathcal{S}) = \mathcal{J}_\omega(\mathcal{S})$, stated in [1, 2], does not hold either, due to the same counter-example.

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);

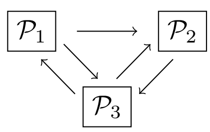
($R2$) for all $(q_0, \lambda, q) \in \Delta$, $\lambda = !a$ for some $a \in \Sigma$ (in other words, there is no receive action labeling a transition from the initial state).

▶ **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. See Appendix A. ◀

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.*

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



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 $a^{1\to 2}$ 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 $\Delta_1 = \{(q, !a^{1\to 2}, q') \mid (q, !a, q') \in \Delta_{\mathcal{A}}\} \cup \{(q, !a^{1\to 3}, q_\delta), (q_\delta, ?a^{3\to 1}, q') \mid \delta = (q, ?a, q') \in \Delta_{\mathcal{A}}\}$. 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 send 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

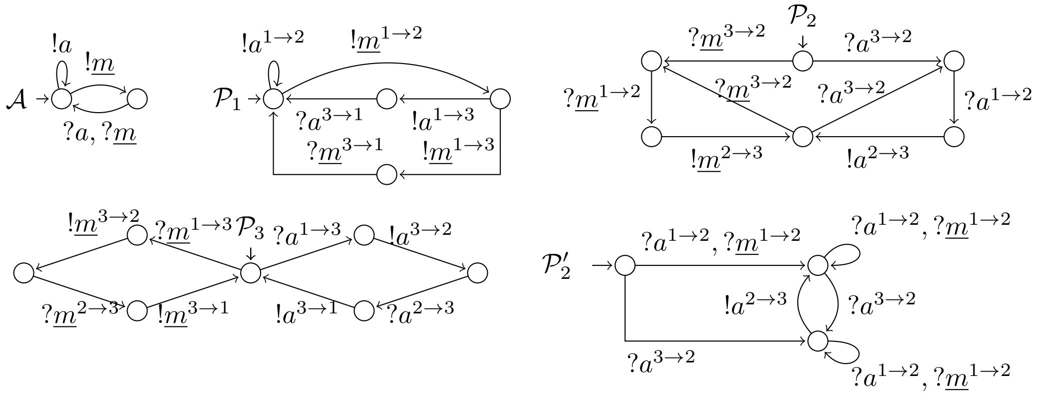


Figure 2 The FIFO automaton $\mathcal{A}$ of Example 5 and its associated systems $\mathcal{S}_\mathcal{A} = \langle \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \rangle$ and $\mathcal{S}'_{\mathcal{A},\underline{m}} = \langle \mathcal{P}_1, \mathcal{P}'_2, \mathcal{P}_3 \rangle$. The sink state $q_\perp$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

a slight complication about the initial state of $\mathcal{P}_2$ (this is motivated by technical reasons that will become clear soon).

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

▶ **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 $k \geq 0$,*

$$\begin{aligned}\mathsf{Traces}_k(\mathcal{S}_\mathcal{A}) = &\downarrow \{h(\tau) \mid \tau \in \mathsf{Traces}_k(\mathcal{A})\} \\ &\cup \downarrow \{h(\tau) \cdot !?a^{1\to3} \cdot !?a^{3\to2} \mid \tau \in \mathsf{Traces}_k(\mathcal{A}), (q_0, \epsilon) \xrightarrow{\tau} (q, w), (q, ?a, q') \in \Delta\}.\end{aligned}$$

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, $\mathcal{J}_0(\mathcal{S}_\mathcal{A}) = \emptyset$ and $\mathcal{J}_k(\mathcal{S}_\mathcal{A}) \neq \emptyset$ 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.

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

▶ **Example 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 $a^{1\to 2} \neq \underline{m}^{1\to 2}$, $\mathcal{P}'_2$ 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 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 $h'(!a) = !?a^{1\to 2}$ 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 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 $h''(!a^{1\to 2}) = !?a^{1\to 2}$, $h''(?a^{1\to 2}) = \epsilon$, 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 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 $v \in \mathcal{I}_k(\mathcal{S}_{\mathcal{A}})$ such that $\underline{m}^{2\to 3}$ occurs in v (take $v = \mathsf{send}(h(\tau \cdot ?\underline{m}))$). By Lemma 6, $v \notin \mathcal{I}_0(\mathcal{S}_{\mathcal{A}}) = \emptyset$, and by Lemma 8, $v \notin \mathcal{I}_0(\mathcal{S}'_{\mathcal{A},\underline{m}})$. Therefore $v \in \mathcal{I}_k(\mathcal{S}''_{\mathcal{A},\underline{m}}) \setminus \mathcal{I}_0(\mathcal{S}''_{\mathcal{A},\underline{m}})$.

(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 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 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

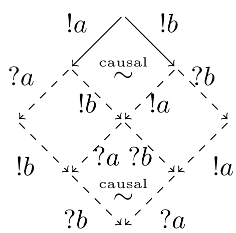
In the previous section we established the undecidability of synchronizability for systems with (at least) three peers. In this section, we show that this result is tight, in the sense that synchronizability is decidable if G_M is an oriented ring, in particular if the system involves two peers only. The proof bases on the fact that 1-synchronizability implies synchronizability for such systems (Theorem 18). In order to show this result, we first establish a trace normalization property. This property implies that 1-synchronizable systems on oriented rings have no unspecified receptions nor orphan messages, and their reachability set is channel-recognizable. We conclude with yet another technical proof that 1-synchronizability implies synchronizability when G_M is an oriented ring.

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 v_1, v_2 are any two of the four different shuffle 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 $\tau \cdot v_1 \overset{\mathcal{S}}{\sim} \tau \cdot v_2$.



Proof. See Appendix B. ◀

▶ Remark. We identified on the diagram the squares that commute thanks to causal equivalence. Not all shuffle are causally equivalent. The left square and the right square do not commute thanks to causal equivalence, but because 1-synchronizability enforces a form of confluence between send and receive transitions on the control flow graph of the peers at every mixed control state that can be reached through a synchronous trace.

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

▶ **Lemma 13.** *Let $\mathcal{S}$ be a 1-synchronizable system. Let $a_1, \dots, a_n, b_1, \dots b_m \in M$ and $\tau \in \mathsf{Traces}_0(\mathcal{S})$ be such that*
1. $\tau \cdot !a_1 \cdots !a_n \in \mathsf{Traces}_1(\mathcal{S})$,
2. $\tau \cdot !b_1 \cdots !b_m \in \mathsf{Traces}_1(\mathcal{S})$, *and*
3. $\mathsf{src}(a_i) \neq \mathsf{src}(b_j)$ *for all* $i \in \{1, \dots, n\}$ *and* $j \in \{1, \dots, m\}$.

Then for any two different shuffle 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. See Appendix B. ◀

▶ **Definition 14** (Normalized trace)**.** A M-trace τ is *normalized* if there is a synchronous M-trace υ and a M-trace θ of the form $!a_1 \cdots !a_n$ such that $\tau = \upsilon \cdot \theta$.

▶ **Lemma 15** (Trace Normalization)**.** *Assume M is such that the communication topology G_M is an oriented ring. Let $\mathcal{S} = \langle \mathcal{P}_1, \dots, \mathcal{P}_p \rangle$ be a 1-synchronizable 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, $\mathsf{norm}(\tau') \cdot \lambda \overset{\mathcal{S}}{\sim} \tau' \cdot \lambda$ 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 $\overset{\mathcal{S}}{\sim}$, there are $\tau_0 \in \mathsf{Traces}_0(\mathcal{S})$, $a_1, \dots, a_n, b_1, \dots, b_m \in M$ 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 $k \in \{1, \dots, n\}$, $\mathsf{src}(b_k) \neq i$ for all $k \in \{1, \dots, m\}$, and $\mathsf{src}(a_1) = i$. Since G_M is an oriented ring, $\mathsf{dst}(a_1) = j$, therefore $a_1 = a$. 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 $?b_1$, $?b_2, \dots ?b_n$ 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 $v_1'' \overset{\mathcal{S}}{\sim} v_2''$. 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, $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} v_2'' \overset{\text{causal}}{\sim} \mathsf{norm}(\tau') \cdot ?a \overset{\text{causal}}{\sim} \tau$, therefore $\mathsf{norm}(\tau) \overset{\mathcal{S}}{\sim} \tau$. ◀

As a consequence, 1-synchronizability implies several interesting properties on the reachability set for oriented rings.

▶ **Definition 16** (Channel-recognizable reachability set [16, 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.** *Assume $\mathcal{S}$ is 1-synchronizable and the communication topology G_M is an oriented ring. 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 $\gamma \xrightarrow{?m_1 \cdots ?m_n} \gamma'$.*

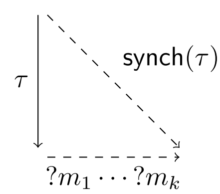
In particular, $\mathcal{S}$ neither has orphan messages nor unspecified receptions [7].

The proof of this result follows from Lemmas 13 and 15 (see Appendix B). We can now prove the result we announced at the beginning of this section.

▶ **Theorem 18.** *Let M be a message set such that G_M is an oriented ring. For all 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 $(C1)$ $\mathsf{synch}(\tau) \in \mathsf{Traces}_0(\mathcal{S})$, $(C2)$ $\tau \cdot ?m_1 \cdots ?m_k \in \mathsf{Traces}_\omega(\mathcal{S})$, and $(C3)$ $\tau \cdot ?m_1 \cdots ?m_k \overset{\mathcal{S}}{\sim} \mathsf{synch}(\tau)$. 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 15, 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, $\mathsf{norm}(\tau') \overset{\mathcal{S}}{\sim} \tau_0 \cdot !m'_1 \cdots !m'_k$. 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 13 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 $\mathsf{norm}(\tau') = \tau_0 \cdot !m'_1 \cdots !m'_k \overset{\mathcal{S}}{\sim} \tau'$ 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 $\upsilon'_2 = \tau_0 \cdot !m'_1 \cdots !m'_k \cdot !a = \mathsf{norm}(\tau') \cdot !a$. 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 $(C3)$. ◀

▶ **Theorem 19.** *Assume G_M is an oriented ring. The problem of deciding whether a given system is synchronizable is decidable.*

5 Extensions

We considered the framework introduced by Basu and Bultan [1] and we showed that synchronizability is not decidable for systems with peer-to-peer FIFO communications. In their more recent work [2], Basu and Bultan considered the question of synchronizability for other communication models. One variant they consider is communications with bags instead of queues, thus allowing to reorder messages. Synchronizability is decidable for such a model of communications: $\mathcal{I}_\omega(\mathcal{S})$ is the language of a Petri net, $\mathcal{I}_0(\mathcal{S})$ is an effective regular language, and whether the language of a Petri is included in a given regular language reduces to the coverability problem. The same argument would hold for lossy communications. Another variant considered in [2] is a communication model based on mailboxes: instead of having distinct queues for messages coming from distinct senders, the peers store in a single queue all the messages they receive. The (un)decidability of synchronizability for this communication model is unclear. At least, the argument that 1-synchronizability implies synchronizability does not hold for this communication model either (see Appendix C for a counter-example), but our results show that it holds for oriented rings, because there is no difference between mailboxes and peer-to-peer queues on such topologies.

Our undecidability result suggests that synchronizability may not be the right notion for all communication topologies, and one might want to find the largest class of communication topologies on which 1-synchronizability implies synchronizability, or on which synchronizability is decidable. Our intention in this work was more limited, and only aimed at explaining why, maybe, the errors in [1, 4, 2] were missed by so many reviewers. We believe it could be more interesting to consider other notions of slack elasticity. For instance, one might consider the property that all traces of a system are causally equivalent to a k-bounded trace. This class of systems ressembles the classes of existentially k-bounded systems [11], but with something like what is called the “non-blocking” assumption in this framework. We leave it for future work to better identify the possible connections of our results with the theory of existentially bounded systems.

References

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

12 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. doi:10.1007/978-3-642-28756-5_34.

13 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.

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

15 Rajit Manohar and Alain J. Martin. *Slack elasticity in concurrent computing*, pages 272–285. doi:10.1007/BFb0054295.

16 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.

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

18 Sarvani Vakkalanka, Anh Vo, Ganesh Gopalakrishnan, and Robert M. Kirby. *Precise Dynamic Analysis for Slack Elasticity: Adding Buffering without Adding Bugs*, pages 152–159. doi:10.1007/978-3-642-15646-5_16.

A Proof of Lemma 4

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 $n \geq 0$, a n-tiling is a function $f : \{1, \ldots, n\} \times \mathbb{N} \to T$ such that

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

The problem of deciding, given a tuple $\mathcal{T} = \langle T, t_0, t_F, H, V\rangle$, Whether there is some $n \geq 0$ for which there exists a n-tiling is undecidable. Note that, due to the presence of the padding tile, this problem is equivalent to the more standard 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.

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

- $Q = \{q_{t,0}, q_{\downarrow=t}, q_{\leftarrow=t}, q_{\leftarrow=t,\downarrow=t'} \mid t \in T, t' \in T \cup \{\$\}\} \cup \{q_0, q_1\}$
- $\Sigma = T \cup \{\$\}$
- $\Delta \subseteq Q \times \mathsf{Act}_\Sigma \times Q$, 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 $n \geq 0$ such that $\mathcal{T}$ admits a n-tiling
2. there is a trace $\tau \in \mathsf{Traces}_\omega(\mathcal{A})$ that contains $?t_F$

B Omitted Proofs of Section 4

B.1 Proof of Lemma 12

Let us first show that $\tau \cdot v \in \mathsf{Traces}_1(\mathcal{S})$ for all shuffle v of $!a \cdot ?a$ with $!b \cdot ?b$. Let $\tau_1 = \tau \cdot !a \cdot ?a \cdot !b \cdot ?b$ and $\tau_2 = \tau \cdot !b \cdot ?b \cdot !a \cdot ?a$. 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, $\tau_1, \tau_2 \in \mathsf{Traces}_0(\mathcal{S})$. It remains to show that for all shuffle v of $a \cdot ?a$ with $!b \cdot ?b$ that start with two sends, $\tau \cdot v$ belongs to $\mathsf{Traces}_1(\mathcal{S})$. By symmetry, and considering that the receptions are executed by two different peers and therefore can be executed in any order, it is enough to show that $\tau_3 := \tau \cdot !a \cdot !b \cdot ?a \cdot ?b \in \mathsf{Traces}_1(\mathcal{S})$. This follows from τ_3 being 1-bounded, and the fact that for any peer $i \in \{1, \ldots, p\}$, $\mathsf{onPeer}_i(\tau_3) \in \{\mathsf{onPeer}_i(\tau_1), \mathsf{onPeer}_i(\tau_2)\}$ is a sequence of actions supported by peer i.

So we proved that for all shuffle v of $!a \cdot ?a$ with $!b \cdot ?b$, $\tau \cdot v \in \mathsf{Traces}_1(\mathcal{S})$. It remains to show that all these traces lead to the same configuration. Since they all lead to a stable configuration, and since $\mathcal{S}$ is 1-synchronizable, the configuration a trace leads to only depends on the order in which the send actions $!a$ and $!b$ are executed in v. But since the two traces $\tau \cdot !a \cdot !b \cdot ?a \cdot ?b$ and $\tau \cdot !b \cdot !a \cdot ?a \cdot ?b$ lead to the same configuration, they all lead to the same configuration.

B.2 Proof of Lemma 13

▶ **Lemma 20.** *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 21.** *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 12, $\tau_n \cdot !?a \cdot !?b_{n+1} \in \mathsf{Traces}_0(\mathcal{S})$, $\tau_n \cdot !?b_{n+1} \cdot !?a \in \mathsf{Traces}_0(\mathcal{S})$, 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 22.** *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*
- $\tau \cdot !?a_1 \cdots !?a_n \cdot !?b_1 \cdots !?b_m \stackrel{\mathcal{S}}{\sim} \tau \cdot !?c_1 \cdots !?c_m$.

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 21, $\tau' \cdot !?b_1 \cdots !?b_m \in \mathsf{Traces}_0(\mathcal{S})$, 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

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

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

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

  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 20.

▶ **Lemma 23.** *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 20, $\tau \cdot !?a_1 \cdots !?a_r \in \mathsf{Traces}_0(\mathcal{S})$, and by induction hypothesis $\tau \cdot !?b_1 \cdots !?b_m \in \mathsf{Traces}_0(\mathcal{S})$, and finally by Lemma 22 $\tau \cdot !?m_1 \cdots !?m_n \in \mathsf{Traces}_0(\mathcal{S})$. ◀

▶ **Lemma 24.** *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 $i \in \{1, \ldots, p\}$ be fixed. For every $m \in M$, let $m^* = !m$ if $\mathsf{dst}(m) \neq i$, otherwise $m^* = ?m$, and let $\overline{m}^* = ?m$ if $\mathsf{dst}(m) \neq i$, otherwise $\overline{m}^* = !m$. Finally, let h be the homomorphism defined by $h(m^*) = !?m$ and $h(\overline{m}^*) = \epsilon$. Then for all M-traces τ, $\mathsf{onPeer}_i(\tau) = \mathsf{onPeer}_i(h(\tau))$. Let τ be a shuffle of $!?a_1 \cdots !?a_n$ with $!?b_1 \cdots !?b_m$. Then there is a shuffle υ of $a_1^* \cdots a_n^*$ with $b_1^* \cdots b_m^*$ such that $h(\upsilon) = h(\tau)$. Therefore, $h(\tau) = !?c_1 \cdots !?c_{n+m}$ for some shuffle $c_1 \cdots c_{n+m}$ of $a_1 \cdots a_n$ with $b_1 \cdots b_m$. ◀

We are now ready to do the proof of Lemma 13.

Proof. (of Lemma 13) Let $\tau \in \mathsf{Traces}_0(\mathcal{S})$ and $a_1, \dots, a_n, b_1, \dots, b_m$, 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 $i \in \{1, \dots, p\}$ a trace τ_i such that

$$\tau_i \in \mathsf{Traces}_\omega(\mathcal{S}) \qquad \text{and} \qquad \mathsf{onPeer}_i(\tau \cdot \upsilon) = \mathsf{onPeer}_i(\tau_i). \tag{1}$$

Let $i \in \{1, \dots, p\}$ be fixed, and let us construct τ_i that validates (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 23,

$$\tau \cdot !?a_1 \cdots !?a_n \in \mathsf{Traces}_0(\mathcal{S}) \text{ and } \tau \cdot !?b_1 \cdots !?b_n \in \mathsf{Traces}_0(\mathcal{S}). \tag{2}$$

On the other hand, by Lemma 24, there is a shuffle $c_1 \dots c_{n+m}$ of $a_1 \cdots a_n$ with $b_1 \cdots b_m$ such that

$$\mathsf{onPeer}_i(\upsilon) = \mathsf{onPeer}_i(!?c_1 \cdots !?c_{n+m}) \tag{3}$$

Let $\tau_i = \tau \cdot !?c_1 \cdots !?c_{n+m}$. By Lemma 22 and (2), $\tau_i \in \mathsf{Traces}_0(\mathcal{S})$, and by (3), the second part of (1) holds. ◀

B.3 Proof of Theorem 17

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 $\mathsf{Reach}^!(\gamma) = \{q_1 \cdots q_p \cdot w_1 \cdots w_p \mid \gamma \xrightarrow{!a_1 \cdots !a_n} (q_1, \dots, q_p, w_1, \dots, w_p), n \geq 0, a_1, \dots a_n \in M\}$ is an effective rational language.
2. Assume $\gamma_0 \xrightarrow{\tau} \gamma$. By Lemma 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 $a_1, \dots, a_n, b_1, b_m$ 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 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'.

C 1-Synchronizability Does not Implie Synchronizability for Communications with Mailboxes

Consider the system of communicating machines depicted in Fig. 3. Assume that the machines communicate via mailboxes, i.e. all messages that are send to peer i wait in a same FIFO queue, and let $\mathcal{J}_k^{*-1}(\mathcal{S})$ denote the k-bounded send traces of $\mathcal{S}$ within this model of communications (see [2] for a precise definition).

The the following holds.

$$\begin{array}{rcll}
\mathcal{J}_0^{*-1}(\mathcal{S}) & = & \downarrow \{ & a^{1\to2} \cdot a^{1\to2} \cdot b^{1\to3} \cdot c^{2\to3} \cdot d^{3\to2}, \\
& & & a^{1\to2} \cdot a^{1\to2} \cdot c^{2\to3} \cdot b^{1\to3}, \\
& & & a^{1\to2} \cdot c^{2\to3} a^{1\to2} \cdot b^{1\to3}, \\
& & & c^{2\to3} \cdot a^{1\to2} \cdot a^{1\to2} \cdot b^{1\to3}\} \\
& = & \mathcal{J}_1^{*-1}(\mathcal{S}) & \\
\mathcal{J}_2^{*-1}(\mathcal{S}) & = & \mathcal{J}_0^{*-1}(\mathcal{S}) \cup \{a^{1\to2} \cdot a^{1\to2} \cdot b^{1\to3} \cdot c^{2\to3} d^{3\to2} \cdot e^{2\to1}\} &
\end{array}$$

Therefore the system of Fig. 3 is 1-synchronizable but not synchronizable.

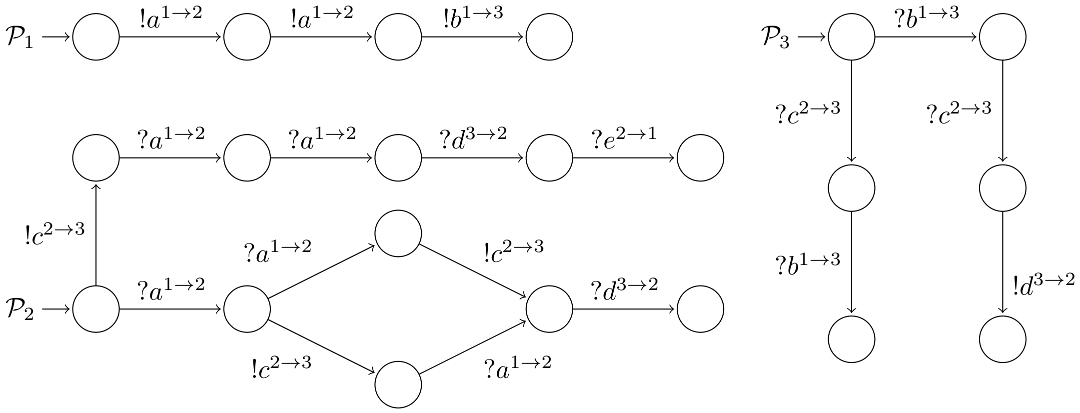


Figure 3 1-synchronizability does not imply synchronizability when processes communicate via mailboxes.