Avoiding Diamonds in Desynchronization

Harsh Beohar and Pieter J.L. Cuijpers

Department of Mathematics and Computer science
Eindhoven University of Technology, Eindhoven, The Netherlands
{H.Beohar,P.J.L.Cuijpers}@tue.nl

Abstract. The design of concurrent systems often assumes synchronous communication between different parts of a system. When system components are physically apart, this assumption becomes inappropriate. Desynchronization is a technique that aims to implement a synchronous design in an asynchronous manner by placing buffers between the components of the synchronous design. When queues are used as buffers, the so-called 'diamond property' (among others) ensures correct operation of the desynchronized design. However, this property is difficult to establish in practice. In this paper, we formally prove that the conditions for desynchronizability can be relaxed, and in particular the diamond property is no longer needed, when half-duplex queues are used as a communication buffer. Furthermore, we discuss how the half-duplex condition can be further relaxed when the diamond property can be partially guaranteed.

1 Introduction

Message passing [14] is a programming paradigm in which software components send and receive messages either synchronously or asynchronously. In synchronous communication components must be physically coupled, making it possible to execute corresponding send and receive messages simultaneously. Asynchronous communication is used when components are placed physically apart. The corresponding send and receive messages are then decoupled and the messages travel via a buffer from a sender to its recipient.

A problem with asynchronous communication is that the presence of buffers makes ensuring the correctness of a system a non-trivial task. In general, if the buffers are modeled to have infinite capacity, analyzing correctness of such systems is undecidable [8]. But also, if the buffers are modeled to have finite capacity, we may still face the state-space explosion problem.

It helps to separate concerns by first designing a correct synchronous system and then desynchronizing it. The challenge is then to design the synchronous system in such a way that the addition of communication buffers does not alter its behavior (in any relevant way) [10]. A synchronous system that is not altered by the addition of communication buffers is called *desynchronizable*.

In the context of web-services [4,5], the focus is on effective analysis (like deadlock freedom, choreography analysis) of an asynchronous system by developing

C.S. Păsăreanu and G. Salaün (Eds.): FACS 2012, LNCS 7684, pp. 36–54, 2013.

synchronizability techniques. The idea is to make an asynchronous system synchronous, which is in contrast to desynchronizability, where a synchronous system is made asynchronous. Thus, synchrononizability techniques are applicable when the components of a system are designed under asynchronous communication from the start (for instance, in web-services), whereas desynchronizatbility techniques are applicable when the components of a system are designed under synchronous communication from the start (for instance, in supervisory control). Despite these differences both approaches aim to establish an equivalence between a synchronous system and its asynchronous version. In [4], the authors showed that weak bisimulation between a deterministic synchronous system and its asynchronous system with one place queues is sufficient and necessary for synchronizability modulo weak bisimulation. In this respect, our work differs from [4] by finding conditions solely on a given synchronous system.

In this paper, we show that the conditions *well-posedness*, *independence of external actions*, *input determinism*, and *diamond property* on a synchronous system are necessary and sufficient for desynchronizability. Intuitively,

- two communicating processes in a synchronous system are well-posed if both the processes are able to receive each other requests.
- the external actions (i.e., actions that are not involved in synchronization) are independent in a synchronous system if a receiver can always delay the execution of its own external action in favour of receiving a sequence of messages, without any consequence on its future behavior of the system.
- input determinism states that the communicating processes should not make nondeterministic choices upon the reception of messages.
- the essence of the diamond property is that when two components both wish to communicate a message, say α and β, then communication of α will not block communication of β, and vice versa. Furthermore, the order of communication will not influence the future behavior of the system.

In previous research [3,7,10,21], well-posedness and the diamond property were already present in the sufficient condition for desynchronizability, while the other two properties are new with respect to these references.

As it turns out, the diamond property is difficult to establish in practice, while in particular well-posedness and input determinism can be easily obtained by construction, at least for supervisory control synthesis [18].

As an example why this leads to practical problems, consider a simplified model of a controlled drive-motor [11]. The drive-motor can move in a forward direction '*fwmove*' or in a backward direction '*bwmove*', and it has a signal *chdir* indicating when it is safe to change this direction. A controller communicates with the drive-motor to ensure that the event '*chdir*' always executes before altering the direction of the motor. The models of the drive motor, the controller, and the synchronous system are shown in Fig. 1, where $!a$ ($?a$) denotes that an action a is sent (received) and a denotes the synchronization of $!a$ and $?a$.

Observe in the synchronous system of drive-motor that the execution of the event *chdir* from state 1 to state 2 disables the execution of the event *fwmove*; thus, violating the commutativity of the traces *chdir.fwmove* and *fwmove.chdir*.

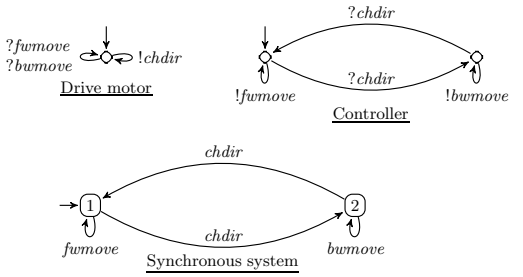

Fig. 1. An illustration showing the impossibility of establishing the diamond property in certain synchronous systems

Similarly, the commutativity of the traces *chdir.bwmove* and *bwmove.chdir* is prohibited in this synchronous system. As a result, the synchronous system in Fig. 1 does not satisfy the diamond property. In fact, the control requirement implicitly requires the diamond property to be broken. Therefore, it is impossible to desynchronize this system unless we adapt the model of plant or supervisor, or we adapt the way in which the desynchronization is performed.

Studying the origin of the diamond property, we notice that it is caused by the type of buffer that is used for communication. The authors of [3,10,21] follow [8] in taking two unidirectional FIFO queues as a means of communication. In [10] a separate unidirectional FIFO queue is used for each type of message, which effectively leads to a *bag*-type of buffering (cf. [7]). Both types of buffer are useful abstractions of a physical communication layer with a protocol layer on top. For example, queues nicely represent the use of the TCP/IP protocol, while bags represent a UDP-like protocol [20]. Note that both approaches require the diamond property, essentially because both approaches allow the messages α and β to be present in the buffer at the same time and arrive in arbitrary order.

Our research hypothesis is that it may be possible to find better desynchronizability conditions by changing the properties of the communication protocol. So far, research has focussed on the properties that the communicating components should have in order to ensure desynchronizability. The buffer is usually taken to be a queue or, incidentally, a bag. In this paper, we reconsider these properties, and alter them by changing the communication protocol if desired.

A first step in that direction is shown in this paper. We prove that the troublesome diamond property can be avoided by changing the type of buffer used for desynchronization to so-called *half-duplex* communication (also used in [9] for model-checking asynchronous systems). In the context of two communicating processes, half-duplex communication means that a component is only allowed to send a message when its input buffer is empty. As a result, the buffering between the two processes alternates in each direction, having to become empty before alternating. We show that in this case a synchronous composition is desynchronizable if and only if it is *well-posed*, *independent of external actions*, and *input deterministic*. These properties are generally weaker than the properties in [3,7,10,21], and we are able to give a general method to adapt systems that are synthesized using supervisory control theory to satisfy these properties. It is

our hope that this paper will initiate discussion on the separation of concerns regarding desynchronization. Our use of a half-duplex buffering strategy indicates that the communication protocol is essential in this separation.

Admittedly, the choice for half-duplex communication is an odd one from the perspective of efficiency. The half-duplex protocol essentially makes components wait for each other, which makes communication slow. In Section 5, we sketch a first step to remedy this by recognizing when actions are independent of each-other. Independent messages satisfy the diamond property and can therefore be processed in a full-duplex way. However, more research is needed to complete this claim and to find out when the half-duplex condition is in fact necessary for desynchronizability, and when it can be dropped for the sake of efficiency.

The methods we use for studying desynchronizability in this paper stem from process algebra and concurrency theory (see e.g. [2]). We do not fix a set of desirable properties a priori, but rather aim for desynchronizability modulo a behavioral equivalence that preserves a large set of possibly desirable properties. The desynchronizability question is therefore posed as: *given two processes p and s, under which conditions are the synchronous composition and the asynchronous composition of p and s behaviorally equivalent?* To be as general as possible, we take *branching bisimulation* as our behavioral equivalence of choice, which is the strongest equivalence used in concurrency theory [12].

Organisation of the Paper. In Section 2, we describe the mathematical notations and formal definitions required to define desynchronizability using two unidirectional FIFO buffers. Section 3 discusses necessary and sufficient conditions for desynchronizability, including the unwanted diamond property. In Section 4, we show how the diamond property can be eliminated by using half-duplex buffers for desynchronization. Lastly, Section 5 discusses ways to relax this half-duplex condition and apply desynchronization in the context of supervisory control.

2 Basic Definitions

In this paper, we model the world as a single transition system in which all behaviors of interest are represented. Components of a system as well as their compositions are called *processes* and are represented by pointing out an *initial state* $q \in \mathbb{P}$ in the labelled transition system. A *process* q is then formed by all reachable states from the initial state $q \in \mathbb{P}$.

Definition 1. *A* labelled transition system *is a tuple* $(\mathbb{P}, A, \rightarrow, \sqcup)$*, where*

- $\mathbb{P}$ *is a set of states.*
- A *is a set of actions.*
- $\rightarrow \subseteq \mathbb{P} \times A \times \mathbb{P}$ *is a transition relation.*
- $\sqcup \subseteq \mathbb{P}$ *is the empty-buffer predicate and its purpose is to observe the states of an asynchronous system that consists of empty buffer contents.*

The notation $q \xrightarrow{\alpha} q'$ *denotes an element* $(q, \alpha, q') \in \rightarrow$*, the notation* $q\sqcup$ *denotes that state* q *satisfies the empty-buffer predicate. For a given initial state* $q \in \mathbb{P}$*, the set of* reachable states $\mathfrak{R}(q)$ *is defined as the smallest set such that:*

– $q \in \mathfrak{R}(q)$, *and*
– $\forall_{q_1,q_2 \in \mathbb{P}} \forall_{\alpha \in A} . \left[\left(q_1 \in \mathfrak{R}(q) \ \wedge q_1 \xrightarrow{\alpha} q_2 \right) \ \Rightarrow \ q_2 \in \mathfrak{R}(q) \right]$.

In what follows, the letter q and its decorations like $q', q_1, q_2, \cdots$ are used to reason about the arbitrary processes, whereas the letters p, s and their corresponding decorations are reserved for special purposes (see the next paragraph).

Considering a synchronous system as depicted in Fig. 2, we identify two basic components p, s, which we assume to be processes in our labeled transition system. These processes are composed into a synchronous process $p \parallel s$. The process $p \parallel s$ can perform four kinds of events; namely, the *external* actions of p and s that belong to the sets E_p and E_s, respectively, and *messages* from p and s that belong to the sets M_p and M_s, respectively.

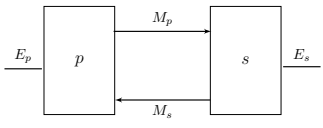

Fig. 2. A synchronous system

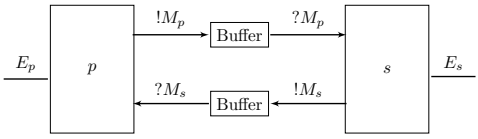

Fig. 3. An asynchronous system

When the system is *desynchronized* we obtain an asynchronous system as depicted in Fig. 3, consisting of the same processes p, s, which are now composed into an asynchronous process $p \,|[\epsilon, \epsilon]|\, s$ (with ϵ indicating initially empty buffer contents). In the asynchronous process, the external actions of p and s remain the same, but we now make a distinction between the sending of a message (modeled for p by the set $!M_p = \{!m \mid m \in M_p\}$) and the receiving of that message (modeled for p by the set $?M_p = \{?m \mid m \in M_p\}$). We assume that the so obtained sets of actions are all part of our alphabet and are all pairwise disjoint: $E_p \uplus E_s \uplus M_p \uplus !M_p \uplus ?M_p \uplus M_s \uplus !M_s \uplus ?M_s \subseteq A$.

Assuming that the processes p and s are already part of our labeled transition system, where p makes use of the actions $!M_p \uplus ?M_p \uplus E_p$ and s makes use of the actions $!M_s \uplus ?M_s \uplus E_s$, we can define the synchronous and asynchronous composition of p and s through structural operational semantic rules (SOS) on the states of the transition system [17]. The premise of each rule states the assumption on the states of the composed processes, and the conclusion gives the resulting transition for the composed state.

In Table 1, we give the SOS rules for synchronous composition and asynchronous composition using two unidirectional lossless FIFO queues. The notation $p \,|[\mu, \nu]|\, s$ denotes the asynchronous composition of states p and s with sequences of messages $\nu \in M_p^*$ and $\mu \in M_s^*$ in the respective queues. Note how the empty-buffer predicate is always true for synchronous compositions, while it is only true for asynchronous compositions if both queues are empty.

As explained in the introduction, a composition $p \parallel s$ is desynchronizable if it is equivalent to its asynchronous composition $p \,|[\epsilon, \epsilon]|\, s$. One problem with defining equivalence between the two is that asynchronous composition needs two actions for the communication of a message while synchronous composition

Table 1. SOS rules for synchronous and asynchronous parallel composition

$$\frac{p_1 \xrightarrow{!m} p_2,\ s_1 \xrightarrow{?m} s_2,\ m \in M_p}{p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_2} \qquad \frac{p_1 \xrightarrow{?n} p_2,\ s_1 \xrightarrow{!n} s_2,\ n \in M_s}{p_1 \parallel s_1 \xrightarrow{n} p_2 \parallel s_2}$$

$$\frac{p_1 \xrightarrow{e} p_2,\ e \in E_p}{p_1 \parallel s_1 \xrightarrow{e} p_2 \parallel s_1} \qquad \frac{s_1 \xrightarrow{e} s_2,\ e \in E_s}{p_1 \parallel s_1 \xrightarrow{e} p_1 \parallel s_2} \qquad \frac{}{(p \parallel s)\sqcup}$$

$$\frac{p \xrightarrow{!m} p',\ m \in M_p}{(p\,|[\mu,\nu]|\,s) \xrightarrow{!m} (p'\,|[\mu,\nu.m]|\,s)} \qquad \frac{s \xrightarrow{!n} s',\ n \in M_s}{(p\,|[\mu,\nu]|\,s) \xrightarrow{!n} (p\,|[\mu.n,\nu]|\,s')}$$

$$\frac{p \xrightarrow{?n} p',\ \mu = n.\mu',\ n \in M_s}{(p\,|[\mu,\nu]|\,s) \xrightarrow{?n} (p'\,|[\mu',\nu]|\,s)} \qquad \frac{s \xrightarrow{?m} s',\ \nu = m.\nu',\ m \in M_p}{(p\,|[\mu,\nu]|\,s) \xrightarrow{?m} (p\,|[\mu,\nu']|\,s')}$$

$$\frac{p \xrightarrow{e} p',\ e \in E_p}{(p\,|[\mu,\nu]|\,s) \xrightarrow{e} (p'\,|[\mu,\nu]|\,s)} \qquad \frac{s \xrightarrow{e} s',\ e \in E_s}{(p\,|[\mu,\nu]|\,s) \xrightarrow{e} (p\,|[\mu,\nu]|\,s')} \qquad \frac{}{(p\,|[\epsilon,\epsilon]|\,s)\sqcup}$$

only needs one. The usual process algebraic way to solve this issue is by defining an abstraction scheme, translating certain actions from the asynchronous system to actions from the synchronous system while hiding others.

In Table 2, we define the abstraction operator $\Delta()$ that maps all the send-messages of the asynchronous system to communicated messages in the synchronous system, while the receive-messages are mapped to a so-called *internal action*, denoted by τ. Subsequently, we define *branching bisimulation* (see [2,12]) as an equivalence between processes that abstracts from internal actions.

Table 2. SOS rules for the abstraction operator $\Delta()$

$$\frac{x_1 \xrightarrow{!m} x_2,\ m \in M_p \cup M_s}{\Delta(x_1) \xrightarrow{m} \Delta(x_2)} \qquad \frac{x_1 \xrightarrow{e} x_2,\ e \in E_p \cup E_s}{\Delta(x_1) \xrightarrow{e} \Delta(x_2)} \qquad \frac{x_1 \xrightarrow{?m} x_2,\ m \in M_p \cup M_s}{\Delta(x_1) \xrightarrow{\tau} \Delta(x_2)} \qquad \frac{x\sqcup}{\Delta(x)\sqcup}$$

Definition 2. *The* reachability relation $\longrightarrow\!\!\!\!\!\rightarrow \subseteq \mathbb{P} \times A^* \times \mathbb{P}$ *is derived from the transition relation* $\rightarrow$ *as the* smallest *relation satisfying:*

$$q_1 \xrightarrow{\epsilon}\!\!\!\!\!\rightarrow q_1 \quad , \quad \frac{q_1 \xrightarrow{w}\!\!\!\!\!\rightarrow q',\ q' \xrightarrow{\tau} q_2}{q_1 \xrightarrow{w}\!\!\!\!\!\rightarrow q_2} \quad , \quad \frac{q_1 \xrightarrow{w}\!\!\!\!\!\rightarrow q',\ q' \xrightarrow{\alpha} q_2,\ \alpha \neq \tau}{q_1 \xrightarrow{w.\alpha}\!\!\!\!\!\rightarrow q_2} \quad .$$

Definition 3. *A binary relation $\mathcal{B} \subseteq \mathbb{P} \times \mathbb{P}$ on the states of the transition system is a* branching bisimulation *relation iff the following conditions are satisfied.*

- $\forall_{q,q_1,q',\alpha}.\Big[\Big((q,q') \in \mathcal{B} \wedge q \xrightarrow{\alpha} q_1\Big) \Rightarrow (\alpha = \tau \wedge (q_1,q') \in \mathcal{B}) \vee \exists_{q'_1,q'_2}.\big[q' \xrightarrow{\epsilon}\!\!\!\!\rightarrow\, q'_1 \xrightarrow{\alpha} q'_2 \wedge (q,q'_1) \in \mathcal{B} \wedge (q_1,q'_2) \in \mathcal{B}\big]\Big]$;
- $\forall_{q,q'}.\Big[\Big((q,q') \in \mathcal{B} \wedge q \sqcup\Big) \Rightarrow \exists_{q''}.\Big[q' \xrightarrow{\epsilon}\!\!\!\!\rightarrow\, q'' \wedge q'' \sqcup \wedge (q,q'') \in \mathcal{B}\Big]\Big]$;
- $\forall_{q,q',q'_1,\alpha}.\Big[\Big((q,q') \in \mathcal{B} \wedge q' \xrightarrow{\alpha} q'_1\Big) \Rightarrow (\alpha = \tau \wedge (q,q'_1) \in \mathcal{B}) \vee \exists_{q_1,q_2}.\big[q \xrightarrow{\epsilon}\!\!\!\!\rightarrow\, q_1 \xrightarrow{\alpha} q_2 \wedge (q_1,q') \in \mathcal{B} \wedge (q_2,q'_1) \in \mathcal{B}\big]\Big]$;
- $\forall_{q,q'}.\Big[\Big((q,q') \in \mathcal{B} \wedge q' \sqcup\Big) \Rightarrow \exists_{q''}.\Big[q \xrightarrow{\epsilon}\!\!\!\!\rightarrow\, q'' \wedge q'' \sqcup \wedge (q'',q') \in \mathcal{B}\Big]\Big]$.

Two processes q and q' are said to be branching bisimilar*, denoted $q \leftrightarrows_{\mathbf{b}} q'$, if there exists a branching bisimulation relation $\mathcal{B}$ such that $(q,q') \in \mathcal{B}$.*

Now we have all the preliminaries that are necessary to define what desychronization formally means.

Definition 4. *A synchronous system $p \parallel s$ is* desynchronizable *if*

$$p \parallel s \leftrightarrows_{\mathbf{b}} \Delta(p\,|[\epsilon,\epsilon]|\,s).$$

3 Properties of Desynchronizable Systems

In this section, we prove a number of properties of desynchronizable systems modulo branching bisimulation. A new result (cf. [3,7,10,21]) is that the observation of the empty-buffer predicate makes that these properties are necessary as well as sufficient for desynchronizability. A technical assumption used to show necessity in this case, is that the desynchronized systems p and s are *concrete*, meaning they do not have internal behavior themselves.

Definition 5. *A process $q \in \mathbb{P}$ is* concrete *if $\nexists_{q',q''}.\Big[q' \in \mathfrak{R}(q) \wedge q' \xrightarrow{\tau} q''\Big]$. A transition $q_1 \xrightarrow{\tau} q_2$ is* inert *modulo $\leftrightarrows_{\mathbf{b}}$ iff $q_1 \leftrightarrows_{\mathbf{b}} q_2$.*

Lemma 1. *Let $p \parallel s$ be a concrete and desynchronizable system. Then, all the τ-transitions in $\Delta(p\,|[\epsilon,\epsilon]|\,s)$ are inert modulo branching bisimulation.*

Proof. Since $p \parallel s$ is a concrete process, none of the τ-transitions in the asynchronous system can be matched by any related state in the synchronous system. Thus, all τ-transitions in the asynchronous system have to be inert [2]. □

A key step in understanding the necessary conditions for desynchronizability, is to see that any reachable state $p' \parallel s' \in \mathfrak{R}(p \parallel s)$ of some desynchronizable system $p \parallel s$ is desynchronizable itself. This property seems both desirable and

trivial, but its proof turned out to be more involved than expected. In particular, the proof turns out to rely on the chosen abstraction scheme, the fact that p and s are concrete processes, disjointness of the message sets, and the fact that we observe the empty-buffer predicate.

Theorem 1. *Let $p \parallel s$ be concrete and desynchronizable, then any $p' \parallel s' \in \mathfrak{R}(p \parallel s)$ is desynchronizable.*

Proof. As a base case, the initial state of $p \parallel s$ is desynchronizable by assumption. By induction, assume that we have a reachable desynchronizable state $p' \parallel s' \in \mathfrak{R}(p \parallel s)$ and consider any p'' and s'' with $p' \parallel s' \xrightarrow{\alpha} p'' \parallel s''$. Following the SOS rules, one of the following transitions must exist in the asynchronous process:

1. a transition $\Delta(p' \,|[\epsilon,\epsilon]|\, s') \xrightarrow{\alpha} \Delta(p'' \,|[\epsilon,\alpha]|\, s')$ with $\alpha \in M_p$, and a hidden transition $\Delta(p'' \,|[\epsilon,\alpha]|\, s') \xrightarrow{\tau} \Delta(p'' \,|[\epsilon,\epsilon]|\, s'')$ that is inert because $p \parallel s$ is concrete, i.e. $\Delta(p'' \,|[\epsilon,\alpha]|\, s') \leftrightarrows_{\mathbf{b}} \Delta(p'' \,|[\epsilon,\epsilon]|\, s'')$;
2. a transition $\Delta(p' \,|[\epsilon,\epsilon]|\, s') \xrightarrow{\alpha} \Delta(p' \,|[\alpha,\epsilon]|\, s'')$ with $\alpha \in M_s$, and a hidden transition $\Delta(p' \,|[\alpha,\epsilon]|\, s'') \xrightarrow{\tau} \Delta(p'' \,|[\epsilon,\epsilon]|\, s'')$ that is inert because $p \parallel s$ is concrete, i.e. $\Delta(p' \,|[\alpha,\epsilon]|\, s') \leftrightarrows_{\mathbf{b}} \Delta(p'' \,|[\epsilon,\epsilon]|\, s'')$;
3. a transition $\Delta(p' \,|[\epsilon,\epsilon]|\, s') \xrightarrow{\alpha} \Delta(p'' \,|[\epsilon,\epsilon]|\, s'')$ with $\alpha \in E_p$ ($\alpha \in E_s$), in which case we find that $s' = s''$ ($p' = p''$).

Because $p' \parallel s' \leftrightarrows_{\mathbf{b}} \Delta(p' \,|[\epsilon,\epsilon]|\, s')$, the properties of branching bisimulation (applied to concrete processes) dictate that we can relate those asynchronous transitions to synchronous transitions. I.e. there exist p''' and s''' such that $p' \parallel s' \xrightarrow{\alpha} p''' \parallel s'''$ and $p''' \parallel s''' \leftrightarrows \Delta(p'' \,|[\epsilon,\epsilon]|\, s'')$. Finally, to prove that this implies that $p'' \parallel s''$ is desynchronizable, we study the relation:

$$\mathcal{S} = \big\{(p_1 \parallel s_1, p_2 \parallel s_2) \mid\ p_1 \parallel s_1 \in \mathfrak{R}(p'' \parallel s'') \wedge \Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2) \in \mathfrak{R}(\Delta(p'' \,|[\epsilon,\epsilon]|\, s'')) \wedge p_1 \parallel s_1 \leftrightarrows_{\mathbf{b}} \Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2)\big\}$$

It remains to show that this is a witnessing branching bisimulation relation for $p''' \parallel s''' \leftrightarrows_{\mathbf{b}} p'' \parallel s''$. For this, consider the following cases.

1. Let $p_1 \parallel s_1 \xrightarrow{m} p_3 \parallel s_3$, $(p_1 \parallel s_1, p_2 \parallel s_2) \in \mathcal{S}$, and $m \in M_p$ (the case when $m \in M_s$ is symmetric). By construction of $\mathcal{S}$ we have $p_1 \parallel s_1 \leftrightarrows_{\mathbf{b}} \Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2)$. By applying concreteness and disjointness of M_p and M_s and the transfer condition of branching bisimulation we get $\Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2) \xrightarrow{m} \Delta(p_4 \,|[\epsilon,m]|\, s_2)$ and $p_3 \parallel s_3 \leftrightarrows_{\mathbf{b}} \Delta(p_4 \,|[\epsilon,m]|\, s_2)$ for some $p_4 \in \mathbb{P}$. Since $p_3 \parallel s_3 \sqcup$, branching bisimulation gives us $\Delta(p_4 \,|[\epsilon,m]|\, s_2) \xrightarrow{\tau} \Delta(p_4 \,|[\epsilon,\epsilon]|\, s_4)$ and $p_3 \parallel s_3 \leftrightarrows_{\mathbf{b}} \Delta(p_4 \,|[\epsilon,\epsilon]|\, s_4)$, for some $s_4 \in \mathbb{P}$. Thus, we derive $p_2 \xrightarrow{!m} p_4$ and $s_2 \xrightarrow{?m} s_4$; hence, $p_2 \parallel s_2 \xrightarrow{m} p_4 \parallel s_4$ and $(p_3 \parallel s_3, p_4 \parallel s_4) \in \mathcal{S}$.
2. Let $p_1 \parallel s_1 \xrightarrow{e} p_3 \parallel s_1$, $(p_1 \parallel s_1, p_2 \parallel s_2) \in \mathcal{S}$, and $e \in E_p$ (the case when $e \in E_s$ is symmetric). By construction of $\mathcal{S}$ we have $p_1 \parallel s_1 \leftrightarrows_{\mathbf{b}} \Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2)$. By concreteness and disjointness of E_p and E_s and the transfer condition of branching bisimulation we get $\Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2) \xrightarrow{e} \Delta(p_4 \,|[\epsilon,\epsilon]|\, s_2)$ and $p_3 \parallel s_1 \leftrightarrows_{\mathbf{b}} \Delta(p_4 \,|[\epsilon,\epsilon]|\, s_2)$. Thus, $p_2 \parallel s_2 \xrightarrow{e} p_4 \parallel s_2$ and $(p_3 \parallel s_1, p_4 \parallel s_2) \in \mathcal{S}$.

3. The cases where the transitions originates from $p_2 \parallel s_2$ when $(p_1 \parallel s_1, p_2 \parallel s_2) \in \mathcal{S}$ can be proved along the above lines.

Finally, by transitivity and symmetry, we get $p'' \parallel s'' \underline{\leftrightarrow}_{\mathbf{b}} \Delta(p'' \,|[\epsilon,\epsilon]|\, s'')$. □

Corollary 1. *If p, s, p', s' are concrete processes and $p \parallel s \underline{\leftrightarrow}_{\mathbf{b}} \Delta(p' \,|[\epsilon,\epsilon]|\, s')$ then $p \parallel s \underline{\leftrightarrow}_{\mathbf{b}} p' \parallel s'$.*

3.1 Well-Posedness

The first actual implication of desynchronizability that we would like to discuss, is that a desynchronizable system is always *well-posed*. This was already observed in [10] for desynchronizability modulo failure equivalence. Well-posedness means that whenever a process p would like to send a message, s should be willing to receive it and vice versa. In a synchronous composition such messages may be blocked, but in an asynchronous composition they lead to *orphans*, i.e., messages that remain forever in the buffer. In turn, orphans lead to deadlocking communication (except in a few pathological cases).

Definition 6. *A binary relation $\mathcal{W} \subseteq \mathbb{P} \times \mathbb{P}$ is called a* well-posedness *relation iff the following conditions are satisfied.*

1. $\forall_{p,s,p',m}.\Big[p \xrightarrow{!m} p' \wedge (p,s) \in \mathcal{W} \Rightarrow \exists_{s'}.\big[s \xrightarrow{?m} s'\big] \wedge \forall_{s'}.\big[s \xrightarrow{?m} s' \Rightarrow (p',s') \in \mathcal{W}\big]\Big]$,
2. $\forall_{p,s,p',e\in E_p}.\Big[p \xrightarrow{e} p' \wedge (p,s) \in \mathcal{W} \Rightarrow (p',s) \in \mathcal{W}\Big]$,
3. *Respectively Conditions 1 and 2 with the role of p and s interchanged.*

A composition $p \parallel s$ is well-posed *if there exists a well-posedness relation $\mathcal{W}$ such that $(p,s) \in \mathcal{W}$.*

Theorem 2. *If $p \parallel s$ is concrete and desynchronizable then it is well-posed.*

Proof. Define a relation $\mathcal{W} = \{(p_1, s_1) \mid \Delta(p_1 \,|[\epsilon,\epsilon]|\, s_1) \in \mathfrak{R}(\Delta(p \,|[\epsilon,\epsilon]|\, s))\}$. To show that $\mathcal{W}$ is a well-posedness relation, let $p_1 \xrightarrow{\alpha} p_2$ and $(p_1, s_1) \in \mathcal{W}$.

1. Let $\alpha \in !M_p$. Then, by the construction of $\mathcal{W}$ we have $\Delta(p_1 \,|[\epsilon,\epsilon]|\, s_1) \in \mathfrak{R}(\Delta(p \,|[\epsilon,\epsilon]|\, s))$ and using $p_1 \xrightarrow{\alpha} p_2$ we get $\Delta(p_1 \,|[\epsilon,\epsilon]|\, s_1) \xrightarrow{m} \Delta(p_2 \,|[\epsilon,m]|\, s_1)$. Since $p \parallel s$ is desynchronizable, we know that there exists $q \in \mathfrak{R}(p \parallel s)$ such that $q \underline{\leftrightarrow}_{\mathbf{b}} \Delta(p_2 \,|[\epsilon,m]|\, s_1)$. Clearly, we have $q\sqcup$. Furthermore by the transfer property of branching bisimulation and under the assumption of concrete processes we get $\exists s_2.\Big[\Delta(p_2 \,|[\epsilon,m]|\, s_1) \xrightarrow{\tau} \Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2) \wedge \Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2)\sqcup\Big]$. Thus, $s_1 \xrightarrow{?m} s_2$. Next, we need to show that for every s_2', whenever $s_1 \xrightarrow{?m} s_2'$ then $(p_2, s_2') \in \mathcal{W}$. So let $s_1 \xrightarrow{?m} s_2'$, thus $\Delta(p_2 \,|[\epsilon,m]|\, s_1) \xrightarrow{\tau} \Delta(p_2 \,|[\epsilon,\epsilon]|\, s_2')$. Hence, by the construction of $\mathcal{W}$ it is clear that $(p_2, s_2') \in \mathcal{W}$.
2. Let $\alpha \in E_p$. Then, by the construction of $\mathcal{W}$ we have $\Delta(p_1 \,|[\epsilon,\epsilon]|\, s_1) \in \mathfrak{R}(\Delta(p \,|[\epsilon,\epsilon]|\, s))$ and using the above transition we get $\Delta(p_1 \,|[\epsilon,\epsilon]|\, s_1) \xrightarrow{e} \Delta(p_2 \,|[\epsilon,\epsilon]|\, s_1)$. Clearly, $(p_2, s_1) \in \mathcal{W}$.

Likewise, the symmetric case can be proved for the process s_1. □

3.2 Independence of External Actions

The second implication of desynchronizability that we would like to discuss is *independence of external actions*. Intuitively, it means that a receiver can always delay the execution of its own external action in favor of receiving a sequence of messages from the other process, without any consequence on its future behavior modulo $\leftrightarrows_{\mathbf{b}}$. The reception of messages becomes *independent* of the external behavior in this way.

In the following, we define independence on the composition $p \parallel s$ rather than on the separate processes p and s because we aim for necessary conditions. The pathological case in which a process p is not independent in a part of its state-space that becomes unreachable when interacting with s has no effects on desynchronizability. Of course, independence of external actions of the separate processes would be a natural part of a sufficient condition for desynchronizability.

Definition 7. *A synchronous system $p \parallel s$ is* independent of external actions modulo $\leftrightarrows_{\mathbf{b}}$ *if the following conditions holds for every $p_1 \parallel s_1 \in \mathfrak{R}(p \parallel s)$.*

1. $\forall_{p_2,p_2',s_2,u,e}.\Big[e \in E_p \wedge u \in (M_s \cup E_s)^* \wedge p_1 \parallel s_1 \xrightarrow{e} p_2 \parallel s_1 \xrightarrow{u}\!\!\!\!\rightarrow\, p_2' \parallel s_2 \Rightarrow$
$\exists_{p_3,p_3'}.\Big[p_1 \parallel s_1 \xrightarrow{u}\!\!\!\!\rightarrow\, p_3 \parallel s_2 \xrightarrow{e} p_3' \parallel s_2 \wedge p_3 \parallel s_2 \leftrightarrows_{\mathbf{b}} p_3' \parallel s_2\Big]\Big]$.
2. $\forall_{p_2,s_2,s_2',v,e}.\Big[e \in E_s \wedge v \in (M_p \cup E_p)^* \wedge p_1 \parallel s_1 \xrightarrow{e} p_1 \parallel s_2 \xrightarrow{v}\!\!\!\!\rightarrow\, p_2 \parallel s_2' \Rightarrow$
$\exists_{s_3,s_3'}.\Big[p_1 \parallel s_1 \xrightarrow{v}\!\!\!\!\rightarrow\, p_2 \parallel s_3 \xrightarrow{e} p_2 \parallel s_3' \wedge p_2 \parallel s_2' \leftrightarrows_{\mathbf{b}} p_2 \parallel s_3'\Big]\Big]$.

Theorem 3. *If $p \parallel s$ is concrete and desynchronizable then it is independent of external actions modulo $\leftrightarrows_{\mathbf{b}}$.*

Proof. Let $x = x_p \uplus x_s$, $yM = yM_p \uplus yM_s$, $x \in \{M, E\}$, and $y \in \{!, ?\}$. By abuse of notations, define two renaming functions $! : (M \cup E)^* \to (!M \cup E)^*$, $? : (M \cup E)^* \to ?M^*$ and a projection function $^-$: $(M \cup E)^* \to M^*$:

1. $?\epsilon = \epsilon$, $?(e.w) = w$, $?(m.w) = ?m.?w$, where $e \in E$ and $w \in (M \cup E)^*$.
2. $!\epsilon = \epsilon$, $!(e.w) = e.!w$, $!(m.w) = !m.!w$, where $e \in E$ and $w \in (M \cup E)^*$.
3. $\bar{\epsilon} = \epsilon$, $\overline{e.w} = \bar{w}$, and $\overline{m.w} = m.\bar{w}$, where $e \in E$ and $w \in (M \cup E)^*$.

Now, assume we have a reachable (Theorem 1) desynchronizable state $p_1 \parallel s_1 \in \mathfrak{R}(p \parallel s)$ with solid transitions as in Fig. 4, where $e \in E_p$ and $u \in (M_s \cup E_s)^*$. Using the above renaming functions and the semantics, we derive $p_1 \xrightarrow{e} p_2$, $s_1 \xrightarrow{!u}\!\!\!\!\rightarrow\, s_2$, and $p_2 \xrightarrow{?u}\!\!\!\!\rightarrow\, p_2'$. As well-posedness is necessary for desynchronizability, we may use it to obtain $p_1 \xrightarrow{?u}\!\!\!\!\rightarrow\, p_3$ (for some p_3). Thus, we get $p_1 \parallel s_1 \xrightarrow{u}\!\!\!\!\rightarrow\, p_3 \parallel s_2$ (dashed in Fig. 4). From these transitions we then derive the solid transitions in the asynchronous system depicted in Fig. 4, where $\mu = \bar{u}$.

Since τ-transitions are inert we have $\Delta(p_1 \,|[\mu, \epsilon]|\, s_2) \leftrightarrows_{\mathbf{b}} \Delta(p_3 \,|[\epsilon, \epsilon]|\, s_2)$. Branching bisimulation, under the assumption of concrete processes and disjointness of the sets E_p, E_s, gives us the existence of p_3' such that $\Delta(p_3 \,|[\epsilon, \epsilon]|\, s_2) \xrightarrow{e}$

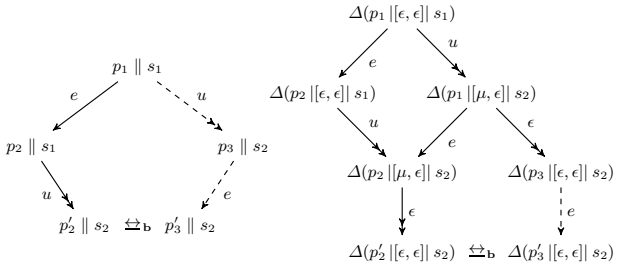

Fig. 4. The role of independence of external actions

$\Delta(p'_3 \,|[\epsilon,\epsilon]|\, s_2)$ and $\Delta(p'_3 \,|[\epsilon,\epsilon]|\, s_2) \ \underline{\leftrightarrow}_{\mathbf{b}} \ \Delta(p_2 \,|[\mu,\epsilon]|\, s_2)$. Thus, by the SOS-rules we get $p_3 \parallel s_2 \xrightarrow{e} p'_3 \parallel s_2$. Next, we need to show that $p'_2 \parallel s_2 \ \underline{\leftrightarrow}_{\mathbf{b}} \ p'_3 \parallel s_2$. From above we have $\Delta(p'_3 \,|[\epsilon,\epsilon]|\, s_2) \ \underline{\leftrightarrow}_{\mathbf{b}} \ \Delta(p_2 \,|[\mu,\epsilon]|\, s_2)$ and since τ-transition are inert we get $\Delta(p_2 \,|[\mu,\epsilon]|\, s_2) \ \underline{\leftrightarrow}_{\mathbf{b}} \ \Delta(p'_2 \,|[\epsilon,\epsilon]|\, s_2)$. Thus, by transitivity we get $\Delta(p'_3 \,|[\epsilon,\epsilon]|\, s_2) \ \underline{\leftrightarrow}_{\mathbf{b}} \ \Delta(p'_2 \,|[\epsilon,\epsilon]|\, s_2)$. By Theorem 1 we get $p'_3 \parallel s_2 \ \underline{\leftrightarrow}_{\mathbf{b}} \ \Delta(p'_3 \,|[\epsilon,\epsilon]|\, s_2)$, $p'_2 \parallel s_2 \ \underline{\leftrightarrow}_{\mathbf{b}} \ \Delta(p'_2 \,|[\epsilon,\epsilon]|\, s_2)$, from which we ultimately conclude $p'_2 \parallel s_2 \ \underline{\leftrightarrow}_{\mathbf{b}} \ p'_3 \parallel s_2$. Likewise, Condition 2 of Definition 7 can be proved. □

3.3 Input Determinism

The next implication of desynchronizability, is that desynchronizable systems should be *input deterministic*. In other words, the synchronous system $p \parallel s$ should not make non-deterministic choices upon the reception of messages. It may perform non-deterministic external behavior, and it may also be non-deterministic when sending messages. The reason for this, is that desynchronization *delays* any non-deterministic choice on the input.

Like in the case of independence of external actions, we define the condition input-determinism on the synchronous process $p \parallel s$ rather than on the individual processes p and s (cf. [1]) because we are aiming for necessary conditions. As before, input-determinism of the individual processes would be a natural part of a sufficient condition for input-determinism of the composition.

Definition 8. *A synchronous system $p \parallel s$ is* input deterministic modulo $\underline{\leftrightarrow}_{\mathbf{b}}$ *if every reachable state $p_1 \parallel s_1 \in \mathfrak{R}(p \parallel s)$ satisfies the following conditions.*

1. *for all p_2, s_2, p_3, whenever $p_1 \parallel s_1 \xrightarrow{u}\!\!\!\!\rightarrow p_2 \parallel s_2$ and $p_1 \parallel s_1 \xrightarrow{u}\!\!\!\!\rightarrow p_3 \parallel s_2$ for some $u \in (M_s \cup E_s)^*$, then $p_2 \parallel s_2 \ \underline{\leftrightarrow}_{\mathbf{b}} \ p_3 \parallel s_2$.*
2. *for all p_2, s_2, s_3, whenever $p_1 \parallel s_1 \xrightarrow{v}\!\!\!\!\rightarrow p_2 \parallel s_2$ and $p_1 \parallel s_1 \xrightarrow{v}\!\!\!\!\rightarrow p_2 \parallel s_3$ for some $v \in (M_p \cup E_p)^*$, then $p_2 \parallel s_2 \ \underline{\leftrightarrow}_{\mathbf{b}} \ p_2 \parallel s_3$.*

Theorem 4. *Let $p \parallel s$ be concrete and desynchronizable, then it is input deterministic modulo $\underline{\leftrightarrow}_{\mathbf{b}}$.*

Proof. Pick a reachable state $p_1 \parallel s_1 \in \mathfrak{R}(p \parallel s)$ (see Fig. 5) such that $p_1 \parallel s_1 \xrightarrow{u}\!\!\!\!\rightarrow p_2 \parallel s_2$ and $p_1 \parallel s_1 \xrightarrow{u}\!\!\!\!\rightarrow p_3 \parallel s_2$, for some $u \in (M_s \cup E_s)^*, p_2, p_3, s_2 \in \mathbb{P}$. By Theorem 1 we have $p_1 \parallel s_1 \leftrightarrows_{\mathbf{b}} \Delta(p_1 \,|[\epsilon, \epsilon]|\, s_1)$. Using the renaming functions from Theorem 3 we have $s_1 \xrightarrow{!u}\!\!\!\!\rightarrow s_2$ and $p_1 \xrightarrow{?u}\!\!\!\!\rightarrow p_2$ and $p_1 \xrightarrow{?u}\!\!\!\!\rightarrow p_3$. For the asynchronous system we then find the transitions as shown in Fig. 5, where $\mu = \bar{u}$. As $p \parallel s$ is concrete, all τ-transitions in the asynchronous system are inert, so we get $\Delta(p_1 \,|[\mu, \epsilon]|\, s_2) \leftrightarrows_{\mathbf{b}} \Delta(p_2 \,|[\epsilon, \epsilon]|\, s_2) \leftrightarrows_{\mathbf{b}} \Delta(p_3 \,|[\epsilon, \epsilon]|\, s_2)$. Finally,

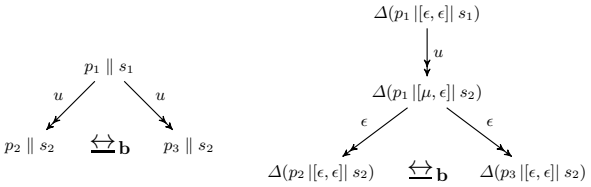

Fig. 5. The role of input-determinism

using Theorem 1 twice we ultimately get $p_2 \parallel s_2 \leftrightarrows_{\mathbf{b}} p_3 \parallel s_2$. Likewise Condition 2 of Definition 8 can be proved. □

3.4 The Diamond Property

The final implication of desynchronizability that we would like is the *diamond property*. Intuitively, the diamond property says that sending a message from one component does not disable the sending of message from the other component. Moreover, any order of execution leads to behaviorally equivalent states.

Definition 9. *A synchronous system $p \parallel s$ has the* diamond property modulo $\leftrightarrows_{\mathbf{b}}$ *if for every reachable state $p_1 \parallel s_1$ and transitions $p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_2$ and $p_1 \parallel s_1 \xrightarrow{n} p_3 \parallel s_3$ with $m \in M_p$ and $n \in M_s$ there exist transitions $p_2 \parallel s_2 \xrightarrow{n} p_4 \parallel s_4$ and $p_3 \parallel s_3 \xrightarrow{m} p_5 \parallel s_5$ with $p_4 \parallel s_4 \leftrightarrows_{\mathbf{b}} p_5 \parallel s_5$.*

Theorem 5. *Let $p \parallel s$ be concrete and desynchronizable, then $p \parallel s$ has the diamond property modulo $\leftrightarrows_{\mathbf{b}}$.*

Proof. Assume a state $p_1 \parallel s_1 \in \mathfrak{R}(p \parallel s)$, $p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_2$ $(m \in M_p)$ and $p_1 \parallel s_1 \xrightarrow{n} p_3 \parallel s_3$ $(n \in M_s)$, as depicted in Fig. 6. From Theorem 1 we know that $p_i \parallel s_i \leftrightarrows_{\mathbf{b}} p_i \,|[\epsilon, \epsilon]|\, s_i$, for $i \in \{1, 2, 3\}$. From the SOS rules we get $p_1 \xrightarrow{!m} p_2$, $s_1 \xrightarrow{?m} s_2$, $p_1 \xrightarrow{?n} p_3$, and $s_1 \xrightarrow{!n} s_3$. Using these transitions we find the transitions at the state $\Delta(p_1 \,|[\epsilon, \epsilon]|\, s_1)$ as shown in Fig. 6.

Since τ-transitions are inert we get $\Delta(p_2 \,|[\epsilon, m]|\, s_1) \leftrightarrows_{\mathbf{b}} \Delta(p_3 \,|[\epsilon, \epsilon]|\, s_3)$. And from Theorem 1 we get $p_2 \parallel s_2 \leftrightarrows_{\mathbf{b}} \Delta(p_2 \,|[\epsilon, \epsilon]|\, s_2)$. Thus, by transitivity we have $p_2 \parallel s_2 \leftrightarrows_{\mathbf{b}} p_2 \,|[\epsilon, m]|\, s_1$. And, the transfer conditions of branching bisimulation gives the dashed transition labeled n shown in Fig. 6 with $p_4 \parallel s_4 \leftrightarrows_{\mathbf{b}} \Delta(p_2 \,|[n, m]|\, s_2)$. Likewise we derive the dashed transition labeled m in Fig. 6 with $p_5 \parallel s_5 \leftrightarrows_{\mathbf{b}} p_2 \,|[n, m]|\, s_3$. Finally, by transitivity $p_4 \parallel s_4 \leftrightarrows_{\mathbf{b}} p_5 \parallel s_5$. □

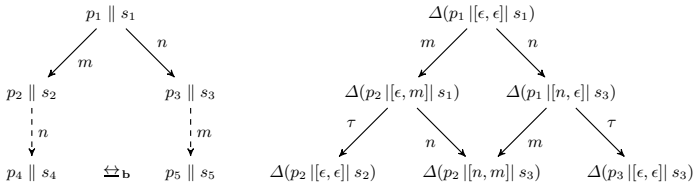

Fig. 6. The role of the diamond property

3.5 Sufficient Conditions for Desynchronizability

Conversely, the four necessary conditions that we discussed in the previous subsections, together form a sufficient condition for desynchronizability.

Theorem 6. *Let $p \parallel s$ be concrete, well-posed, independent of external actions, input deterministic, and have the diamond property, then $p \parallel s \leftrightarrows_{\mathbf{b}} \Delta(p \,|[\epsilon,\epsilon]|\, s)$.*

Proof. See [6]. □

4 Half-Duplex Communication Eliminates the Diamonds

In the previous section, we showed that the diamond property is a necessary condition for desynchronizability, while we expressed a desire in the introduction to desynchronize systems that do not possess this property as well. This leads us to rethink our model of desynchronization.

Changing the notion of equivalence or the observation of the predicate is not likely to help. Previous research [3,10] has been performed on weaker notions of equivalence, and although the diamond property was not identified as a necessary condition there, it did come up as a natural sufficient condition that the authors could not work around. This is why we decided to experiment with the properties of the buffer instead.

Inspired by the observation that the problem occurs when both communicating parties would like to send a message at the same time, we decided to see if *half-duplex communication*, in which only one party can communicate at a time, would give a solution. We model half-duplex communication between processes p and s as a process $p \,|[\epsilon,\epsilon]|_{\mathrm{h}}\, s$, of which the structured operational semantics are given in Table 3. Observe that the rules are similar to those we used before, except that either the left or the right queue remains empty at all times.

Definition 10. *A synchronous system $p \parallel s$ is* half-duplex desynchronizable *if $p \parallel s \leftrightarrows_{\mathbf{b}} \Delta(p \,|[\epsilon,\epsilon]|_{\mathrm{h}}\, s)$.*

Next, we find that the diamond property can be dropped from the necessary and sufficient conditions.

Theorem 7. *Let $p \parallel s$ be concrete and half-duplex desynchronizable, then it is well-posed, independent of external actions, and input deterministic.*

Table 3. SOS rules for asynchronous systems with half-duplex queues

$$\frac{p \xrightarrow{!m} p'}{(p\,|[\epsilon,\nu]|_{\mathrm{h}}\,s) \xrightarrow{!m} (p'\,|[\epsilon,\nu.m]|_{\mathrm{h}}\,s)} \qquad \frac{s \xrightarrow{!n} s'}{(p\,|[\mu,\epsilon]|_{\mathrm{h}}\,s) \xrightarrow{!n} (p\,|[\mu.n,\epsilon]|_{\mathrm{h}}\,s')}$$

$$\frac{p \xrightarrow{?n} p',\ \mu = n.\mu',\ n \in M_s}{(p\,|[\mu,\nu]|_{\mathrm{h}}\,s) \xrightarrow{?n} (p'\,|[\mu',\nu]|_{\mathrm{h}}\,s)} \qquad \frac{s \xrightarrow{?m} s',\ \nu = m.\nu',\ m \in M_p}{(p\,|[\mu,\nu]|_{\mathrm{h}}\,s) \xrightarrow{?m} (p\,|[\mu,\nu']|_{\mathrm{h}}\,s')}$$

$$\frac{p \xrightarrow{e} p',\ e \in E_p}{(p\,|[\mu,\nu]|_{\mathrm{h}}\,s) \xrightarrow{e} (p'\,|[\mu,\nu]|_{\mathrm{h}}\,s)} \qquad \frac{s \xrightarrow{e} s',\ e \in E_s}{(p\,|[\mu,\nu]|_{\mathrm{h}}\,s) \xrightarrow{e} (p\,|[\mu,\nu]|_{\mathrm{h}}\,s')} \qquad \frac{}{(p\,|[\epsilon,\epsilon]|_{\mathrm{h}}\,s)\sqcup}.$$

Proof. Along the same lines as the proofs in the previous section. □

Theorem 8. *Suppose a concrete process $p \parallel s$ is well-posed, independent of external actions, and input deterministic, then it is half-duplex desynchronizable.*

Proof. See [6]. □

5 Discussion

5.1 Relaxing the Half-Duplex Condition

As already mentioned, the half-duplex mechanism leads to an inefficient design of an asynchronous system because a sender is prohibited to send messages while its input queue is non-empty. Moreover, we required half-duplex communication because we could not guarantee the diamond property for our synchronous system. In essence, the half-duplex property ensures a certain level of synchronization over the communication buffer. Half-duplex communication, namely, can only be implemented if some kind of semaphore is in place on top of the physical layer.

Table 4. SOS rules for semi-duplex communication over a set I

$$\frac{p \xrightarrow{!m} p',\ m \in M_p,\ (\mu \in I^* \vee m \in I)}{(p\,|[\mu,\nu]|\,s) \xrightarrow{!m} (p'\,|[\mu,\nu.m]|_I\,s)} \qquad \frac{s \xrightarrow{!n} s',\ n \in M_s,\ (\nu \in I^* \vee n \in I)}{(p\,|[\mu,\nu]|\,s) \xrightarrow{!n} (p\,|[\mu.n,\nu]|_I\,s')}$$

$$\frac{p \xrightarrow{?n} p',\ \mu = n.\mu',\ n \in M_s}{(p\,|[\mu,\nu]|\,s) \xrightarrow{?n} (p'\,|[\mu',\nu]|_I\,s)} \qquad \frac{s \xrightarrow{?m} s',\ \nu = m.\nu',\ m \in M_p}{(p\,|[\mu,\nu]|\,s) \xrightarrow{?m} (p\,|[\mu,\nu']|_I\,s')}$$

$$\frac{p \xrightarrow{\alpha} p',\ \alpha \in E_p}{(p\,|[\mu,\nu]|\,s) \xrightarrow{\alpha} (p'\,|[\mu,\nu]|_I\,s)} \qquad \frac{s \xrightarrow{\alpha} s',\ \alpha \in E_s}{(p\,|[\mu,\nu]|\,s) \xrightarrow{\alpha} (p\,|[\mu,\nu]|_I\,s')} \qquad \frac{}{(p\,|[\epsilon,\epsilon]|_I\,s)\sqcup}$$

Now, suppose that we do have the diamond property for certain pairs of actions in the synchronous system. In such a case, a specialized semaphore could be put in place that verifies whether there are actions in the incoming buffer that conflict with a specific outgoing action. For example, suppose we can identify a subset $I \subseteq M_p \cup M_s$ of actions that satisfy the diamond property with respect to all other messages in $M_p \cup M_s$. As long as there are only actions from I in the buffer, it is safe to send any message, and at any time it is safe to send actions from I. Such a type of communication is captured in the SOS rules of Table 4.

We conjecture that the necessary and sufficient conditions for desynchronization using such a buffer are well-posedness, independence of external actions, input determinism, and the diamond property for pairs of messages modeled by the set I. We actually expect the proof to be along the same lines of Theorem 6.

However, before going into detailed proofs of such theorems, we would like to point out that the selection of a semi-duplex buffering strategy does not only depend on the particular diamonds that can be proven, but also on the particular kinds of semaphores / semi-duplex buffering strategies that are implementable. If we want to distinguish different classes of messages that share the diamond property, we also need to use different semaphores to ensure the associated semi-duplex buffer (reminiscent of [16]). Which semaphores are actually implementable is highly dependent on the application domain, so we would like to concentrate future research on finding out which possibilities we have in practice (in our case, in practical cases of supervisory control) to put semaphores on a communication buffer.

5.2 Desynchronization in Supervisory Control

Regarding supervisory control theory, we should still check whether the conditions we have gotten so far are reasonable. That is the topic of this subsection.

Supervisory control theory [18] aims at controlling the behavior of a *plant* p to fit a requirement r by synthesizing a *supervisor* s such that $p \parallel s \leftrightarrows_{\mathbf{b}} r$. For this purpose, a plant and its supervisor perform two kinds of actions: *controllable* and *uncontrollable* actions. We model uncontrollable actions as the send messages from a plant to its supervisor, while the controllable actions are modeled as the send messages from the supervisor to the plant.

To make supervisory control synthesis feasible, it is usually assumed that p is deterministic. The result of the synthesis is then also a deterministic s.

In order to synthesize a supervisor that is well-posed, consider the procedure of taking the process $p \parallel s$ and renaming all communication actions to send-actions if they originated from s and to receive actions if they originated from p. In other words, define a function $\gamma : \mathbb{P} \to \mathbb{P}$ such that

$$\gamma(m) = \begin{cases} !m & ;\text{if } m \in M_s \\ ?m & ;\text{if } m \in M_p \\ m & ;\text{otherwise} \end{cases}$$

and consider the process $\gamma(p \parallel s)$ defined using the SOS rules of Table 5.

Table 5. SOS rules for renaming using a function γ

$$\frac{p \xrightarrow{m} p'}{\gamma(p) \xrightarrow{\gamma(m)} \gamma(p')} \qquad \frac{p\!\sqcup}{\gamma p\!\sqcup}$$

We obtain the following theorem, which gives us a well-posed and input-deterministic supervisor for p.

Theorem 9. *If p and s are deterministic, then p and $\gamma(p \parallel s)$ are well-posed, $p \parallel \gamma(p \parallel s)$ is input-deterministic, and $p \parallel s \leftrightarrows_{\mathbf{b}} p \parallel \gamma(p \parallel s) \leftrightarrows_{\mathbf{b}} r$.*

Proof. It is easy, but tedious, to verify that well-posedness of $p \parallel \gamma(p \parallel s)$ follows from the witnessing relation $\mathcal{W} = \{(p_1, \gamma(p_1 \parallel s_1)) \mid p_1 \parallel s_1 \in \mathfrak{R}(p \parallel s)\}$, while the fact that we have constructed a valid supervisor is witnessed by the branching bisimulation relation $\mathcal{B} = \{(p_1 \parallel s_1, p_1 \parallel \gamma(p_1 \parallel s_1)) \mid p_1 \parallel s_1 \in \mathfrak{R}(p \parallel s)\}$. Both witnesses rely on determinism of p and s, but we have to leave out the details for reasons of space. Obviously, if p and s are deterministic so is $\gamma(p \parallel s)$ (using disjointness of the message sets), hence it is input-deterministic. □

The issue of ensuring independence of external actions is more involved. Intuitively, independence of external actions says that an external action can always be delayed in favor of an internal communication. Of course, since the role of a supervisor is just to limit the behavior of the plant, it has no direct need for external actions. However, if the plant's communication is dependent on the external behavior – for example, external behavior is processed with higher priority than internal communication – desynchronizability is still at risk.

5.3 Desynchronization of Non-Concrete Synchronous Systems

In this subsection, we focus on the desynchronization of synchronous systems that allow τ-transitions in their definitions.

The introduction of τ-transitions in a synchronous system makes it impossible to know from the semantics whether the process p or[1] s performed a τ-transition, whenever the synchronous system $p \parallel s$ executes the τ-transition. Such an information is vital in the definition of witnessing branching bisimulation relation between a synchronous system, and its asynchronous version.

One way to circumvent this problem is by renaming the label τ of every τ-transitions present in the processes p and s by the labels τ_p and τ_s, respectively. Furthermore, by assuming that the labels τ_p, τ_s are present in the external actions of the processes p, s, respectively, the conditions of Theorem 6 (Theorem 8) can still be used to assert whether a non-concrete synchronous system is desynchronizable (half-duplex desynchronizable) or not. However, despite this

[1] The word 'or' is used in the exclusive sense.

soundness result, more research is required in order to examine to what extent are these conditions necessary in the absence of concreteness assumption.

5.4 Conclusions

In this paper, we studied necessary and sufficient conditions for desynchronizability modulo branching bisimulation, and we showed that reverting to half-duplex communication, or variants of it, can help in avoiding a troublesome condition known as the diamond property. To the best of our knowledge, this is the first characterization of desynchronizability modulo branching bisimulation; moreover, the previous works (cf. [3,7,10,21]) on weaker equivalences focused on giving sufficient conditions for desynchronizability.

Our results indicate that the study of desynchronizability should no longer focus on the properties one needs to retain equivalence of behavior in a certain communication context, but rather should focus on changing the communication context in such a way that these properties actually become attainable. Furthermore, we have shown that reasonable desynchronizability results can be obtained even for the finest equivalence in the van Glabbeek spectrum. Perhaps some of the necessary conditions can be relaxed by weakening this equivalence. For example, we know that we can eliminate the need for input determinism by studying desynchronizability modulo contra-simulation [12]. But so far the properties obtained using weaker equivalences are very similar to the ones we found, which indicates that there is not much to be gained there.

Another observation we made is that the choice of abstraction scheme is crucial in obtaining useful results. On the one hand, if we had chosen to abstract from outputs rather than from inputs in our definition of the operator $\Delta()$, there would have been an additional necessary condition saying that at any reachable state of $p \parallel s$ only one send-transition is allowed (the details of this are outside the scope of this paper, see [6]). On the other hand, we obtained interesting results in [7] using an abstraction scheme that abstracted from send- and receive actions from the plant using bags as a communication buffer, but that abstraction scheme did not work out for queues.

For deterministic supervisory control, we showed that it is possible to synthesize a controller that satisfies the well-posedness property by construction. For other systems, however, this may not be so easy. Therefore, it would be beneficial if tools for model checking asynchronous systems, like mCRL2 [13] and CADP [15], could be optimized to check for well-posedness as well.

Finally, we observe a similarity between our work and the work on choreographies and contracts, which turns out to be useful in model checking of asynchronous systems [4,5,19]. Basically, such choreographies serve to restrict the occurrence of diamonds in an asynchronous system, which means that it becomes synchronizable [5]. Perhaps it is also possible to use this idea in the other direction, i.e., to desynchronize a system using a choreography on the communication buffer. It would be interesting to see if, for example, the proposed semi-duplex buffer discussed in Section 5 can be implemented using a choreography.

Acknowledgements. The authors thank the anonymous reviewers for their feedbacks on an earlier draft of this paper. The authors also thank Jos Baeten, Koos Rooda, Bert van Beek, and Damian Nadales, for various discussions regarding this work and for putting us on the track of this problem.

This work has been performed as part of the "Integrated Multi-formalism Tool Support for the Design of Networked Embedded Control Systems" (MULTIFORM) project, supported by the Seventh Research Framework Programme of the European Commission (Grant agreement number: INFSO-ICT-224249).

References

1. Alfaro, L., Henzinger, T.: Interface-Based Design. In: Broy, M., Grünbauer, J., Harel, D., Hoare, C.A.R. (eds.) Engineering Theories of Software Intensive Systems. NATO Science Series, vol. 195, pp. 83–104. Springer Netherlands (2005)
2. Baeten, J.C.M., Basten, T., Reniers, M.A.: Process Algebra: Equational Theories of Communicating Processes, 1st edn. Cambridge University Press, New York (2009)
3. Balemi, S.: Control of Discrete Event Systems: Theory And Application. Ph.D. thesis, Swiss Federal Institute of Technology, Automatic Control Laboratory, ETH Zurich (May 1992)
4. Basu, S., Bultan, T.: Choreography conformance via synchronizability. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, pp. 795–804. ACM, New York (2011)
5. Basu, S., Bultan, T., Ouederni, M.: Synchronizability for Verification of Asynchronously Communicating Systems. In: Kuncak, V., Rybalchenko, A. (eds.) VMCAI 2012. LNCS, vol. 7148, pp. 56–71. Springer, Heidelberg (2012)
6. Beohar, H.: Refinement of communication and states in models of embedded systems. Ph.D. thesis, Eindhoven university of technology (in preparation)
7. Beohar, H., Cuijpers, P.J.L.: Desynchronizability of (partial) synchronous closed loop systems. Scientific Annals of Computer Science 21, 5–38 (2011)
8. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM 30, 323–342 (1983)
9. Cécé, G., Finkel, A.: Verification of programs with half-duplex communication. Inf. Comput. 202, 166–190 (2005)
10. Fischer, C., Janssen, W.: Synchronous Development of Asynchronous Systems. In: Montanari, U., Sassone, V. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 735–750. Springer, Heidelberg (1996)
11. Forschelen, S.T.J.: Supervisory control of theme park vehicles. Master's thesis, Eindhoven University of Technology, System Engineering Group, Dept. of Mechanical Engineering (2010)
12. van Glabbeek, R.J.: The Linear Time - branching Time Spectrum II. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 66–81. Springer, Heidelberg (1993)
13. Groote, J.F., Mathijssen, A., Reniers, M., Usenko, Y., van Weerdenburg, M.: The formal specification language mCRL2. In: MMOSS 2006 (2006)
14. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM 21(8), 666–677 (1978)
15. Mateescu, R.: Specification and Analysis of Asynchronous Systems using CADP, pp. 141–169. ISTE (2010), http://dx.doi.org/10.1002/9780470611012.ch5

16. Peters, K., Schicke, J.-W., Nestmann, U.: Synchrony vs causality in asynchronous pi-calculus. In: Luttik, B., Valencia, F. (eds.) 18th International Workshop on Expressiveness in Concurrency, EXPRESS. EPTCS, vol. 64, pp. 89–103 (2011)
17. Plotkin, G.D.: A Structural Approach to Operational Semantics. Tech. Rep. DAIMI FN-19, University of Aarhus (1981)
18. Ramadge, P.J., Wonham, W.M.: Supervisory control of a class of discrete event processes. SIAM Journal on Control and Optimization 25(1), 206–230 (1987)
19. Salaün, G., Bultan, T.: Realizability of Choreographies Using Process Algebra Encodings. In: Leuschel, M., Wehrheim, H. (eds.) IFM 2009. LNCS, vol. 5423, pp. 167–182. Springer, Heidelberg (2009)
20. Tanenbaum, A.: Computer Networks, 4th edn. Prentice Hall Professional Technical Reference (2002)
21. Udding, J.: Classification and Composition of Delay-Insensitive Circuits. Ph.D. thesis, Eindhoven University of Technology, Eindhoven (1984)