On Composing Communicating Systems*

Franco Barbanera
Dept. of Mathematics and Computer Science, University of Catania (Italy)

Ivan Lanese
Focus Team, University of Bologna/INRIA (Italy)

Emilio Tuosto
Gran Sasso Science Institute (Italy)

Communication is an essential element of modern software, yet programming and analysing communicating systems are difficult tasks. A reason for this difficulty is the lack of compositional mechanisms that preserve relevant communication properties.

This problem has been recently addressed for the well-known model of *communicating systems*, that is sets of components consisting of finite-state machines capable of exchanging messages. The main idea of this approach is to take two systems, select a participant from each of them, and derive from those participants a pair of coupled gateways connecting the two systems. More precisely, a message directed to one of the gateways is forwarded to the gateway in the other system, which sends it to the other system. It has been shown that, under some suitable *compatibility* conditions between gateways, this composition mechanism preserves deadlock freedom for asynchronous as well as symmetric synchronous communications (where sender and receiver play the same part in determining which message to exchange).

This paper considers the case of *asymmetric synchronous communications* where senders decide independently which message should be exchanged. We show here that preservation of lock freedom requires sequentiality of gateways, while this is not needed for preservation of either deadlock freedom or strong lock freedom.

1 Introduction

Communication is an essential constitutive element of modern software due to the fact that applications are increasingly developed in distributed architectures (e.g., service-oriented architectures, microservices, cloud, etc.). In practice, APIs and libraries featuring different communication mechanisms are available for practically any programming language.

Reasoning on and developing communicating systems are difficult endeavours. Indeed, several models have been used to study interactions between systems (e.g., process algebras, transition systems, Petri nets, logical frameworks, etc.). The so-called *business logic*, necessary to determine *what* has to be communicated, needs to be complemented with the so-called *application level protocol* specifying *how* information spreads across a system. Conceptual and programming errors may occur in the realisation of application level protocols. For instance, it may happen that some components in a system are prevented to communicate because all the expected partners terminated their execution (deadlock). Other typical errors occur when a system is not lock-free, that is when some components cannot progress because all their partners are perpetually involved in other communications. These kinds of problems can arise

* Research partly supported by the EU H2020 RISE programme under the Marie Skłodowska-Curie grant agreement No 778233, by the MIUR project PRIN 2017FTXR7S "IT-MaTTerS" (Methods and Tools for Trustworthy Smart Systems) and by the Progetto di Ateneo "Piaceri". The authors have also been partially supported by INdAM as members of GNCS (Gruppo Nazionale per il Calcolo Scientifico). The authors thanks the reviewers for their helpful comments and also Mariangiola Dezani for her support.

C. Aubert, C. Di Giusto, L. Safina & A. Scalas (Eds.):
15th Interaction and Concurrency Experience (ICE 2022)
EPTCS 365, 2022, pp. 53–68, doi:10.4204/EPTCS.365.4

when a system can evolve in different ways depending on some conditions and components have inconsistent "views" of the state of the system. If this happens, some components may reach a state no longer "compatible" with the state of their partners and therefore communications do not happen as expected.

We illustrate these problems with some simple examples for deadlock freedom (similar examples may be given for lock freedom). Suppose we want to model a client-server system where clients' requests are acknowledged either with an answer or with "unknown" from servers. Due to its popularity, we choose CCS [17] to introduce this scenario, so take the following agents:

$$\mathsf{C} = \bar{\mathsf{r}}.\mathsf{a} + \bar{\mathsf{r}}.\mathsf{u} \qquad \text{and} \qquad \mathsf{D} = \mathsf{r}.\bar{\mathsf{a}} + \mathsf{r}.\bar{\mathsf{u}} \tag{1}$$

where ports r, a, and u are respectively used to commmunicate requests, answers, and unknowns. (Recall that in CCS $\mathsf{a}.\_$ and $\_ + \_$ represent respectively prefix and non-deterministic choice.) The common interpretation of agents in (1) is that x and $\bar{\mathsf{x}}$ respectively represent an input and an output on port x. It is a simple observation that the system $\mathsf{C} \mid \mathsf{D}$ where C and D run in parallel can evolve to e.g., the deadlock state $\mathsf{a} \mid \bar{\mathsf{u}}$ where each party is waiting for the other to progress. The problem is that the choice of what communication should happen after a request is taken independently by C and D instead of letting D to take the decision and drive C "on the right" branch. This is attempted in the next version:

$$\mathsf{C} = \bar{\mathsf{r}}.(\mathsf{a} + \mathsf{u}) \qquad \text{and} \qquad \mathsf{D} = \mathsf{r}.(\bar{\mathsf{a}} + \bar{\mathsf{u}}) \tag{2}$$

A key difference with the agents in (1) is that the server D in (2) decides what to reply to the client C, which becomes aware of the choice through the interaction with D after the request has been made. Assume now that D acts as a proxy to another server, say D'. When D cannot return an answer to the client C it interacts with D' on port p. Answers are sent directly to C if D' can compute them, otherwise D' returns an unknown on port u' to D which forwards it to C. This is modelled by the agents

$$\mathsf{D} = \mathsf{r}.(\bar{\mathsf{a}} + \bar{\mathsf{p}}.\mathsf{u}'.\bar{\mathsf{u}}) \qquad \text{and} \qquad \mathsf{D}' = \mathsf{p}.(\bar{\mathsf{a}} + \bar{\mathsf{u}'}) \tag{3}$$

Note that this change is completely transparent to agent C, which in fact stays as in (2). It is now more difficult to ascertain if these choices may lead to a deadlock since the decision of D may involve also D'. Indeed, the parallel composition of agents in (3) may deadlock because, when C and D interact on port a, D' hangs on port p and, likewise, if C and D' interact on port a then D hangs on port u'.

A reason for this difficulty is that it is hard to define compositional mechanisms that preserve relevant communication properties such as deadlock or lock freedom. Recently, an approach to the composition of concurrent and distributed systems has been proposed in [2, 3] for the well-known model of systems of *communicating finite-state machines* (CFSMs) [12], that is sets of finite-state automata capable of exchanging messages. The compositional mechanism is based on the idea that two given systems, say S and S', are composed by transforming two CFSMs, say H in S and K in S', into "coupled forwarders". Basically, each message that H receives from a machine in S is forwarded to K and vice versa. It has been shown that, under suitable *compatibility* conditions between H and K, this composition mechanism preserves deadlock freedom for asynchronous as well as symmetric synchronous communications (where sender and receiver play the same part in determining which message to exchange). The compatibility condition identified in [2, 3] consists in exhibiting essentially dual behaviours: gateway H is able to receive a message whenever gateway K is willing to send one and vice versa. As observed in [4], a remarkable feature of such an approach is that it enables the composition of systems originally designed as *closed* systems. As far as two compatible machines can be found, any two systems can be composed by transforming as hinted above the compatible machines.

The results in [2, 3] are developed in the asynchronous semantics of CFSMs. These results have been transferred in [5] to a setting where CFSMs communicate synchronously much like as the communication mechanisms considered for instance in process algebras like CCS, ACP, etc. This model assumes a perfect symmetry between sender and receiver in synchronous communications. Let us again discuss this with an example. Consider the agents

$$\mathsf{T} = \bar{\mathsf{a}}.\mathsf{P} + \bar{\mathsf{b}}.\mathsf{Q} \qquad \text{and} \qquad \mathsf{R} = \mathsf{a}.\mathsf{P}' + \mathsf{b}.\mathsf{Q}' \tag{4}$$

According to the standard semantics of CCS [17], system $(\mathsf{T} \mid \mathsf{R}) \setminus \{\mathsf{a},\mathsf{b}\}$ has two possible evolutions:

$$(\mathsf{T} \mid \mathsf{R}) \setminus \{\mathsf{a},\mathsf{b}\} \xrightarrow{\tau} \mathsf{P} \mid \mathsf{P}' \qquad \text{and} \qquad (\mathsf{T} \mid \mathsf{R}) \setminus \{\mathsf{a},\mathsf{b}\} \xrightarrow{\tau} \mathsf{Q} \mid \mathsf{Q}'$$

namely, either both T and R opt for the "leftmost" branch (synchronising on a) or they both choose the "rightmost" one (synchronising on b). (Recall that in CCS $\_ \setminus X$ is the hiding of ports in the set X and that τ represents an internal action.) This means that, the resolution of the choice is implicit in the communication mechanism: a branch is taken as soon as T and R synchronise on the corresponding port. Intuitively, no distinction is made between sender and receiver (formally they are indeed interchangeable); this implies that the communication mechanism is at the very core of choice resolution [5].

Interestingly, for synchronous communications an alternative interpretation is actually possible. Indeed, this perfect symmetry is not assumed so that sender and receiver play different roles in choice resolution while still relying on synchronous communication. Let us explain this interpretation using again CCS. Consider a variant of CCS where outputs must be enabled before being fired. One could formally specify that with the following reduction rules:

$$\bar{\mathsf{a}}.\mathsf{P} + \mathsf{P}' \xrightarrow{\tau} \bar{\bar{\mathsf{a}}}.\mathsf{P} \qquad \text{and} \qquad \bar{\bar{\mathsf{a}}}.\mathsf{P} \mid (\mathsf{a}.\mathsf{Q} + \mathsf{Q}') \xrightarrow{\tau} \mathsf{P}|\mathsf{Q} \tag{5}$$

whereby the leftmost rule *chooses* one of the possible outputs of the sender (the chosen output is marked by the double bar in our notation) and the rightmost rule actually synchronises sender and receiver. This semantics is an abstract model of *asymmetric* communications (used e.g., in [8, 18]), where silent steps taken using the left rule model some internal computation of the sender to decide what to communicate to the partner. In other words, now the choice is entirely resolved on "one side" while the communication is a mere interaction of complementary actions, the output and the input. This asymmetry, at the core of asynchronous communication, can therefore also carry for synchronous communication.

It is worth observing that asymmetric communications abstract a rather common programming pattern where sending components may choose the output to execute *depending* on some internal computation. For instance, elaborating on the proxy scenario in (3), D could decide to directly send unknowns to normal clients while reserving the use of D' only for "privileged" clients.

Contributions. This paper transfers the composition by gateway mechanism of [2, 3] to the case of asymmetric synchronous communication of CFSMs. The main technical results are that, in the asymmetric case, gateway composition

- preserves deadlock freedom (as well as a strong version of lock freedom) provided that systems are composable (the relation of compatibility – one of the requirements for systems to be composable – in the present paper is less restrictive than the one used in [5]);
- preserves lock freedom if systems are composable and gateways are *sequential*, namely each state has at most one outgoing transition.

Interestingly, preservation of deadlock freedom can be guaranteed under milder conditions than in the symmetric case. In fact, sequentiality of gateways is not necessary to preserve deadlock freedom in the asymmetric case, while it is in the symmetric one.

Structure of the paper. Section 2 introduces systems of (asymmetric synchronous) CFSMs, related notions and communication properties. Composition by gateways is introduced and discussed in Section 3 together with the compatibility relation. Section 4 discusses the issues that prevent gateway composition to preserve communication properties. Section 5 is devoted to the preservation of communication properties. Conclusions, related and future work are discussed in Section 6. For space limitation, full proofs are reported in [6].

2 Background

Communicating Finite State Machines (CFSMs) [12] are Finite State Automata (FSAs) where transitions are labelled by communications. We recall basic notions on FSAs.

A *finite state automaton* (FSA) is a tuple $A = \langle \mathscr{S}, q_0, \mathscr{L}, \rightarrow \rangle$ where

- $\mathscr{S}$ is a finite set of states (ranged over by lowercase italic Latin letters);
- $q_0 \in \mathscr{S}$ is the *initial state*;
- $\mathscr{L}$ is a finite set of labels
- $\rightarrow \subseteq \mathscr{S} \times (\mathscr{L} \cup \{\tau\}) \times \mathscr{S}$ is a set of transitions.

Hereafter, we let λ range over $\mathscr{L} \cup \{\tau\}$ when it is immaterial to specify the set of labels or it is understood. We use the usual notation $q_1 \xrightarrow{\lambda} q_2$ for the transition $(q_1, \lambda, q_2) \in \rightarrow$, and $q_1 \rightarrow q_2$ when there exists a label λ such that $q_1 \xrightarrow{\lambda} q_2$. Let $\_\cdot\_$ be the concatenation operation on labels and write $p \xrightarrow{\pi} q$ where $\pi = \lambda_1 \cdot \lambda_2 \cdot \ldots \cdot \lambda_n$ whenever $p \xrightarrow{\lambda_1} p_1 \xrightarrow{\lambda_2} \ldots \xrightarrow{\lambda_n} p_n = q$. We let $\pi, \psi, \ldots$ range over $\mathscr{L}^\star$ (i.e., sequences of labels) and define the set of *reachable states in A from q* as

$$\mathscr{R}(A,q) = \{\, p \mid \text{there is } \pi \in \mathscr{L}^\star \text{ such that } q \xrightarrow{\pi} p \,\}$$

The set of *reachable states in A* is $\mathscr{R}(A) = \mathscr{R}(A, q_0)$. For succinctness, $q \xrightarrow{\lambda} q' \in A$ means that the transition belongs to (the set of transitions of) A; likewise, $q \in A$ means that q belongs to the states of A. We say that $q \xrightarrow{\lambda} q'$ is an *outgoing* (resp. *incoming*) transition of q (resp. q'). Since we use FSAs to formalise communicating systems, accepting states are disregarded (as also done in [12]).

We adapt the definitions in [12] to cast CFSMs in our context. Let $\mathfrak{P}$ be a set of *participants* (or *roles*, ranged over by A, B, etc.) and $\mathscr{M}$ a set of *messages* (ranged over by m, n, etc.). We take $\mathfrak{P}$ and $\mathscr{M}$ disjoint. An *output label* is written as $\mathsf{A\,B!m}$ and represents the willingness of A to send message m to B; likewise, an *input label* is written as $\mathsf{A\,B?m}$ and represents the willingness of B to receive message m from A. The *subjects* of an output label $\mathsf{A\,B!m}$ and of an input label $\mathsf{A\,B?m}$ are A and B, respectively.

Definition 2.1 (CFSMs). *Let* $\mathscr{L}_{act} = \{\mathsf{A\,B!m}, \mathsf{A\,B?m} \mid \mathsf{A} \neq \mathsf{B} \in \mathfrak{P}, \mathsf{m} \in \mathscr{M}\}$. *A* communicating finite-state machine *(CFSM) is an FSA M with labels* $\mathscr{L}_{act} \cup \{\tau\}$ *such that, for any transition* $p \xrightarrow{\lambda} q$ *of M,*

- *if* λ *is an output label then* $p \neq q$ *and p has exactly one incoming transition, and such transition is labelled by* τ*;*
- *if* $\lambda = \tau$ *then* $p \neq q$ *and q has exactly one outgoing transition, and such transition is labelled by an output label.*

A state of M is

- terminal, *if it has no outgoing transition; we define* $\mathsf{T}(M) = \{\, p \in M \mid p \text{ is terminal in } M \,\}$

- sending, *if it is not terminal and all its outgoing transitions have output labels*
- receiving, *if it is not terminal and all its outgoing transitions have input labels*
- mixed, *if it has a silent outgoing transition and an outgoing transition with an input label.*

A CFSM is A-local *if all its non* τ *transitions have subject* A.

Unlike in [5], the transitions of our CFSMs can also be labelled by the silent action τ. Definition 2.1 can be looked at as the CFSM counterpart of the τC contracts described in [7]. Imposing the no-mixed state condition on our CFSM, turns them into the communicating automata counterpart of the processes (contracts) called "session behaviours"[1] in e.g., [11, 1, 8]. These processes are in turn the process counterpart of (binary) session types [14]. As we shall see below (and also shown in [3] and [5]), the absence of mixed states is necessary in order to get the preservation of properties by composition. As a matter of fact, we could drop the conditions related to τ-transitions in case a transition like $p \xrightarrow{\mathsf{XY!z}} q$ is the only outgoing transition from p, namely when no actual choice of output actions is possible in p. We however prefer to avoid this distinction because (*i*) our uniform treatment of transitions allows us to immediately adapt definitions in a more abstract setting and (*ii*) secondly, uniformity allows us to simplify some technicalities. Said that, all proofs in the present paper could easily be adapted to the above mentioned alternative definition of CFSM.

Definition 2.2 (Communicating systems)**.** *A* (communicating) system over $\mathscr{P}$ *is a map* $\mathsf{S} = (M_{\mathsf{A}})_{\mathsf{A}\in\mathscr{P}}$ *assigning an* A*-local CFSM* M_{A} *to each participant* $\mathsf{A} \in \mathscr{P}$ *where* $\mathscr{P} \subseteq \mathfrak{P}$ *is finite and any participant occurring in a transition of* M_{A} *is in* $\mathscr{P}$.

Note that Definition 2.2 requires that any input or output label does refer to participants belonging to the system itself. In other words, Definition 2.2 restricts to *closed* systems.

We define the synchronous semantics of communicating systems as an FSA (differently from the asynchronous case, where the set of states can be infinite). Hereafter, $\mathrm{dom}(f)$ denotes the domain of a function f and $f[x \mapsto y]$ denotes the update of f in a point $x \in \mathrm{dom}(f)$ with the value y.

Definition 2.3 (Asymmetric synchronisations)**.** *Let* S *be a communicating system. A* configuration of S *is a map* $s = (q_{\mathsf{A}})_{\mathsf{A}\in\mathrm{dom}(\mathsf{S})}$ *assigning a* local state $q_{\mathsf{A}} \in \mathsf{S}(\mathsf{A})$ *to each* $\mathsf{A} \in \mathrm{dom}(\mathsf{S})$.

The asymmetric synchronisations *of* S *is the FSA* $[\![\mathsf{S}]\!] = \langle \mathscr{S}, s_0, \mathscr{L}_{int} \cup \{\tau\}, \to \rangle$ *where*

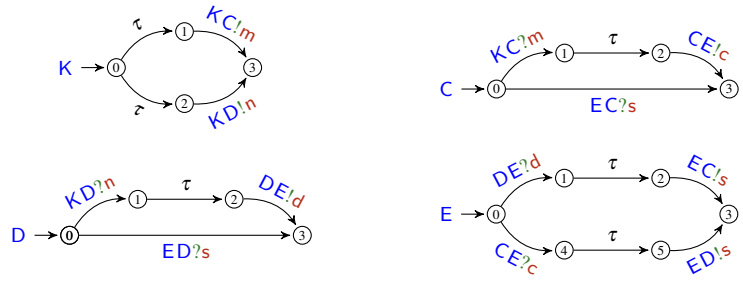
- $\mathscr{S}$ *is the set of synchronous configurations of* S*, as defined above;*
- $s_0 = (q_{0\mathsf{A}})_{\mathsf{A}\in\mathrm{dom}(\mathsf{S})} \in \mathscr{S}$ *is the* initial *configuration where, for each* $\mathsf{A} \in \mathrm{dom}(\mathsf{S})$, $q_{0\mathsf{A}}$ *is the initial state of* $\mathsf{S}(\mathsf{A})$*;*
- $\mathscr{L}_{int} = \{\mathsf{A}{\to}\mathsf{B}\colon \mathsf{m} \mid \mathsf{A} \neq \mathsf{B} \in \mathfrak{P} \text{ and } \mathsf{m} \in \mathscr{M}\}$ *is a set of interaction labels;*
- $s \xrightarrow{\mathsf{A}\to\mathsf{B}\colon \mathsf{m}} s[\mathsf{A} \mapsto q, \mathsf{B} \mapsto q'] \in [\![\mathsf{S}]\!]$ *if* $s(\mathsf{A}) \xrightarrow{\mathsf{AB!m}} q \in \mathsf{S}(\mathsf{A})$ *and* $s(\mathsf{B}) \xrightarrow{\mathsf{AB?m}} q' \in \mathsf{S}(\mathsf{B})$*;*
- $s \xrightarrow{\tau} s[\mathsf{A} \mapsto q] \in [\![\mathsf{S}]\!]$ *if* $s(\mathsf{A}) \xrightarrow{\tau} q \in \mathsf{S}(\mathsf{A})$*;*

Configuration s enables A in S *if* $s(\mathsf{A})$ *has at least an outgoing transition.*

As expected, an interaction $\mathsf{A}{\to}\mathsf{B}\colon \mathsf{m}$ occurs when A performs an output $\mathsf{AB!m}$ (which has been previously chosen) and B the corresponding input $\mathsf{AB?m}$.

[1] Actually different variations of this name are used in the listed references.

Example 2.4. *Let us consider the communicating system* $\mathsf{S} = (M_{\mathsf{X}})_{\mathsf{X}\in\{\mathsf{K},\mathsf{C},\mathsf{D},\mathsf{E}\}}$*, where*



A sequence of transitions of $[\![\mathsf{S}]\!]$ *out of* s_0 *is, according to Definition 2.3,*

$$\begin{array}{lll} s_0 = (0_{\mathsf{K}},0_{\mathsf{C}},0_{\mathsf{D}},0_{\mathsf{E}}) & \xrightarrow{\tau} & (1_{\mathsf{K}},0_{\mathsf{C}},0_{\mathsf{D}},0_{\mathsf{E}}) \xrightarrow{\mathsf{K}\to\mathsf{C}:\,\mathsf{m}} (3_{\mathsf{K}},1_{\mathsf{C}},0_{\mathsf{D}},0_{\mathsf{E}}) \\ & \xrightarrow{\tau} & (3_{\mathsf{K}},2_{\mathsf{C}},0_{\mathsf{D}},0_{\mathsf{E}}) \xrightarrow{\mathsf{C}\to\mathsf{E}:\,\mathsf{c}} (3_{\mathsf{K}},3_{\mathsf{C}},0_{\mathsf{D}},4_{\mathsf{E}}) \\ & \xrightarrow{\tau} & (3_{\mathsf{K}},3_{\mathsf{C}},0_{\mathsf{D}},5_{\mathsf{E}}) \xrightarrow{\mathsf{E}\to\mathsf{D}:\,\mathsf{s}} (3_{\mathsf{K}},3_{\mathsf{C}},3_{\mathsf{D}},3_{\mathsf{E}}) \end{array} \qquad \diamond$$

The symmetric synchronisation in [5] for systems without τ-transitions can be readily obtained from the above definition by disregarding the clause for the τ-transitions.

In the following, $\mathsf{ptp}(\tau) = \emptyset$ and $\mathsf{ptp}(\mathsf{A}\to\mathsf{B}\colon \mathsf{m}) = \mathsf{ptp}(\mathsf{A}\,\mathsf{B}!\mathsf{m}) = \mathsf{ptp}(\mathsf{A}\,\mathsf{B}?\mathsf{m}) = \{\mathsf{A},\mathsf{B}\}$ and, for a sequence $\pi = \lambda_1\cdot\cdots\cdot\lambda_n$, we let $\mathsf{ptp}(\pi) = \cup_{1\le i\le n}\mathsf{ptp}(\lambda_i)$.

As discussed in Section 1, we shall study the preservation of communication properties under composition. We shall consider the following relevant properties: deadlock freedom, lock freedom and strong lock freedom. The definitions below adapt the ones in [13] to a synchronous setting (as done also in [16, 19, 5]).

Definition 2.5 (Communication properties)**.** *Let* S *be a communicating system on* $\mathscr{P}$*. We say that a participant* $\mathsf{A}\in\mathscr{P}$ *is* involved *in a run* $s \xrightarrow{\lambda_1} s_1 \ldots \xrightarrow{\lambda_n} s_n$ *of* S *if there is* $1\le i\le n$ *such that either* $\mathsf{A}\in \mathit{ptp}(\lambda_i)$ *or* $\lambda_i = \tau$*,* $s_i(\mathsf{A}) \xrightarrow{\tau} q$ *in* $\mathsf{S}(\mathsf{A})$*, and* $s_{i+1} = s_i[\mathsf{A}\mapsto q]$*.*

Deadlock freedom *A configuration* $s\in\mathscr{R}([\![\mathsf{S}]\!])$ *is a* deadlock *if*

- *s has no outgoing transitions in* $[\![\mathsf{S}]\!]$ *and*
- *there exists* $\mathsf{A}\in\mathscr{P}$ *such that* $s(\mathsf{A})$ *enables* A *in* S*.*

A system is deadlock-free *if none of its configurations is a deadlock.*

Lock freedom *Let* $\mathsf{A}\in\mathscr{P}$*. A configuration* $s\in\mathscr{R}([\![\mathsf{S}]\!])$ *is a* lock *for* A *if*

- $s(\mathsf{A})$ *has outgoing transitions; and*
- A *is not involved in any run from s.*

A system is lock-free *if none of its configurations is a lock for any of its participants.*

Strong lock freedom *System* S *is* strongly lock-free for $\mathsf{A}\in\mathscr{P}$ *if for each* $s\in\mathscr{R}([\![\mathsf{S}]\!])$ *enabling* A *in* S *then* A *is involved in all maximal sequences from s.*
A system is strongly lock free *if it is strongly lock free for each of its participants.*

Proposition 2.6. *1. Lock-freedom implies deadlock-freedom;*

2. Strong lock freedom implies lock freedom.

Example 2.7. *Let us consider the system* S *of Example 2.4. The only other maximal transition sequence in* $[\![\mathsf{S}]\!]$ *out of* s_0*, besides the one described in Example 2.4, is*

$$\begin{array}{rcl} s_0 = (0_\mathsf{K}, 0_\mathsf{C}, 0_\mathsf{D}, 0_\mathsf{E}) & \xrightarrow{\tau} & (2_\mathsf{K}, 0_\mathsf{C}, 0_\mathsf{D}, 0_\mathsf{E}) \xrightarrow{\mathsf{K}\to\mathsf{D}:\mathsf{n}} (3_\mathsf{K}, 0_\mathsf{C}, 1_\mathsf{D}, 0_\mathsf{E}) \\ & \xrightarrow{\tau} & (3_\mathsf{K}, 0_\mathsf{C}, 2_\mathsf{D}, 0_\mathsf{E}) \xrightarrow{\mathsf{D}\to\mathsf{E}:\mathsf{d}} (3_\mathsf{K}, 0_\mathsf{C}, 3_\mathsf{D}, 1_\mathsf{E}) \\ & \xrightarrow{\tau} & (3_\mathsf{K}, 0_\mathsf{C}, 3_\mathsf{D}, 2_\mathsf{E}) \xrightarrow{\mathsf{E}\to\mathsf{C}:\mathsf{s}} (3_\mathsf{K}, 3_\mathsf{C}, 3_\mathsf{D}, 3_\mathsf{E}) \end{array}$$

These two sequences are both maximal and contain all the elements of $\mathscr{R}([\![\mathsf{S}]\!])$*. By the above observations it is possible to check* S *to be strongly lock free.* ◇

3 Composition via Gateways

We now discuss the composition of systems of CFSMs via gateways (cf. [2, 3]) and study its properties under asymmetric synchronisation. The main idea is that two systems of CFSMs, say S_1 and S_2, can be composed by transforming one participant in each of them into gateways connected to each other.

3.1 Building gateways

Hereafter, H and K denote the selected participant in S_1 and S_2 respectively selected for the composition. The gateways for H and K are connected to each other and act as forwarders: each message sent to the gateway for H by a participant from the original system S_1 is now forwarded to the gateway for K, that in turn forwards it to the same participant to which K sent it in the original system S_2. The dual will happen to messages that the gateway for K receives from S_2. A main advantage of this approach is that no extension of the CFSM model is needed to transform systems of CFSMs, which are normally closed systems, into open systems that can be composed. Another advantage is that the composition is fully transparent to all participants different from H and K.

We will now define composition via gateways on systems of CFSMs, following the intuition above.

Definition 3.1 (Gateway)**.** *Given a* H*-local CFSM* M *and a participant* K*, the* gateway of M towards K *is the CFSM* $\mathsf{gw}(M,\mathsf{K})$ *obtained by replacing in* M

- *each pair of consecutive transitions* $p \xrightarrow{\tau} q \xrightarrow{\mathsf{H}\mathsf{A}!\mathsf{m}} r$ *with*

$$p \xrightarrow{\mathsf{K}\mathsf{H}?\mathsf{m}} p' \xrightarrow{\tau} q \xrightarrow{\mathsf{H}\mathsf{A}!\mathsf{m}} r \qquad \text{for some fresh state } p' \tag{6}$$

- *each transition* $p \xrightarrow{\mathsf{A}\mathsf{H}?\mathsf{m}} r$ *with*

$$p \xrightarrow{\mathsf{A}\mathsf{H}?\mathsf{m}} p' \xrightarrow{\tau} p'' \xrightarrow{\mathsf{H}\mathsf{K}!\mathsf{m}} r \qquad \text{for some fresh states } p' \text{ and } p'' \tag{7}$$

We shall call external *the states like* p *and* r *and* internal *the states like* p'*,* p'' *and* q*.*

Note that gateways execute "segments" of the form described in (6) and (7) in the above definition. Also, by very construction, we have the following

Fact 3.2. *Given a* H*-local CFSM* M *and a participant* K*, each state of* $\mathsf{gw}(M,\mathsf{K})$ *has at most one incoming or outgoing* τ *transition.*

We compose systems with disjoint participants taking all the participants of the original systems but H and K, whereas H and K are replaced by their respective gateways.

Given two functions f and g such that $\mathrm{dom}(f) \cap \mathrm{dom}(g) = \emptyset$, we let $f+g$ denote the function behaving as function f on $\mathrm{dom}(f)$ and as function g on $\mathrm{dom}(g)$.

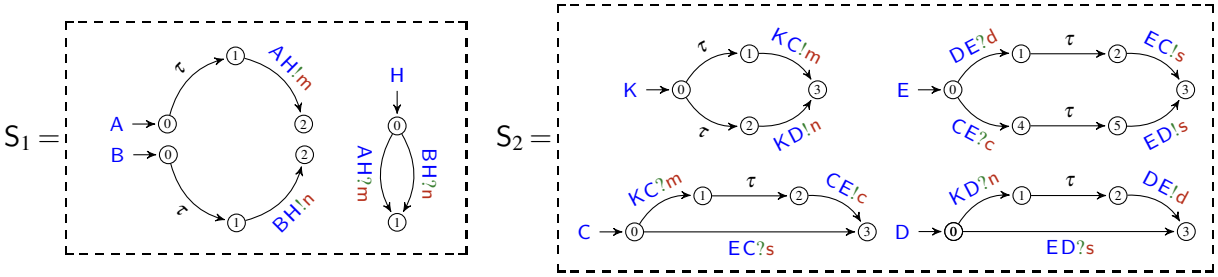
Definition 3.3 (System composition)**.** *Let* S_1 *and* S_2 *be two systems with disjoint domains. The* composition of S_1 and S_2 via $\mathsf{H} \in \mathrm{dom}(\mathsf{S}_1)$ and $\mathsf{K} \in \mathrm{dom}(\mathsf{S}_2)$ *is defined as*

$$\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}}\, \mathsf{S}_2 = \mathsf{S}_1[\mathsf{H} \mapsto \mathsf{gw}(\mathsf{S}_1(\mathsf{H}), \mathsf{K})] + \mathsf{S}_2[\mathsf{K} \mapsto \mathsf{gw}(\mathsf{S}_2(\mathsf{K}), \mathsf{H})]$$

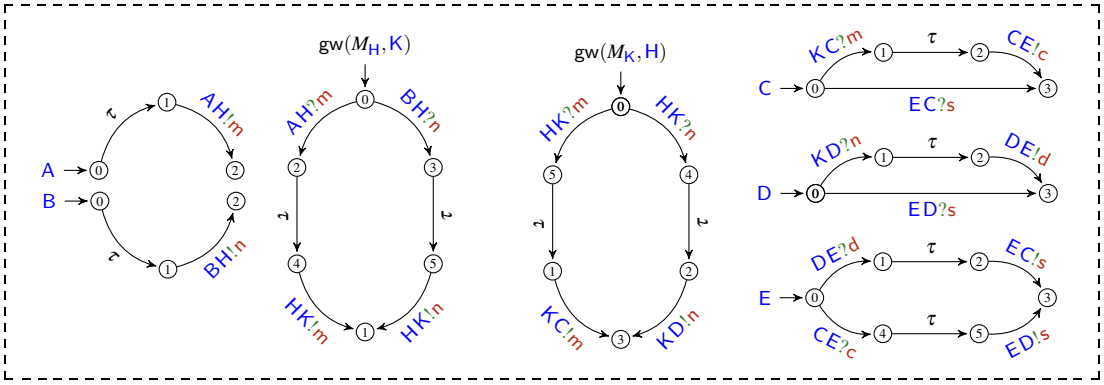
(Note that $\mathrm{dom}(\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}}\, \mathsf{S}_2) = \mathrm{dom}(\mathsf{S}_1) \cup \mathrm{dom}(\mathsf{S}_2)$*.)*

We remark that, by the above approach for composition, we do not actually need to formalise the notion of *open* system. In fact any closed system can be looked at as open by choosing two suitable participants in the "to-be-connected" systems and transforming them into two forwarders. Also, note that the notion of composition above is structural: corresponding notions of behaviourals composition have been studied in [4] and in [15] for multiparty session types.

Example 3.4. *Let us take the following two communicating systems.*



The system $\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}}\, \mathsf{S}_2$ *is*



Note that the CFSMs A*,* B*,* C*,* D*, and* E *remain unchanged.* ◇

3.2 Compatibility

A few simple auxiliary notions are useful. Let $\mathscr{L}_{\mathrm{i/o}} = \{\, ?\mathsf{m}, !\mathsf{m} \mid \mathsf{m} \in \mathscr{M} \,\}$ and define the functions $\mathsf{io} : \mathscr{L}_{\mathrm{act}} \to \mathscr{L}_{\mathrm{i/o}}$ and $\overline{(\cdot)} : \mathscr{L}_{\mathrm{i/o}} \to \mathscr{L}_{\mathrm{i/o}}$ as follows

$$\mathsf{io}(\mathsf{A}\,\mathsf{B}?\mathsf{m}) = ?\mathsf{m} \qquad \mathsf{io}(\mathsf{A}\,\mathsf{B}!\mathsf{m}) = !\mathsf{m} \qquad \text{and} \qquad \overline{?\mathsf{m}} = !\mathsf{m} \qquad \overline{!\mathsf{m}} = ?\mathsf{m}$$

These functions extend to CFSMs in the obvious way: given a CFSM $M = \langle \mathscr{S}, q_0, \mathscr{L}_{\mathrm{act}}, \to \rangle$ we define $\mathsf{io}(M) = \langle \mathscr{S}, q_0, \mathscr{L}_{\mathrm{i/o}}, \to' \rangle$ where $\to' = \{\, q \xrightarrow{\mathsf{io}(\lambda)} q' \mid q \xrightarrow{\lambda} q' \in M, \lambda \in \mathscr{L}_{\mathrm{act}} \,\} \cup \{\, q \xrightarrow{\tau} q' \mid q \xrightarrow{\tau} q' \in M \,\}$ and likewise for $\overline{M}$.

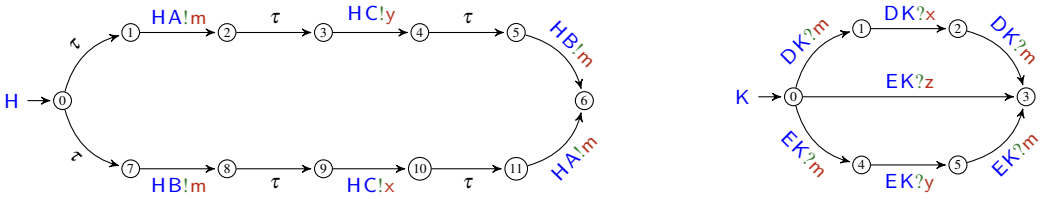
Informally, two CFSMs M_1 and M_2 are *compatible* if each output of M_1 has a corresponding input in M_2 and vice versa once the identities of communicating partners are blurred away.

Definition 3.5 (Compatibility)**.** *Let M and M' be two FSAs on $\mathscr{L}_{i/o}$. An* io-correspondence *is a relation R between states of M and those of M' such that whenever $(q,q') \in R$:*

- *$q \in \mathsf{T}(M)$ if, and only if, $q' \in \mathsf{T}(M')$ (cf. Definition 2.1)*
- *if $q \xrightarrow{!\mathsf{m}} r \in M$ then there is $q' \xrightarrow{?\mathsf{m}} r' \in M'$ such that $(r,r') \in R$*
- *if $q' \xrightarrow{!\mathsf{m}} r' \in M'$ then there is $q \xrightarrow{?\mathsf{m}} r \in M$ such that $(r,r') \in R$*
- *if $q \xrightarrow{\tau} r \in M$ then $(r,q') \in R$*
- *if $q' \xrightarrow{\tau} r' \in M'$ then $(q,r') \in R$*

Two CFSMs M and M' are compatible *(in symbols $M \asymp M'$) if there is an io-correspondence relating the initial states of $\mathsf{io}(M_1)$ and $\mathsf{io}(M')$.*

Example 3.6 (Compatibility)**.** *The machines* H *and* K *of Example 3.4 are compatible. For a more complex example, consider the following CFSMs*



The above H *and* K *are compatible. Apart for τ actions preceding them,* H *can only perform output actions, whereas* K *can only perform input actions. By disregarding the names of the receivers in the actions of* H*, and of the senders in those of* K*, any output action after its corresponding τ can find a matching input in* K*. The vice versa does not hold, since none of the possible output actions that can occur after a τ from* 0 *(i.e. the outputs from* 1 *and* 7 *in* H*) can actually match the input action* EK?z *from* 0 *in* K*. Such a possibility is in fact allowed by our definition of compatibility.* ◇

Definition 3.5 transfers the notion of compatibility given in [4] for processes in multiparty sessions. Also, Definition 3.5 differs from the notions of compatibility in [5] and in [2, 3] which are defined as bisimulations and do not involve τ-transitions.

Definition 3.7. *An* A*-local CFSM M is:*

1. ?-deterministic *if $p \xrightarrow{\mathsf{XA?m}} q$ and $p \xrightarrow{\mathsf{YA?m}} r \in M$ implies $q = r$;*
2. !-deterministic *if $p \xrightarrow{\tau} \xrightarrow{\mathsf{AX!m}} q$ and $p \xrightarrow{\tau} \xrightarrow{\mathsf{AY!m}} r \in M$ implies $q = r$;*
3. ?!-deterministic *if it is both ?-deterministic and !-deterministic;*

A non-terminal state $q \in M$ is *asymmetric sending* (resp. *receiving*) if all its outgoing transitions have τ (resp. receiving) labels; q is a *asymmetric mixed* state if it is neither asymmetric sending nor receiving.

Example 3.8. *Machines* H *and* K *in Example 3.6 are, respectively non !-deterministic and non ?-deterministic. In particular, conditions (2) and (1) of Definition 3.7 fail for, respectively, state* 0 *of* H *and state* 0 *of* K*.*

We require a stronger condition then compatibity for two systems to be composable.

Definition 3.9 ((H,K)-composability)**.** *Let S_1 and S_2 be two systems with disjoint domains. We say that S_1 and S_2 are* (H,K)-composable *if $\mathsf{H} \in \mathrm{dom}(\mathsf{S}_1)$ and $\mathsf{K} \in \mathrm{dom}(\mathsf{S}_2)$ are two compatible ?!-deterministic machines with no asymmetric mixed states.*

4 Composition Related Issues

It is known that under symmetric synchronisation composition spoils deadlock-freedom; this is shown by the example below, borrowed from [5].

Example 4.1 (Symmetric synchronisation spoils deadlock-freedom preservation)**.** *Take the systems*

Clearly, S_1 *and* S_2 *are* (H,K)*-composable and deadlock-free, yet their composition* $\mathsf{S} = \mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2$ *has a deadlock. In fact, when the gateway for* K *receives* m*, it tries to synchronise with participant* B *on message* m *while* B *is waiting only for* x*. For* S_2 *in isolation, this is not a deadlock, since* B *and* K *synchronise on* x *under the symmetric semantics.* ◇

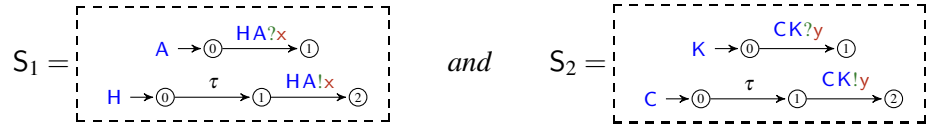
Notice that the counterexample of Example 4.1 does not apply in an asynchronous setting. Indeed, S_2 could deadlock due to the fact that K could send m without synchronising with B. Likewise, the counterexample of Example 4.1 does not apply in our asymmetric setting. Even if communication is still synchronous, the τ-transitions introduced to resolve internal choices (i.e., those prefixing outputs) allow S_2 to reach a deadlock configuration by choosing the τ-transition leading to the output $\mathsf{K}\,\mathsf{B}!\mathsf{m}$.

Now, one may think that analogously to what happens in [2, 3], if two systems are (H,K)-composable and deadlock-free then their composition is deadlock-free too.

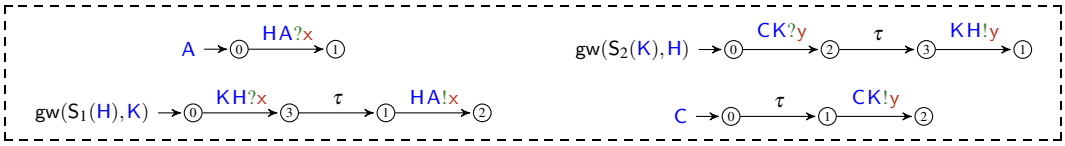
In Section 5 we shall prove that in our setting lock-freedom is preserved by composition, without any further condition beside (H,K)-composability. Before doing that, we give examples showing the necessity of our conditions for deadlock freedom preservation.

Let us begin with compatibility. Properties cannot be preserved under composition without compatibility, as shown in the next example.

Example 4.2 (Lack of compatibility spoils deadlock freedom preservation)**.** *The systems*



are trivially deadlock free. However, H *and* K *are not compatible, since there is no corresponding input in* K *for the output from* H*. The composition of* S_1 *and* S_2 *via* H *and* K *yields*



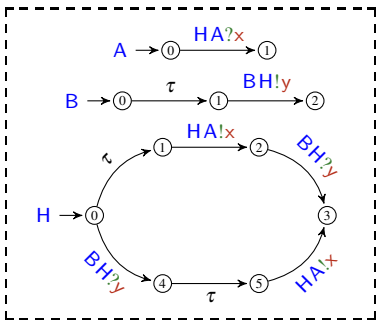
Starting from the initial configuration of $\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2$*, the following transitions are possible in* $[\![\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2]\!]$

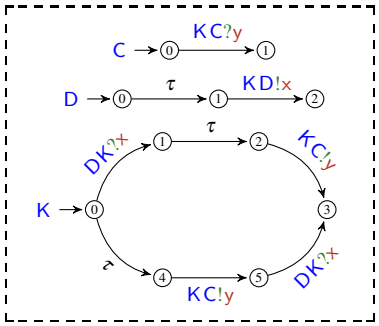
$$(0_\mathsf{A},0_\mathsf{H},0_\mathsf{K},0_\mathsf{C}) \xrightarrow{\tau} (0_\mathsf{A},0_\mathsf{H},0_\mathsf{K},1_\mathsf{C}) \xrightarrow{\mathsf{C}\to\mathsf{K}:\,\mathsf{y}} (0_\mathsf{A},0_\mathsf{H},2_\mathsf{K},2_\mathsf{C}) \xrightarrow{\tau} (0_\mathsf{A},0_\mathsf{H},3_\mathsf{K},2_\mathsf{C}) \not\to$$

where $(0_\mathsf{A},0_\mathsf{H},3_\mathsf{K},2_\mathsf{C})$ *is a deadlock configuration for* $[\![\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2]\!]$ *since* K *wishes to send* y *to* H*, which is instead waiting for message* x*.* ◇

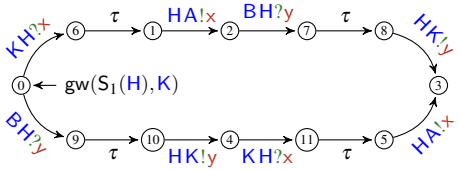
The following example, casting in our setting an example given in [3] for the asynchronous semantics, illustrates that asymmetric mixed states must be avoided to preserve properties under composition.

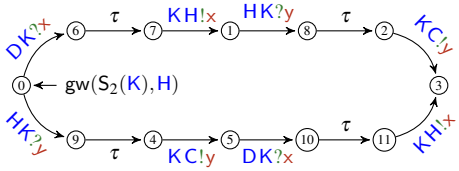
Example 4.3 (Asymmetric mixed-states spoil deadlock freedom preservation)**.** *Let* S_1 *and* S_2 *be*

$\mathsf{S}_1 =$ 

$\mathsf{S}_2 =$ 

Noticeably, the initial states are asymmetric mixed and H *and* K *are compatible. The gateways are*

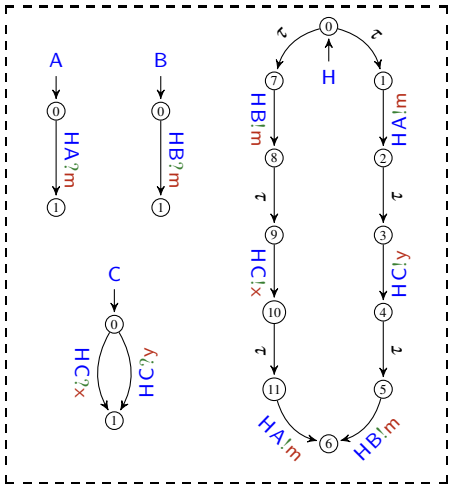




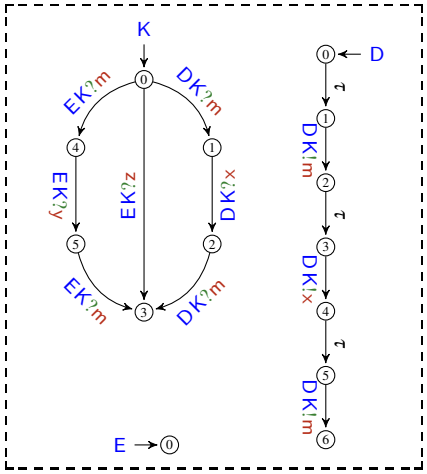
The composed system $\mathsf{S}_1 \mathsf{H}{\leftrightarrow}\mathsf{K} \mathsf{S}_2$ *deadlocks when* $\mathsf{gw}(\mathsf{S}_1(\mathsf{H}),\mathsf{K})$ *receives from* B *while* $\mathsf{gw}(\mathsf{S}_2(\mathsf{K}),\mathsf{H})$ *receives from* D *since both gateways reach an output state (respectively states* 10 *and* 7*).* ◇

The following example shows that!?-nondeterminism is problematic too.

Example 4.4. *Consider the two deadlock-free systems*

$\mathsf{S}_1 =$ 

and

$\mathsf{S}_2 =$ 

The deadlock freedom of S_1 *follows from the fact that, from its initial configuration* $(0_\mathsf{A},0_\mathsf{B},0_\mathsf{C},0_\mathsf{H})$*,* S_1 *can only branch over the two* τ*-transitions of* H *reaching either of the following configurations* $(0_\mathsf{A},0_\mathsf{B},0_\mathsf{C},7_\mathsf{H})$ *or* $(0_\mathsf{A},0_\mathsf{B},0_\mathsf{C},1_\mathsf{H})$*. From the former (resp. latter) configuration* S_1 *can only reach configurations where* H *synchronises with* C *and then with* A *(resp.* B*). In either case* S_1 *reaches the*

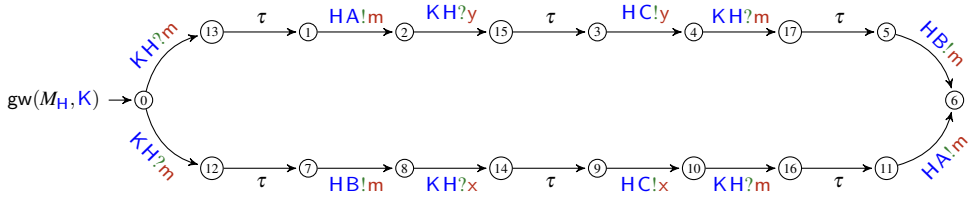
terminal configuration $(1_A, 1_B, 1_C, 6_H)$. *Let us take a look at* S_2. *The only possible transitions of* S_2 *can involve* K *and* D *only since* E *cannot synchronise being it terminated. We therefore have that*

$$(0_K,0_D,0_E) \xrightarrow{\tau} (0_K,1_D,0_E) \xrightarrow{D\to K:\,m} (1_K,2_D,0_E) \xrightarrow{\tau} (1_K,3_D,0_E) \xrightarrow{D\to K:\,x} (2_K,4_D,0_E)$$
$$\xrightarrow{\tau} (2_K,5_D,0_E) \xrightarrow{D\to K:\,m} (3_K,6_D,0_E)$$

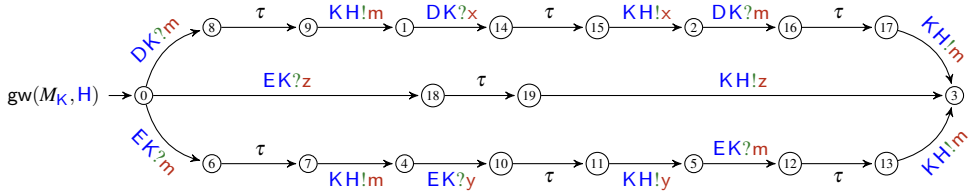
is the only possible execution from the initial configuration $(0_K,0_D,0_E)$ *of* S_2, *leading to the terminal configuration* $(3_K,6_D,0_E)$. ◇

The next example shows that the compositions of the systems S_1 and S_2 in Example 4.4 can deadlock.

Example 4.5 (?!-determinism is necessary)**.** *The CFSMs* H *and* K *in Example 4.4 are compatible as seen in Example 3.6. Hence, we can build the composed system* $S_1 {}^{H\leftrightarrow K} S_2$ *through the gateways*



and



Now, from the initial configuration $s_0 = (0_A,0_B,0_C,0_{gw(M_H,K)},0_{gw(M_K,H)},0_D,0_E)$ *of* $S_1 {}^{H\leftrightarrow K} S_2$ *we have the following run*

$$\begin{aligned}
s_0 &\xrightarrow{\tau} (0_A,0_B,0_C,0_{gw(M_H,K)},0_{gw(M_K,H)},1_D,0_E)\\
&\xrightarrow{D\to K:\,m} \xrightarrow{\tau} (0_A,0_B,0_C,0_{gw(M_H,K)},9_{gw(M_K,H)},2_D,0_E)\\
&\xrightarrow{K\to H:\,m} \xrightarrow{\tau} (0_A,0_B,0_C,13_{gw(M_H,K)},1_{gw(M_K,H)},3_D,0_E) \qquad (8)\\
&\xrightarrow{D\to K:\,x} (0_A,0_B,0_C,13_{gw(M_H,K)},14_{gw(M_K,H)},4_D,0_E)\\
&\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau} (0_A,0_B,0_C,1_{gw(M_H,K)},15_{gw(M_K,H)},5_D,0_E)\\
&\xrightarrow{H\to A:\,m} (1_A,0_B,0_C,2_{gw(M_H,K)},15_{gw(M_K,H)},5_D,0_E) \qquad (9)
\end{aligned}$$

where the τ*-transition of* D *enables the synchronisation of* $gw(M_K,H)$ *and* D *with label* $D\to K:\,m$ *that leads the gateway in state* 9 *after its* τ*-transition from state* 8. *Now, the two gateways can communicate and exchange message* m. *Due to* ?!*-nondeterminism of* S_1, *from state* 0 $gw(M_H,K)$ *can move either to state* 12 *or to state* 13. *Fatally, transition* (8) *leads to a deadlock: after* $gw(M_K,H)$ *and* D *synchronise to exchange message* x *the system goes into a configuration from where* $gw(M_H,K)$ *forwards* m *to* A *and reaches the last configuration* (9). *This is a deadlock for* $S_1 {}^{H\leftrightarrow K} S_2$, *since no CFSM can do a* τ*-transitions, the only enabled output action is from* $gw(M_K,H)$ *which tries to send message* x *to* $gw(M_H,K)$*; however,* $gw(M_H,K)$ *can only receive message* y *from* K *and hence these actions cannot synchronise.* ◇

5 Preserving Properties by Composition

Composition via gateways does not ensure the preservation of communication properties. We provide below sufficient conditions for this to happen. Recall that (H,K)-composability requires absence of asymmetric mixed states and ?!-determinism.

Theorem 5.1 (Deadlock freedom preservation)**.** *Let* S_1 *and* S_2 *be two* (H,K)*-composable and deadlock-free systems. Then the composed system* $\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2$ *is deadlock-free.*

Proof sketch. The proof relies on the fact that the reachable configurations of $\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2$ can be projected on reachable configurations of S_1 and S_2. This implies that a deadlock in $\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2$ corresponds to a deadlock in S_1 or in S_2. See the appendix for the detailed proof. □

Example 5.2. *We can infer deadlock-freedom of the system* $\mathsf{S} = \mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2$ *of Example 3.4 by the result above, since* S_1 *and* S_1 *are* (H,K)*-composable and deadlock-free.*

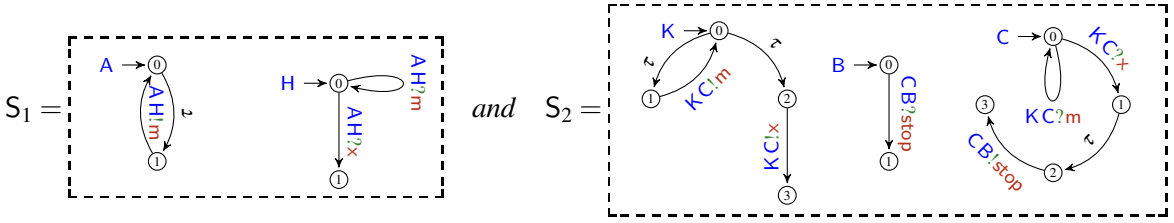
Somehow surprisingly, in the symmetric case preservation of deadlock freedom requires stricter conditions on gateways than in the asymmetric case. In fact, in the asymmetric case, deadlock freedom preservation requires only absence of asymmetric mixed states and ?!-determinism while the symmetric case requires the (stronger) condition of *sequentiality*.

Definition 5.3 (Sequential CFSM)**.** *A CFSM is* sequential *if each of its states has at most one outgoing transition. A participant* A *of a system* S *is sequential if* $\mathsf{S}(\mathsf{A})$ *is so.*

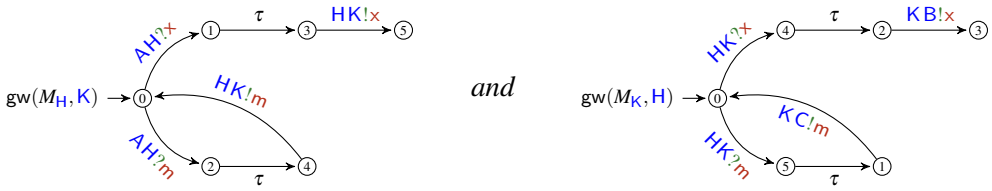
As we will see (cf. Theorem 5.5), sequentiality is necessary to preserve lock-freedom also in the asymmetric case. We note that sequentiality implies absence of asymmetric mixed states and ?!-determinism, while the converse does not hold.

As mentioned before, the property of lock freedom is not preserved in general by composition, as shown by the following example.

Example 5.4 (Composability does not preserve lock-freedom)**.** *Take the communicating systems*



Note that both S_1 *and* S_2 *are lock-free and that* H *and* K *are compatible. The gateways are*



Hence, the composed system $\mathsf{S}_1 {}^{\mathsf{H}\leftrightarrow\mathsf{K}} \mathsf{S}_2$ *is non lock-free because e.g. the configuration*

$$s = (0_{\mathsf{A}}, 0_{\mathsf{gw}(M_{\mathsf{H}},\mathsf{K})}, 0_{\mathsf{gw}(M_{\mathsf{K}},\mathsf{H})}, 0_{\mathsf{B}}, 0_{\mathsf{C}})$$

is a lock for B*, since the only outgoing transition from* 0_{B} *could be fired only in case the transition* CB!stop *is enabled. However, this is impossible since* $\mathsf{gw}(M_{\mathsf{H}},\mathsf{K})$ *forwards only message* m*; hence, the run* $s \xrightarrow{\tau} \xrightarrow{\mathsf{A}\to\mathsf{H}:\,\mathsf{m}} (0_{\mathsf{A}}, 2_{\mathsf{gw}(M_{\mathsf{H}},\mathsf{K})}, 0_{\mathsf{gw}(M_{\mathsf{K}},\mathsf{H})}, 0_{\mathsf{B}}, 0_{\mathsf{C}}) \xrightarrow{\tau} \xrightarrow{\mathsf{A}\to\mathsf{H}:\,\mathsf{m}} s\cdots$ *(which does not involve* B*) is perpetually executed.* ◇

We show that the problem of Example 5.4 cannot happen in case we restrict to sequential gateways, as done for deadlock freedom in the symmetric case (cf. [5]). As usual, $f\,|_X$ denotes the restriction of a function f on a subset X of its domain.

Theorem 5.5 (Lock-freedom preservation)**.** *Let* S_1 *and* S_2 *be two* (H,K)*-composable and lock-free systems with* H *and* K *sequential. Then the composed system* $\mathsf{S}_1{}^{\mathsf{H}\leftrightarrow\mathsf{K}}\,\mathsf{S}_2$ *is lock-free.*

Proof sketch. The proof goes as the one of Theorem 5.1 noticing that we have to reconstruct "backward" the sequence of interactions. This exploits sequentiality and lock-freedom of S_1 and S_2 in order to guarantee the reconstruction when we "cross" the two composed systems through the gateways. □

We turn now our attention to strong lock-freedom. In this case, as for deadlock freedom, (H,K)-composability suffices for preservation by composition; we shall see that this is not the case for lock freedom preservation.

Theorem 5.6 (Strong lock freedom preservation)**.** *Let* S_1 *and* S_2 *be two* (H,K)*-composable and strongly lock free systems. Then the composed system* $\mathsf{S}_1{}^{\mathsf{H}\leftrightarrow\mathsf{K}}\,\mathsf{S}_2$ *is strongly lock free.*

Proof sketch. The proof is similar to the one of Theorem 5.5 but for the use of strong lock freedom of S_1 and S_2 instead of their deadlock freedom. □

6 Conclusions and Future Work

We introduce an asymmetric synchronous semantics of communicating systems which breaks the symmetry between senders and receivers. In fact, our semantics decouples communication from choice resolution as in standard semantics of communicating systems (and other models). We then adapted the gateway composition mechanism defined in [2, 3] to our asymmetric semantics and gave conditions for the preservation of some communication properties under this notion of composition.

In *contract automata* [10, 9] transitions express "requests" and "offers" among participants. The composition mechanism is based on "trimming" a product of contract automata according to relevant *agreement* properties. This yields controllers that preserve deadlocks. Contract automata do not consider asymmetric synchronous semantics. Our composition mechanism does not introduce orchestrators which, under some conditions, can be avoided also for contract automata [10, 9].

Modular approaches to the development of concurrent systems can be exploited even for systems designed using formalisms intrinsically dealing with *closed* systems. Indeed, given two systems, any two components – one per system – exhibiting *compatible* behaviours can be replaced by two coupled forwarders (gateways) connecting the systems, as investigated initially in [2, 3] for an asynchronous interaction model. The investigation on the composition-by-gateways technique was shifted in [5] towards synchronous symmetric interactions. We pushed a step forward such an investigation, by considering *asymmetric* synchronous interactions. Interestingly, deadlock freedom preservation in the synchronous asymmetric case we consider does not require sequentiality of gateways, like in the asynchronous case, and differently from the synchronous symmetric case. Notably, sequentiality is needed here for lock-freedom preservation, but not for strong-lock freedom preservation.

While the path of investigation above is quite homogeneous, the different analyses present some methodological differences. For instance, [5] considers also another form of composition, where one single gateway (interacting with both the composed systems) is used. On the other side, [5] focused only on deadlocks, disregarding other properties we consider. A first item of future research consist in filling the bits missing due to the mismatches above.

A more challenging direction for future work is looking for refined composition mechanisms in order to get preservation of relevant properties under weaker conditions.

References

[1] F. Barbanera & U. de'Liguoro (2015): *Sub-behaviour relations for session-based client/server systems*. *MSCS* 25(6), pp. 1339–1381, doi:10.1017/S096012951400005X.

[2] Franco Barbanera, Ugo de'Liguoro & Rolf Hennicker (2018): *Global Types for Open Systems*. In Massimo Bartoletti & Sophia Knight, editors: *ICE*, *EPTCS* 279, pp. 4–20, doi:10.4204/EPTCS.279.4.

[3] Franco Barbanera, Ugo de'Liguoro & Rolf Hennicker (2019): *Connecting open systems of communicating finite state machines*. *JLAMP* 109, doi:10.1016/j.jlamp.2019.07.004.

[4] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese & Emilio Tuosto (2021): *Composition and decomposition of multiparty sessions*. *JLAMP* 119, p. 100620, doi:10.1016/j.jlamp.2020.100620.

[5] Franco Barbanera, Ivan Lanese & Emilio Tuosto (2020): *Composing Communicating Systems, Synchronously*. In Tiziana Margaria & Bernhard Steffen, editors: *ISoLA 2020*, *LNCS* 12476, Springer, pp. 39–59, doi:10.1007/978-3-030-61362-4_3.

[6] Franco Barbanera, Ivan Lanese & Emilio Tuosto (2022): *On Composing Communicating Systems. Full Version*. Available at `https://emwww.github.io/home/slides/ice22TR.pdf`.

[7] Massimo Bartoletti, Tiziana Cimoli & Roberto Zunino (2015): *Compliance in Behavioural Contracts: A Brief Survey*. In Chiara Bodei, Gian Luigi Ferrari & Corrado Priami, editors: *Programming Languages with Applications to Biology and Security - Essays Dedicated to Pierpaolo Degano on the Occasion of His 65th Birthday*, *Lecture Notes in Computer Science* 9465, Springer, pp. 103–121, doi:10.1007/978-3-319-25527-9_9.

[8] Massimo Bartoletti, Alceste Scalas & Roberto Zunino (2014): *A Semantic Deconstruction of Session Types*. In Paolo Baldan & Daniele Gorla, editors: *CONCUR 2014*, *LNCS* 8704, Springer, pp. 402–418, doi:10.1007/978-3-662-44584-6_28.

[9] Davide Basile, Maurice H. ter Beek & Rosario Pugliese (2020): *Synthesis of Orchestrations and Choreographies: Bridging the Gap between Supervisory Control and Coordination of Services*. *LMCS* 16(2), doi:10.23638/LMCS-16(2:9)2020.

[10] Davide Basile, Pierpaolo Degano, Gian Luigi Ferrari & Emilio Tuosto (2016): *Playing with Our CAT and Communication-Centric Applications*. In Elvira Albert & Ivan Lanese, editors: *FORTE*, *LNCS* 9688, Springer, pp. 62–73, doi:10.1007/978-3-319-39570-8_5.

[11] G. Bernardi & M. Hennessy (2012): *Modelling session types using contracts*. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, ACM, New York, NY, USA, pp. 1941–1946, doi:10.1145/2231936.2232097.

[12] Daniel Brand & Pitro Zafiropulo (1983): *On Communicating Finite-State Machines*. *J. ACM* 30(2), pp. 323–342, doi:10.1145/322374.322380.

[13] Gérard Cécé & Alain Finkel (2005): *Verification of programs with half-duplex communication*. *I&C* 202(2), pp. 166–190, doi:10.1016/j.ic.2005.05.006.

[14] K. Honda, V. T. Vasconcelos & M. Kubo (1998): *Language Primitives and Type Disciplines for Structured Communication-based Programming*. In Chris Hankin, editor: *ESOP*, *LNCS* 1381, Springer, pp. 22–138, doi:10.1007/BFb0053567.

[15] H. Hüttel et al. (2016): *Foundations of Session Types and Behavioural Contracts*. *ACM Comput. Surv.* 49(1), pp. 3:1–3:36, doi:10.1145/2873052.

[16] Julien Lange, Emilio Tuosto & Nobuko Yoshida (2015): *From Communicating Machines to Graphical Choreographies*. In Sriram K. Rajamani & David Walker, editors: *POPL*, ACM, pp. 221–232, doi:10.1145/2676726.2676964.

[17] Robin Milner (1980): *A Calculus of Communicating Systems*. *LNCS* 92, Springer, Berlin, doi:10.1007/3-540-10235-3.

[18] L. Padovani (2010): *Contract-Based Discovery of Web Services Modulo Simple Orchestrators*. *Theoretical Computer Science* 411, pp. 3328–3347, doi:10.1016/j.tcs.2010.05.002.

[19] Emilio Tuosto & Roberto Guanciale (2018): *Semantics of global view of choreographies*. *JLAMP* 95, pp. 17–40, doi:10.1016/j.jlamp.2017.11.002.