Logical Methods in Computer Science
Vol. 14(1:1)2018, pp. 1–31
https://lmcs.episciences.org/

Submitted Nov. 14, 2016
Published Jan. 09, 2018

COMPATIBILITY PROPERTIES OF SYNCHRONOUSLY AND ASYNCHRONOUSLY COMMUNICATING COMPONENTS

ROLF HENNICKER AND MICHEL BIDOIT

Ludwig-Maximilians-Universität München, Germany
e-mail address: hennicker@ifi.lmu.de

LSV, CNRS and ENS de Cachan, France
e-mail address: bidoit@lsv.ens-cachan.fr

Abstract. We study interacting components and their compatibility with respect to synchronous and asynchronous composition. The behavior of components is formalized by I/O-transition systems. Synchronous composition is based on simultaneous execution of shared output and input actions of two components while asynchronous composition uses unbounded FIFO-buffers for message transfer. In both contexts we study compatibility notions based on the idea that any output issued by one component should be accepted as an input by the other. We distinguish between strong and weak versions of compatibility, the latter allowing the execution of internal actions before a message is accepted. We consider open systems and study conditions under which (strong/weak) *synchronous* compatibility is sufficient and necessary to get (strong/weak) *asynchronous* compatibility. We show that these conditions characterize half-duplex systems. Then we focus on the verification of weak asynchronous compatibility for possibly non half-duplex systems and provide a decidable criterion that ensures weak asynchronous compatibility. We investigate conditions under which this criterion is complete, i.e. if it is not satisfied then the asynchronous system is not weakly asynchronously compatible. Finally, we discuss deadlock-freeness and investigate relationships between deadlock-freeness in the synchronous and in the asynchronous case.

1. Introduction

Distributed systems consist of sets of components which are deployed on different nodes and communicate through certain media. In this work we consider reactive components with a well defined behavior which communicate by message exchange. Each single component has a life cycle during which it sends and receives messages and it can also perform internal actions in between. For the correct functioning of the overall system it is essential that no communication errors occur during component interactions. Two prominent classes of communication errors can be distinguished: The first one concerns situations, in which an output of one component is not accepted as an input by the other. The second one occurs if a component waits for an input which is never delivered. In this paper we focus on the former kind of communication error and we consider systems consisting of two

2012 ACM CCS: [**Theory of computation**]: Concurrency.
Key words and phrases: distributed systems, synchronous and asynchronous communication, verification of interaction compatibility.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.23638/LMCS-14(1:1)2018

components. We assume that outputs are autonomous actions and we call two components compatible if any output issued by one component is accepted by its communication partner. In our study we deal with bidirectional, peer to peer communication and with synchronous and asynchronous message exchange. The former is based on a rendezvous mechanism such that two components must execute shared output and input actions together while the latter uses potentially unbounded FIFO-buffers which hold the messages sent by one component and received by the other. We consider FIFO-buffers since these are used in well-known communication models, like CFSMs (Communicating Finite State Machines [4]), but also many concrete technologies rely on FIFO-communication, like the TCP protocol, the Java Messaging Service (being part of the Java Enterprise Edition as a message oriented middleware) and the Microsoft Message Queuing Service for service-oriented architectures. While compatibility of synchronously communicating components is decidable, see, e.g., [1], it is undecidable if unbounded FIFO-buffers are used as communication channels [4]. Therefore we are interested to investigate effective proof techniques for the verification of compatibility in the asynchronous case.

For this purpose we study, in the first part of this paper, relationships between synchronous and asynchronous compatibility. In each case we consider two versions, a strong and a weak compatibility notion. For the formalization of component behaviors we use I/O-transition systems (IOTSes) and call two IOTses A and B *strongly synchronously compatible* if in any reachable state of the synchronous product of A and B, if one component, say A, has a transition enabled with output action a then B must have a transition enabled with input action a. In many practical examples it turns out that before interacting with the sending component the receiving component should still be able to perform some internal actions in between. This leads to our notion of *weak synchronous compatibilty.* In the asynchronous context, components communicate via unbounded message queues. The idea of asynchronous compatibility is to require that whenever a message queue is not empty, then the receiver component must be able to take the next element of the queue; a property called specified reception in [4]. We distinguish again between *strong* and *weak* versions of *asynchronous compatibility.* In the asynchronous context the weak compatibility notion is particularly powerful since it allows a component, before it inputs a message waiting in the queue, still to put itself messages in its output queue (since we consider such enqueue actions as internal).

An obvious question is to what extent synchronous and asynchronous compatibility notions can be related to each other and, if this is not possible, which proof techniques can be used to verify asynchronous compatibility. We contribute to these issues with the following results:

(1) We establish a relationship between strong/weak synchronous and asynchronous compatibility of two components (Sects. 4.1 and 4.2). As a main result (Cor. 4.10) we get that strong (weak) synchronous compatibility is equivalent to strong (weak) asynchronous compatibility if the system enjoys the *half-duplex* property [10]. This means that in the asynchronous system at any time at most one message queue is not empty.

(2) In the second part of this work (Sect. 5), we consider general, possibly non half-duplex systems and study the verification of weak asynchronous compatibility in such cases. In Sect. 5.1 we investigate a decidable and powerful criterion, called WAC-criterion, for weak asynchronous compatibility (Thm. 5.3). The criterion cannot be necessary, since for unbounded FIFO-buffers the problem is undecidable, i.e., our proof method cannot be complete. In Sect. 5.2 we discuss how far we are away from completeness. To this end

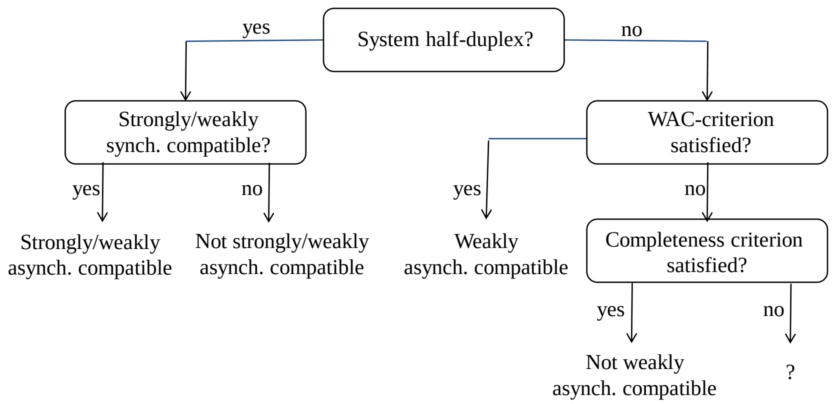

Figure 1: Verification method for compatibility

we develop a decidable completeness criterion (Thm. 5.8). If this criterion is satisfied then we can even disprove weak asynchronous compatibility.

Our results lead to the verification methodology for compatibility checking summarized in Fig. 1. Assume given two asynchronously communicating components A and B, each one having finitely many local states. First, we check whether the system is half-duplex, which is decidable [10]. If the answer is positive, then we check whether A and B are strongly (weakly) synchronously compatible, which is decidable as well [3]. Then the equivalence in Cor. 4.10 shows that if the answer is positive, then A and B are strongly (weakly) asynchronously compatible, otherwise they are not. If the system is not half-duplex we proceed as follows: We check whether the WAC-criterion (Thm. 5.3) is satisfied, which is decidable. If the answer is positive, then A and B are weakly asynchronously compatible. Otherwise we check the completeness criterion (Thm. 5.8), which is decidable as well. If the answer is positive, we know that A and B are not weakly asynchronously compatible. Otherwise we don't know.

In Sect. 6, we discuss deadlock-freeness of synchronous and asynchronous systems and show that deadlock-freeness is neither sufficient nor necessary for compatibility. We show how to perform a deadlock analysis for asynchronous systems following again the idea to prove properties of the synchronous system in order to get properties of the asynchronous one. Sect. 7 discusses related work and Sect. 8 summarizes our results and future work.

This paper is a significantly revised and extended version of the conference paper [13]. In Sect. 5 we have simplified the criterion for weak asynchronous compatibility. We have also added Sect. 5.2, which discusses completeness of the criterion. Moreover, the analysis of weak asynchronous compatibility is complemented in the new Sect. 6 by a deadlock analysis.

2. I/O-Transition Systems and Their Compositions

We start with the definitions of I/O-transition systems and their synchronous and asynchronous compositions which are the basis of the subsequent study.

Definition 2.1 (IOTS)**.** An *I/O-transition system* is a quadruple $A = (states_A, start_A, act_A, \longrightarrow_A)$ consisting of a set of states $states_A$, an initial state $start_A \in states_A$, a set

$act_A = in_A \cup out_A \cup int_A$ of actions being the disjoint union of sets in_A, out_A and int_A of input, output and internal actions resp., and a transition relation $\longrightarrow_A \subseteq states_A \times act_A \times states_A$.

We write $s \xrightarrow{a}_A s'$ instead of $(s, a, s') \in \longrightarrow_A$. For $X \subseteq act_A$ we write $s \xrightarrow{X}{}^*_A s'$ if there exists a (possibly empty) sequence of transitions $s \xrightarrow{a_1}_A s_1 \dots s_{n-1} \xrightarrow{a_n}_A s'$ involving only actions of X, i.e. $a_1, \dots, a_n \in X$. A state $s \in states_A$ is *reachable* if $start_A \xrightarrow{act_A}{}^*_A s$. The set of reachable states of A is denoted by $\mathcal{R}(A)$.

Two IOTSes A and B are *(syntactically) composable* if their actions only overlap on complementary types, i.e. $act_A \cap act_B = (in_A \cap out_B) \cup (in_B \cap out_A)$. The *set of shared actions* $act_A \cap act_B$ is denoted by $shared(A, B)$. The *synchronous composition* of two IOTSes A and B is defined as the product of transition systems with synchronization on shared actions which become internal actions in the composition. Shared actions can only be executed together; they are blocked if the other component is not ready for communication. In contrast, internal actions and non-shared input and output actions can always be executed by a single component in the composition. These (non-shared) actions are called *free actions* in the following.

Definition 2.2 (Synchronous composition)**.** Let A and B be two composable IOTSes. The *synchronous composition of* A *and* B is the IOTS $A \otimes B = (states_A \times states_B, (start_A, start_B), act_{A\otimes B}, \longrightarrow_{A\otimes B})$ where $act_{A\otimes B}$ is the disjoint union of the input actions $in_{A\otimes B} = (in_A \cup in_B) \setminus shared(A, B)$, the output actions $out_{A\otimes B} = (out_A \cup out_B) \setminus shared(A, B)$, and the internal actions $int_{A\otimes B} = int_A \cup int_B \cup shared(A, B)$. The transition relation of $A \otimes B$ is the smallest relation such that

- for all $a \in act_A \setminus shared(A, B)$, if $s \xrightarrow{a}_A s'$, then $(s, t) \xrightarrow{a}_{A\otimes B} (s', t)$ for all $t \in states_B$,
- for all $a \in act_B \setminus shared(A, B)$, if $t \xrightarrow{a}_B t'$, then $(s, t) \xrightarrow{a}_{A\otimes B} (s, t')$ for all $s \in states_A$, and
- for all $a \in shared(A, B)$, if $s \xrightarrow{a}_A s'$ and $t \xrightarrow{a}_B t'$, then $(s, t) \xrightarrow{a}_{A\otimes B} (s', t')$.

The synchronous composition of two IOTSes A and B yields a *closed* system if it has no input and output actions, i.e. $(in_A \cup in_B) \setminus shared(A, B) = \emptyset$ and $(out_A \cup out_B) \setminus shared(A, B) = \emptyset$, otherwise the system is *open*.

In distributed applications, implemented, e.g., with a message-passing middleware, usually an asynchronous communication pattern is used. In this paper, we consider asynchronous communication via unbounded message queues. In Fig. 2 two asynchronously communicating IOTSes A and B are depicted. A sends a message a to B by putting it, with action $a^{\triangleright}$, into a queue which stores the outputs of A. Then B can receive a by removing it, with action a, from the queue. In contrast to synchronous communication, the sending of a message cannot be blocked if the receiver is not ready to accept it. Similarly, B can send a message b to A by using a second queue which stores the outputs of B. The system in Fig 2 is open: A has an open output x to the environment and an open input y for messages coming from the environment. Similarly B has an open input u and an open output v. Additionally, A and B may have some internal actions.

To formalize asynchronous communication, we equip each communicating IOTS with an "output queue", which leads to a new IOTS indicated in Fig. 2 by $\Omega(A)$ and $\Omega(B)$ respectively. The motivation for using output queues should become clear when we define asynchronous compatibility in the next section. Formally, we represent an output queue as an (infinite) IOTS and then, in the case of A, we compose it with a renamed version of A where all outputs a of A (to be stored in the queue) are renamed to enqueue actions of the form $a^{\triangleright}$.

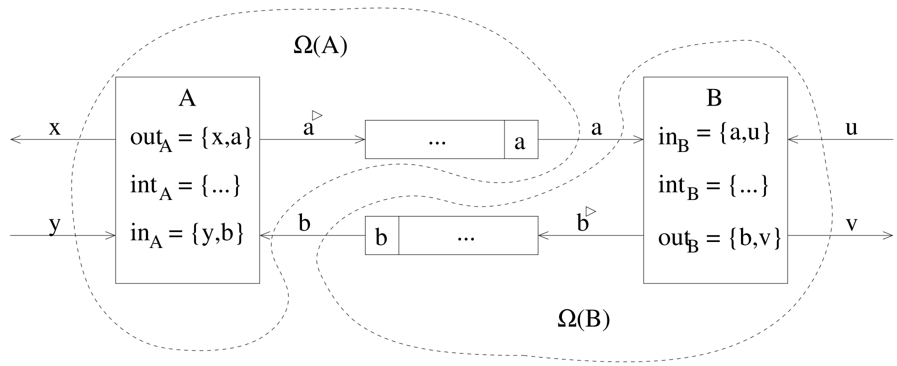

Figure 2: Asynchronous communication

Definition 2.3 (IOTS with output queue)**.**

(1) Let M be a set of names and $M^{\triangleright} = \{a^{\triangleright} \mid a \in M\}$. The *queue IOTS for* M is $Q_M = (M^*, \epsilon, act_{Q_M}, \longrightarrow_{Q_M})$ where the set of states is the set M^* of all words over M, the initial state $\epsilon \in M^*$ is the empty word, and the set of actions act_{Q_M} is the disjoint union of input actions $in_{Q_M} = M^{\triangleright}$, output actions $out_{Q_M} = M$ and with no internal action. The transition relation $\longrightarrow_{Q_M}$ is the smallest relation such that

- for all $a^{\triangleright} \in M^{\triangleright}$ and states $q \in M^*$: $q \xrightarrow{a^{\triangleright}}_{Q_M} qa$ (enqueue on the right),
- for all $a \in M$ and states $q \in M^*$: $aq \xrightarrow{a}_{Q_M} q$ (dequeue on the left).

(2) Let A be an IOTS such that $M \subseteq out_A$ and $M^{\triangleright} \cap act_A = \emptyset$. Let $A^{\triangleright}_M$ be the renamed version of A where all $a \in M$ are renamed to $a^{\triangleright}$. The *IOTS* A *equipped with output queue for* M is given by the synchronous composition $\Omega_M(A) = A^{\triangleright}_M \otimes Q_M$. (Note that $A^{\triangleright}_M$ and Q_M are composable.)

The states of $\Omega_M(A)$ are pairs (s, q) where s is a state of A and q is a word over M. The initial state is $(start_A, \epsilon)$. For the actions we have $in_{\Omega_M(A)} = in_A$, $out_{\Omega_M(A)} = out_A$, and $int_{\Omega_M(A)} = int_A \cup M^{\triangleright}$. Transitions in $\Omega_M(A)$ are:

- if $a \in in_A$ and $s \xrightarrow{a}_A s'$ then $(s, q) \xrightarrow{a}_{\Omega_M(A)} (s', q)$,
- if $a \in out_A \setminus M$ and $s \xrightarrow{a}_A s'$ then $(s, q) \xrightarrow{a}_{\Omega_M(A)} (s', q)$,
- if $a \in M \subseteq out_A$ then $(s, aq) \xrightarrow{a}_{\Omega_M(A)} (s, q)$,
- if $a \in int_A$ and $s \xrightarrow{a}_A s'$ then $(s, q) \xrightarrow{a}_{\Omega_M(A)} (s', q)$,
- if $a^{\triangleright} \in M^{\triangleright}$ and $s \xrightarrow{a}_A s'$ (i.e. $s \xrightarrow{a^{\triangleright}}_{A^{\triangleright}_M} s'$) then $(s, q) \xrightarrow{a^{\triangleright}}_{\Omega_M(A)} (s', qa)$.

To define the asynchronous composition of two IOTSes A and B, we assume that A and B are *asynchronously composable* which means that A and B are composable (as before) and $shared(A, B)^{\triangleright} \cap (act_A \cup act_B) = \emptyset$, i.e. no name conflict can arise when we rename a shared action a to $a^{\triangleright}$. Concerning A we consider the output actions of A which are shared with input actions of B and denote them by $out_{AB} = out_A \cap in_B$. These are the messages of A directed to B. Then, according to Def. 2.3, the IOTS A equipped with output queue for out_{AB} is given by $\Omega_{out_{AB}}(A) = A^{\triangleright}_{out_{AB}} \otimes Q_{out_{AB}}$. Note that $A^{\triangleright}_{out_{AB}}$ is the renamed version of A where all actions $a \in out_{AB}$ are renamed to $a^{\triangleright}$. Similarly, we consider the output actions of B which are shared with input actions of A, denote them by $out_{BA} = out_B \cap in_A$

and construct the IOTS $\Omega_{out_{BA}}(B) = B^{\triangleright}_{out_{BA}} \otimes Q_{out_{BA}}$ which represents the component B equipped with output queue for out_{BA}. The IOTSes $\Omega_{out_{AB}}(A)$ and $\Omega_{out_{BA}}(B)$ are then synchronously composed which gives the asynchronous composition of A and B.

Definition 2.4 (Asynchronous composition)**.** Let A, B be two asynchronously composable IOTSes. The *asynchronous composition* of A and B is defined by $A \otimes_{as} B = \Omega_{out_{AB}}(A) \otimes \Omega_{out_{BA}}(B)$.[1]

In the sequel we will briefly write $\Omega(A)$ for $\Omega_{out_{AB}}(A)$ and $\Omega(B)$ for $\Omega_{out_{BA}}(B)$. The states of $\Omega(A) \otimes \Omega(B)$ are pairs $((s_A, q_A), (s_B, q_B))$ where s_A is a state of A, the queue q_A stores elements of out_{AB}, s_B is a state of B, and the queue q_B stores elements of out_{BA}. The initial state is $((start_A, \epsilon), (start_B, \epsilon))$. For the actions we have $in_{\Omega(A)\otimes\Omega(B)} = in_{A\otimes B}$, $out_{\Omega(A)\otimes\Omega(B)} = out_{A\otimes B}$, and $int_{\Omega(A)\otimes\Omega(B)} = int_{A\otimes B} \cup shared(A, B)^{\triangleright}$. For the transitions in $\Omega(A) \otimes \Omega(B)$ we have two main cases:

(1) Transitions which can freely occur in A or in B without involving any output queue. These transitions change just the local state of A or of B. An example would be a transition $s_A \xrightarrow{a}_A s'_A$ with action $a \in out_A \setminus in_B$ which induces a transition $((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A), (s_B, q_B))$.
(2) Transitions which involve the output queue of A. There are two sub-cases concerning dequeue and enqueue actions which are internal actions in $\Omega(A) \otimes \Omega(B)$:
   (a) $a \in out_{AB}$ (hence $a \in out_{Q_{out_{AB}}}$) and $s_B \xrightarrow{a}_B s'_B$
   then $((s_A, aq_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s_A, q_A), (s'_B, q_B))$.
   (b) $a^{\triangleright} \in out^{\triangleright}_{AB}$ (hence $a^{\triangleright} \in in_{Q_{out_{AB}}}$) and $s_A \xrightarrow{a}_A s'_A$
   then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a^{\triangleright}}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A a), (s_B, q_B))$.
   Transitions which involve the output queue of B are analogous.

A detailed description of the form of the transitions of $\Omega(A) \otimes \Omega(B)$ is given in Appendix A.

3. Compatibility Notions

In this section we review our compatibility notions introduced in [3] for the synchronous and in [2] for the asynchronous case. For synchronous compatibility the idea is that whenever two synchronously cooperating components reach a state, in which one of the components wants to send an output a, i.e., a is enabled in the local state of the component, and if this action a belongs to the input actions of the other component, then the other component should be ready to receive a. This means that the other component should be in a local state such that an outgoing transition labeled with a exists. An implicit assumption behind this definition is that outputs are autonomously selected by the sending component and therefore its communication partner should accept (as an input) any possible output.

Definition 3.1 (Strong synchronous compatibility)**.** Two IOTSes A and B are *strongly synchronously compatible*, denoted by $A \leftrightarrow B$, if they are composable and if for all reachable states $(s_A, s_B) \in \mathcal{R}(A \otimes B)$,

(1) $\forall a \in out_A \cap in_B : \; s_A \xrightarrow{a}_A s'_A \Longrightarrow \exists \; s_B \xrightarrow{a}_B s'_B$,
(2) $\forall a \in out_B \cap in_A : \; s_B \xrightarrow{a}_B s'_B \Longrightarrow \exists \; s_A \xrightarrow{a}_A s'_A$.

[1]Note that $\Omega_{out_{AB}}(A)$ and $\Omega_{out_{BA}}(B)$ are composable.

In [3] we have introduced a weak version of compatibility such that a component can delay the required input and perform some internal actions before. We have shown in [3] that this fits well to weak refinement of component specifications in the sense that weak refinement (in particular, weak bisimulation) preserves weak compatibility while it does not preserve strong compatibility. Refinement is, however, not a topic of this work.

Definition 3.2 (Weak synchronous compatibility)**.** Two IOTSes A and B are *weakly synchronously compatible*, denoted by $A \dashleftarrow\!\dashrightarrow B$, if they are composable and if for all reachable states $(s_A, s_B) \in \mathcal{R}(A \otimes B)$,

(1) $\forall a \in out_A \cap in_B : \ s_A \xrightarrow{a}_A s'_A \Longrightarrow \exists\ s_B \xrightarrow{int_B}{}^{*}_{B}\ \bar{s}_B \xrightarrow{a}_B s'_B$,

(2) $\forall a \in out_B \cap in_A : \ s_B \xrightarrow{a}_B s'_B \Longrightarrow \exists\ s_A \xrightarrow{int_A}{}^{*}_{A}\ \bar{s}_A \xrightarrow{a}_A s'_A$,

Now we turn to compatibility of asynchronously communicating components A and B. In this case outputs of a component are stored in a queue from which they can be consumed by the receiver component. Therefore, in the asynchronous context, compatibility means that whenever a queue is not empty, the receiver component must be ready to take (i.e. input) the next removable element from the queue. Since we have enhanced components by output queues (rather than input queues) this idea can be easily formalized by reduction to synchronous compatibility of the components $\Omega(A)$ and $\Omega(B)$. Indeed, $\Omega(A)$ has an output a enabled iff a is the first element of the output queue of A and the same holds symmetrically for $\Omega(B)$.

Definition 3.3 (Strong and weak asynchronous compatibility)**.** Let A and B be two asynchronously composable I/O-transition systems. A and B are *strongly asynchronously compatible*, denoted by $A \overset{\mathrm{a}}{\longleftrightarrow} B$, if $\Omega(A) \longleftrightarrow \Omega(B)$. A and B are *weakly asynchronously compatible*, denoted by $A \overset{\mathrm{a}}{\dashleftarrow\!\dashrightarrow} B$, if $\Omega(A) \dashleftarrow\!\dashrightarrow \Omega(B)$.

Example 3.4. Fig. 3 shows the behavior of a `Maker` and a `User` process. Here and in the subsequent drawings we use the following notations: Initial states are denoted by 0, input actions a are indicated by $a?$, output actions a by $a!$, and internal actions a by τ_a. The maker expects some material from the environment (input action `material`), constructs some item (internal action `make`), and then it signals either that the item is ready (output action `ready`) or that the production did fail (output action `fail`). Both actions are shared with input actions of the user. When the user has received the ready signal it uses the item (internal action `use`). `Maker` and `User` are weakly synchronously compatible but not strongly synchronously compatible. The critical state in the synchronous product `Maker` $\otimes$ `User` is $(2,1)$ which can be reached with the transitions

$$(0,0) \xrightarrow{\text{material}} (1,0) \xrightarrow{\text{make}} (2,0) \xrightarrow{\text{ready}} (0,1) \xrightarrow{\text{material}} (1,1) \xrightarrow{\text{make}} (2,1).$$

In this state the maker wants to send `ready` or `fail` but the user must first perform its internal `use` action before it can receive the corresponding input. The asynchronous composition `Maker` $\otimes_{as}$ `User` has infinitely many states since the maker can be faster then the user. We will see, as an application of the forthcoming results, that `Maker` and `User` are also weakly asynchronously compatible.

4. Relating Synchronous and Asynchronous Compatibility

As pointed out in Sect. 1, it is generally undecidable whether two IOTSes are asynchronously compatible. In this section we study relationships between synchronous and asynchronous

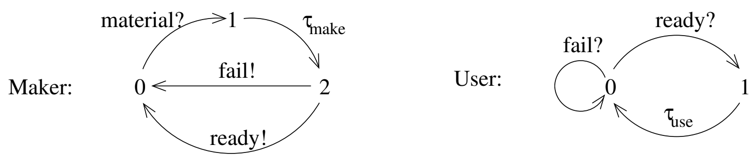

Figure 3: `Maker` and `User`

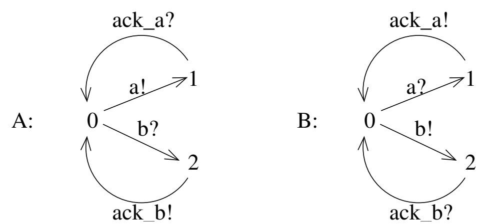

Figure 4: $\mathtt{A} \longleftrightarrow \mathtt{B}$ but <u>not</u> $\mathtt{A} \overset{a}{\longleftrightarrow} \mathtt{B}$

compatibility and conditions under which both are equivalent. Under these conditions we can reduce asynchronous compatibility checking to synchronous compatibility checking which is decidable for finite state components.

4.1. **From Synchronous to Asynchronous Compatibility.** We are interested in conditions under which it is sufficient to check strong (weak) synchronous compatibility to ensure strong (weak) asynchronous compatibility. In general this implication does not hold. As an example consider the two IOTSes `A` and `B` in Fig. 4. Obviously, `A` and `B` are strongly synchronously compatible. They are, however, neither strongly nor weakly asynchronously compatible since `A` may first put `a` in its output queue, then `B` can output `b` in its queue and then both are blocked (`A` can only accept `ack_a` while `B` can only accept `ack_b`). In Fig. 4 each IOTS has a state (the initial state) where a choice between an output and an input action is possible. We will see (Cor. 4.6) that if such situations are avoided synchronous compatibility implies asynchronous compatibility, and we will even get more general criteria (Thm. 4.3) for which the following property $\mathcal{P}$ is important.

Property $\mathcal{P}$: Let A and B be two asynchronously composable IOTSes. The asynchronous system $A \otimes_{as} B$ satisfies property $\mathcal{P}$ if for each reachable state $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$ one of the following conditions holds:

(i) $q_A = q_B = \epsilon$ and $(s_A, s_B) \in \mathcal{R}(A \otimes B)$.

(ii) $q_A = a_1 \dots a_m \neq \epsilon$ and $q_B = \epsilon$ and there exists $r_A \in \mathit{states}_A$ such that: $(r_A, s_B) \in \mathcal{R}(A \otimes B)$ and $r_A \overset{a_1}{\Longrightarrow}_A \dots \overset{a_m}{\Longrightarrow}_A s_A$.

(iii) $q_A = \epsilon$ and $q_B = b_1 \dots b_m \neq \epsilon$ and there exists $r_B \in \mathit{states}_B$ such that: $(s_A, r_B) \in \mathcal{R}(A \otimes B)$ and $r_B \overset{b_1}{\Longrightarrow}_B \dots \overset{b_m}{\Longrightarrow}_B s_B$.

To define the notation $\overset{a}{\Longrightarrow}_A$, let $a \in \mathit{out}_A \cap \mathit{in}_B$ and $F_A = \mathit{act}_A \setminus \mathit{shared}(A, B)$ be the set of the free actions of A. Then $s \overset{a}{\Longrightarrow}_A s'$ holds if there exist transitions $s \xrightarrow{F_A}{}^{*}_{A} \bar{s} \xrightarrow{a}_A \bar{s}' \xrightarrow{F_A}{}^{*}_{A} s'$. Recall from Sect. 2 that $t \xrightarrow{F_A}{}^{*}_{A} t'$ stands for a (possibly empty) sequence of transitions involving only actions of F_A. Hence, $s \overset{a}{\Longrightarrow}_A s'$ stands for a sequence of transitions such that a single transition with $a \in out_A \cap in_B$ is surrounded by arbitrary transitions in A involving only free actions of A. The notation $\overset{b}{\Longrightarrow}_B$ is defined analogously.

Looking more closely to property $\mathcal{P}$ we see that it requires two things: (a) In each reachable state of the asynchronous composition at least one of the two queues is empty, since one of the cases (i) - (iii) must always hold which entails emptiness of at least one queue. (b) The state of the component where the output queue is not empty can be reached from a reachable state in the *synchronous product* by outputting the actions stored in the queue, possibly interleaved with free actions. Part (a) specifies a well-known class of asynchronous systems called *half-duplex* systems; see, e.g., [10].

Definition 4.1. Let A and B be two asynchronously composable IOTSes. The asynchronous system $A \otimes_{as} B$ is *half-duplex*, if for all reachable states $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$ it holds that $q_A = \epsilon$ or $q_B = \epsilon$.

It turns out that part (b) explained above is not really an extra condition. It is already a consequence of part (a), i.e., of being a half-duplex system. Hence, property $\mathcal{P}$ and being half-duplex are equivalent conditions, as shown in the crucial lemma below. The direction (1) $\Rightarrow$ (2) is trivial since property $\mathcal{P}$ entails (a) from above, i.e., the half-duplex property. The difficult direction is (2) $\Rightarrow$ (1). The induction proof for this direction relies on condition (3) of Lem. 4.2 which is an easy consequence of being half-duplex. A nice side-effect of Lem. 4.2 is the equivalence (2) $\Leftrightarrow$ (3). It shows how the half-duplex property can be decided, since (3) is decidable for finite state components. This corresponds to a result in [10] where it is shown that membership to the class of half-duplex systems is decidable.

Lemma 4.2 (Crucial lemma)**.** *Let A and B be two asynchronously composable IOTSes. The following conditions are equivalent:*

(1) *The asynchronous system $A \otimes_{as} B$ satisfies property $\mathcal{P}$.*
(2) *The asynchronous system $A \otimes_{as} B$ is half-duplex.*
(3) *For each reachable state $(s_A, s_B) \in \mathcal{R}(A \otimes B)$ and each transitions $s_A \xrightarrow{a}_A s'_A$ and $s_B \xrightarrow{b}_B s'_B$ either $a \notin out_A \cap in_B$ or $b \notin out_B \cap in_A$.*

Proof. (1) $\Rightarrow$ (2) is trivial. (2) $\Rightarrow$ (3) is proved by contradiction: Assume (3) does not hold. Then there exist a reachable state $(s_A, s_B) \in \mathcal{R}(A \otimes B)$ and transitions $s_A \xrightarrow{a}_A s'_A$ and $s_B \xrightarrow{b}_B s'_B$ such that $a \in out_A \cap in_B$ and $b \in out_B \cap in_A$. Now we allow us a forward reference to Lem. 4.8, which shows $((s_A, \epsilon), (s_B, \epsilon)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. Since $s_A \xrightarrow{a}_A s'_A$ we get a transition

$$((s_A, \epsilon), (s_B, \epsilon)) \xrightarrow{a^{\triangleright}}_{\Omega(A) \otimes \Omega(B)} ((s'_A, a), (s_B, \epsilon)).$$

Since $s_B \xrightarrow{b}_B s'_B$ we get a transition

$$((s'_A, a), (s_B, \epsilon)) \xrightarrow{b^{\triangleright}}_{\Omega(A) \otimes \Omega(B)} ((s'_A, a), (s'_B, b))$$

and therefore the system is not half-duplex.

The direction (3) $\Rightarrow$ (1) is proved by induction on the length of the derivation to reach $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. It involves a complex case distinction on the form of the transitions in the asynchronous composition. Interestingly only the case of transitions with enqueue actions needs the assumption (3). The compete proof of (3) $\Rightarrow$ (1) is given in Appendix B. The interesting case in this proof is Case 5 (iii). $\square$

Theorem 4.3 (Synch2Asynch)**.** *Let A and B be two asynchronously composable IOTSes such that one (and hence all) of the conditions in Lemma 4.2 are satisfied. Then the following holds:*
(1) $A \leftrightarrow B \Longrightarrow A \overset{a}{\leftrightarrow} B$.
(2) $A \leftarrow\!\!\rightarrow B \Longrightarrow A \overset{a}{\leftarrow\!\!\rightarrow} B$.

Proof. The proof uses Lem. 4.2 for both cases.
(1) Assume $A \leftrightarrow B$. We have to show $\Omega(A) \leftrightarrow \Omega(B)$. We prove condition (1) of Def. 3.1. Condition (2) is proved analogously.
Let $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$, $a \in out_{\Omega(A)} \cap in_{\Omega(B)}$ and $(s_A, q_A) \xrightarrow{a}_{\Omega(A)} (s'_A, q'_A)$. Then q_A has the form $aa_2 \dots a_m$. By assumption, $\Omega(A) \otimes \Omega(B)$ satisfies property $\mathcal{P}$. Hence, there exists $r_A \in states_A$ such that $(r_A, s_B) \in \mathcal{R}(A \otimes B)$ and $r_A \overset{a}{\Longrightarrow}_A \overline{r}_A \overset{a_2}{\Longrightarrow}_{\dots} \overset{a_m}{\Longrightarrow}_A s_A$. Thereby $r_A \overset{a}{\Longrightarrow}_A \overline{r}_A$ is of the form $r_A \xrightarrow{F_A}{}^*_A s \xrightarrow{a}_A s' \xrightarrow{F_A}{}^*_A \overline{r}_A$. Since F_A involves only free actions of A (not shared with B), and since $(r_A, s_B) \in \mathcal{R}(A \otimes B)$ we have that $(s, s_B) \in \mathcal{R}(A \otimes B)$. Now we can use the assumption $A \leftrightarrow B$ which says that there exists $s_B \xrightarrow{a}_B s'_B$. Since $a \in in_B$, we get a transition $(s_B, q_B) \xrightarrow{a}_{\Omega(B)} (s'_B, q_B)$ and we are done.
(2) The weak case is a slight generalization of the proof of (1). The first part of the proof is the same but then we use the assumption $A \leftarrow\!\!\rightarrow B$ which says that there exists $s_B \xrightarrow{int_B}{}^*_B \overline{s}_B \xrightarrow{a}_B s'_B$ consisting of a sequence of internal transitions of B followed by $\overline{s}_B \xrightarrow{a}_B s'_B$ with $a \in in_B$. Therefore we get transitions $(s_B, q_B) \xrightarrow{int_B}{}^*_{\Omega(B)} (\overline{s}_B, q_B) \xrightarrow{a}_{\Omega(B)} (s'_B, q_B)$ and, since $int_B \subseteq int_{\Omega(B)}$, we are done. □

Note that the half-duplex property is not necessary for getting implication (1) and (2) of the last theorem. An example would be two components A and B such that $out_A \cap in_B = \{a\}$, $out_B \cap in_A = \{b\}$ are the only actions, A has one state and two looping transitions labeled with a and b, and the same holds for B. Then A and B are synchronously and asynchronously strongly and weakly compatible, but the system is not half-duplex. In fact, condition (3) of Lem. 4.2 is violated.

We come back to our discussion at the beginning of this section where we have claimed that for I/O-transition systems which do not show states where input and output actions are both enabled, synchronous compatibility implies asynchronous compatibility. We must, however, be careful whether we consider the strong or the weak case which leads us to two versions of I/O-separation.

Definition 4.4 (I/O-separated transition systems)**.** Let A be an IOTS.
(1) A is called *I/O-separated* if for all reachable states $s \in \mathcal{R}(A)$ it holds: If there exists a transition $s \xrightarrow{a}_A s'$ with $a \in out_A$ then there is no transition $s \xrightarrow{a'}_A s''$ with $a' \in in_A$.
(2) A is called *observationally I/O-separated* if for all reachable states $s \in \mathcal{R}(A)$ it holds: If there exists a transition $s \xrightarrow{a}_A s'$ with $a \in out_A$ then there is no sequence of transitions $s \xrightarrow{int_A}{}^*_A \overline{s}_A \xrightarrow{a'}_A s''$ with $a' \in in_A$.

Obviously, observational I/O-separation implies I/O-separation but not the other way round; cf. Ex. 4.7.

Lemma 4.5. *Let A and B be two asynchronously composable IOTSes.*
(1) *If A and B are I/O-separated and $A \leftrightarrow B$, then one (and hence all) of the conditions in Lemma 4.2 are satisfied.*

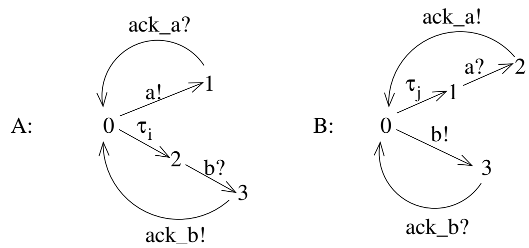

Figure 5: I/O-separated and $\mathtt{A} \dashleftarrow\!\!\dashrightarrow \mathtt{B}$ but <u>not</u> $\mathtt{A} \overset{a}{\dashleftarrow\!\!\dashrightarrow} \mathtt{B}$

(2) *If A and B are observationally I/O-separated and $A \dashleftarrow\!\!\dashrightarrow B$, then one (and hence all) of the conditions in Lemma 4.2 are satisfied.*

Proof. (1) By contradiction: Assume condition (3) of Lem. 4.2 does not hold. Then there are a reachable state $(s_A, s_B) \in \mathcal{R}(A \otimes B)$ and transitions $s_A \xrightarrow{a}_A s'_A$ and $s_B \xrightarrow{b}_B s'_B$ such that $a \in out_A \cap in_B$ and $b \in out_B \cap in_A$. Since $A \longleftrightarrow B$ there is a transition $s_B \xrightarrow{a}_B s''_B$ with $a \in in_B$. Therefore B is not I/O-separated.
(2) is proved similarly by contradiction: Assume that condition (3) of Lem. 4.2 does not hold. This gives us again a reachable state $(s_A, s_B) \in \mathcal{R}(A \otimes B)$ and transitions $s_A \xrightarrow{a}_A s'_A$ and $s_B \xrightarrow{b}_B s'_B$ such that $a \in out_A \cap in_B$ and $b \in out_B \cap in_A$. Since $A \dashleftarrow\!\!\dashrightarrow B$ there exist transitions $s_B \xrightarrow{int_B}{}^{*}_{B} \overline{s}_B \xrightarrow{a}_B s''_B$ with $a \in in_B$. Therefore B is not observationally I/O-separated. $\square$

The notion of I/O-separation appears in a more strict version, called *input-separation*, in [14] and similarly as *system without local mixed states* in [10]. [21] introduces internal choice labeled transition systems which are particular versions of I/O-separated transition systems. The difference is that I/O-separation still allows internal actions as an alternative to an input. Part (1) of Lem. 4.5 can be considered as a generalization of Lemma 4 in [14] which has shown that input-separated IOTSes which are strongly compatible and form a closed system are half-duplex. This result was in turn a generalization of Thm. 35 in [10]. Open systems and weak compatibility were not an issue in these approaches. With Theorem 4.3 and Lemma 4.5 we get:

Corollary 4.6. *Let A and B be two asynchronously composable IOTSes.*
(1) *If A and B are I/O-separated and $A \longleftrightarrow B$, then $A \overset{a}{\longleftrightarrow} B$.*
(2) *If A and B are observationally I/O-separated and $A \dashleftarrow\!\!\dashrightarrow B$, then $A \overset{a}{\dashleftarrow\!\!\dashrightarrow} B$.*

As an application of Cor. 4.6 we refer to Ex. 3.4. `Maker` and `User` are observationally I/O-separated, they are weakly synchronously compatible and therefore, by Cor. 4.6(2), they are also weakly asynchronously compatible.

Example 4.7. It may be interesting to note that part (2) of Cor. 4.6 and of Lem. 4.5 would not hold, if we would only assume I/O-separation. Fig. 5 shows two I/O-separated IOTSes `A` and `B` with internal actions `i` and `j` resp., such that `A` and `B` are not observationally I/O-separated. `A` and `B` are weakly synchronously compatible but not weakly asynchronously compatible and the asynchronous system $A \otimes_{as} B$ is also not half-duplex.

4.2. **From Asynchronous to Synchronous Compatibility.** This section studies the other direction, i.e. whether asynchronous compatibility can imply synchronous compatibility. It turns out that for the strong case this is indeed true without any further assumption while for the weak case this holds under the equivalent conditions of Lem. 4.2. In any case, we need for the proof the following lemma which shows that all reachable states in the synchronous product are reachable in the asynchronous product with empty output queues.

Lemma 4.8. *Let A and B be two asynchronously composable IOTSes. For any state $(s_A, s_B) \in \mathcal{R}(A \otimes B)$, the state $((s_A, \epsilon), (s_B, \epsilon))$ belongs to $\mathcal{R}(\Omega(A) \otimes \Omega(B))$.*

Proof. The proof is straightforward by induction on the length of the derivation of $(s_A, s_B) \in \mathcal{R}(A \otimes B)$. It is given in the Appendix. □

Theorem 4.9 (Asynch2Synch)**.** *For asynchronously composable IOTSes A and B it holds:*

1. $A \overset{a}{\longleftrightarrow} B \Longrightarrow A \longleftrightarrow B$.
2. *If one (and hence all) of the conditions in Lemma 4.2 are satisfied, then* $A \overset{a}{\dashleftarrow\!\dashrightarrow} B \Longrightarrow A \dashleftarrow\!\dashrightarrow B$.

Proof. (1) Assume $A \overset{\mathrm{a}}{\longleftrightarrow} B$, i.e. $\Omega(A) \longleftrightarrow \Omega(B)$. We have to show $A \longleftrightarrow B$. We prove condition (1) of Def. 3.1. Condition (2) is proved analogously.
Let $(s_A, s_B) \in \mathcal{R}(A \otimes B), a \in out_A \cap in_B$ and $s_A \xrightarrow{a}_A s'_A$. By Lem. 4.8, $((s_A, \epsilon), (s_B, \epsilon)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. Since $s_A \xrightarrow{a}_A s'_A$, we have a transition in $\Omega(A) \otimes \Omega(B)$ with enqueue action for a: $((s_A, \epsilon), (s_B, \epsilon)) \xrightarrow{a^{\rhd}}_{\Omega(A)\otimes\Omega(B)} ((s'_A, a), (s_B, \epsilon))$ and it holds $((s'_A, a), (s_B, \epsilon)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. Then, there is a transition $(s'_A, a) \xrightarrow{a}_{\Omega(A)} (s'_A, \epsilon)$. Since $\Omega(A) \longleftrightarrow \Omega(B)$ there must be a transition $(s_B, \epsilon) \xrightarrow{a}_{\Omega(B)} (s'_B, \epsilon)$. This transtion must be caused by a transition $s_B \xrightarrow{a}_B s'_B$ and we are done.
(2) Assume $A \overset{\mathrm{a}}{\dashleftarrow\!\dashrightarrow} B$, i.e. $\Omega(A) \dashleftarrow\!\dashrightarrow \Omega(B)$. We have to show $A \dashleftarrow\!\dashrightarrow B$. We prove condition (1) of Def. 3.2. Condition (2) is proved analogously.
Let $(s_A, s_B) \in \mathcal{R}(A \otimes B), a \in out_A \cap in_B$ and $s_A \xrightarrow{a}_A s'_A$. With the same reasoning as in case (1) we get $((s'_A, a), (s_B, \epsilon)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$ and we get a transition $(s'_A, a) \xrightarrow{a}_{\Omega(A)} (s'_A, \epsilon)$. Since $\Omega(A) \dashleftarrow\!\dashrightarrow \Omega(B)$ there are transitions $(s_B, \epsilon) \xrightarrow{int_{\Omega(B)}}{}^{*}_{\Omega(B)} (\overline{s}_B, \overline{q}_B) \xrightarrow{a}_{\Omega(B)} (s'_B, \overline{q}_B)$. Since internal transitions of $\Omega(B)$ do not involve any steps of $\Omega(A)$, we have $((s'_A, a), (\overline{s}_B, \overline{q}_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. Due to the assumption that the conditions in Lemma 4.2 are satisfied, $\Omega(A) \otimes \Omega(B)$ is half-duplex and therefore $\overline{q}_B$ must be empty and the same holds for all intermediate queues reached by the transitions in $(s_B, \epsilon) \xrightarrow{int_{\Omega(B)}}{}^{*}_{\Omega(B)} (\overline{s}_B, \overline{q}_B)$. Therefore no enqueue action can occur in these transitions. Noticing that $int_{\Omega(B)} = int_B \cup (out_B \cap in_A)^{\rhd}$, we get $(s_B, \epsilon) \xrightarrow{int_B}{}^{*}_{\Omega(B)} (\overline{s}_B, \epsilon) \xrightarrow{a}_{\Omega(B)} (s'_B, \epsilon)$ and all these transtions must be induced by transitions $s_B \xrightarrow{int_B}{}^{*}_B \overline{s}_B \xrightarrow{a}_B s'_B$, i.e. we are done. □

As a consequence of Thms. 4.3 and 4.9 we see that under the equivalent conditions of Lem. 4.2, in particular when the asynchronous system is half-duplex, (weak) synchronous compatibility is equivalent to (weak) asynchronous compatibility.

Corollary 4.10 (SynchIFFAsynch)**.** *Let A and B be two asynchronously composable IOTSes such that one (and hence all) of the conditions in Lemma 4.2 are satisfied. Then the following holds:*

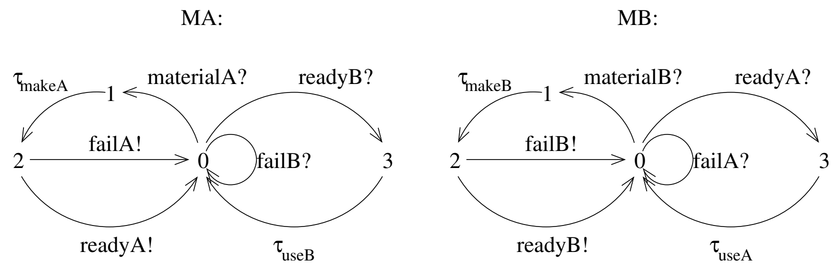

Figure 6: $\mathtt{MA} \overset{a}{\leftarrow\!\!-\!\!\rightarrow} \mathtt{MB}$ but <u>not</u> $\mathtt{MA} \leftarrow\!-\!\rightarrow \mathtt{MB}$.

(1) $A \longleftrightarrow B \iff A \overset{a}{\longleftrightarrow} B$.
(2) $A \leftarrow\!-\!\rightarrow B \iff A \overset{a}{\leftarrow\!-\!\rightarrow} B$.

5. Weak Asynchronous Compatibility: The General Case

In this section we are interested in the verification of asynchronous compatibility in the general case, where at the same time both queues of the communicating components may be not empty. We focus here on *weak* asynchronous compatibility since non-half duplex systems are often weakly asynchronously compatible but not weakly synchronously compatible.[2] A simple example would be two components which both start to send a message to each other and after that each component takes the message addressed to it from the buffer. Such a system would be weakly asynchronously compatible but not weakly synchronously compatible.

Example 5.1. Fig. 6 shows two IOTSes `MA` and `MB` which produce items for each other. After reception of some material from the environment (input action `materialA`), `MA` produces an item (internal action `makeA`) followed by either a signal that the item is ready for use (output `readyA`) or a signal that the production did fail (output `failA`). Whenever `MA` reaches its initial state it can also accept an input `readyB` and then use the item produced by `MB` (internal action `useB`) or it can accept a signal that the production of its partner did fail (input `failB`). The behavior of `MB` is analogous. Note that `materialA` (resp. `materialB`) is a non-shared input action of `MA` (resp. `MB`). They are open to the environment after composition of `MA` and `MB`. The asynchronous composition of `MA` and `MB` is not half-duplex; both processes can produce and signal concurrently. Clearly, the system is not weakly synchronously compatible. For instance, the state (2,2) is reachable in the synchronous product and in this state each of the two components wants to output an action but the other one is not able to synchronize with a corresponding input. We will prove below that the system is weakly asynchronously compatible. Let us note that the system considered here is neither synchronizable in the sense of [19] nor desynchronizable in the sense of [6, 7]. The reason is simple: The synchronous system is blocked in state (2,2) while the asynchronous system can always proceed with putting messages in the buffers, consuming them and firing transitions for the free actions open to the environment. Therefore there cannot be a branching bisimulation between the synchronous and the asynchronous versions of the system as required for synchronizability in [19] and for desynchronizability in [6, 7].

[2] This is in contrast to the strong case where strong asynchronous compatibility implies strong synchronous compatibility; see Thm. 4.9(1).

In general, the problem of weak asynchronous compatibility is undecidable due to potentially unbounded message queues. For instance, in the asynchronous composition of `MA` and `MB` in Ex. 5.1 both output queues can grow without upper bound. In Sect. 5.1 we develop a criterion for proving weak asynchronous compatibility in the general case (allowing for non half-duplex systems with unbounded message queues). The criterion is decidable if the underlying IOTSes are finite. In Sect. 5.2 we investigate properties under which the criterion is even complete, i.e. if the criterion is not satisfied, then the system is not weakly asynchronously compatible.

5.1. **A Criterion for Weak Asynchronous Compatibility.** Let A and B be two asynchronously composable IOTSes. The idea for proving weak asynchronous compatibility of A and B is again to use synchronous products, but not the standard synchronous composition of A and B but variants of it. First we focus only on one direction of compatibility concerning the outputs of A which should be received by B. Due to the weak compatibility notion B can, before it takes an input message, execute internal actions. In particular, it can put outputs directed to A in its output queue. (Remember that enqueue actions are internal). To simulate these autonomous enqueue actions in a *synchronous* product with A, we consider the renamed version $B^{\triangleright}_{out_{BA}}$ of B where all actions $b \in out_{BA} = out_B \cap in_A$ are renamed to $b^{\triangleright}$. Thus they become non-shared actions which can be freely executed in the synchronous product of A and $B^{\triangleright}_{out_{BA}}$ (just as the enqueue actions $b^{\triangleright}$ in the asynchronous product of A and B). At the same time all previously shared input actions of A become free. Now we require that in each reachable state of the *synchronous* product $A \otimes B^{\triangleright}_{out_{BA}}$ if A wants to send an output a addressed to B then $B^{\triangleright}_{out_{BA}}$ can execute some internal actions and/or free output actions $b^{\triangleright} \in out^{\triangleright}_{BA}$ before it accepts a. This idea is formalized in the following condition (a). A symmetric condition concerning the compatibility in the direction from B to A is formalized in condition (b).

(a) For all reachable states $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$, $\forall a \in out_{AB} = out_A \cap in_B$: $s_A \xrightarrow{a}_A s'_A \Longrightarrow \exists\, s_B \xrightarrow{int_B \cup out^{\triangleright}_{BA}}{}^{*}_{B^{\triangleright}_{out_{BA}}} \bar{s}_B \xrightarrow{a}_B s'_B$.[3]

(b) For all reachable states $(s_A, s_B) \in \mathcal{R}(A^{\triangleright}_{out_{AB}} \otimes B)$, $\forall b \in out_{BA} = out_B \cap in_A$: $s_B \xrightarrow{b}_B s'_B \Longrightarrow \exists\, s_A \xrightarrow{int_A \cup out^{\triangleright}_{AB}}{}^{*}_{A^{\triangleright}_{out_{AB}}} \bar{s}_A \xrightarrow{b}_A s'_A$.

Notation 5.2. We write $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ if condition (a) holds and $B \dashrightarrow A^{\triangleright}_{out_{AB}}$ if condition (b) holds.

We call the conditions $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ and $B \dashrightarrow A^{\triangleright}_{out_{AB}}$ the *WAC-criterion* since they are sufficient for weak asynchronous compatibility.

Theorem 5.3 (WAC-criterion)**.** *Let A and B be two asynchronously composable IOTSes such that $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ and $B \dashrightarrow A^{\triangleright}_{out_{AB}}$ holds. Then A and B are weakly asynchronously compatible, i.e. $A \overset{a}{\leftarrow\!-\!\rightarrow} B$.*

The proof of this theorem needs an auxiliary, technical lemma which establishes a relationship between the reachable states of the asynchronous composition of A and B and

[3]Note that $int_B = int_{B^{\triangleright}_{out_{BA}}}$ and $\bar{s}_B \xrightarrow{a}_B s'_B$ is equivalent to $\bar{s}_B \xrightarrow{a}_{B^{\triangleright}_{out_{BA}}} s'_B$, since $a \in out_A \cap in_B$ is not renamed.

the reachable states considered in the synchronous products $A \otimes B^{\triangleright}_{out_{BA}}$ and $A^{\triangleright}_{out_{AB}} \otimes B$ respectively.

Lemma 5.4. *For any two asynchronously composable IOTSes A and B it holds that A and $B^{\triangleright}_{out_{BA}}$ as well as $A^{\triangleright}_{out_{AB}}$ and B are synchronously composable and* both *of the following two properties $\mathcal{Q}_A$ and $\mathcal{Q}_B$ are satisfied.*

Property $\mathcal{Q}_A$: For each reachable state $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$ one of the following two conditions holds:

(i) *$q_A = \epsilon$ and $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$,*

(ii) *$q_A = a_1 \dots a_m \neq \epsilon$ and there exists $r_A \in states_A$ such that:*
$(r_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ and $r_A \overset{a_1}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$.

The notation $s \overset{a}{\Rrightarrow}_A s'$ stands for an arbitrary sequence of transitions in A which contains exactly one transition with an output action in $out_A \cap in_B$ and this output action is a.

Property $\mathcal{Q}_B$: For each reachable state $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$ one of the following two conditions holds:

(i) *$q_B = \epsilon$ and $(s_A, s_B) \in \mathcal{R}(A^{\triangleright}_{out_{AB}} \otimes B)$,*

(ii) *$q_B = b_1 \dots b_m \neq \epsilon$ and there exists $r_B \in states_B$ such that:*
$(s_A, r_B) \in \mathcal{R}(A^{\triangleright}_{out_{AB}} \otimes B)$ and $r_B \overset{b_1}{\Rrightarrow}_B \dots \overset{b_m}{\Rrightarrow}_B s_B$.

The notation $\overset{b}{\Rrightarrow}_B$ is defined analogously to $\overset{a}{\Rrightarrow}_A$.

Proof. Since A and B are asynchronously composable they are synchronously composable and $shared(A, B)^{\triangleright} \cap (act_A \cup act_B) = \emptyset$. Hence, A and $B^{\triangleright}_{out_{BA}}$ as well as $A^{\triangleright}_{out_{AB}}$ and B are synchronously composable.

The initial state $((start_A, \epsilon), (start_B, \epsilon))$ satisfies $\mathcal{Q}_A$ and $\mathcal{Q}_B$. Then we consider transitions

$$((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A) \otimes \Omega(B)} ((s'_A, q'_A), (s'_B, q'_B))$$

and show that if $((s_A, q_A), (s_B, q_B))$ satisfies $\mathcal{Q}_A$ ($\mathcal{Q}_B$ resp.) then $((s'_A, q'_A), (s'_B, q'_B))$ satisfies $\mathcal{Q}_A$ ($\mathcal{Q}_B$ resp.). Then the result follows by induction on the length of the derivation to reach $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. The complete proof is given in the Appendix. □

Property $\mathcal{Q}_A$(i) expresses that whenever a global state $((s_A, \epsilon), (s_B, q_B))$ is reachable in the asynchronous composition of A and B, then (s_A, s_B) is already reachable in the synchronous composition $A \otimes B^{\triangleright}_{out_{BA}}$. Property $\mathcal{Q}_A$(ii) expresses that whenever a global state $((s_A, q_A), (s_B, q_B))$ with $q_A \neq \epsilon$ is reachable in the asynchronous composition of A and B there exists a state r_A of A such that (r_A, s_B) is reachable in the synchronous composition $A \otimes B^{\triangleright}_{out_{BA}}$ and the local control state s_A of A can be reached from r_A by outputting the actions stored in the queue, possibly interleaved with arbitrary other actions of A which are not output actions directed to B. Properties $\mathcal{Q}_B$(i) and (ii) are the symmetric properties concerning the output queue of B.

The properties $\mathcal{Q}_A$ and $\mathcal{Q}_B$ have a pattern similar to property $\mathcal{P}$ in Sect. 4.1 which has related reachable states of the asynchronous composition of A and B with reachable states in the synchronous product $A \otimes B$. Such a relation was only possible under the half-duplex assumption while $\mathcal{Q}_A$ and $\mathcal{Q}_B$ are generally valid. The intuitive reason is that $A \otimes B^{\triangleright}_{out_{BA}}$ as well as $A^{\triangleright}_{out_{AB}} \otimes B$ can have significantly more reachable states than $A \otimes B$. This is demonstrated in Ex. 5.6 and this is also the reason why our proof technique is in general not complete; see Sect. 5.2. We are now prepared to prove Thm. 5.3.

Proof of Theorem 5.3: By definition of $A \overset{a}{\leftarrow\!\!\!\dashrightarrow} B$ we have to show $\Omega(A) \leftarrow\!\!\!\dashrightarrow \Omega(B)$. We prove condition (1) of Def. 3.2. Condition (2) is proved analogously.
Let $((s_A,q_A),(s_B,q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. To prove condition (1) of Def. 3.2 we assume given $a \in \mathit{out}_{\Omega(A)} \cap \mathit{in}_{\Omega(B)}$ and $(s_A,q_A) \xrightarrow{a}_{\Omega(A)} (s'_A,q'_A)$ and we must show that there exist transitions $(s_B,q_B) \xrightarrow{\mathit{int}_{\Omega(B)}}{}^{*}_{\Omega(B)} (\overline{s}_B,\overline{q}_B) \xrightarrow{a}_{\Omega(B)} (s'_B,q'_B)$.

By the assumption, q_A must have the form $aa_2 \dots a_m$. By Lem. 5.4, property $\mathcal{Q}_A$(ii) holds for $((s_A,q_A),(s_B,q_B))$. Hence, there exists $r_A \in \mathit{states}_A$ such that $(r_A,s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{\mathit{out}_{BA}})$ and $r_A \overset{a}{\Rrightarrow}_A \overline{r}_A \overset{a_2}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$. Thereby $r_A \overset{a}{\Rrightarrow}_A \overline{r}_A$ is of the form $r_A \xrightarrow{Y_A}{}^{*}_{A} s \xrightarrow{a}_A s' \xrightarrow{Y_A}{}^{*}_{A} \overline{r}_A$ with $a \in \mathit{out}_{\Omega(A)} \cap \mathit{in}_{\Omega(B)} = \mathit{out}_A \cap \mathit{in}_B = \mathit{out}_{AB}$ and Y_A involves no action in out_{AB}. Since out_{AB} are the only shared actions of A and $B^{\triangleright}_{\mathit{out}_{BA}}$, the transitions in $r_A \xrightarrow{Y_A}{}^{*}_{A} s$ induce transitions in $A \otimes B^{\triangleright}_{\mathit{out}_{BA}}$ without involving $B^{\triangleright}_{\mathit{out}_{BA}}$. Therefore, since $(r_A,s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{\mathit{out}_{BA}})$, we get $(s,s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{\mathit{out}_{BA}})$. Now we can use the assumption $A \dashrightarrow B^{\triangleright}_{\mathit{out}_{BA}}$ which says that there exists a sequence of transitions

$$s_B \xrightarrow{\mathit{int}_B \cup \mathit{out}^{\triangleright}_{BA}}{}^{*}_{B^{\triangleright}_{\mathit{out}_{BA}}} \overline{s}_B \xrightarrow{a}_B s'_B.$$

The actions in $\mathit{int}_B \cup \mathit{out}^{\triangleright}_{BA}$ are internal actions of $\Omega(B)$ such that we get transitions

$$(s_B,q_B) \xrightarrow{\mathit{int}_{\Omega(B)}}{}^{*}_{\Omega(B)} (\overline{s}_B,\overline{q}_B) \xrightarrow{a}_{\Omega(B)} (s'_B,\overline{q}_B)$$

where $\overline{q}_B$ extends q_B according to the elements that have been enqueued with actions in $\mathit{out}^{\triangleright}_{BA}$. Thus $\Omega(B)$ accepts a, possibly after some internal actions, and we are done. $\square$

Example 5.5. To apply Thm. 5.3 to Ex. 5.1 we have to prove $\mathtt{MA} \dashrightarrow \mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$ and $\mathtt{MB} \dashrightarrow \mathtt{MA}^{\triangleright}_{\{\mathtt{readyA},\mathtt{failA}\}}$. For the former case, Fig. 7 shows the IOTS `MA` and the IOTS $\mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$ obtained by renaming of its outputs. We will check only this case, the other one is analogous. We have to consider the reachable states in the synchronous product $\mathtt{MA} \otimes \mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$ and when an output `readyA` or `failA` is possible in `MA`. These states are (2,0), (2,1) and (2,2) since `materialA`, `materialB` are non-shared input actions and `makeA`, `makeB` are internal actions. (Note that state (2,3) is not reachable in $\mathtt{MA} \otimes \mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$ because `readyA` is a shared action of `MA` and $\mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$.)

In state (2,0) any output `readyA` or `failA` is immediately accepted by $\mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$. In state (2,1), $\mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$ can perform first the internal action `makeB`, then the free output action $\mathtt{readyB}^{\triangleright}$ or $\mathtt{failB}^{\triangleright}$ and then it can accept the input `readyA` or `failA`. In state (2,2), $\mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$ can perform the free output action $\mathtt{readyB}^{\triangleright}$ or $\mathtt{failB}^{\triangleright}$ and then accept the input. Therefore we have shown $\mathtt{MA} \overset{a}{\leftarrow\!\!\!\dashrightarrow} \mathtt{MB}$. Note that with the free output actions we have simulated in the synchronous product the (internal) enqueue actions $\mathtt{readyB}^{\triangleright}$ and $\mathtt{failB}^{\triangleright}$ that can be executed by `MB` in the asynchronous composition.[4]

[4] Our technique would also work for the non-synchronizable system example in [19], Fig. 4.

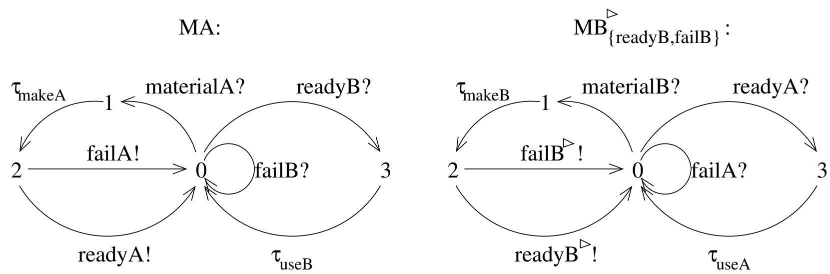

Figure 7: Compatibility check: $\mathtt{MA} \dashrightarrow \mathtt{MB}^{\triangleright}_{\{\mathtt{readyB},\mathtt{failB}\}}$

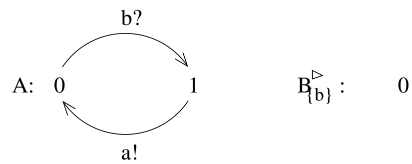

Figure 8: Compatibility check $\mathtt{A} \dashrightarrow \mathtt{B}^{\triangleright}_{\{\mathtt{b}\}}$ fails but $\mathtt{A} \overset{\mathtt{a}}{\dashleftarrow\!\dashrightarrow} \mathtt{B}$.

5.2. **On the Completeness of the Compatibility Criterion.** The compatibility criterion of the last section relies on the two conditions (a) and (b) required for all reachable states of $A \otimes B^{\triangleright}_{out_{BA}}$ and $A^{\triangleright}_{out_{AB}} \otimes B$ respectively. If A and B are finite then the compatibility criterion is decidable while weak asynchronous compatibility is, in general, not decidable. Hence the compatibility criterion cannot be complete. In this section we first discuss in which situations it can happen that the compatibility criterion is not necessary for weak asynchronous compatibility and then we establish a condition under which the compatibility criterion is even complete for proving or disproving weak asynchronous compatibility.

Example 5.6. The following very simple example illustrates the issue. We consider two components A and B such that $in_A = \{b\}$, $out_A = \{a\}$, $int_A = \emptyset$ and $in_B = \{a\}$, $out_B = \{b\}$, $int_B = \emptyset$. The transitions of A are shown in Fig. 8. The component B and hence $B^{\triangleright}_{out_{BA}}$ has no transitions; i.e. their actions are never enabled. Then it is trivial that A and B are weakly asynchronously compatible, since in the asynchronous composition A will never receive a message from B, i.e. $\Omega(A) \otimes \Omega(B)$ will never reach a state $(s_A, q_A), (s_B, q_B)$ with $s_A = 1$. Therefore A will never put a in its output buffer. However, our condition (a), $A \dashrightarrow B^{\triangleright}_{out_{BA}}$, is not satisfied since b is a free input action of A in $A \otimes B^{\triangleright}_{out_{BA}}$ and therefore the state $(1, 0)$ is reachable in $A \otimes B^{\triangleright}_{out_{BA}}$. Then $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ would require that $B^{\triangleright}_{out_{BA}}$ is able to receive b in its state 0 which is not the case.

The problem encountered in Ex. 5.6 is that $A \otimes B^{\triangleright}_{out_{BA}}$ (or, symmetrically, $A^{\triangleright}_{out_{AB}} \otimes B$) may have more reachable states than necessary to be considered in the asynchronous composition $A \otimes_{as} B$. These states are reached by open inputs in $A \otimes B^{\triangleright}_{out_{BA}}$ (or $A^{\triangleright}_{out_{AB}} \otimes B$) which are never served in the asynchronous composition where the inputs are shared actions. More precisely, our conjecture is that the criterion of Thm. 5.3 may not be complete only if either

(i) there are states (s_A, s_B) reachable in $A \otimes B^{\triangleright}_{out_{BA}}$ such that A has an output in state s_A but the local state s_A is not reachable in the asynchronous composition with B and hence irrelevant for proving asynchronous compatibility in the direction from A to B, or

(ii) there are states (s_A, s_B) reachable in $A^{\triangleright}_{out_{AB}} \otimes B$ such that B has an output in state s_B but the local state s_B is not reachable in the asynchronous composition with A and hence irrelevant for proving asynchronous compatibility in the direction from B to A.

The subsequent theorem shows that our conjecture is right. It relies on the definition of locally reachable states.

Definition 5.7. Let A and B be two synchronously composable IOTSes. A state s_A of A is *locally reachable* in $A \otimes B$, if there exists a state s_B of B such that $(s_A, s_B) \in \mathcal{R}(A \otimes B)$. Local reachability for states of B is defined analogously.

Theorem 5.8 (Completeness criterion)**.** *Let A and B be two asynchronously composable IOTSes such that the following two properties $\mathcal{X}_A$ and $\mathcal{X}_B$ are satisfied.*
Property $\mathcal{X}_A$: For any state s_A of A for which a transition $s_A \xrightarrow{a}_A s'_A$ exists with $a \in out_{AB}$ the following holds: If s_A is locally reachable in $A \otimes B^{\triangleright}_{out_{BA}}$ then (s_A, ϵ) is locally reachable in $\Omega(A) \otimes \Omega(B)$.[5]
Property $\mathcal{X}_B$: For any state s_B of B for which a transition $s_B \xrightarrow{b}_B s'_B$ exists with $b \in out_{BA}$ the following holds: If s_B is locally reachable in $A^{\triangleright}_{out_{AB}} \otimes B$ then (s_B, ϵ) is locally reachable in $\Omega(A) \otimes \Omega(B)$.

Then $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ and $B \dashrightarrow A^{\triangleright}_{out_{AB}}$ holds if, and only if, A and B are weakly asynchronously compatible, i.e. $A \overset{a}{\leftarrow\!\!\rightarrow} B$.

Proof. Taking into account Thm. 5.3, it remains to show that under the assumptions $\mathcal{X}_A$ and $\mathcal{X}_B$ we have that $A \overset{a}{\leftarrow\!\!\rightarrow} B$ implies $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ and $B \dashrightarrow A^{\triangleright}_{out_{AB}}$. Let $A \overset{a}{\leftarrow\!\!\rightarrow} B$, i.e. $\Omega(A) \leftarrow\!\!\rightarrow \Omega(B)$. We show that then $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ holds. The proof of $B \dashrightarrow A^{\triangleright}_{out_{AB}}$ is analogous.

Let $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$. To show $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ we assume $a \in out_{AB} = out_A \cap in_B$ and $s_A \xrightarrow{a}_A s'_A$. According to the meaning of $A \dashrightarrow B^{\triangleright}_{out_{BA}}$ (as defined in the notation before), we have to show that there exist transitions

$$(*) s_B \xrightarrow{int_B \cup out^{\triangleright}_{BA}}{}^{*}_{B^{\triangleright}_{out_{BA}}} \bar{s}_B \xrightarrow{a}_B s'_B.$$

Since $\mathcal{X}_A$ is valid, there exist s_B, q_B such that $((s_A, \epsilon), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. The transition $s_A \xrightarrow{a}_A s'_A$ induces an enqueue transition $((s_A, \epsilon), (s_B, q_B)) \xrightarrow{a^{\triangleright}}_{\Omega(A)\otimes\Omega(B)} ((s'_A, a), (s_B, q_B))$ such that a is the only element in the output queue of A. Obviously, $((s'_A, a), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$ and there is a transition $(s'_A, a) \xrightarrow{a}_{\Omega(A)} (s'_A, \epsilon)$. Since $\Omega(A) \leftarrow\!\!\rightarrow \Omega(B)$ there are transitions

$$(s_B, q_B) \xrightarrow{int_{\Omega(B)}}{}^{*}_{\Omega(B)} (\bar{s}_B, \bar{q}_B) \xrightarrow{a}_{\Omega(B)} (s'_B, \bar{q}_B).$$

Since $int_{\Omega(B)} = int_B \cup out^{\triangleright}_{BA}$ and $a \in in_{\Omega(B)} = in_B$ there are transitions (*) and we are done. □

[5] In other words, there exist s_B, q_B such that $((s_A, \epsilon), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$.

Example 5.9. In this example we want to construct two components whose asynchronous composition has infinitely many states but weak asynchronous compatibility is decidable. As a tool we want to apply Thm. 5.8. Therefore our components should satisfy properties $\mathcal{X}_A$ and $\mathcal{X}_B$ of Thm. 5.8. Consider the two components `MA` and `MB` in Fig. 6. We remove the transition $0 \xrightarrow{\texttt{failA}}_{\texttt{MB}} 0$ with input action `failA` from `MB` which gives us the component `MB'`. Now we show that the properties $\mathcal{X}_A$ and $\mathcal{X}_B$ are satisfied for `MA` and `MB'`. To check $\mathcal{X}_A$ we must consider the state 2 of `MA` in which an output is enabled and which is locally reachable in $\texttt{MA} \otimes \texttt{MB'}^{\triangleright}_{\{\texttt{readyB},\texttt{failB}\}}$ (since, e.g., (2,0) is reachable in $\texttt{MA} \otimes \texttt{MB'}^{\triangleright}_{\{\texttt{readyB},\texttt{failB}\}}$). Obviously, the state (2,ϵ) of $\Omega(\texttt{MA})$ is locally reachable in $\Omega(\texttt{MA}) \otimes \Omega(\texttt{MB'})$. (For instance, ((2,$\epsilon$),(0,$\epsilon$)) is reachable in $\Omega(\texttt{MA}) \otimes \Omega(\texttt{MB'})$.) Property $\mathcal{X}_B$ is checked analogously. According to Thm. 5.8 we can therefore decide whether `MA` and `MB'` are weakly asynchronously compatible. In this example, $\texttt{MA} \dashrightarrow \texttt{MB'}^{\triangleright}_{\{\texttt{readyB},\texttt{failB}\}}$ does *not* hold since in state (2,0) the component `MA` can output `failA` which cannot be accepted by `MB'`. Therefore, by Thm. 5.8, we have proved that `MA` and `MB'` are *not* weakly asynchronously compatible.

6. Deadlock Analysis for Communicating Components

Another property which is important when analysing system behaviours concerns deadlock-freeness. We are interested here in the analysis of deadlock-freeness for communicating components A and B.

Definition 6.1. Let A and B be two asynchronously composable IOTSes.

(1) A *deadlock state* of the synchronous system $A \otimes B$ is a state $(s_A, s_B) \in \mathcal{R}(A \otimes B)$ such that there exists no outgoing transition from (s_A, s_B) in $A \otimes B$. If $A \otimes B$ has no deadlock state then it is *synchronously deadlock-free*, denoted by *df*$(A \otimes B)$.
(2) A *deadlock state* of the asynchronous system $A \otimes_{as} B$ is a state $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$ such that there exists no outgoing transition from $((s_A, q_A), (s_B, q_B))$ in $\Omega(A) \otimes \Omega(B)$. If $A \otimes_{as} B$ has no deadlock state then it is *asynchronously deadlock-free*, denoted by *df*$(A \otimes_{as} B)$.

For finite IOTSes A and B synchronous deadlock-freeness is decidable while asynchronous deadlock-freeness is generally undecidable. In this section we study possibilities for verification of deadlock-freeness for asynchronous systems. First, we want to point out that deadlock-freeness and (weak) asynchronous compatibility are different properties. None of the two implies the other.

Example 6.2.

(1) $A \overset{\mathrm{a}}{\dashleftarrow\dashrightarrow} B$ does not imply *df*$(A \otimes_{as} B)$: We consider two components A and B such that $in_A = \{b\}$, $out_A = \{a\}$, $int_A = \emptyset$ and $in_B = \{a\}$, $out_B = \{b\}$, $int_B = \emptyset$. The transitions of A and B are shown in Fig. 9. Component A is always ready to accept b and B is always ready to accept a but none of the two ever sends a message to the other. Hence $A \overset{\mathrm{a}}{\dashleftarrow\dashrightarrow} B$ (and also $A \overset{\mathrm{a}}{\longleftrightarrow} B$) holds trivially but, since no message is sent, the initial state of $\Omega(A) \otimes \Omega(B)$ is a deadlock state.
(2) *df*$(A \otimes_{as} B)$ does not imply $A \overset{\mathrm{a}}{\dashleftarrow\dashrightarrow} B$: Let A and B be two components with the actions defined in part (1) above. The transitions of A and B are shown in Fig. 10. The asynchronous composition $A \otimes_{as} B$ is deadlock-free since A puts continuously message a in its output queue while B does the same with message b. Since A (resp. B) never

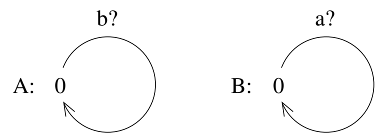

Figure 9: $A \overset{\mathrm{a}}{\leftrightarrow} B$ but <u>not</u> $\mathit{df}(A \otimes_{as} B)$

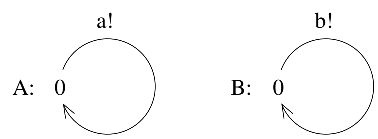

Figure 10: $\mathit{df}(A \otimes_{as} B)$ but <u>not</u> $A \overset{\mathrm{a}}{\leftrightarrow} B$

takes the message addressed to it, the system is not weakly asynchronously compatible (and also not strongly asynchronously compatible).

In the following of this section we assume that we have already checked that A and B are weakly asynchronously compatible and that we now want to prove deadlock-freeness of the asynchronous system. The half-duplex property is again useful for this case. In fact, if the asynchronous system is half-duplex, then deadlock freeness of the asynchronous system is equivalent to deadlock-freeness of the synchronous composition.

Theorem 6.3. *Let A and B be two asynchronously composable and weakly asynchronously compatible IOTSes. If the asynchronous system $A \otimes_{as} B$ is half-duplex, then $\mathit{df}(A \otimes B)$ holds if, and, only if $\mathit{df}(A \otimes_{as} B)$ holds.*

Proof. $\Rightarrow$: Let $((s_A, q_A), (s_B, q_B))$ be an arbitrary state in $\mathcal{R}(\Omega(A) \otimes \Omega(B))$.
Case 1: $q_A \neq \epsilon$ or $q_B \neq \epsilon$. By assumption, $A \overset{\mathrm{a}}{\leftrightarrow} B$. Hence, any element being in one of the queues will be consumed and therefore $((s_A, q_A), (s_B, q_B))$ is not a deadlock state of $A \otimes_{as} B$.
Case 2: Let $q_A = q_B = \epsilon$. Since $A \otimes_{as} B$ is half-duplex we then know, by Lem. 4.2, that $A \otimes_{as} B$ satisfies property $\mathcal{P}(i)$. Therefore $(s_A, s_B) \in \mathcal{R}(A \otimes B)$. Since $\mathit{df}(A \otimes B)$ holds, there exists a transition $(s_A, s_B) \xrightarrow{x}_{A \otimes B} (s'_A, s'_B)$. If x is a non-shared action of A or of B then this transition is induced by a transition of A or B which in turn induces a transition of $\Omega(A) \otimes \Omega(B)$ starting in $((s_A, \epsilon), (s_B, \epsilon))$. In fact, if x is not a shared action of A and B this is clear. If x is a shared action of A and B there are two cases: $x \in \mathit{out}_A \cap \mathit{in}_B$ or $x \in \mathit{out}_B \cap \mathit{in}_A$. W.l.o.g. let $x \in \mathit{out}_A \cap \mathit{in}_B$. Then $(s_A, s_B) \xrightarrow{x}_{A \otimes B} (s'_A, s'_B)$ is induced by transitions $s_A \xrightarrow{x}_A s'_A$ and $s_B \xrightarrow{x}_B s'_B$. The transition $s_A \xrightarrow{x}_A s'_A$ induces a transition $((s_A, \epsilon), (s_B, \epsilon)) \xrightarrow{x^{\triangleright}}_{\Omega(A) \otimes \Omega(B)} ((s'_A, x), (s_B, \epsilon))$. Hence, $((s_A, \epsilon), (s_B, \epsilon))$ is not a deadlock state of $A \otimes_{as} B$. Thus, in all possible cases $((s_A, q_A), (s_B, q_B))$ is not a deadlock state of $A \otimes_{as} B$ and therefore $\mathit{df}(A \otimes_{as} B)$ holds.

$\Leftarrow$: Let (s_A, s_B) be an arbitrary state in $\mathcal{R}(A \otimes B)$. By Lem. 4.8, $((s_A, \epsilon), (s_B, \epsilon))$ belongs to $\mathcal{R}(\Omega(A) \otimes \Omega(B))$. Since $\mathit{df}(A \otimes_{as} B)$ holds, there exists a transition

$$((s_A, \epsilon), (s_B, \epsilon)) \xrightarrow{x}_{\Omega(A) \otimes \Omega(B)} ((s'_A, q_A), (s'_B, q_B)).$$

If x is an action of A or of B which is not shared between A and B, then this transition is induced by a transition of A or of B which in turn induces a transition of $A \otimes B$

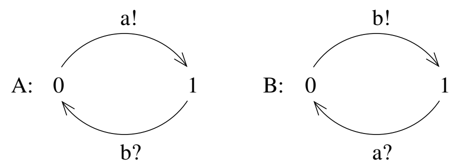

Figure 11: $df(A \otimes_{as} B)$ but not $df(A \otimes B)$

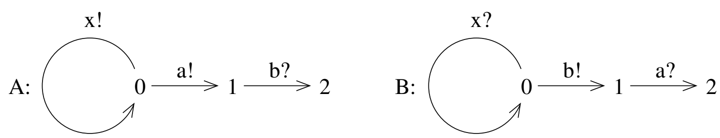

Figure 12: $df(A \otimes B)$ but not $df(A \otimes_{as} B)$

starting in (s_A, s_B). Hence, (s_A, s_B) is not a deadlock state of $A \otimes B$. Otherwise there are four cases: (i) $x \in out_A \cap in_B$, (ii) $x \in out_B \cap in_A$, or (iii) x is of the form $a^{\triangleright}$ with $a \in out_A \cap in_B$ or (iv) x is of the form $b^{\triangleright}$ with $b \in out_B \cap in_A$. Cases (i) and (ii) are not possible since, e.g., case (i) relies on an input action of B which is not possible since the output queue of A is empty. For the remaining two cases we consider, w.l.o.g., case (iii). Then $((s_A, \epsilon), (s_B, \epsilon)) \xrightarrow{a^{\triangleright}}_{\Omega(A)\otimes\Omega(B)} ((s'_A, a), (s_B, \epsilon))$ is induced by a transition $s_A \xrightarrow{a}_A s'_A$ with $a \in out_A \cap in_B$. Since, by assumption, $A \overset{a}{\dashleftarrow\dashrightarrow} B$ holds and $A \otimes_{as} B$ is half-duplex, we know, by Thm. 4.9(2), that $A \dashleftarrow\dashrightarrow B$ holds. Therefore there exist transitions $s_B \xrightarrow{int_B}{}^{*}_{B} \overline{s}_B \xrightarrow{a}_B s'_B$ which induce transitions $(s_A, s_B) \xrightarrow{int_B}{}^{*}_{A\otimes B} (s_A, \overline{s}_B) \xrightarrow{a}_{A\otimes B} (s'_A, s'_B)$. Hence, (s_A, s_B) is not a deadlock state of $A \otimes B$. Thus, in all possible cases (s_A, s_B) is not a deadlock state of $A \otimes B$ and therefore $df(A \otimes B)$ holds. □

The next example shows that Thm. 6.3 would not hold without the half-duplex assumption.

Example 6.4.

(1) $df(A \otimes_{as} B)$ does not imply $df(A \otimes B)$: Let A and B be two components with actions as in Ex. 6.2. The transitions of A and B are shown in Fig. 11. $A \otimes_{as} B$ is not half-duplex. Obviously, $A \otimes_{as} B$ is deadlock-free but $A \otimes B$ is not.

(2) $df(A \otimes B)$ does not imply $df(A \otimes_{as} B)$: Let A and B be two components with the actions as above but with an additional shared action x being an output action of A and an input action of B. The transitions of A and B are shown in Fig. 12. $A \otimes_{as} B$ is not half-duplex. Obviously, $A \otimes B$ is deadlock-free but $A \otimes_{as} B$ is not.

We are now interested in verifying deadlock-freeness in the general case where $A \otimes_{as} B$ is not half-duplex. Similarly to the technique proposed for verifying weak asynchronous compatibility we rely again on a criterion which uses the synchronous products $A \otimes B^{\triangleright}_{out_{BA}}$ and $A^{\triangleright}_{out_{AB}} \otimes B$; see Sect. 5.

Definition 6.5. Let A and B be two asynchronously composable IOTSes. $A \otimes B^{\triangleright}_{out_{BA}}$ is *autonomously deadlock free* if for each reachable state $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ there exists a transition $(s_A, s_B) \xrightarrow{a}_{A\otimes B^{\triangleright}_{out_{BA}}} (s'_A, s'_B)$ with $a \notin in_A \cap out_B$. Autonomous deadlock-freeness of $A^{\triangleright}_{out_{AB}} \otimes B$ is defined analogously.

Note that the condition $a \notin in_A \cap out_B$ is needed in the next theorem to ensure $df(A \otimes_{as} B)$. Otherwise, if we have a transition $(s_A, s_B) \xrightarrow{a}_{A \otimes B^{\rhd}_{out_{BA}}} (s'_A, s'_B)$ with $a \in in_A \cap out_B$, then this would be a free input of A in the composition $A \otimes B^{\rhd}_{out_{BA}}$. But in the asynchronous composition $A \otimes_{as} B$, a would be a shared input which can only be performed if the message a is available in the output queue of B. But this may not be the case and therefore $A \otimes_{as} B$ could be in a state in which it cannot continue with a while $A \otimes B^{\rhd}_{out_{BA}}$ could continue due to the freeness of a. Therefore deadlock-freeness of $A \otimes B^{\rhd}_{out_{BA}}$ or $A^{\rhd}_{out_{AB}} \otimes B$ would not be sufficient and that's why we have introduced the stronger version of autonomous deadlock-freeness above.

Theorem 6.6. *Let A and B be two asynchronously composable and weakly asynchronously compatible IOTSes. If $A \otimes B^{\rhd}_{out_{BA}}$ or $A^{\rhd}_{out_{AB}} \otimes B$ is autonomously deadlock free, then $df(A \otimes_{as} B)$.*

Proof. As in the proof of Thm. 6.3, direction "⇒", the critical cases are states $((s_A, q_A), (s_B, q_B)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$ with $q_A = q_B = \epsilon$. (Otherwise, the assumption $A \overset{a}{\leftrightarrow} B$ guarantees progress.) W.l.o.g. let $A \otimes B^{\rhd}_{out_{BA}}$ be autonomously deadlock free. Since $q_A = \epsilon$, we know, by Lem. 5.4, that $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\rhd}_{out_{BA}})$. Then, by assumption, there exists a transition $(s_A, s_B) \xrightarrow{x}_{A \otimes B^{\rhd}_{out_{BA}}} (s'_A, s'_B)$ with $x \notin in_A \cap out_B$. If x is an action of A or of B which is not shared between A and B, then this transition is induced by a transition of A or of B which in turn induces a transition of $\Omega(A) \otimes \Omega(B)$ starting in $((s_A, \epsilon), (s_B, \epsilon))$. Hence, $((s_A, \epsilon), (s_B, \epsilon))$ is not a deadlock state of $A \otimes_{as} B$. Otherwise (i) $x \in out_A \cap in_B$ or (ii) x is of the form $b^{\rhd}$ with $b \in out_B \cap in_A$. (Note that $x \in in_A \cap out_B$ is not possible due to the assumption.)

(i): If $x \in out_A \cap in_B$, then $(s_A, s_B) \xrightarrow{x}_{A \otimes B^{\rhd}_{out_{BA}}} (s'_A, s'_B)$ is induced by transitions $s_A \xrightarrow{x}_A s'_A$ and $s_B \xrightarrow{x}_{B^{\rhd}_{out_{BA}}} s'_B$. The transition $s_A \xrightarrow{x}_A s'_A$ induces a transition

$$((s_A, \epsilon), (s_B, \epsilon)) \xrightarrow{x^{\rhd}}_{\Omega(A) \otimes \Omega(B)} ((s'_A, x), (s_B, \epsilon)).$$

Hence, $((s_A, \epsilon), (s_B, \epsilon))$ is not a deadlock state of $A \otimes_{as} B$.

(ii): If x is of the form $b^{\rhd}$ with $b \in out_B \cap in_A$, then $(s_A, s_B) \xrightarrow{b^{\rhd}}_{A \otimes B^{\rhd}_{out_{BA}}} (s_A, s'_B)$ is induced by a transition $s_B \xrightarrow{b^{\rhd}}_{B^{\rhd}_{out_{BA}}} s'_B$. Hence, there exists a transition

$$((s_A, \epsilon), (s_B, \epsilon)) \xrightarrow{b^{\rhd}}_{\Omega(A) \otimes \Omega(B)} ((s_A, \epsilon), (s'_B, b))$$

and therefore $((s_A, \epsilon), (s_B, \epsilon))$ is not a deadlock state of $A \otimes_{as} B$. In summary, there is no deadlock state of $A \otimes_{as} B$ and therefore $df(A \otimes_{as} B)$ holds. □

Example 6.7. Consider the two components `MA` and `MB` in Fig. 6. We have shown, in Ex. 5.5, that $\mathtt{MA} \overset{a}{\leftrightarrow} \mathtt{MB}$ holds. It is easy to check that $\mathtt{MA} \otimes \mathtt{MB}^{\rhd}_{\{\mathtt{readyB,failB}\}}$ is autonomously deadlock-free, since in any reachable state an action different from `readyB,failB` can be executed. Therefore, we can apply Th. 6.6 and get $df(\mathtt{MA} \otimes_{as} \mathtt{MB})$.

7. Related Work

Compatibility notions are mostly considered for synchronous systems, since in this case compatibility checking is easier manageable and even decidable if the behaviors of local components have finitely many states. Some approaches use process algebras to study compatibility, like [8] using the π-calculus, others investigate interface theories with binary compatibility relations preserved by refinement, see, e.g., interface automata [1] or modal interfaces [20, 16]. Others consider n-ary compatibility in multi-component systems like, e.g., team automata [9]. A prominent example of multi-component systems with asynchronous communication via unbounded FIFO-buffers are CFSMs [4], for which many problems, like absence of unspecified reception, are undecidable. Exceptions where decidability is ensured are half-duplex systems consisting of two components; see, e.g., [10] and [17], or systems whose network topologies are acyclic; see [15]. Bag structures are typically used for modeling asynchronous communication with Petri nets where the reachability problem, and therefore many compatibility problems [12], are decidable. In [11] decidable topologies are studied for systems which contain both FIFO and bag channels for communication.

There is, however, not much work on relationships between synchronous and asynchronous compatibility. Exceptions are approaches based on *synchronizability* [19] and on *desynchronizability* [6]. Despite of the different terminologies in both cases the idea is to establish a branching bisimulation between the synchronous and the asynchronous versions of a system with message consumption from buffers considered internal. Under the assumption of synchronizability [19] proposes methods to prove compatibility of asynchronously communicating peers by checking synchronous compatibility. The central notion is (synchronous/asynchronous) UR compatibility which corresponds to our weak (synchronous/asynchronous) compatibility plus deadlock-freeness. Comparing our work to [19], obvious differences are that [19] considers multi-component systems while we study compatibility for two components only. On the other hand, [19] considers closed systems while we allow open systems. Also our method for checking asynchronous compatibility is very different. In the first part of our work we rely on half-duplex systems instead of synchronizability and in the second part we drop any assumptions and investigate powerful and decidable criteria for asynchronous compatibility of systems which are neither half-duplex nor synchronizable.

The desynchronization approach in [6, 7] suggests a variant of asynchronous composition which enforces the half-duplex property by blocking outputs to a buffer if there are inputs waiting in the other buffer. [7] shows that for such systems desynchronizability implies freedom of orphans, which means that buffers can always be emptied and therefore no message loss can occur. Moreover, conditions for the synchronous system are provided which characterize desynchronizability[6]. Among them is the condition of well-posedness which coincides with our notion of synchronous strong compatibility. The results in [6, 7] are established for concrete systems, i.e., the underlying components do not have silent transitions. Under this assumption strong and weak synchronous compatibility are the same. Comparing our work to [6, 7], obvious differences are that we allow internal transitions of the underlying components which (a) leads to the notion of weak synchronous compatibility and (b) is also necessary to scale to larger systems where subsystems are asynchronously composed and therefore introduce silent transitions anyway. Moreover, in the first part of our work we

[6]For the definition of a desynchronizable system [6, 7] use a variant of branching bisimulation which is sensitive w.r.t. emptyness of buffers.

do not enforce half-duplex queues (which are not supported by standard implementation technologies either) but study asynchronous systems which have by themselves the half-duplex property. As already mentioned above, in the second part we drop any assumptions such that we can treat also systems which are not desynchronizable.

Another issue concerns checking the correctness of implementations of reactive components against their specifications. Although this is not really a topic of this work, it is still very relevant that compatibility proved on specification level should also hold for (possibly distributed) implementations. Since in practice implementations often use asynchronous message passing, it would be nice if one could check compatibility for *synchronous* composition of specifications and infer from this compatibility for *asynchronous* composition of implementations. A pragmatic solution has been studied in [22] where programming strategies have been proposed to ensure that implementations of reactive components collaborate correctly if their component protocols are compatible in the sense of strong synchronous compatibility as considered here. A formal treatment for implementation correctness using a testing approach has been studied in [21] and, developed further, in [18]. The testing approach for input-output conformance is motivated by the fact that a formal model of a concrete implementation may not be available and therefore the implementation can only be checked by running a set of test cases against it. Since implementations are often only accessible through asynchronous communication channels the observations obtained by testing an implementation rely on an asynchronous interaction. On the other hand, test cases and component specifications can be represented by transition systems with input-output labels and their interaction could be most easily observed by a synchronous composition of the two. Then the question at hand is whether observations obtained by synchronous composition of a set of test cases with a component specification allow us to infer that an implementation tested in an asynchronous environment conforms to the specification. [21] and [18] develop conditions under which such an approach is feasible. Interestingly the conditions are very much related to properties studied in the first part of our paper. For instance, an internal choice input-output labeled transition system in [21, 18] is an I/O-separated transition system according to our Def. 4.4(1). An internal choice test case in the sense of [18] stimulates an implementation-under-test only when "quiescence has been observed", i.e., the implementation does not send itself an output to the test case in the current state. This is similar to condition (3) in our Lem. 4.2 which expresses that two synchronously composed components cannot reach a state in which each of them has an output enabled. We have shown in Lem. 4.2 that this condition characterizes half-duplex systems. Similarly, Lemma 4 in [18] states when executing an internal choice test case on an implementation behaving as internal choice input-output labeled transition system, the input and output queues cannot be empty simultaneously, i.e., we get half-duplex communication. Despite of these technical similarities the goals of our work are quite different since we study safe communication and not correctness of implementations w.r.t. specifications.

Last not least let us point out that the first part of our work is closely related to the study of half-duplex systems by Cécé and Finkel [10]. Due to their decidability result concerning unspecified reception (for two communicating CFSMs) it is not really surprising that we get an effective characterization of asynchronous compatibility and a way to decide it for components with finitely many states. A main difference to [10] is that we consider also synchronous systems and relate their compatibility properties to the asynchronous versions. Moreover, we deal with open systems as well and consider a weak variant of asynchronous compatibility, which we believe adds much power to the strong version. The same differentiation applies to [17]. Finally, as explained above, a significant part of our work deals with systems which are not necessarily half-duplex and this was not an issue in [10].

8. Conclusion

We have proposed techniques to verify asynchronous compatibility and deadlock-freeness of communicating components by using criteria that are based on synchronous composition and hence decidable (if the components are finite state). We have shown that strong (weak) synchronous and strong (weak) asynchronous compatibility are equivalent if the asynchronous system is half-duplex. For non-half duplex systems we have provided decidable conditions which are sufficient for weak asynchronous compatibility.

The IOTSes used here for modeling component behaviors are special cases of modal I/O-transition systems [16], for which synchronous composition and synchronous compatibility checking is implemented in the MIO Workbench [3], an Eclipse-based verification tool. Since the verification conditions studied in this paper involve only synchronous compatibility checking, we can use the MIO Workbench for this purpose.

The most important issues for future research concern (a) the consideration of other compatibility problems, e.g., that a component waiting for some input will eventually get it [5], and (b) the extension of our approach to treat multi-component systems. The latter is a particularly challenging task. For instance, the results on two component half-duplex systems cannot be directly extended since systems with $n > 2$ components and pairwise half-duplex communication have the power of Turing machines; see [10].

Acknowledgment

We are grateful to Alexander Knapp for his suggestion to use output queues (instead of input queues) for the formalization of asynchronous compatibility. Moreover, we would like to thank anonymous reviewers for many useful comments and suggestions.

References

[1] Luca de Alfaro and Thomas A. Henzinger. Interface Automata. Proc. *9th ACM SIGSOFT Ann. Symp. Foundations of Software Engineering (FSE'01)*, 109–120. ACM Press, 2001.

[2] Sebastian S. Bauer, Rolf Hennicker, and Stephan Janisch. Interface Theories for (A)synchronously Communicating Modal I/O-Transition Systems. Proc. *Foundations for Interface Technologies, FIT'10*, EPTCS 46, 1–8, 2010.

[3] Sebastian S. Bauer, Philip Mayer, Andreas Schroeder, and Rolf Hennicker. On Weak Modal Compatibility, Refinement, and the MIO Workbench. Proc. *16th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10)*, Lecture Notes in Computer Science 6015, 175–189. Springer, 2010.

[4] Daniel Brand and Pitro Zafiropulo. On Communicating Finite-State Machines. *J. ACM*, 30(2), 323–342, 1983.

[5] Maurice H. ter Beek, Josep Carmona, Rolf Hennicker, and Jetty Kleijn. Communication Requirements for Team Automata. Proc. *19th IFIP Int. Conf. on Coordination Models and Languages (COORDINATION'17)*, Lecture Notes in Computer Science. Springer, to appear 2017.

[6] Harsh Beohar, and Pieter J. L. Cuijpers. Avoiding Diamonds in Desynchronisation. *Sci. Comput. Program.*, 91, 45-69, 2014.

[7] Harsh Beohar. Refinement of Communication and States in Models of Embedded Systems. Faculty of Mathematics and Computer Science, Technische Universiteit Eindhoven, 2013.

[8] Carlos Canal, Ernesto Pimentel, and José M. Troya. Compatibility and Inheritance in Software Architectures. *Sci. Comput. Program.*, 41(2), 105–138, 2001.

[9] Josep Carmona and Jetty Kleijn. Compatibility in a Multi-component Environment. *Theor. Comput. Sci.*, 484, 1–15, 2013.

[10] Gérard Cécé and Alain Finkel. Verification of Programs with Half-duplex Communication. *Inf. Comput.*, 202(2), 166–190, 2005.

[11] Lorenzo Clemente, Frédéric Herbreteau, and Grégoire Sutre. Decidable Topologies for Communicating Automata with FIFO and Bag Channels. Proc. *25th Int. Conf. on Concurrency Theory (CONCUR'14)*, Lecture Notes in Computer Science 8704, 281–296. Springer, 2014.

[12] Serge Haddad, Rolf Hennicker, and Mikael H. Møller. Channel Properties of Asynchronously Composed Petri Nets. Proc. *Application and Theory of Petri Nets and Concurrency*, Lecture Notes in Computer Science 7927, 369–388. Springer, 2013.

[13] Rolf Hennicker, Michel Bidoit, and Thanh-Son Dang. On Synchronous and Asynchronous Compatibility of Communicating Components. Proc. *18th IFIP Int. Conf. on Coordination Models and Languages (COORDINATION'16)*, Lecture Notes in Computer Science 9686, 138–156. Springer, 2016.

[14] Rolf Hennicker, Stephan Janisch and Alexander Knapp. Refinement of Components in Connection-Safe Assemblies with Synchronous and Asynchronous Communication. *Foundations of Computer Software. Future Trends and Techniques for Development, 15th Monterey Workshop 2008*, Lecture Notes in Computer Science 6028, 154–180. Springer, 2008.

[15] Salvatore La Torre, P. Madhusudan, and Gennaro Parlato. Context-Bounded Analysis of Concurrent Queue Systems. Proc. *14th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*, Lecture Notes in Computer Science 4963, 299–314. Springer, 2008.

[16] Kim Guldstrand Larsen, Ulrik Nyman, and Andrzej Wasowski. Modal I/O Automata for Interface and Product Line Theories. Proc.*16th European Symposium on Programming, ESOP'07*, Lecture Notes in Computer Science 4421, 64–79. Springer, 2007.

[17] Étienne Lozes and Jules Villard. Reliable Contracts for Unreliable Half-Duplex Communications. Proc. *8th International Workshop on Web Services and Formal Methods WS-FM'11*, Lecture Notes in Computer Science 7176, 2–16. Springer, 2011.

[18] Neda Noroozi, Ramtin Khosravi, Mohammad Reza Mousavi, and Tim A. C. Willemse. Synchrony and Asynchrony in Conformance Testing. *Software and System Modeling*, 14 (1), 149-172, 2015.

[19] Meriem Ouederni, Gwen Salaün, and Tevfik Bultan. Compatibility Checking for Asynchronously Communicating Software. Proc. *Formal Aspects of Component Software - 10th International Symposium, FACS'13*, Lecture Notes in Computer Science 8348, 310–328. Springer, 2013.

[20] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud, Axel Legay, and Roberto Passerone. A Modal Interface Theory for Component-based Design. *Fundam. Inform.*, 108 (1-2), 119–149, 2011.

[21] Martin Weiglhofer. Automated Software Conformance Testing. PhD thesis, TU Graz, 2009.

[22] Daniel M. Yellin, and Robert E. Strom. Protocol Specifications and Component Adaptors. *ACM Trans. Program. Lang. Syst.*, 19 (2), 292-333, 1997.

Appendix A. Transitions of $\Omega(A) \otimes \Omega(B)$

- If $a \in in_{\Omega(A)\otimes\Omega(B)}$:
  - $a \in in_A \setminus out_B$ and $s_A \xrightarrow{a}_A s'_A$
    then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A), (s_B, q_B))$,
  - $a \in in_B \setminus out_A$ and $s_B \xrightarrow{a}_B s'_B$
    then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s_A, q_A), (s'_B, q_B))$.

- If $a \in out_{\Omega(A)\otimes\Omega(B)}$:
  - $a \in out_A \setminus in_B$ and $s_A \xrightarrow{a}_A s'_A$
    then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A), (s_B, q_B))$,

– $a \in out_B \setminus in_A$ and $s_B \xrightarrow{a}_B s'_B$
then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s_A, q_A), (s'_B, q_B))$.

- If $a \in int_{A\otimes B} = int_A \cup int_B \cup (out_A \cap in_B) \cup (out_B \cap in_A)$:
  – $a \in int_A$ and $s_A \xrightarrow{a}_A s'_A$
  then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A), (s_B, q_B))$,
  – $a \in int_B$ and $s_B \xrightarrow{a}_B s'_B$
  then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s_A, q_A), (s'_B, q_B))$.
  – $a \in out_A \cap in_B$ (hence $a \in out_{Q_{out_{AB}}}$) and $s_B \xrightarrow{a}_B s'_B$
  then $((s_A, aq_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s_A, q_A), (s'_B, q_B))$,
  – $a \in out_B \cap in_A$ (hence $a \in out_{Q_{out_{BA}}}$) and $s_A \xrightarrow{a}_A s'_A$
  then $((s_A, q_A), (s_B, aq_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A), (s_B, q_B))$.

- If $a^{\triangleright} \in shared(A, B)^{\triangleright} = (out_A \cap in_B)^{\triangleright} \cup (out_B \cap in_A)^{\triangleright}$:
  – $a^{\triangleright} \in (out_A \cap in_B)^{\triangleright}$ (hence $a^{\triangleright} \in in_{Q_{out_{AB}}}$) and $s_A \xrightarrow{a}_A s'_A$
  then $(s_A, q_A) \xrightarrow{a^{\triangleright}}_{\Omega(A)} (s'_A, q_A a)$ and
  then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a^{\triangleright}}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A a), (s_B, q_B))$,
  – $a^{\triangleright} \in (out_B \cap in_A)^{\triangleright}$ (hence $a \in in_{Q_{out_{BA}}}$) and $s_B \xrightarrow{a}_B s'_B$
  then $(s_B, q_B) \xrightarrow{a^{\triangleright}}_{\Omega(B)} (s'_B, q_B a)$ and
  then $((s_A, q_A), (s_B, q_B)) \xrightarrow{a^{\triangleright}}_{\Omega(A)\otimes\Omega(B)} ((s_A, q_A), (s'_B, q_B a))$.

Appendix B. Proofs

Proof of Lemma 4.2:
It remains to prove (3) ⇒ (1): We have to show that for each reachable state in $\mathcal{R}(\Omega(A) \otimes \Omega(B))$ one of the conditions (i), (ii), or (iii) in the definition of property $\mathcal{P}$ is valid. The initial state $((start_A, \epsilon), (start_B, \epsilon))$ satisfies (i). Now assume given an arbitrary transition

$$(*) \qquad ((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q'_A), (s'_B, q'_B))$$

with reachable state $((s_A, q_A), (s_B, q_B))$. It is sufficient to show that for any kind of action $a \in act_{\Omega(A)\otimes\Omega(B)}$, if $((s_A, q_A), (s_B, q_B))$ satisfies one of the conditions (i), (ii), or (iii) then $((s'_A, q'_A), (s'_B, q'_B))$ satisfies (i), (ii), or (iii). The proof is done by case distinction on the form of the action a.

Case 1 : In this case we consider actions $a \in act_A \setminus shared(A, B)$ which can freely occur in A, i.e. without involving B or the output queue of A. This covers the cases $a \in in_A \setminus out_B$, $a \in out_A \setminus in_B$, and $a \in int_A$. In all these cases the transition (*) has the form

$$((s_A, q_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A), (s_B, q_B))$$

and is induced by a transition $s_A \xrightarrow{a}_A s'_A$. It is trivial that, if $((s_A, q_A), (s_B, q_B))$ satisfies (i) ((iii) resp.), then $((s'_A, q_A), (s_B, q_B))$ satisfies (i) ((iii) resp.). If the state $((s_A, q_A), (s_B, q_B))$ satisfies (ii), then $q_A = a_1 \dots a_m \neq \epsilon$ and $q_B = \epsilon$ and there exists $r_A \in states_A$ such that:

$(r_A, s_B) \in \mathcal{R}(A \otimes B)$ and $r_A \stackrel{a_1}{\Longrightarrow}_A \ldots \stackrel{a_m}{\Longrightarrow}_A s_A$. Since $\stackrel{a_m}{\Longrightarrow}_A$ can involve, besides a_m, arbitrary free actions of A and $s_A \stackrel{a}{\longrightarrow}_A s'_A$ is such a free action, we obtain $r_A \stackrel{a_1}{\Longrightarrow}_A \ldots \stackrel{a_m}{\Longrightarrow}_A s'_A$. Thus $((s'_A, q_A), (s_B, q_B))$ satisfies (ii).
Case 2 : In this case we consider actions $a \in act_B \smallsetminus shared(A, B)$ which can freely occur in B, i.e. without involving A or the output queue of B. This case is proved analogously to case 1.
Case 3 : $a \in out_A \cap in_B$. Then the transition (*) has the form

$$((s_A, aq_A), (s_B, q_B)) \stackrel{a}{\longrightarrow}_{\Omega(A) \otimes \Omega(B)} ((s_A, q_A), (s'_B, q_B))$$

and is induced by a transition $s_B \stackrel{a}{\longrightarrow}_B s'_B$. In this case $((s_A, aq_A), (s_B, q_B))$ can only satisfy (ii) such that: $q_A = aa_2 \ldots a_m \neq \epsilon$ and $q_B = \epsilon$ and there exists $r_A \in states_A$ such that $(r_A, s_B) \in \mathcal{R}(A \otimes B)$ and $r_A \stackrel{a}{\Longrightarrow}_A \overline{r}_A \stackrel{a_2}{\Longrightarrow}_{\ldots} \stackrel{a_m}{\Longrightarrow}_A s_A$. Thereby $r_A \stackrel{a}{\Longrightarrow}_A \overline{r}_A$ is of the form $r_A \stackrel{F_A}{\longrightarrow}{}^*_A s \stackrel{a}{\longrightarrow}_A s' \stackrel{F_A}{\longrightarrow}{}^*_A \overline{r}_A$. Since F_A involves only free actions of A (not shared with B), and since $(r_A, s_B) \in \mathcal{R}(A \otimes B)$ we have that $(s, s_B) \in \mathcal{R}(A \otimes B)$. Now the two transitions $s \stackrel{a}{\longrightarrow}_A s'$ and $s_B \stackrel{a}{\longrightarrow}_B s'_B$ synchronize and reach $(s', s'_B) \in \mathcal{R}(A \otimes B)$. Obviously, $s' \stackrel{a_2}{\Longrightarrow}_{\ldots} \stackrel{a_m}{\Longrightarrow}_A s_A$. If $m = 0$ then $q_A = q_B = \epsilon$ and $s' \stackrel{F_A}{\longrightarrow}{}^*_A s_A$. Thus $(s_A, s'_B) \in \mathcal{R}(A \otimes B)$ and condition (i) is valid for $((s_A, q_A), (s'_B, q_B))$. Otherwise, since $(s', s'_B) \in \mathcal{R}(A \otimes B)$ and $s' \stackrel{a_2}{\Longrightarrow}_{\ldots} \stackrel{a_m}{\Longrightarrow}_A s_A$, condition (ii) holds for $((s_A, q_A), (s'_B, q_B))$.
Case 4 : $a \in out_B \cap in_A$. This case is analogous to case 3.
Case 5 : $a^{\triangleright} \in (out_A \cap in_B)^{\triangleright}$. Then the transition (*) has the form

$$((s_A, q_A), (s_B, q_B)) \stackrel{a^{\triangleright}}{\longrightarrow}_{\Omega(A) \otimes \Omega(B)} ((s'_A, q_A a), (s_B, q_B))$$

and is induced by a transition $s_A \stackrel{a}{\longrightarrow}_A s'_A$ with $a \in out_A \cap in_B$.

If $((s_A, q_A), (s_B, q_B))$ satisfies condition (i), then $q_A = q_B = \epsilon$ and $(s_A, s_B) \in \mathcal{R}(A \otimes B)$. Since $s_A \stackrel{a}{\longrightarrow}_A s'_A$ we have $s_A \stackrel{a}{\Longrightarrow}_A s'_A$. Thus, taking $r_A = s_A$ condition (ii) is satisfied for $((s'_A, q_A a), (s_B, q_B))$.

If $((s_A, q_A), (s_B, q_B))$ satisfies condition (ii), then $q_A = a_1 \ldots a_m \neq \epsilon$ and $q_B = \epsilon$ and there exists $r_A \in states_A$ such that: $(r_A, s_B) \in \mathcal{R}(A \otimes B)$ and $r_A \stackrel{a_1}{\Longrightarrow}_A \ldots \stackrel{a_m}{\Longrightarrow}_A s_A$. Since $s_A \stackrel{a}{\longrightarrow}_A s'_A$ we get a sequence $r_A \stackrel{a_1}{\Longrightarrow}_A \ldots \stackrel{a_m}{\Longrightarrow}_A s_A \stackrel{a}{\Longrightarrow}_A s'_A$. Thus $((s'_A, q_A a), (s_B, q_B))$ satisfies condition (ii).

If $((s_A, q_A), (s_B, q_B))$ would satisfy condition (iii), then $q_A = \epsilon$ and $q_B = b_1 \ldots b_m \neq \epsilon$ and there exists $r_B \in states_B$ such that: $(s_A, r_B) \in \mathcal{R}(A \otimes B)$ and $r_B \stackrel{b_1}{\Longrightarrow}_B \overline{r}_B \stackrel{b_2}{\Longrightarrow}_{\ldots} \stackrel{b_m}{\Longrightarrow}_B s_B$. Here $r_B \stackrel{b_1}{\Longrightarrow}_B \overline{r}_B$ has the form $r_B \stackrel{F_B}{\longrightarrow}{}^*_B s \stackrel{b_1}{\longrightarrow}_B s' \stackrel{F_B}{\longrightarrow}{}^*_B \overline{r}_B$. Since F_B involves only free actions of B (not shared with A), and since $(s_A, r_B) \in \mathcal{R}(A \otimes B)$ we get $(s_A, s) \in \mathcal{R}(A \otimes B)$. Now we have two transitions $s_A \stackrel{a}{\longrightarrow}_A s'_A$ with $a \in (out_A \cap in_B)$ and $s \stackrel{b_1}{\longrightarrow}_B s'$ with $b_1 \in (out_B \cap in_A)$ which contradicts the assumption (3). Hence $((s_A, q_A), (s_B, q_B))$ cannot satisfy condition (iii).
Case 6 : $a^{\triangleright} \in (out_B \cap in_A)^{\triangleright}$. This case is analogous to case 5. $\square$

Proof of Lemma 4.8:
The proof is by induction on the length of the derivation of $(s_A, s_B) \in \mathcal{R}(A \otimes B)$. For the initial state $(start_A, start_B)$ of $A \otimes B$ we have $((start_A, \epsilon), (start_B, \epsilon)) \in \mathcal{R}(\Omega(A) \otimes \Omega(B))$. For the induction step it is enough to show that whenever a state $(s_A, s_B) \in \mathcal{R}(A \otimes B)$

satisfies $((s_A,\epsilon),(s_B,\epsilon)) \in \mathcal{R}(\Omega(A)\otimes\Omega(B))$, then for any possible transition

$$(*) \quad (s_A,s_B)\xrightarrow{a}_{A\otimes B}(s'_A,s'_B)$$

the successor state (s'_A,s'_B) satisfies $((s'_A,\epsilon),(s'_B,\epsilon)) \in \mathcal{R}(\Omega(A)\otimes\Omega(B))$. The proof is done by case distinction on the form of the action a.

Case 1 : $a \in act_A \smallsetminus shared(A,B)$. Then the transition (*) has the form

$$(s_A,s_B)\xrightarrow{a}_{A\otimes B}(s'_A,s_B)$$

and is induced by a transition $s_A\xrightarrow{a}_A s'_A$. We assume $((s_A,\epsilon),(s_B,\epsilon)) \in \mathcal{R}(\Omega(A)\otimes\Omega(B))$. Since a is not shared with B, the transition $s_A\xrightarrow{a}_A s'_A$ induces a transition $((s_A,\epsilon),(s_B,\epsilon))\xrightarrow{a}_{\Omega(A)\otimes\Omega(B)}((s'_A,\epsilon),(s_B,\epsilon))$. Since $((s'_A,\epsilon),(s_B,\epsilon)) \in \mathcal{R}(\Omega(A)\otimes\Omega(B))$, (s'_A,s_B) satisfies the desired property.

Case 2 : $a \in act_B \smallsetminus shared(A,B)$. The proof is symmetric to Case 1.

Case 3 : $a \in out_A \cap in_B$. Then the transition (*) is induced by two transition $s_A\xrightarrow{a}_A s'_A$ with $a \in out_A$ and $s_B\xrightarrow{a}_B s'_B$ with $a \in in_B$. We assume $((s_A,\epsilon),(s_B,\epsilon)) \in \mathcal{R}(\Omega(A)\otimes\Omega(B))$. Since $a \in out_A \cap in_B$, we get a transition

$$((s_A,\epsilon),(s_B,\epsilon))\xrightarrow{a^{\triangleright}}_{\Omega(A)\otimes\Omega(B)}((s'_A,a),(s_B,\epsilon))$$

with enqueue action $a^{\triangleright}$. On the other hand, the transition $s_B\xrightarrow{a}_B s'_B$ gives rise to a transition $((s'_A,a),(s_B,\epsilon))\xrightarrow{a}_{\Omega(A)\otimes\Omega(B)}((s'_A,\epsilon),(s'_B,\epsilon))$ with dequeue action a. Since $((s'_A,\epsilon),(s'_B,\epsilon)) \in \mathcal{R}(\Omega(A)\otimes\Omega(B))$, (s'_A,s'_B) satisfies the desired property.

Case 4 : $a \in out_B \cap in_A$. The proof is symmetric to Case 3. □

Proof of Lemma 5.4:

The initial state $((start_A,\epsilon),(start_B,\epsilon))$ satisfies $\mathcal{Q}_A$ and $\mathcal{Q}_B$. Then we consider transitions

$$(*) \quad ((s_A,q_A),(s_B,q_B))\xrightarrow{a}_{\Omega(A)\otimes\Omega(B)}((s'_A,q'_A),(s'_B,q'_B))$$

and show that if $((s_A,q_A),(s_B,q_B))$ satisfies $\mathcal{Q}_A$ and $\mathcal{Q}_B$ then $((s'_A,q'_A),(s'_B,q'_B))$ satisfies $\mathcal{Q}_A$ and $\mathcal{Q}_B$. The proof is performed by case distinction on the form of the action a. Then the result follows by induction on the length of the sequence of transitions to reach an arbitrary state $((s_A,q_A),(s_B,q_B)) \in \mathcal{R}(\Omega(A)\otimes\Omega(B))$. In the following we show that property $\mathcal{Q}_A$ is preserved by transitions (*). For $\mathcal{Q}_B$ the proof is completely analogous.

Case 1 : In this case we consider actions $a \in act_A \smallsetminus shared(A,B)$ which can freely occur in A, i.e. without involving B or the output queue of A. This covers the cases $a \in in_A \smallsetminus out_B$, $a \in out_A \smallsetminus in_B$, and $a \in int_A$. In all these cases the transition (*) has the form

$$((s_A,q_A),(s_B,q_B))\xrightarrow{a}_{\Omega(A)\otimes\Omega(B)}((s'_A,q_A),(s_B,q_B))$$

and is induced by a transition $s_A\xrightarrow{a}_A s'_A$. If $((s_A,q_A),(s_B,q_B))$ satisfies (i), then $q_A=\epsilon$ and $(s_A,s_B) \in \mathcal{R}(A\otimes B^{\triangleright}_{out_{BA}})$. Since $a \in act_A \smallsetminus shared(A,B)$ and A and B are asynchronously composable, $a \in act_A \smallsetminus shared(A,B^{\triangleright}_{out_{BA}})$. Hence, since $s_A\xrightarrow{a}_A s'_A$, also $(s'_A,s_B) \in \mathcal{R}(A\otimes B^{\triangleright}_{out_{BA}})$ and therefore $((s'_A,q_A),(s_B,q_B))$ satisfies (i).

If $((s_A,q_A),(s_B,q_B))$ satisfies (ii), then $q_A = a_1\ldots a_m \neq \epsilon$ and there exists $r_A \in states_A$ such that: $(r_A,s_B) \in \mathcal{R}(A\otimes B^{\triangleright}_{out_{BA}})$ and $r_A \overset{a_1}{\Rrightarrow}_A \ldots \overset{a_m}{\Rrightarrow}_A s_A$. Since $\overset{a_m}{\Rrightarrow}_A$ can involve, besides

a_m, arbitrary actions of A which are not in $out_A \cap in_B$ and $s_A \xrightarrow{a}_A s'_A$ is such a free action, we obtain $r_A \overset{a_1}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s'_A$. Thus $((s'_A, q_A), (s_B, q_B))$ satisfies (ii).

Case 2 : In this case we consider actions $b \in act_B \setminus shared(A, B)$ which can freely occur in B, i.e. without involving A or the output queue of B. This covers the cases $b \in in_B \setminus out_A$, $b \in out_B \setminus in_A$, and $b \in int_B$. In all these cases the transition (*) has the form

$$((s_A, q_A), (s_B, q_B)) \xrightarrow{b}_{\Omega(A) \otimes \Omega(B)} ((s_A, q_A), (s'_B, q_B))$$

and is induced by a transition $s_B \xrightarrow{b}_B s'_B$. If $((s_A, q_A), (s_B, q_B))$ satisfies (i), then $q_A = \epsilon$ and $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$. Since $b \in act_B \setminus shared(A, B)$ and A and B are asynchronously composable, $b \in act_B \setminus shared(A, B^{\triangleright}_{out_{BA}})$. Hence, since $s_B \xrightarrow{b}_B s'_B$, also $s_B \xrightarrow{b}_{B^{\triangleright}_{out_{BA}}} s'_B$ and $(s_A, s'_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$. Therefore $((s_A, q_A), (s'_B, q_B))$ satisfies (i).

If $((s_A, q_A), (s_B, q_B))$ satisfies (ii), then $q_A = a_1 \dots a_m \neq \epsilon$ and there exists $r_A \in states_A$ such that: $(r_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ and $r_A \overset{a_1}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$. Since $s_B \xrightarrow{b}_B s'_B$ involves only a free action of B and hence of $B^{\triangleright}_{out_{BA}}$, $(r_A, s'_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ and therefore $((s_A, q_A), (s'_B, q_B))$ satisfies (ii).

Case 3 : $a \in out_A \cap in_B$. Then the transition (*) has the form

$$((s_A, aq_A), (s_B, q_B)) \xrightarrow{a}_{\Omega(A) \otimes \Omega(B)} ((s_A, q_A), (s'_B, q_B))$$

and is induced by a transition $s_B \xrightarrow{a}_B s'_B$. In this case $((s_A, aq_A), (s_B, q_B))$ can only satisfy (ii) such that: $q_A = aa_2 \dots a_m \neq \epsilon$ and there exists $r_A \in states_A$ such that: $(r_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ and $r_A \overset{a}{\Rrightarrow}_A \overline{r}_A \overset{a_2}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$. Thereby $r_A \overset{a}{\Rrightarrow}_A \overline{r}_A$ is of the form $r_A \xrightarrow{Y_A}{}^{*}_A s \xrightarrow{a}_A s' \xrightarrow{Y_A}{}^{*}_A \overline{r}_A$. Since Y_A involves arbitrary actions of A but no action in $out_A \cap in_B$, and since $(r_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ we have that $(s, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$.[7] Now the transition $s \xrightarrow{a}_A s'$ can synchronize with $s_B \xrightarrow{a}_B s'_B$ and therefore also with $s_B \xrightarrow{a}_{B^{\triangleright}_{out_{BA}}} s'_B$. Thus $(s', s'_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$. Obviously, $s' \overset{a_2}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$. If $m < 2$ then $q_A = \epsilon$ and $s' \xrightarrow{Y_A}{}^{*}_A s_A$. Thus $(s_A, s'_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ and condition (i) is valid for $((s_A, q_A), (s'_B, q_B))$. Otherwise, since $(s', s'_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ and $s' \overset{a_2}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$, condition (ii) holds for $((s_A, q_A), (s'_B, q_B))$.

Case 4 : $a \in out_B \cap in_A$. Then the transition (*) has the form

$$((s_A, q_A), (s_B, aq_B)) \xrightarrow{a}_{\Omega(A) \otimes \Omega(B)} ((s'_A, q_A), (s_B, q_B))$$

and is induced by a transition $s_A \xrightarrow{a}_A s'_A$. If $((s_A, q_A), (s_B, aq_B))$ satisfies (i), then $q_A = \epsilon$ and $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$. Since $s_A \xrightarrow{a}_A s'_A$ and a is not a shared action of A and $B^{\triangleright}_{out_{BA}}$, since $out_{BA} = out_B \cap in_A$ has been renamed to $B^{\triangleright}_{out_{BA}}$, also $(s'_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ and therefore $((s'_A, q_A), (s_B, q_B))$ satisfies (i).

If $((s_A, q_A), (s_B, aq_B))$ satisfies (ii), then $q_A = a_1 \dots a_m \neq \epsilon$ and there exists $r_A \in states_A$ such that: $(r_A, s_B) \in \mathcal{R}(A \otimes B^{\triangleright}_{out_{BA}})$ and $r_A \overset{a_1}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$. Since $s_A \xrightarrow{a}_A s'_A$ and a is not in $out_A \cap in_B$ we get $r_A \overset{a_1}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s'_A$. Thus $((s'_A, q_A), (s_B, q_B))$ satisfies (ii).

[7] Note that the shared actions of A and $B^{\triangleright}_{out_{BA}}$ are $out_A \cap in_B$.

Case 5 : $a^{\rhd} \in (out_A \cap in_B)^{\rhd}$. Then the transition (*) has the form

$$((s_A, q_A), (s_B, q_B)) \xrightarrow{a^{\rhd}}_{\Omega(A)\otimes\Omega(B)} ((s'_A, q_A a), (s_B, q_B))$$

and is induced by a transition $s_A \xrightarrow{a}_A s'_A$ with $a \in out_A \cap in_B$.

If $((s_A, q_A), (s_B, q_B))$ satisfies (i), then $q_A = \epsilon$ and $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\rhd}_{out_{BA}})$. Since $s_A \xrightarrow{a}_A s'_A$ we have $s_A \overset{a}{\Rrightarrow}_A s'_A$. Thus, taking $r_A = s_A$ condition (ii) is satisfied for $((s'_A, q_A a), (s_B, q_B))$.

If $((s_A, q_A), (s_B, q_B))$ satisfies (ii), then $q_A = a_1 \dots a_m \neq \epsilon$ and there exists $r_A \in states_A$ such that: $(r_A, s_B) \in \mathcal{R}(A \otimes B^{\rhd}_{out_{BA}})$ and $r_A \overset{a_1}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$. Since $s_A \xrightarrow{a}_A s'_A$ we get a sequence $r_A \overset{a_1}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A \overset{a}{\Rrightarrow}_A s'_A$. Thus $((s'_A, q_A a), (s_B, q_B))$ satisfies condition (ii).

Case 6 : $b^{\rhd} \in (out_B \cap in_A)^{\rhd} = out^{\rhd}_{BA}$. Then the transition (*) has the form

$$((s_A, q_A), (s_B, q_B)) \xrightarrow{b^{\rhd}}_{\Omega(A)\otimes\Omega(B)} ((s_A, q_A), (s'_B, q_B b))$$

and is induced by a transition $s_B \xrightarrow{b}_B s'_B$ with $b \in out_B \cap in_A = out_{BA}$.

If $((s_A, q_A), (s_B, q_B))$ satisfies (i), then $q_A = \epsilon$ and $(s_A, s_B) \in \mathcal{R}(A \otimes B^{\rhd}_{out_{BA}})$. Since $s_B \xrightarrow{b}_B s'_B$ we have $s_B \xrightarrow{b^{\rhd}}_{B^{\rhd}_{out_{BA}}} s'_B$. Moreover, since A and B are asynchronously composable, $b^{\rhd}$ is not a shared action with A. Hence $(s_A, s'_B) \in \mathcal{R}(A \otimes B^{\rhd}_{out_{BA}})$. Thus $((s_A, q_A), (s'_B, q_B b))$ satisfies (i).

If $((s_A, q_A), (s_B, q_B))$ satisfies (ii), then $q_A = a_1 \dots a_m \neq \epsilon$ and there exists $r_A \in states_A$ such that: $(r_A, s_B) \in \mathcal{R}(A \otimes B^{\rhd}_{out_{BA}})$ and $r_A \overset{a_1}{\Rrightarrow}_A \dots \overset{a_m}{\Rrightarrow}_A s_A$. Since $s_B \xrightarrow{b}_B s'_B$ we have $s_B \xrightarrow{b^{\rhd}}_{B^{\rhd}_{out_{BA}}} s'_B$ and since $b^{\rhd}$ is not a shared action with A we get $(r_A, s'_B) \in \mathcal{R}(A \otimes B^{\rhd}_{out_{BA}})$. Thus $((s_A, q_A), (s'_B, q_B b))$ satisfies (ii). $\square$