
Article:
Beohar, H. orcid.org/0000-0001-5256-1334 and Cuijpers, P.J.L. (2014) Avoiding diamonds in desynchronisation. Science of Computer Programming, 91, Part A. pp. 45-69. ISSN 0167-6423

https://doi.org/10.1016/j.scico.2013.12.002

Avoiding Diamonds in Desynchronisation

H. Beohar[a,*], P. J. L. Cuijpers[b]

[a] *Center for Research on Embedded Systems,*
Halmstad University,
301 18, Halmstad, Sweden
[b] *Department of Mathematics and Computer science,*
Eindhoven University of Technology,
PO Box 513, 5600 MB, Eindhoven, The Netherlands

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. Desynchronisation 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 desynchronised design. However, this property is difficult to establish in practice.

In this paper, we give sufficient and necessary conditions under which a concrete synchronous design (i.e., without the unobservable action) is equivalent to an asynchronous design and formally prove that the diamond property is no longer needed for desynchronisation 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. To illustrate how this theory may be applied, we desynchronise the synchronous systems that are synthesised using supervisory control theory.

Keywords: Synchrony to Asynchrony, Desynchronisation, Branching Bisimulation, Equivalence Checking of Infinite State Systems

1. Introduction

Message passing [1] 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 modelled to have infinite capacity, analysing correctness of such systems is undecidable [2]. But also, if the buffers are modelled 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 desynchronising it. The challenge is then to design the synchronous system in such a way that the addition of communication buffers does not alter its behaviour (in any relevant way) [3]. A synchronous system that is not altered by the addition of communication buffers is called *desynchronisable*.

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 *synchronisability* techniques. Their idea is to make an asynchronous system synchronous, which is in contrast to desynchronisability, where a synchronous system is made asynchronous.

*Corresponding author
Email addresses: harsh.beohar@hh.se (H. Beohar), P.J.L.Cuijpers@tue.nl (P. J. L. Cuijpers)

Thus, synchronisability techniques are applicable when the components of a system are designed under asynchronous communication from the start (for instance, in web-services), whereas desynchronisability techniques are applicable when the components of a system are designed under synchronous communication from the start (for instance, in supervisory control [6]).

Despite these differences both approaches aim to establish an equivalence between a synchronous system and its asynchronous version. In [4], the authors showed that the existence of a weak bisimulation relation between a deterministic synchronous system and its asynchronous system with one place queues is sufficient and necessary for synchronisability modulo weak bisimulation. Our work differs from [4] as we propose conditions for desynchronisability solely on a given (possibly nondeterministic) 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 desynchronisability. Intuitively,

- two communicating processes in a synchronous system are well-posed if both the processes are able to receive each other's requests.
- the external actions (i.e., actions that are not involved in synchronisation) are independent in a synchronous system if the communicating processes can always delay the execution of their own external actions in favour of receiving a sequence of messages, without any consequence on the future behaviour 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 behaviour of the system.

In previous research [3, 7–9], well-posedness and the diamond property were already present as sufficient conditions for desynchronisability, while the other two properties are new with respect to these references because external actions were not allowed in the specification of communicating processes and the equivalences studied in these papers were coarser than branching bisimulation[1]. Table 1 provides the list of buffers and equivalences studied previously in the context of desynchronisation.

References	Buffers	Equivalences
Udding [9]	Wires	Trace equivalence
Balemi [7]	Queues	Trace equivalence
Fischer and Janssen [3]	Bags	Failure equivalence
Beohar and Cuijpers [8]	Bags	Branching bisimulation

Table 1: Buffers and equivalences studied previously in the context of desynchronisation.

As it turns out, the diamond property is difficult to establish in practice, while in particular input determinism and independence of external actions can be easily obtained by construction, at least for supervisory control synthesis [6]. As an example why diamond property 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 Figure 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

[1] We claim that input determinism as a sufficient condition disappears when studying desynchronisability modulo any equivalence coarser than contra-simulation cf. [10, Chapter 4].

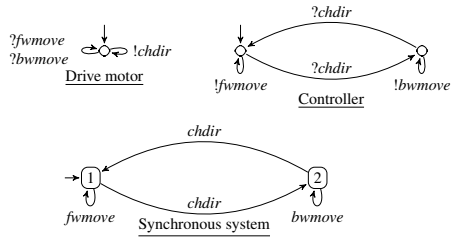


Figure 1: An illustration showing the impossibility of establishing the diamond property in certain synchronous systems.

fwmove.chdir. Similarly, the commutativity of the traces *chdir.bwmove* and *bwmove.chdir* is prohibited in this synchronous system. As a result, the synchronous system in Figure 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 desynchronise this system unless we adapt the model of plant or supervisor, or we adapt the way in which the desynchronisation 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, 7, 9] follow [2] in taking two unidirectional FIFO queues as a means of communication. In [3] a separate unidirectional FIFO queue is used for each type of message, which effectively leads to a *bag*-type of buffering (cf. [8]). 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 [12]. 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 desynchronisability 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 desynchronisability. 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 desynchronisation to so-called *half-duplex* communication (also used in [13] 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 desynchronisable 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, 7–9], and we are able to give a general method to adapt systems that are synthesised using supervisory control theory to satisfy these properties. In Section 4, we show how to obtain a well-posed supervisor for a given plant, whenever there exists a supervisor for the same plant. Our motivation in selecting supervisory control theory of Ramadge and Wonham [6] as our application domain is due to the inexact synchronisation problem, which prevents in applying this theory in practice (see [14]). The issue is that the existing supervisory control theory assumes synchronous interaction between a plant and its supervisor, whereas in practice the interactions between a plant and its supervisor are implemented asynchronously.

The use of half-duplex communication to restrict the behaviour of an asynchronous system is not new; see, for instance, [13, 15]. Furthermore, the authors of [13, 15] also gave conditions under which the behaviour of an asynchronous system is insensitive with the use of half-duplex or full-duplex queues. Intuitively, these conditions prevent the so-called *mixed states* (a state is mixed [13] if an input and output action is enabled from that state) in the specification of the communicating processes. Thus, this provides an alternative way to avoid diamonds in an asynchronous system. Nevertheless, we consider the prohibition of mixed states in a specification restrictive, at-least for control purposes [6], where the plants and their supervisors are allowed to have mixed states (for instance, see the case-study described in [11]). In addition, these mixed states also arise naturally when the communicating processes itself are composed with the parallel composition operator, which we do allow in our setup.

It is our hope that this paper will initiate discussion on the separation of concerns regarding desynchronisation. 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 recognising 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 desynchronisability, and when it can be dropped for the sake of efficiency.

The methods we use for studying desynchronisability in this paper stem from process algebra and concurrency theory (see e.g. [16]). We do not fix a set of desirable properties a priori, but rather aim for desynchronisability modulo a behavioural equivalence that preserves a large set of possibly desirable properties. The desynchronisability 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 behaviourally equivalent?* To be as general as possible, we take *branching bisimulation* as our behavioural equivalence of choice, which is the strongest equivalence used in interleaving semantics [17]. Note, that in many of the earlier works, weaker equivalences were used, but this did not lead to weaker desynchronisability conditions. Therefore, using a stronger equivalence does not pose a restriction here, while it does improve the applicability of the desynchronisation theorem.

In this paper, we show that our sufficient conditions for desynchronisability are also necessary for concrete (without silent steps) synchronous systems. To establish this result, we need to observe whether there are no more pending messages in a channel of an asynchronous system. We model this observation by a predicate, called *empty-buffer* predicate. Now by enriching the transfer condition of branching bisimilarity (just like how the termination predicate is handled in the definition of branching bisimulation; see [16]) we ensure that an asynchronous system constructed from a desynchronisable synchronous system has the following communication property: "every sent message is eventually received". Note that this property was already present in the previous works [3, 8, 9] on desynchronisability; although, it was never made explicit with the respective equivalences studied previously. For instance, the communication property was called *absence of computation interference* in [9]; while, [3] used *sender domination* to satisfy the above property. Thus, by making the communication property explicit in our setup we are able to show that our conditions are also necessary for desynchronisability, which was previously absent in [3, 7–9].

Organisation of the paper.. In Section 2, we describe the mathematical notations and formal definitions required to define desynchronisability using two unidirectional FIFO buffers. Section 3 discusses necessary and sufficient conditions for desynchronisability, including the unwanted diamond property and show how it can be eliminated by using half-duplex buffers for desynchronisation. Section 4 considers how to apply desynchronisation in the context of supervisory control. Lastly, Section 5 motivates the choice of abstraction scheme used to construct asynchronous systems and discusses desynchronisation of non-concrete synchronous systems.

This article extends [18] in the following four aspects.

- First, this article contains proofs of all the lemmas and theorems, which were either sketched or left out in [18].
- Second, a new simplified sufficient condition for desynchronisability modulo branching bisimulation is given in Subsection 3.6, which is derived from the characterisation obtained in [18].
- Third, a more detailed account of desynchronisability for synchronous systems that are synthesised using supervisory control theory is given in Section 4. In contrast to [18], we allow external actions and nondeterminism in the specification of the supervisors.
- Fourth, the different abstraction schemes of [8] are compared in Subsection 5.1 with the objective to find out the abstraction scheme that yields in weakest conditions for desynchronisation (modulo branching bisimulation) in the presence of (half-duplex) queues.

2. Basic definitions

In this paper, we model the world as a single transition system in which all behaviour of interest is 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 $q \in \mathbb{P}$.

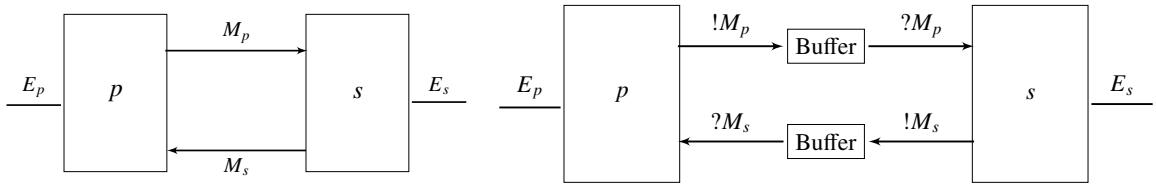
Definition 1. A *labelled transition system* is a tuple $(\mathbb{P}, A, \rightarrow, \sqcup)$, where

- $\mathbb{P}$ is a set of states.
- A is a set of actions.
- $\rightarrow \subseteq \mathbb{P} \times (A \cup \{\tau\}) \times \mathbb{P}$ 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 in which all buffers are empty.

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 state $q \in \mathbb{P}$, the set of *reachable states* $\Re(q)$ is defined as the smallest set such that:

- $q \in \Re(q)$, and
- $\forall q_1, q_2 \in \mathbb{P}, \alpha \in A \cup \{\tau\}. \left[\left(q_1 \in \Re(q) \ \wedge q_1 \xrightarrow{\alpha} q_2\right) \Rightarrow \ q_2 \in \Re(q)\right]$.

In what follows, the letter q and its decorations like $q', q_1, q_2, \cdots$ are used to reason about arbitrary processes, whereas the letters p, s and their corresponding decorations are reserved to denote the communicating processes of a given synchronous system.



(a) A synchronous system.

(b) An asynchronous system.

Figure 2: From Synchrony to Asynchrony.

Considering a synchronous system as depicted in Figure 2(a), we identify two components p, s, which we assume to be processes in our labelled 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.

When the system is *desynchronised* we obtain an asynchronous system as depicted in Figure 2(b), 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 (i.e., the set $!M_p = \{!m \mid m \in M_p\}$) and the receiving of a message from p sent by s (i.e., the set $?M_s = \{?m \mid m \in M_s\}$). 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 labelled transition system, where p makes use of the actions $!M_p \uplus ?M_s \uplus E_p$ and s makes use of the actions $!M_s \uplus ?M_p \uplus E_s$, we can define the synchronous and asynchronous composition of p and s through Structural Operational Semantic (SOS) rules on the states of the transition system [19]. 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 2, we give the SOS rules for synchronous composition and asynchronous composition using two unidirectional lossless FIFO queues. The notation $p \parallel s$ denotes the synchronous composition of processes p and s. Note that one can construct the synchronous composition of processes, say p' and $p \parallel s$, by renaming the synchronous interaction (e.g. the renaming operator from TCP process algebra [16]) of $p \parallel s$ into a send message or receive message

Table 2: SOS rules for synchronous and asynchronous parallel composition.

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

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

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

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

depending on the architecture. The notation $p\,|[\mu,\nu]|\,s$ denotes the asynchronous composition of processes p and s with sequences of messages $\nu \in M_p^*, \mu \in M_s^*$ 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 desynchronisable 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 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 3, 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 [16, 17]) as an equivalence between processes that abstracts from internal actions. In Subsection 5.1, we discuss the impact of alternative choices of abstraction schemes on desynchronisability modulo branching bisimulation.

Table 3: SOS rules for the abstraction operator $\Delta()$.

$$\frac{q_1 \xrightarrow{!m} q_2,\ m \in M_p \cup M_s}{\Delta(q_1) \xrightarrow{m} \Delta(q_2)} \qquad \frac{q_1 \xrightarrow{e} q_2,\ e \in E_p \cup E_s}{\Delta(q_1) \xrightarrow{e} \Delta(q_2)} \qquad \frac{q_1 \xrightarrow{?m} q_2,\ m \in M_p \cup M_s}{\Delta(q_1) \xrightarrow{\tau} \Delta(q_2)} \qquad \frac{q\sqcup}{\Delta(q)\sqcup}$$

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:

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

Let $x = x_p \uplus x_s$ and $yM = yM_p \uplus yM_s$, where $x \in \{M, E\}$ 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 $^- : (M \cup E)^* \to M^*$ in the following way:

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 $w \in (M \cup E)^*$.

Given a sequence of messages and external actions $w \in (M \cup E)^*$, then, $!w$ denotes a sequence of send messages and external actions, $?w$ denotes a sequence of receive messages, and $\bar{w}$ denotes the projection of the sequence w onto the set of messages M.

Proposition 1. *Let $p \parallel s$ be a synchronous system.*

1. *If $u \in (M_s \cup E_s)^*$ and $p \parallel s \xrightarrow{u}\!\!\!\!\rightarrow p' \parallel s'$ then $p \xrightarrow{?u}\!\!\!\!\rightarrow p' \wedge s \xrightarrow{!u}\!\!\!\!\rightarrow s'$.*
2. *If $v \in (M_p \cup E_p)^*$ and $p \parallel s \xrightarrow{v}\!\!\!\!\rightarrow p' \parallel s'$ then $s \xrightarrow{?v}\!\!\!\!\rightarrow s' \wedge p \xrightarrow{!v}\!\!\!\!\rightarrow p'$.*

Proof. Straightforward by induction on u and v, respectively. □

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.

1. $\forall q, q_1, q', \alpha. \Big[\Big((q, q') \in \mathcal{B} \wedge q \xrightarrow{\alpha} q_1\Big) \Rightarrow \Big(\alpha = \tau \wedge (q_1, q') \in \mathcal{B}\Big) \vee \exists q'_1, q'_2. \Big[q' \xrightarrow{\epsilon}\!\!\!\!\rightarrow q'_1 \xrightarrow{\alpha} q'_2 \wedge (q, q'_1) \in \mathcal{B} \wedge (q_1, q'_2) \in \mathcal{B}\Big]\Big]$,
2. $\forall q, q'. \Big[\Big((q, q') \in \mathcal{B} \wedge q \sqcup\Big) \Rightarrow \exists q''. \Big[q' \xrightarrow{\epsilon}\!\!\!\!\rightarrow q'' \wedge q'' \sqcup \wedge (q, q'') \in \mathcal{B}\Big]\Big]$,
3. $\forall q, q', q'_1, \alpha. \Big[\Big((q, q') \in \mathcal{B} \wedge q' \xrightarrow{\alpha} q'_1\Big) \Rightarrow \Big(\alpha = \tau \wedge (q, q'_1) \in \mathcal{B}\Big) \vee \exists q_1, q_2. \Big[q \xrightarrow{\epsilon}\!\!\!\!\rightarrow q_1 \xrightarrow{\alpha} q_2 \wedge (q_1, q') \in \mathcal{B} \wedge (q_2, q'_1) \in \mathcal{B}\Big]\Big]$,
4. $\forall q, q'. \Big[\Big((q, q') \in \mathcal{B} \wedge q' \sqcup\Big) \Rightarrow \exists q''. \Big[q \xrightarrow{\epsilon}\!\!\!\!\rightarrow q'' \wedge q'' \sqcup \wedge (q'', q') \in \mathcal{B}\Big]\Big]$.

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

Note that Conditions 1 and 3 are the conventional transfer properties of branching bisimulation, while Conditions 2 and 4 are reminiscent of the transfer properties involved with the termination predicate cf. [16]. Here, we interpret Condition 2 (likewise Condition 4) as follows: if two processes q, q' are branching bisimilar and q has empty buffer, then q' can empty its buffer contents by performing invisible actions and become branching bisimilar to q.

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

Definition 4. A synchronous system $p \parallel s$ is *desynchronisable* if $p \parallel s \leftrightarrow_{\mathbf{b}} \Delta(p\,|[\epsilon, \epsilon]|\, s)$.

3. Properties of desynchronisable systems

In this section, we prove a number of properties of desynchronisable systems modulo branching bisimulation. In particular, Definition 5, Lemmas 1-2, and Theorem 1 are only required to establish the necessity of well-posedness, input-determinism, independence of external actions, and diamond property for desynchronisation. A new result (cf. [3, 7–9]) is that the observation of the empty-buffer predicate makes that these properties are necessary as well as sufficient for desynchronisability. A technical assumption used to show necessity in this case, is that the components p and s are *concrete*, meaning they do not have internal behavior themselves. Furthermore, in subsection 3.7, we show that by just dropping the diamond property from the general characterisation of desynchronisation results in the characterisation of desynchronisation when half-duplex queues are used to construct asynchronous systems.

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

Henceforth, due to the following proposition, we use strong bisimulation[2] equivalence $\leftrightarrows$ in favour of branching bisimulation equivalence between any two concrete processes because of its simpler transfer properties.

Proposition 2. *For any two concrete processes $q, q' \in \mathbb{P}$, we have $q \leftrightarrows_{\mathbf{b}} q' \iff q \leftrightarrows q'$.*

Lemma 1. *Let $p \parallel s$ be a concrete and desynchronisable 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 [16]. □

A key step in understanding the necessary conditions for desynchronisability, is to see that any reachable state $p' \parallel s' \in \Re(p \parallel s)$ of some desynchronisable system $p \parallel s$ is desynchronisable 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.

Lemma 2. *Let p_1, s_1, p_2, s_2 be any four concrete processes. Then,*

$$p_1 \parallel s_1 \leftrightarrows_{\mathbf{b}} p_2 \,|[\epsilon, \epsilon]|\, s_2 \Rightarrow p_1 \parallel s_1 \leftrightarrows_{\mathbf{b}} p_2 \parallel s_2 \ .$$

Proof. Due to Proposition 2, it is sufficient to show that the above implication holds w.r.t strong bisimulation $\leftrightarrows$, i.e., $p_1 \parallel s_1 \leftrightarrows_{\mathbf{b}} p_2 \,|[\epsilon, \epsilon]|\, s_2 \Rightarrow p_1 \parallel s_1 \leftrightarrows p_2 \parallel s_2$. Define the following relation $\mathcal{S}$:

$$\mathcal{S} = \left\{ (p_3 \parallel s_3, p_4 \parallel s_4) \mid p_3 \parallel s_3 \in \Re(p_1 \parallel s_1) \wedge p_4 \,|[\epsilon, \epsilon]|\, s_4 \in \Re(p_2 \,|[\epsilon, \epsilon]|\, s_2) \wedge p_3 \parallel s_3 \leftrightarrows_{\mathbf{b}} p_4 \,|[\epsilon, \epsilon]|\, s_4 \right\}.$$

Next, we show that the relation $\mathcal{S}$ is indeed a strong bisimulation relation.

1. Let $p_3 \parallel s_3 \xrightarrow{m} p_5 \parallel s_5$ and $(p_3 \parallel s_3, p_4 \parallel s_4) \in \mathcal{S}$ for some $m \in M_p, p_5, s_5 \in \mathbb{P}$. Then, by the construction of $\mathcal{S}$ we have $p_3 \parallel s_3 \leftrightarrows_{\mathbf{b}} \Delta(p_4 \,|[\epsilon, \epsilon]|\, s_4)$. Applying the transfer conditions of branching bisimulation under the assumption of concrete processes and disjointness of sets M_p, M_s we get $\Delta(p_4 \,|[\epsilon, \epsilon]|\, s_4) \xrightarrow{m} \Delta(p_6 \,|[\epsilon, m]|\, s_4)$ and $p_5 \parallel s_5 \leftrightarrows_{\mathbf{b}} \Delta(p_6 \,|[\epsilon, m]|\, s_4)$ for some $p_6 \in \mathbb{P}$. Since $(p_5 \parallel s_5)\sqcup$, branching bisimulation under concreteness assumption gives us $\Delta(p_6 \,|[\epsilon, m]|\, s_4) \xrightarrow{\tau} \Delta(p_6 \,|[\epsilon, \epsilon]|\, s_6)$ and $p_5 \parallel s_5 \leftrightarrows_{\mathbf{b}} \Delta(p_6 \,|[\epsilon, \epsilon]|\, s_6)$, for some $s_6 \in \mathbb{P}$. Thus, we derive $p_4 \xrightarrow{!m} p_6$ and $s_4 \xrightarrow{?m} s_6$; hence, $p_4 \parallel s_4 \xrightarrow{m} p_6 \parallel s_6$ and $(p_5 \parallel s_5, p_6 \parallel s_6) \in \mathcal{S}$.
2. Let $p_3 \parallel s_3 \xrightarrow{\alpha} p_5 \parallel s_5$ and $(p_3 \parallel s_3, p_4 \parallel s_4) \in \mathcal{S}$ for some $\alpha \in E_p \cup E_s \cup M_s$. Similar to the previous case.
3. Let $p_4 \parallel s_4 \xrightarrow{m} p_6 \parallel s_6$ and $(p_3 \parallel s_3, p_4 \parallel s_4) \in \mathcal{S}$ for some $m \in M_p, p_6, s_6 \in \mathbb{P}$. Then, by the semantics we have $p_4 \xrightarrow{!m} p_6$ and $s_4 \xrightarrow{?m} s_6$. Also, by the construction of $\mathcal{S}$ we have $p_3 \parallel s_3 \leftrightarrows_{\mathbf{b}} \Delta(p_4 \,|[\epsilon, \epsilon]|\, s_4)$. Using the above transitions at the state $\Delta(p_4 \,|[\epsilon, \epsilon]|\, s_4)$ we derive the following transitions: $\Delta(p_4 \,|[\epsilon, \epsilon]|\, s_4) \xrightarrow{m} \Delta(p_6 \,|[\epsilon, m]|\, s_4) \xrightarrow{\tau} \Delta(p_6 \,|[\epsilon, \epsilon]|\, s_6)$. Applying the transfer conditions of branching bisimulation under concreteness assumption we get $\exists p_5, s_5. \left[p_3 \parallel s_3 \xrightarrow{m} p_5 \parallel s_5 \wedge p_5 \parallel s_5 \leftrightarrows_{\mathbf{b}} \Delta(p_6 \,|[\epsilon, m]|\, s_4) \right]$. Now consider the transition $\Delta(p_6 \,|[\epsilon, m]|\, s_4) \xrightarrow{\tau} \Delta(p_6 \,|[\epsilon, \epsilon]|\, s_6)$. Note that the synchronous system $p_1 \parallel s_1$ is concrete, thus this τ-transition can be only mapped by zero τ-steps from the state $p_5 \parallel s_5$. Thus, $p_5 \parallel s_5 \leftrightarrows_{\mathbf{b}} \Delta(p_6 \,|[\epsilon, \epsilon]|\, s_6)$. Hence, $(p_5 \parallel s_5, p_6 \parallel s_6) \in \mathcal{S}$.
4. Let $p_4 \parallel s_4 \xrightarrow{\alpha} p_6 \parallel s_6$ and $(p_3 \parallel s_3, p_4 \parallel s_4) \in \mathcal{S}$ for some $\alpha \in E_p \cup E_s \cup M_s$. Similar to the previous case.

[2] see [16] for a formal definition.

5. The transfer property for the empty buffer predicate $\sqcup$ holds trivially because every state in the composition $\|$ satisfies the predicate $\sqcup$ by definition. □

Theorem 1. *Let $p \parallel s$ be concrete and desynchronisable, then any $p' \parallel s' \in \Re(p \parallel s)$ is desynchronisable.*

Proof. As a base case, the initial state of $p \parallel s$ is desynchronisable by assumption. By induction, assume that we have a reachable desynchronisable state $p' \parallel s' \in \Re(p \parallel s)$ 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 τ-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 τ-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 $\Delta(p' \,|[\epsilon, \epsilon]|\, s') \xrightarrow{e} \Delta(p'' \,|[\epsilon, \epsilon]|\, s'')$ with $e \in E_p$, in which case we find that $s' = s''$.
4. a transition $\Delta(p' \,|[\epsilon, \epsilon]|\, s') \xrightarrow{e} \Delta(p'' \,|[\epsilon, \epsilon]|\, s'')$ with $e \in E_s$, in which case we find that $p' = p''$.

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 $p''' \parallel s''' \leftrightarrows_{\mathbf{b}} \Delta(p'' \,|[\epsilon, \epsilon]|\, s'')$. Furthermore, from Lemma 2 we get $p'' \parallel s'' \leftrightarrows_{\mathbf{b}} p''' \parallel s'''$. And by transitivity we conclude that $p'' \parallel s''$ is desynchronisable. □

3.1. Well-posedness

The first actual implication of desynchronisability that we would like to discuss, is that a desynchronisable system is always *well-posed*. This was already observed in [3] for desynchronisability 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. $\forall p_1, s_1, p_2, m. \left[p_1 \xrightarrow{!m} p_2 \wedge (p_1, s_1) \in \mathcal{W} \Rightarrow \exists s_2. \left[s_1 \xrightarrow{?m} s_2 \right] \wedge \forall s_2. \left[s_1 \xrightarrow{?m} s_2 \Rightarrow (p_2, s_2) \in \mathcal{W} \right] \right]$,
2. $\forall p_1, s_1, p_2, e \in E_p. \left[p_1 \xrightarrow{e} p_2 \wedge (p_1, s_1) \in \mathcal{W} \Rightarrow (p_2, s_1) \in \mathcal{W} \right]$,
3. $\forall p_1, s_1, s_2, m. \left[s_1 \xrightarrow{!m} s_2 \wedge (p_1, s_1) \in \mathcal{W} \Rightarrow \exists p_2. \left[p_1 \xrightarrow{?m} p_2 \right] \wedge \forall p_2. \left[p_1 \xrightarrow{?m} p_2 \Rightarrow (p_2, s_2) \in \mathcal{W} \right] \right]$,
4. $\forall p_1, s_1, s_2, e \in E_s. \left[s_1 \xrightarrow{e} s_2 \wedge (p_1, s_1) \in \mathcal{W} \Rightarrow (p_1, s_2) \in \mathcal{W} \right]$.

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

Proposition 3. *Let $p \parallel s$ be a concrete and a well-posed synchronous system with $\mathcal{W}$ being the witnessing well-posedness relation, i.e., $(p, s) \in \mathcal{W}$. Then,*

$$\forall p_1, s_1. \left[p_1 \parallel s_1 \in \Re(p \parallel s) \Rightarrow (p_1, s_1) \in \mathcal{W} \right].$$

Lemma 3 (Generalised well-posedness)**.** *Let $p \parallel s$ be a well-posed and concrete synchronous system.*

1. *If $p_1 \parallel s_1 \in \Re(p \parallel s)$, $u \in (M_s \cup E_s)^*$, and $s_1 \xrightarrow{!u} s_2$, then $\exists p_2. \left[p_1 \parallel s_1 \xrightarrow{u} p_2 \parallel s_2 \right]$.*
2. *If $p_1 \parallel s_1 \in \Re(p \parallel s)$, $v \in (M_p \cup E_p)^*$, and $p_1 \xrightarrow{!v} p_2$, then $\exists s_2. \left[p_1 \parallel s_1 \xrightarrow{v} p_2 \parallel s_2 \right]$.*

Proof. Straightforward from the induction on u (v) and application of well-posedness definition. □

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

Proof. Define a relation $\mathcal{W} = \{(p_1, s_1) \mid \Delta(p_1 \,|[\epsilon, \epsilon]|\, s_1) \in \Re(\Delta(p \,|[\epsilon, \epsilon]|\, s))\}$. To show that $\mathcal{W}$ is a well-posedness relation, assume a transition $p_1 \xrightarrow{\alpha} p_2$ and $(p_1, s_1) \in \mathcal{W}$, for some $p_1, s_1, p_2 \in \mathbb{P}$.

1. Let $\alpha \in !M_p$. Then, by the construction of $\mathcal{W}$ we have $\Delta(p_1 \,|[\epsilon, \epsilon]|\, s_1) \in \Re(\Delta(p \,|[\epsilon, \epsilon]|\, s))$ and using $p_1 \xrightarrow{!m} p_2$ we get $\Delta(p_1 \,|[\epsilon, \epsilon]|\, s_1) \xrightarrow{m} \Delta(p_2 \,|[\epsilon, m]|\, s_1)$. Since $p \parallel s$ is desynchronisable, we know that there exists $q \in \Re(p \parallel s)$ such that $q \leftrightarrows_{\mathbf{b}} \Delta(p_2 \,|[\epsilon, m]|\, s_1)$. 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, we showed that there exists s_2 such that $s_1 \xrightarrow{?m} s_2$ and by the construction of $\mathcal{W}$ it is clear that $(p_2, s_2) \in \mathcal{W}$. But, well-posedness also asserts that for every s_2' such that $s_1 \xrightarrow{?m} s_2'$ and $(p_2, s_2') \in \mathcal{W}$. So pick a $s_2' \in \mathbb{P}$ such that $s_1 \xrightarrow{?m} s_2'$. Then we have $\Delta(p_2 \,|[\epsilon, m]|\, s_1) \xrightarrow{\tau} \Delta(p_2 \,|[\epsilon, \epsilon]|\, s_2')$. Thus, $(p_2, s_2') \in \mathcal{W}$.
2. Let $\alpha \in E_p$. Then, by the construction of $\mathcal{W}$ we have $\Delta(p_1 \,|[\epsilon, \epsilon]|\, s_1) \in \Re(\Delta(p \,|[\epsilon, \epsilon]|\, s))$ and using the above transition we get $\Delta(p_1 \,|[\epsilon, \epsilon]|\, s_1) \xrightarrow{e} \Delta(p_2 \,|[\epsilon, \epsilon]|\, s_1)$. And, from the construction of $\mathcal{W}$ we conclude that $(p_2, s_1) \in \mathcal{W}$.

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

3.2. Independence of external actions

The second implication of desynchronisability 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 e in favour of receiving a sequence of messages u from the other process, without any consequence on its future behaviour modulo $\leftrightarrows_{\mathbf{b}}$, i.e., the traces $e.u$ and $u.e$ commute up to branching bisimulation. The reception of messages becomes *independent* of the external behaviour 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 desynchronisability. Of course, independence of external actions of the separate processes would be a natural part of a sufficient condition for desynchronisability.

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

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

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

Proof. Assume we have a reachable and desynchronisable (Theorem 1) state $p_1 \parallel s_1 \in \Re(p \parallel s)$ with solid transitions as in Figure 3, where $e \in E_p$ and $u \in (M_s \cup E_s)^*$. From Proposition 1 we get $p_1 \xrightarrow{e} p_2$, $s_1 \xrightarrow{!u} s_2$, and $p_2 \xrightarrow{?u} p_2'$. As well-posedness is necessary for desynchronisability, we may use it to obtain $p_1 \xrightarrow{?u} p_3$ (for some p_3) from Lemma 3. Using the SOS-rules we get $p_1 \parallel s_1 \xrightarrow{u} p_3 \parallel s_2$ (dashed in Figure 3). From these transitions we then derive the transitions in the asynchronous system depicted as solid lines in Figure 3, where $\bar{u} = \mu$.

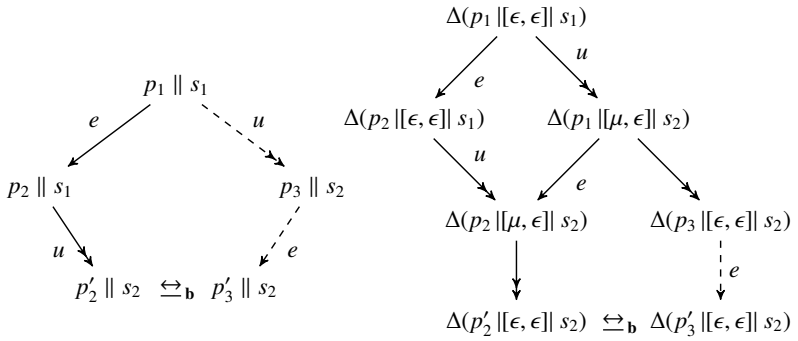


Figure 3: An illustration showing that an external action e by the process p_1 can be delayed in favour of receiving the messages u such that the traces $e.u$ and $u.e$ commute up to branching bisimulation.

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 and E_s, gives us the existence of p_3' such that $\Delta(p_3 \,|[\epsilon, \epsilon]|\, s_2) \xrightarrow{e} \Delta(p_3' \,|[\epsilon, \epsilon]|\, s_2)$ and $\Delta(p_3' \,|[\epsilon, \epsilon]|\, s_2) \ \leftrightarrows_{\mathbf{b}} \ \Delta(p_2 \,|[\mu, \epsilon]|\, s_2)$. By the SOS-rules we get $p_3 \xrightarrow{e} p_3'$, thus, $p_3 \parallel s_2 \xrightarrow{e} p_3' \parallel s_2$. Next, we show that $p_2' \parallel s_2 \ \leftrightarrows_{\mathbf{b}} \ p_3' \parallel s_2$. From above we have $\Delta(p_3' \,|[\epsilon, \epsilon]|\, s_2) \ \leftrightarrows_{\mathbf{b}} \ \Delta(p_2 \,|[\mu, \epsilon]|\, s_2)$ and since τ-transition are inert we have $\Delta(p_2 \,|[\mu, \epsilon]|\, s_2) \ \leftrightarrows_{\mathbf{b}} \ \Delta(p_2' \,|[\epsilon, \epsilon]|\, s_2)$. By transitivity we get $\Delta(p_3' \,|[\epsilon, \epsilon]|\, s_2) \ \leftrightarrows_{\mathbf{b}} \ \Delta(p_2' \,|[\epsilon, \epsilon]|\, s_2)$. By Theorem 1 we have $p_3' \parallel s_2 \ \leftrightarrows_{\mathbf{b}} \ \Delta(p_3' \,|[\epsilon, \epsilon]|\, s_2)$ and $p_2' \parallel s_2 \ \leftrightarrows_{\mathbf{b}} \ \Delta(p_2' \,|[\epsilon, \epsilon]|\, s_2)$, from which we ultimately conclude $p_2' \parallel s_2 \ \leftrightarrows_{\mathbf{b}} \ p_3' \parallel s_2$. Likewise, Condition 2 of Definition 7 can be proved. □

3.3. Input determinism

The next implication of desynchronisability, is that desynchronisable 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 behaviour, and it may also be non-deterministic when sending messages. The reason for this, is that desynchronisation *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. [20]) 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* $\leftrightarrows_{\mathbf{b}}$ if every reachable state $p_1 \parallel s_1 \in \Re(p \parallel s)$ satisfies the following conditions.

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

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

Proof. Pick a reachable state $p_1 \parallel s_1 \in \Re(p \parallel s)$ such that $p_1 \parallel s_1 \xrightarrow{u} p_2 \parallel s_2$ and $p_1 \parallel s_1 \xrightarrow{u} p_3 \parallel s_2$, for some $u \in (M_s \cup E_s)^*, p_2, p_3, s_2 \in \mathbb{P}$ (see Figure 4). By Theorem 1 we have $p_1 \parallel s_1 \ \leftrightarrows_{\mathbf{b}} \ \Delta(p_1 \,|[\epsilon, \epsilon]|\, s_1)$. Using the given

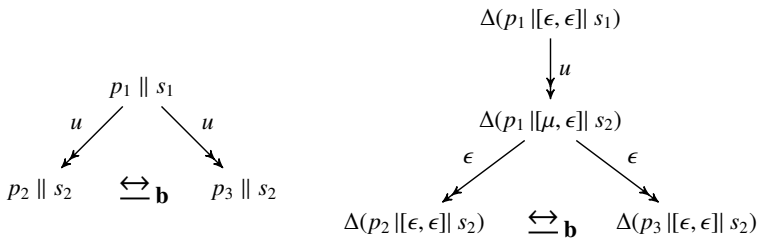


Figure 4: An illustration showing that nondeterministic reception of messages u in a desynchronisable synchronous system leads to branching bisimilar states $p_2 \parallel s_2, p_3 \parallel s_2$.

transitions in Proposition 1 we get $s_1 \xrightarrow{!u} s_2$, $p_1 \xrightarrow{?u} p_2$ and $p_1 \xrightarrow{?u} p_3$. For the asynchronous system we then find the transitions as shown in Figure 4, where $\mu = \bar{u}$.

As $p \parallel s$ is concrete, all τ-transitions in the asynchronous system are inert (Lemma 1), so we get $\Delta(p_1 \,[\![\mu, \epsilon]\!]\, s_2) \leftrightarrow_{\mathbf{b}} \Delta(p_2 \,[\![\epsilon, \epsilon]\!]\, s_2) \leftrightarrow_{\mathbf{b}} \Delta(p_3 \,[\![\epsilon, \epsilon]\!]\, s_2)$. Finally, using Theorem 1 twice we ultimately conclude that $p_2 \parallel s_2 \leftrightarrow_{\mathbf{b}} p_3 \parallel s_2$. Likewise, Condition 2 of Definition 8 can be proved. □

3.4. The diamond property

The final implication of desynchronisability 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 behaviourally equivalent states.

Definition 9. A synchronous system $p \parallel s$ has the *diamond property modulo* $\leftrightarrow_{\mathbf{b}}$ 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 $p_4 \parallel s_4 \leftrightarrow_{\mathbf{b}} p_5 \parallel s_5$.

Lemma 4 (Generalised diamond property)**.** *Let $p \parallel s$ be a concrete synchronous system such that it satisfies Definitions 6-9. If $p_1 \parallel s_1 \in \Re(p \parallel s)$, $u \in (M_s \cup E_s)^*$, $v \in (M_p \cup E_p)^*$, $p_1 \parallel s_1 \xrightarrow{u\!\!\rightarrow} p_2 \parallel s_2$, and $p_1 \parallel s_1 \xrightarrow{v\!\!\rightarrow} p_3 \parallel s_3$ then $\exists p_4, s_4, p_5, s_5. \left[p_2 \parallel s_2 \xrightarrow{v\!\!\rightarrow} p_4 \parallel s_4 \wedge p_3 \parallel s_3 \xrightarrow{u\!\!\rightarrow} p_5 \parallel s_5 \wedge p_4 \parallel s_4 \leftrightarrow_{\mathbf{b}} p_5 \parallel s_5\right]$.*

Proof. We first prove the following claim if $p_1 \parallel s_1 \xrightarrow{u\!\!\rightarrow} p_2 \parallel s_2$, $p_1 \parallel s_1 \xrightarrow{\alpha} p_3 \parallel s_3$, $u \in (M_s \cup E_s)^*$, and $\alpha \in M_p \cup E_p$ then $\exists p_4, s_4, p_5, s_5. \left[p_2 \parallel s_2 \xrightarrow{\alpha} p_4 \parallel s_4 \wedge p_3 \parallel s_3 \xrightarrow{u\!\!\rightarrow} p_5 \parallel s_5 \wedge p_4 \parallel s_4 \leftrightarrow_{\mathbf{b}} p_5 \parallel s_5\right]$. Without loss of generality, assume that $u = u'.\alpha'$ and $p_1 \parallel s_1 \xrightarrow{u'\!\!\rightarrow} p_2' \parallel s_2' \xrightarrow{\alpha'} p_2 \parallel s_2$. Then, by induction hypothesis we have

$$\exists p_4', s_4', p_5', s_5'. \left[p_2' \parallel s_2' \xrightarrow{\alpha} p_4' \parallel s_4' \wedge p_3 \parallel s_3 \xrightarrow{u'\!\!\rightarrow} p_5' \parallel s_5' \wedge p_4' \parallel s_4' \leftrightarrow_{\mathbf{b}} p_5' \parallel s_5'\right].$$

We identify the following cases based on the types of α, α'.

1. Let $\alpha = e$, for some $e \in E_p$, $\alpha' = e'$, for some $e' \in E_s$. Trivial.
2. Let $\alpha = e$, for some $e \in E_p$, $\alpha' = n$, for some $n \in M_s$. The single step transitions from the above inductive hypothesis are shown as solid lines in Figure 5. Note that $s_2' = s_4'$ because of Rule 2. From the transition $p_2' \parallel s_2' \xrightarrow{n} p_2 \parallel s_2$ we have $s_2' \xrightarrow{!n} s_2$. And from well-posedness (Definition 6) we get $\exists p_4. \left[p_4' \xrightarrow{?n} p_4\right]$. Thus, $p_4' \parallel s_2' \xrightarrow{n} p_4 \parallel s_2$.
   Now, applying independence of external actions (Definition 7) at the state $p_2' \parallel s_2'$ we get (see Figure 5)

$$\exists p_6, p_6'. \left[p_2' \parallel s_2' \xrightarrow{n} p_6 \parallel s_2 \xrightarrow{e} p_6' \parallel s_2 \wedge p_6' \parallel s_2 \leftrightarrow_{\mathbf{b}} p_4 \parallel s_2\right].$$

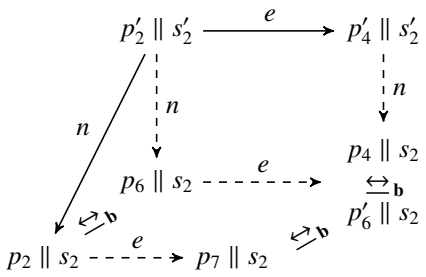


Figure 5: Case 2 in Lemma 4.

From input-determinism (Definition 8) we get $p_2 \parallel s_2 \leftrightarrow_{\mathbf{b}} p_6 \parallel s_2$. And from transfer conditions of branching bisimulation under the assumption of concrete processes and disjointness of the sets E_p, E_s we get

$$\exists p_7. \Big[p_2 \parallel s_2 \xrightarrow{e} p_7 \parallel s_2 \wedge p_6' \parallel s_2 \leftrightarrow_{\mathbf{b}} p_7 \parallel s_2 \Big].$$

Thus, by transitivity we get $p_7 \parallel s_2 \leftrightarrow_{\mathbf{b}} p_4 \parallel s_2$. Recall from induction hypothesis that $p_4' \parallel s_2' \leftrightarrow_{\mathbf{b}} p_5' \parallel s_5'$. Again, from the transfer conditions of branching bisimulation under the assumption of concrete processes we get $\exists p_5, s_5. \Big[p_5' \parallel s_5' \xrightarrow{n} p_5 \parallel s_5 \wedge p_4 \parallel s_2 \leftrightarrow_{\mathbf{b}} p_5 \parallel s_5 \Big]$. Finally, by transitivity we have $p_7 \parallel s_2 \leftrightarrow_{\mathbf{b}} p_5 \parallel s_5$.

3. Let $\alpha = m$, for some $m \in M_p$, $\alpha' = e'$, for some $e' \in E_s$. Similar to the previous case.
4. Let $\alpha = m$, for some $m \in M_p$, $\alpha' = n$, for some $n \in M_s$. Similar to Case 2, use Definition 9 instead of Definition 7.

Likewise, the main statement can be proved by assuming $v = v'.\alpha$ for some $\alpha \in E_p \cup M_p$. □

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

Proof. Assume a state $p_1 \parallel s_1 \in \Re(p \parallel s)$, $p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_2$ ($m \in M_p$) and $p_1 \parallel s_1 \xrightarrow{n} p_3 \parallel s_3$ ($n \in M_s$), as depicted in Figure 6. From Theorem 1 we know that $p_i \parallel s_i \leftrightarrow_{\mathbf{b}} p_i \,|[\epsilon, \epsilon]|\, s_i$, 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 Figure 6.

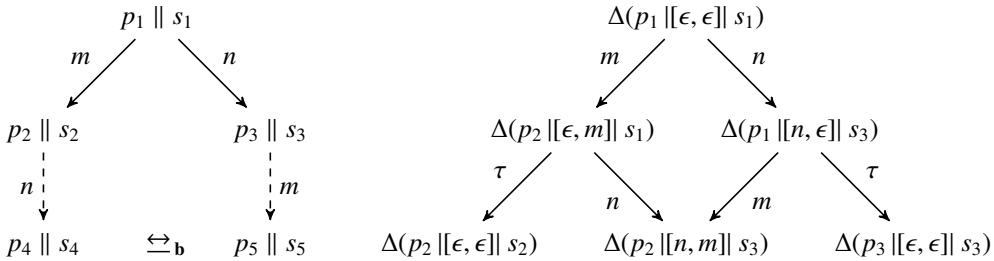


Figure 6: An illustration showing that the communication of two send messages m, n by different processes can be done in any order in a desynchronisable synchronous system.

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

3.5. Sufficient conditions for desynchronisability

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

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

Proof. We define a relation $\mathcal{B}$ in the following way:

$$\begin{aligned}\mathcal{B} = \Big\{ & (p_1 \parallel s_1, \Delta(p_2\,|[\mu,\nu]|\, s_2)) \mid p_1 \parallel s_1 \in \Re(p \parallel s) \wedge \Delta(p_2\,|[\mu,\nu]|\, s_2) \in \Re(\Delta(p\,|[\epsilon,\epsilon]|\, s)) \wedge \\ & \exists p_2', s_2', q, q', u, v. \Big[u \in (M_s \cup E_s)^* \wedge v \in (M_p \cup E_p)^* \wedge \bar{u} = \mu \wedge \bar{v} = \nu \wedge p_2 \parallel s_2' \in \Re(p \parallel s) \wedge \\ & p_2' \parallel s_2 \in \Re(p \parallel s) \wedge p_2 \parallel s_2' \xrightarrow{u} q \leftrightarrow_{\mathbf{b}} p_1 \parallel s_1 \leftrightarrow_{\mathbf{b}} q' \xleftarrow{v} p_2' \parallel s_2 \Big] \Big\}.\end{aligned}$$

Intuitively, two states $p_1 \parallel s_1, \Delta(p_2\,|[\mu,\nu]|\, s_2)$ are $\mathcal{B}$-related if

1. the above states are reachable from their respective initial states, i.e., $p_1 \parallel s_1 \in \Re(p \parallel s), \Delta(p_2\,|[\mu,\nu]|\, s_2) \in \Re(\Delta(p\,|[\epsilon,\epsilon]|\, s)$; and
2. it is possible to read the contents of the input queues (because of the transitions $p_2 \parallel s_2' \xrightarrow{u} q, p_2' \parallel s_2 \xrightarrow{v} q'$ in the construction of $\mathcal{B}$) such that an asynchronous state $\mathcal{B}$-related to $p_1 \parallel s_1$ is reached.

Next, we show that $\mathcal{B}$ is a witnessing branching bisimulation. We first enumerate the cases that show every transition from the state $p_1 \parallel s_1$ is simulated by the state $\Delta(p_2\,|[\mu,\nu]|\, s_2)$ when $(p_1 \parallel s_1, \Delta(p_2\,|[\mu,\nu]|\, s_2)) \in \mathcal{B}$.

1. Let $p_1 \parallel s_1 \xrightarrow{\alpha} p_3 \parallel s_3$, $(p_1 \parallel s_1, \Delta(p_2\,|[\mu,\nu]|\, s_2)) \in \mathcal{B}$, and $\alpha \in M_p$ (the case when $\alpha \in M_s \cup E_s \cup E_p$ is similar). From the construction of $\mathcal{B}$ we have $p_1 \parallel s_1 \in \Re(p \parallel s)$ and there exists $p_2', s_2', p_4, p_4', s_4, s_4', u, v$ such that

$$p_2 \parallel s_2' \xrightarrow{u} p_4 \parallel s_4' \leftrightarrow_{\mathbf{b}} p_1 \parallel s_1 \leftrightarrow_{\mathbf{b}} p_4' \parallel s_4 \xleftarrow{v} p_2' \parallel s_2. \tag{1}$$

   Applying Proposition 1 we get $p_2 \xrightarrow{?u} p_4$ and $s_2 \xrightarrow{?v} s_4$. Thus, we get $\Delta(p_2\,|[\mu,\nu]|\, s_2) \xrightarrow{\epsilon} \Delta(p_4\,|[\epsilon,\epsilon]|\, s_4)$. But from above we have $p_4 \parallel s_4' \leftrightarrow_{\mathbf{b}} p_1 \parallel s_1 \leftrightarrow_{\mathbf{b}} p_4' \parallel s_4$. Now instantiating transfer properties of branching bisimulation and since the processes p, s are concrete we get:

$$p_4 \parallel s_4' \xrightarrow{m} p_5 \parallel s_5' \leftrightarrow_{\mathbf{b}} p_3 \parallel s_3 \leftrightarrow_{\mathbf{b}} p_5' \parallel s_5 \xleftarrow{m} p_4' \parallel s_4,$$

   for some $p_5, p_5', s_5, s_5' \in \mathbb{P}$. Thus, $\Delta(p_4\,|[\epsilon,\epsilon]|\, s_4) \xrightarrow{m} \Delta(p_5\,|[\epsilon,m]|\, s_4)$. Finally, using the facts $p_5 \parallel s_5' \leftrightarrow_{\mathbf{b}} p_3 \parallel s_3 \leftrightarrow_{\mathbf{b}} p_5' \parallel s_5$ and $p_4' \parallel s_4 \xrightarrow{m} p_5' \parallel s_5$ in the construction of $\mathcal{B}$ we conclude that $(p_3 \parallel s_3, \Delta(p_5\,|[\epsilon,m]|\, s_4)) \in \mathcal{B}$.
2. Let $(p_1 \parallel s_1, \Delta(p_2\,|[\mu,\nu]|\, s_2)) \in \mathcal{B}$ and $(p_1 \parallel s_1)\sqcup$. By following the arguments of Case 1 we can derive $\Delta(p_2\,|[\mu,\nu]|\, s_2) \xrightarrow{\epsilon} \Delta(p_4\,|[\epsilon,\epsilon]|\, s_4)$, for some $p_4, s_4 \in \mathbb{P}$. Clearly, $(\Delta(p_4\,|[\epsilon,\epsilon]|\, s_4))\sqcup$. Now using the facts $p_4 \parallel s_4' \leftrightarrow_{\mathbf{b}} p_1 \parallel s_1 \leftrightarrow_{\mathbf{b}} p_4' \parallel s_4$ from (1) in the construction of $\mathcal{B}$ we get $(p_1 \parallel s_1, \Delta(p_4\,|[\epsilon,\epsilon]|\, s_4)) \in \mathcal{B}$.

The proof for the other direction is an intricate one and we distinguish the following cases.

1. Let $\Delta(p_2\,|[\mu,\nu]|\, s_2) \xrightarrow{\alpha} \Delta(p_4\,|[\mu',\nu']|\, s_4)$, $(p_1 \parallel s_1, \Delta(p_2\,|[\mu,\nu]|\, s_2)) \in \mathcal{B}$, and $\alpha \in E_s$ (the case when $\alpha \in E_p$ is symmetric). Then, by semantics we get $p_2 = p_4, s_2 \xrightarrow{e} s_4, \mu' = \mu, \nu' = \nu$. From the construction of $\mathcal{B}$ we get the solid transitions depicted in Figure 7. Using the transition $s_2 \xrightarrow{e} s_4$ we get the dashed transition (1) in Figure 7.

   Consider the transition $p_2' \parallel s_2 \xrightarrow{v} p_3' \parallel s_3$. From Proposition 1 we have $p_2' \xrightarrow{!v} p_3'$. By generalised well-posedness we get the dashed transition (2) in Figure 7. By independence of external actions we get the two dashed transitions labelled as (3) in Figure 7. From input-determinism we get $p_3' \parallel s_6' \leftrightarrow_{\mathbf{b}} p_3' \parallel s_3$. And from the transfer conditions of branching bisimulation (under the concreteness assumption) we get the remaining dashed transitions (4) and (5).

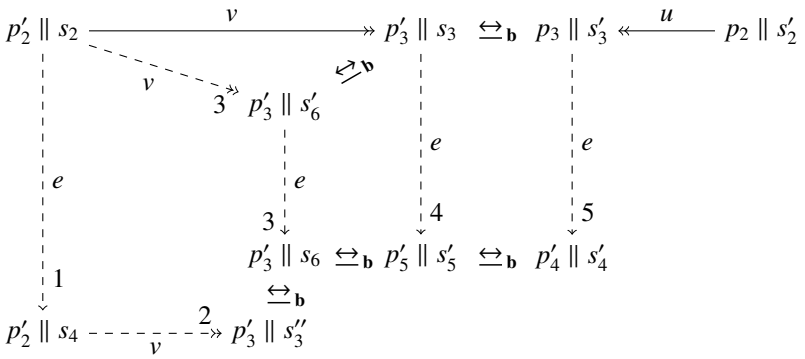


Figure 7: Case 1 of Theorem 6.

Furthermore, from the construction of $\mathcal{B}$ we have $p_3 \parallel s'_3 \leftrightarrows_{\mathbf{b}} p_1 \parallel s_1$. And, from the transfer conditions of a branching bisimulation (under concreteness assumption) we get there exists p'_1, s'_1 such that $p_1 \parallel s_1 \xrightarrow{e} p'_1 \parallel s'_1$ and $p'_1 \parallel s'_1 \leftrightarrows_{\mathbf{b}} p'_4 \parallel s'_4$. Finally, using the transitions $p_2 \parallel s'_2 \xrightarrow{u.e} p'_4 \parallel s'_4$, $p'_2 \parallel s_4 \xrightarrow{v} p'_3 \parallel s''_3$ and the fact that $p'_3 \parallel s''_3 \leftrightarrows_{\mathbf{b}} p'_4 \parallel s'_4 \leftrightarrows_{\mathbf{b}} p'_1 \parallel s'_1$ in the construction of $\mathcal{B}$ we conclude that $(p'_1 \parallel s'_1, \Delta(p_2 \,|[\mu, \nu]|\, s_4)) \in \mathcal{B}$.

2. Let $\Delta(p_2 \,|[\mu, \nu]|\, s_2) \xrightarrow{\alpha} \Delta(p_4 \,|[\mu', \nu']|\, s_4)$, $(p_1 \parallel s_1, \Delta(p_2 \,|[\mu, \nu]|\, s_2)) \in \mathcal{B}$, and $\alpha \in M_s$ (the case when $\alpha \in M_p$ is symmetric). Then, $p_2 = p_4, \mu' = \mu.n, \nu' = \nu, s_2 \xrightarrow{!n} s_4$, where $\alpha = n$, for $n \in M_s$. The remainder of the proof is similar to the previous case; except we use the diamond property instead of applying independence of external actions when both μ and ν are non-empty.
3. Let $\Delta(p_2 \,|[\mu, \nu]|\, s_2) \xrightarrow{\tau} \Delta(p_4 \,|[\mu', \nu']|\, s_4)$, $(p_1 \parallel s_1, \Delta(p_2 \,|[\mu, \nu]|\, s_2)) \in \mathcal{B}$. Since the communicating components are concrete, the above transition is due to the removal of an element, either from μ or from ν. Thus,
   (a) In case an element is removed from μ, then, $p_2 \xrightarrow{?n} p_4, \mu = n.\mu', \nu = \nu', s_2 = s_4$, for some $n \in M_s$. From the construction of $\mathcal{B}$ we have the transitions $p_2 \parallel s'_2 \xrightarrow{u} p_3 \parallel s'_3$ and $p'_2 \parallel s_2 \xrightarrow{v} p'_3 \parallel s_3$. Since $\bar{u} = \mu = n.\mu'$, we can decompose the transition $p_2 \parallel s'_2 \xrightarrow{u} p_3 \parallel s'_3$ as shown by the solid lines in Figure 8, where $\bar{u}_1 = \epsilon$, and $\bar{u}_2 = \mu'$. As $p_2 \xrightarrow{?n} p_4$, we get the dashed transition (1) in Figure 8. By generalised well-posedness, we get the dashed transition (2) in Figure 8, for some $p_5 \in \mathbb{P}$. And from input-determinism we get $p_5 \parallel s'_3 \leftrightarrows_{\mathbf{b}} p_3 \parallel s'_3$. Finally, by using the transitions $p_4 \parallel s''_4 \xrightarrow{u_2} p_5 \parallel s'_3$, $p'_2 \parallel s_2 \xrightarrow{v} p'_3 \parallel s_3$ and the facts $\bar{u}_2 = \mu'$ and $p_5 \parallel s'_3 \leftrightarrows_{\mathbf{b}} p'_3 \parallel s_3 \leftrightarrows_{\mathbf{b}} p_1 \parallel s_1$ in the construction of $\mathcal{B}$ we conclude that $(p_1 \parallel s_1, \Delta(p_4 \,|[\mu', \nu]|\, s_2)) \in \mathcal{B}$.
   (b) In case an element is removed from ν, the proof is symmetric.
4. Let $(\Delta(p_2 \,|[\mu, \nu]|\, s_2))\sqcup$ and $(p_1 \parallel s_1, \Delta(p_2 \,|[\mu, \nu]|\, s_2)) \in \mathcal{B}$. Trivial. □

The next corollary states that substituting branching bisimulation equivalence by syntactical equivalence in the conditions of Theorem 6 does not affect the desynchronisability of a synchronous system.

Corollary 1. *Let $p \parallel s$ be concrete, well-posed, independent of external actions modulo $=$, input deterministic modulo $=$, and satisfies the diamond property modulo $=$, then $p \parallel s \leftrightarrows_{\mathbf{b}} \Delta(p \,|[\epsilon, \epsilon]|\, s)$.*

Note that the definition of the relation $\mathcal{B}$ given in the proof of Theorem 6 is independent of the size of queues. This leads us to claim, without further proof, that the conditions of Theorem 6 are also sufficient in case we use lossless queues of finite size with back-pressure to construct our asynchronous system (just like the main theorems of [8]).

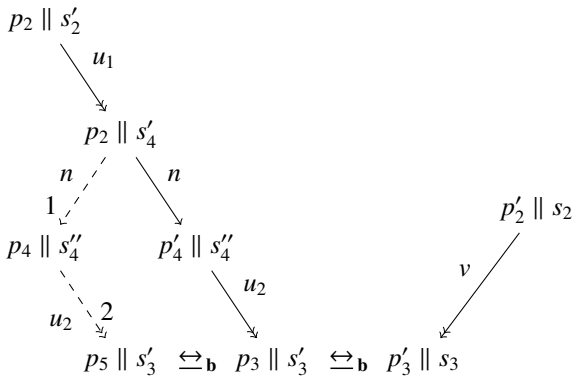


Figure 8: Case 3 of Theorem 6.

3.6. A simplified sufficient condition for desynchronisability

In this subsection, we give an equivalent sufficient condition for desynchronisability which is much simpler to verify on a given synchronous system than the preconditions of Corollary 1. The motive is to find the equivalent formulations of independence of external actions (modulo =) and input determinism (modulo =) using the single step transition relation $\rightarrow$, instead of using the multiple steps transition relation $\longrightarrow\!\!\!\!\!\rightarrow$.

Definition 10. A synchronous system $p \parallel s$ is *locally* independent of external actions if the following conditions holds for every $p_1 \parallel s_1 \in \Re(p \parallel s)$.

1. $\forall p_2, p_2', s_2, n, e.\Big[\Big(e \in E_p \wedge n \in M_s \wedge p_1 \parallel s_1 \xrightarrow{e} p_2 \parallel s_1 \xrightarrow{n} p_2' \parallel s_2\Big) \Rightarrow \exists p_3.\Big[p_1 \parallel s_1 \xrightarrow{n} p_3 \parallel s_2 \xrightarrow{e} p_2' \parallel s_2\Big]\Big]$.
2. $\forall p_2, s_2, s_2', m, e.\Big[\Big(e \in E_s \wedge m \in M_p \wedge p_1 \parallel s_1 \xrightarrow{e} p_1 \parallel s_2 \xrightarrow{m} p_2 \parallel s_2'\Big) \Rightarrow \exists s_3.\Big[p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_3 \xrightarrow{e} p_2 \parallel s_2'\Big]\Big]$.

Lemma 5. *A concrete synchronous system $p \parallel s$ is locally independent of external actions iff it is independent of external actions modulo =.*

Proof. The *only-if* part follows directly from Definition 7. To prove the *if* part, assume the transitions $p_1 \parallel s_1 \xrightarrow{e} p_2 \parallel s_1 \xrightarrow{u} p_2' \parallel s_2$, where $p_1 \parallel s_1 \in \Re(p \parallel s)$, $e \in E_p$, and $u \in (M_s \cup E_s)^*$. We show by induction on u that there exists p_3 such that $p_1 \parallel s_1 \xrightarrow{u} p_3 \parallel s_2 \xrightarrow{e} p_2' \parallel s_2$. Without loss of generality, assume $u = u'.\alpha$ such that $p_2 \parallel s_1 \xrightarrow{u'} p_3' \parallel s_2' \xrightarrow{\alpha} p_2' \parallel s_2$, for some $p_3', s_2' \in \mathbb{P}$ and $u' \in (M_s \cup E_s)^*$ and $\alpha \in M_s \cup E_s$. Then, by induction hypothesis we have $p_1 \parallel s_1 \xrightarrow{u'} p_3 \parallel s_2' \xrightarrow{e} p_3' \parallel s_2'$, for some $p_3 \in \mathbb{P}$. Now performing case distinction on α we get the following cases.

1. Let $\alpha = e'$ for some $e' \in E_s$. Then, by the semantics we have $p_3' = p_2'$ and $s_2' \xrightarrow{e'} s_2$. Using this transition at the state $p_3 \parallel s_2'$ we get $p_3 \parallel s_2' \xrightarrow{e'} p_3 \parallel s_2$. But, from inductive hypothesis we have $p_3 \xrightarrow{e} p_2'$. Thus, $p_3 \parallel s_2 \xrightarrow{e} p_2' \parallel s_2$; hence, $p_1 \parallel s_1 \xrightarrow{u} p_3 \parallel s_2 \xrightarrow{e} p_2' \parallel s_2$ as required.
2. Let $\alpha = n$ for some $n \in M_s$. Then, by applying Definition 10 we get $p_3 \parallel s_2' \xrightarrow{n} p_4 \parallel s_2 \xrightarrow{e} p_2' \parallel s_2$, for some $p_4 \in \mathbb{P}$. Thus, $p_1 \parallel s_1 \xrightarrow{u} p_4 \parallel s_2 \xrightarrow{e} p_2' \parallel s_2$ as required. □

Definition 11. A synchronous system $p \parallel s$ is *locally* input deterministic if every reachable state $p_1 \parallel s_1 \in \Re(p \parallel s)$ satisfies the following conditions.

1. $\forall p_2, s_2, p_3, n.\Big[\Big(p_1 \parallel s_1 \xrightarrow{n} p_2 \parallel s_2 \wedge p_1 \parallel s_1 \xrightarrow{n} p_3 \parallel s_2 \wedge n \in M_s\Big) \Rightarrow p_2 \parallel s_2 = p_3 \parallel s_2\Big]$.

2. $\forall p_2, s_2, s_3, m.\Big[\Big(p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_2 \wedge p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_3 \wedge m \in M_p\Big) \Rightarrow p_2 \parallel s_2 = p_2 \parallel s_3\Big]$.

Lemma 6. *A concrete synchronous system $p \parallel s$ is locally input deterministic iff it is input deterministic modulo $=$.*

Proof. The *only-if* part follows directly from Definition 8. The *if* part follows by performing induction on sequences $u \in (M_s \cup E_s)^*$ ($v \in (M_p \cup E_p)^*$) and instantiating Definition 11. □

Now, we are ready to prove the main result of this subsection.

Theorem 7. *Let $p \parallel s$ be concrete, well-posed, locally independent of external actions, locally input deterministic, and satisfies the diamond property modulo $=$, then $p \parallel s \leftrightarrows_{\mathbf{b}} \Delta(p\,|[\epsilon,\epsilon]|\,s)$.*

Proof. By Lemmas 5,6 and Corollary 1. □

3.7. Half-duplex communication eliminates the diamonds

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

Changing the notion of equivalence or the observation of the predicate is not likely to help. Previous research [3, 7] 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 4. 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.

Table 4: SOS rules for asynchronous systems with half-duplex queues.

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

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

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

Definition 12. A synchronous system $p \parallel s$ is *half-duplex desynchronisable* 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 8. *Let $p \parallel s$ be concrete and half-duplex desynchronisable, then it is well-posed, independent of external actions, and input deterministic.*

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

Theorem 9. *Suppose a concrete process $p \parallel s$ is well-posed, independent of external actions, and input deterministic, then it is half-duplex desynchronisable.*

Proof. Recall the relation $\mathcal{B}$ from Theorem 6 and replace the full-duplex operator with our new half-duplex operator. This relation will serve as a witness for our theorem along the same lines set out in Theorem 6, except that in Case 2 we find for the queue contents that either μ or ν will be empty due to the half-duplex condition. As a result, the necessity of the diamond property in this proof disappears. □

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 not allowed to send messages while its input queue is non-empty. Furthermore, our only reason for wanting half-duplex communication is that we could not guarantee the diamond property for our synchronous system.

In essence, the half-duplex condition just ensures a certain level of synchronisation over the communication buffer. In practice, half-duplex communication can only be implemented if some kind of semaphore is in place on top of the physical layer.

Table 5: SOS rules for semi-duplex communication over a set I.

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

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

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

Now, suppose that we do have the diamond property for certain pairs of actions in the synchronous system. In such a case, a specialised 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 5. We call this *semi-duplex* communication.

We conjecture that the necessary and sufficient conditions for desynchronisation using such a buffer are well-posedness, independence of external actions, input determinism, and the diamond property for pairs of messages, one of which is in the subset I. We actually expect the proof to be along the same lines as the other ones in this paper.

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 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 [21]). 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.

4. Desynchronisation in supervisory control

Regarding supervisory control theory of Ramadge and Wonham [6], we should still check whether the conditions we have gotten so far are reasonable. That is the topic of this section. Furthermore, we are also interested in exploring to what extent the assumptions of supervisory control theory of [6] can be relaxed without affecting the obtained conditions of (half-duplex) desynchronisability. In this section, we will extend Theorem 9 of [18] by allowing external actions and nondeterminism in the specification of a supervisor, which are usually prohibited in supervisory control theory.

The three basic entities in supervisory control theory are a *plant*, a *supervisor*, and a *requirement*. Intuitively, a plant is specified by an automaton that models the behaviour of a hardware that is to be controlled, a supervisor is an automaton that forces the plant to meet the requirement by synchronously interacting with it, and a requirement is an automaton that models the legal behaviour which the plant should perform.

Supervisory control theory [6] aims at controlling the behaviour of a plant to fit a requirement by synthesising a supervisor such that the synchronous composition of a plant and its supervisor is equivalent to the requirement. In supervisory control theory, the control problem is usually studied up to language equivalence (cf. [6]). Nevertheless, the synthesis techniques developed in this paper are insensitive to the choice of equivalence used in the control equation, provided that this equivalence is coarser than branching bisimulation.

To define a notion of controllability, a plant and its supervisor perform two kinds of actions: *controllable* and *uncontrollable* actions. The idea behind this partition is that a supervisor can enable or disable controllable actions of a plant, but it cannot disable the uncontrollable actions of a plant. In this paper, we follow the *input-output interpretation* [7] between a plant and its supervisor, i.e., the uncontrollable actions are modelled as the send messages from a plant to its supervisor, while the controllable actions are modelled as the send messages from the supervisor to the plant.

Definition 13. A concrete process $p \in \mathbb{P}$ is called a *RW-plant*[3] if p is deterministic and $E_p = \emptyset$. Similarly, a concrete process $s \in \mathbb{P}$ is called a *RW-supervisor* if s is deterministic and $E_s = \emptyset$.

Observe that a RW-plant and RW-supervisor do not contain external actions in their alphabet. This is because in the theory of Ramadge and Wonham [6] a plant can perform only either controllable or uncontrollable actions. Furthermore, the synthesis procedure ensures that a supervisor restricts the behaviour of a plant by synchronising with some of the controllable actions and no extra transitions are introduced in a supervisor that a plant cannot execute. Therefore, all the send messages from a plant are read by its supervisor; however, all the send messages from a supervisor are not *necessarily* read by its plant. Thus, the synthesised supervisor in general is not well-posed with its plant. For example, we generated the supervisor described in [11] and found that the generated supervisor was not well-posed with plant.

4.1. Ensuring well-posedness

In order to synthesise 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 6. Now the idea is that the process $\gamma(p \parallel s)$ is the new supervisor for the given plant p, which is always well-posed with p. Before we prove this result, we recall an important property which every supervisor in the supervisory control theory of Ramadge and Wonham [6] satisfies.

Definition 14. A process p is *controllable* [6, 22] by a process s if

$$\forall p_1, s_1, p_2, m. \left[p_1 \parallel s_1 \in \mathfrak{R}(p \parallel s) \wedge p_1 \xrightarrow{!m} p_2 \Rightarrow \exists s_2. \left[s_1 \xrightarrow{?m} s_2 \right] \right].$$

[3] The prefix RW stands for Ramadge and Wonham [6].

Table 6: SOS rules for renaming using a function γ.

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

Note that, in general, controllability does not imply well-posedness between a plant and its supervisor. This is why we construct a new supervisor in the form of $\gamma(p \parallel s)$, which always returns a well-posed supervisor for the plant p. The following lemma proves this fact. Furthermore, observe that in the following lemma neither we require the supervisor to be deterministic nor the set of external actions of the supervisor needs to be empty.

Lemma 7. *Let p be a RW-plant and let s be a process such that p is controllable by s. Then p and $\gamma(p \parallel s)$ are well-posed.*

Proof. Define a binary relation $\mathcal{W} = \{(p_1, \gamma(p_1 \parallel s_1)) \mid p_1 \parallel s_1 \in \Re(p \parallel s)\}$. Next, we show that $\mathcal{W}$ is a well-posedness relation.

1. Let $p_1 \xrightarrow{!m} p_2$ and $(p_1, \gamma(p_1 \parallel s_1)) \in \mathcal{W}$. Since p is controllable by s, then there exists $s_2 \in \mathbb{P}$ such that $s_1 \xrightarrow{?m} s_2$. Thus, $p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_2$. Using the definition of renaming function γ we get $\gamma(p_1 \parallel s_1) \xrightarrow{?m} \gamma(p_2 \parallel s_2)$. Thus, $(p_2, \gamma(p_2 \parallel s_2)) \in \mathcal{W}$.

   Furthermore, we need to show that for every p_3, s_3 such that $\gamma(p_1 \parallel s_1) \xrightarrow{?m} \gamma(p_3 \parallel s_3)$, we have $(p_2, \gamma(p_3 \parallel s_3)) \in \mathcal{W}$. Assume a transition $\gamma(p_1 \parallel s_1) \xrightarrow{?m} \gamma(p_3 \parallel s_3)$. Then by the semantics of $\parallel$ and γ we have $p_1 \xrightarrow{!m} p_3$. But, the process p is deterministic, thus, $p_2 = p_3$. Hence, $(p_2, \gamma(p_3 \parallel s_3)) \in \mathcal{W}$.
2. Let $\gamma(p_1 \parallel s_1) \xrightarrow{!n} \gamma(p_2 \parallel s_2)$ and $(p_1, \gamma(p_1 \parallel s_1)) \in \mathcal{W}$. By definition of renaming function γ we have $p_1 \parallel s_1 \xrightarrow{n} p_2 \parallel s_2$ and $n \in M_s$. Since the sets M_p, M_s are disjoint we have $p_1 \xrightarrow{?n} p_2$ and $s_1 \xrightarrow{!n} s_2$. And, using the construction of $\mathcal{W}$ we get $(p_2, \gamma(p_2 \parallel s_2)) \in \mathcal{W}$. Furthermore, we know that the plant p is deterministic. Thus, for every $p_3 \in \mathbb{P}$ such that $p_2 \xrightarrow{?n} p_3$ we have $p_2 = p_3$.
3. Let $\gamma(p_1 \parallel s_1) \xrightarrow{\alpha} \gamma(p_1 \parallel s_2)$, $\alpha \in E_s \cup \{\tau\}$ and $(p_1, \gamma(p_1 \parallel s_1)) \in \mathcal{W}$. Clearly, by the construction of $\mathcal{W}$ we have $(p_1, \gamma(p_1 \parallel s_2))$. □

The next lemma states that a RW-plant p cannot distinguish the behaviour between its old supervisor s and the new well-posed supervisor $\gamma(p \parallel s)$ up to strong bisimulation.

Lemma 8. *Let p be a RW-plant and let s be a process. Then, $p \parallel s \leftrightarrows p \parallel \gamma(p \parallel s)$.*

Proof. Define a witnessing relation $\mathcal{S} = \{(p_1 \parallel s_1, p_1 \parallel \gamma(p_1 \parallel s_1)) \mid p_1 \parallel s_1 \in \Re(p \parallel s)\}$. Next, we need to show that $\mathcal{S}$ is a strong bisimulation relation.

1. Let $p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_2$, for some $m \in M_p$ and $(p_1 \parallel s_1, p_1 \parallel \gamma(p_1 \parallel s_1)) \in \mathcal{S}$. Then, by the semantics we have $p_1 \xrightarrow{!m} p_2$ and $s_1 \xrightarrow{?m} s_2$. And from the definition of the renaming function γ we get $\gamma(p_1 \parallel s_1) \xrightarrow{?m} \gamma(p_2 \parallel s_2)$. Clearly, $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{m} p_2 \parallel \gamma(p_2 \parallel s_2)$. Thus, we get $(p_2 \parallel s_2, p_2 \parallel \gamma(p_2 \parallel s_2))$ as desired.
2. Let $p_1 \parallel s_1 \xrightarrow{\alpha} p_2 \parallel s_2$, for some $\alpha \in M_s \cup E_s$, and $(p_1 \parallel s_1, p_1 \parallel \gamma(p_1 \parallel s_1)) \in \mathcal{S}$. Similar to the previous case.
3. Let $p_1 \parallel s_1 \xrightarrow{\tau} p_2 \parallel s_2$ and $(p_1 \parallel s_1, p_1 \parallel \gamma(p_1 \parallel s_1)) \in \mathcal{S}$. Note that the process p is a RW-plant, thus this τ-transition is due to a move by s_1. Then, by semantics we have $p_1 = p_2$ and $s_1 \xrightarrow{\tau} s_2$. Clearly, we have $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{\tau} p_1 \parallel \gamma(p_1 \parallel s_2)$ and $(p_1 \parallel s_2, p_1 \parallel \gamma(p_1 \parallel s_2)) \in \mathcal{S}$.

4. Let $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{m} p_2 \parallel \gamma(p_2' \parallel s_2)$, for some $m \in M_p$ and $(p_1 \parallel s_1, p_1 \parallel \gamma(p_1 \parallel s_1)) \in \mathcal{S}$. Using the fact that the sets M_p, M_s are disjoint in the semantics, we get $p_1 \xrightarrow{!m} p_2$ and $\gamma(p_1 \parallel s_1) \xrightarrow{?m} \gamma(p_2' \parallel s_2)$. Since $m \in M_p$ we have $p_1 \xrightarrow{!m} p_2'$ and $s_1 \xrightarrow{?m} s_2$. But, the RW-plant process p is deterministic, i.e., $p_2 = p_2'$. Furthermore, $p_1 \parallel s_1 \xrightarrow{m} p_2 \parallel s_2$ and by the construction of $\mathcal{S}$ we conclude that $(p_2 \parallel s_2, p_2 \parallel \gamma(p_2 \parallel s_2)) \in \mathcal{S}$.
5. Let $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{n} p_2 \parallel \gamma(p_2' \parallel s_2)$, for some $n \in M_s$ and $(p_1 \parallel s_1, p_1 \parallel \gamma(p_1 \parallel s_1)) \in \mathcal{S}$. Similar to the previous case.
6. Let $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{\alpha} p_2 \parallel \gamma(p_2' \parallel s_2)$, for some $\alpha \in E_s \cup \{\tau\}$ and $(p_1 \parallel s_1, p_1 \parallel \gamma(p_1 \parallel s_1)) \in \mathcal{S}$. Note that the process p is concrete and $E_p = \emptyset$. Thus, this α-transition is due to a move by s_1. Then, by the semantics we have $p_1 = p_2 = p_2'$ and $s_1 \xrightarrow{\alpha} s_2$. Clearly, we have $p_1 \parallel s_1 \xrightarrow{\alpha} p_1 \parallel s_2$ and $(p_1 \parallel s_2, p_1 \parallel \gamma(p_1 \parallel s_2)) \in \mathcal{S}$. □

4.2. Ensuring input-determinism, independence of external actions, and diamond property

Next we will show that the remaining conditions of (half-duplex) desynchronisability (i.e., input-determinism, independence of external actions, and diamond property) are satisfied by the new synchronous system $p \parallel \gamma(p \parallel s)$, whenever the old synchronous system $p \parallel s$ satisfies these conditions. In other words, we will show that input-determinism, independence of external actions, and diamond property are strong bisimulation-closed. However, to prove these results we need the following result (Lemma 10): if $p \parallel s \leftrightarrows p \parallel \gamma(p \parallel s)$ then $p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1)$, for every $p_1 \parallel s_1 \in \Re(p \parallel s)$.

Lemma 9. *Let p be a RW-plant. Let s, s' be any two processes. If $p \parallel s \leftrightarrows p \parallel \gamma(p \parallel s')$ then $p \parallel s \leftrightarrows p \parallel s'$.*

Proof. Define a relation $\mathcal{S}$ in the following way:

$$\mathcal{S} = \Big\{(p_1 \parallel s_1, p_1 \parallel s_1') \mid p_1 \parallel s_1 \in \Re(p \parallel s) \wedge p_1 \parallel s_1' \in \Re(p \parallel s') \wedge p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1')\Big\}.$$

Next, we show that the relation $\mathcal{S}$ is a witnessing strong bisimulation relation.

1. Let $p_1 \parallel s_1 \xrightarrow{\alpha} p_2 \parallel s_2$ and $(p_1 \parallel s_1, p_1 \parallel s_1') \in \mathcal{W}$. Then, from the construction of $\mathcal{S}$ we have

$$\begin{aligned}
& p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1') \\
\Rightarrow\ & \exists p_2, p_2', s_2'.\Big[p_1 \parallel \gamma(p_1 \parallel s_1') \xrightarrow{\alpha} p_2 \parallel \gamma(p_2' \parallel s_2') \wedge p_2 \parallel s_2 \leftrightarrows p_2 \parallel \gamma(p_2' \parallel s_2')\Big] \quad \text{(Definition of } \leftrightarrows\text{)} \\
\Rightarrow\ & p_2 = p_2' \quad (p \text{ is deterministic}) \\
\Rightarrow\ & p_1 \parallel s_1' \xrightarrow{\gamma^{-1}(\alpha)} p_2 \parallel s_2' \quad \text{(Semantics of } \parallel \text{ and } \gamma\text{)} \\
\Rightarrow\ & (p_2 \parallel s_2, p_2 \parallel s_2') \in \mathcal{W} \quad \text{(Using } p_2 \parallel s_2 \leftrightarrows p_2 \parallel \gamma(p_2 \parallel s_2') \text{ in the construction of } \mathcal{S}\text{)}.
\end{aligned}$$

2. Let $p_1 \parallel s_1' \xrightarrow{\alpha} p_2 \parallel s_2'$ and $(p_1 \parallel s_1, p_1 \parallel s_1') \in \mathcal{W}$. Then based on the type of α we identify the following cases:
   (a) Let $\alpha = m$, for some $m \in M_p$. Then from the semantics of $\parallel$ we have

$$\begin{aligned}
& p_1 \xrightarrow{!m} p_2 \wedge s_1' \xrightarrow{?m} s_2' \\
\Rightarrow\ & p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1') \quad \text{(Inductive hypothesis)} \\
\Rightarrow\ & p_1 \parallel \gamma(p_1 \parallel s_1') \xrightarrow{m} p_2 \parallel \gamma(p_2 \parallel s_2') \quad \text{(Semantics of } \parallel \text{ and } \gamma\text{)} \\
\Rightarrow\ & \exists p_3, s_3.\Big[p_1 \parallel s_1 \xrightarrow{m} p_3 \parallel s_3 \wedge p_3 \parallel s_3 \leftrightarrows p_2 \parallel \gamma(p_2 \parallel s_2')\Big] \quad \text{(Definition of } \leftrightarrows \text{ and } m \in M_p\text{)} \\
\Rightarrow\ & p_1 \xrightarrow{!m} p_3 \wedge p_2 = p_3 \quad \text{(Semantics of } \parallel \text{ and } p \text{ is deterministic)} \\
\Rightarrow\ & (p_2 \parallel s_3, p_2 \parallel s_2') \in \mathcal{W} \quad \text{(Using } p_2 \parallel s_3 \leftrightarrows p_2 \parallel \gamma(p_2 \parallel s_2') \text{ in the construction of } \mathcal{S}\text{)}.
\end{aligned}$$

(b) Let $\alpha \in E_{s'} \cup M_{s'} \cup \{\tau\}$. Similar to the previous case.

Note that the transfer conditions for the predicate $\sqcup$ are trivially satisfied by the construction of $\mathcal{S}$. □

Lemma 10. *Let p be a RW-plant and s be a process. If $p \parallel s \leftrightarrows p \parallel \gamma(p \parallel s)$ then*

$$\forall p_1, s_1. \left[p_1 \parallel s_1 \in \Re(p \parallel s) \Rightarrow p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1)\right].$$

Proof. Without loss of generality, assume $p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1)$ and $p_1 \parallel s_1 \xrightarrow{\alpha} p_2 \parallel s_2$, for some $p_1 \parallel s_1 \in \Re(p \parallel s)$ and $p_2, s_2 \in \mathbb{P}$. Then, by the semantics we have $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{\alpha} p_2 \parallel \gamma(p_2 \parallel s_2)$. And from the transfer conditions of strong bisimulation we have there exists p_2', s_2' such that $p_1 \parallel s_1 \xrightarrow{\alpha} p_2' \parallel s_2'$ and $p_2' \parallel s_2' \leftrightarrows p_2 \parallel \gamma(p_2 \parallel s_2)$.

1. Let $\alpha \in E_s \cup \{\tau\}$. Then by the semantics of $\parallel$ under the assumption that p is a RW-plant we have $p_1 = p_2'$ and $p_1 = p_2$. Thus, $p_1 \parallel s_2' \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_2)$. And, from Lemma 9 we have $p_1 \parallel s_2' \leftrightarrows p_1 \parallel s_2$. Finally, from transitivity we conclude that $p_1 \parallel s_2 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_2)$.
2. Let $\alpha \in M_p \cup M_s$. Then we have $p_2 = p_2'$ because the process p is deterministic. Thus, $p_2 \parallel s_2' \leftrightarrows p_2 \parallel \gamma(p_2 \parallel s_2)$. From Lemma 9 we get $p_2 \parallel s_2' \leftrightarrows p_2 \parallel s_2$. Finally, from transitivity we conclude that $p_2 \parallel s_2 \leftrightarrows p_2 \parallel \gamma(p_2 \parallel s_2)$. □

The next lemma states that if the given synchronous system $p \parallel s$ is input-deterministic then the new synchronous system $p \parallel \gamma(p \parallel s)$ is also input-deterministic.

Lemma 11. *Let p be a RW-plant and s be a concrete process. If $p \parallel s$ is input-deterministic modulo $\leftrightarrows$ then $p \parallel \gamma(p \parallel s)$ is also input-deterministic modulo $\leftrightarrows$.*

Proof. Recall the definition of input-determinism modulo $\leftrightarrows$ (Definition 8). Thus, we need to prove the following statements:

1. If $p_1 \parallel \gamma(p_1 \parallel s_1) \in \Re(p \parallel \gamma(p \parallel s))$, $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{u}\!\!\!\!\rightarrow p_2 \parallel \gamma(p_2' \parallel s_2)$, and $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{u}\!\!\!\!\rightarrow p_3 \parallel \gamma(p_2' \parallel s_2)$, then $p_2 \parallel \gamma(p_2' \parallel s_2) \leftrightarrows p_3 \parallel \gamma(p_2' \parallel s_2)$, for some $u \in (E_s \cup M_s)^*$.
2. If $p_1 \parallel \gamma(p_1 \parallel s_1) \in \Re(p \parallel \gamma(p \parallel s))$, $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{v}\!\!\!\!\rightarrow p_2 \parallel \gamma(p_2' \parallel s_2)$, and $p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{v}\!\!\!\!\rightarrow p_2 \parallel \gamma(p_3' \parallel s_3)$, then $p_2 \parallel \gamma(p_2' \parallel s_2) \leftrightarrows p_2 \parallel \gamma(p_3' \parallel s_3)$, for some $v \in (E_p \cup M_p)^*$.

The proof of the first statement follows directly from the reflexivity of $\leftrightarrows$ because the process p is deterministic. In other words, for the transitions $p_1 \xrightarrow{?u}\!\!\!\!\rightarrow p_2$ and $p_1 \xrightarrow{?u}\!\!\!\!\rightarrow p_3$ we have $p_2 = p_3$. Thus, $p_2 \parallel \gamma(p_2' \parallel s_2) \leftrightarrows p_2 \parallel \gamma(p_2' \parallel s_2)$.

To prove the second statement, using Lemma 10 we have $p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1)$. Thus, from the transfer conditions of strong bisimulation we have there exists p_4, s_4 such that $p_1 \parallel s_1 \xrightarrow{v}\!\!\!\!\rightarrow p_4 \parallel s_4$ and $p_4 \parallel s_4 \leftrightarrows p_2 \parallel \gamma(p_2' \parallel s_2)$. Likewise, we can derive $p_1 \parallel s_1 \xrightarrow{v}\!\!\!\!\rightarrow p_5 \parallel s_5$ and $p_5 \parallel s_5 \leftrightarrows p_2 \parallel \gamma(p_3' \parallel s_3)$, for some p_5, s_5. From the above transitions we have $p_1 \xrightarrow{!v}\!\!\!\!\rightarrow p_4$ and $p_1 \xrightarrow{!v}\!\!\!\!\rightarrow p_5$. But the process p is deterministic and thus we have $p_4 = p_5$. Since the synchronous system is input-deterministic modulo $\leftrightarrows$ we have $p_4 \parallel s_4 \leftrightarrows p_4 \parallel s_5$. And from transitivity we get the desired result, i.e., $p_2 \parallel \gamma(p_2' \parallel s_2) \leftrightarrows p_2 \parallel \gamma(p_3' \parallel s_3)$ □

The next lemma states that if the given synchronous system $p \parallel s$ is independent of external actions then the new synchronous system $p \parallel \gamma(p \parallel s)$ is also independent of external actions.

Lemma 12. *Let p be a RW-plant and s be a concrete process. If $p \parallel s$ is independent of external actions modulo $\leftrightarrows$ then $p \parallel \gamma(p \parallel s)$ is also independent of external actions modulo $\leftrightarrows$.*

Proof. Note that $E_p = \emptyset$ since p is a RW-plant. As a result, Condition 1 of Definition 7 is vacuously satisfied. To show Condition 2 of Definition 7 assume that we are given the following transitions

$$p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{e} p_1 \parallel \gamma(p_1 \parallel s_2) \xrightarrow{v}\!\!\!\!\rightarrow p_2 \parallel \gamma(p_2' \parallel s_2),$$

for some $e \in E_s$, $v \in {M_p}^*$. Note that $p_2 = p'_2$ because the process p is deterministic.

By Lemma 10 we know that $p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1)$. Then, from the transfer conditions of strong bisimulation we can derive the following transitions:

$$\exists p'_1, s'_3, p_3, s_3. \Big[p_1 \parallel s_1 \xrightarrow{e} p'_1 \parallel s'_3 \xrightarrow{v}\!\!\!\!\rightarrow p_3 \parallel s_3 \wedge p'_1 \parallel s'_3 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_2) \wedge p_3 \parallel s_3 \leftrightarrows p_2 \parallel \gamma(p_2 \parallel s_2) \Big]. \tag{2}$$

Since $e \in E_s$ and $E_p = \emptyset$ we know that $p_1 = p'_1$.

Now instantiating Condition 2 of independence of external actions on the state $p_1 \parallel s_1$ we get

$$\exists s'_4, s_4. \Big[p_1 \parallel s_1 \xrightarrow{v}\!\!\!\!\rightarrow p_3 \parallel s'_4 \xrightarrow{e} p_3 \parallel s_4 \wedge p_3 \parallel s_4 \leftrightarrows p_3 \parallel s_3 \Big]. \tag{3}$$

Again using the transfer conditions of strong bisimulation to simulate the transitions derived in Equation (3) we get $\exists p_5, s_5, p'_5, s'_5. \Big[p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{v}\!\!\!\!\rightarrow p_5 \parallel \gamma(p'_5 \parallel s_5) \xrightarrow{e} p_5 \parallel \gamma(p'_5 \parallel s'_5) \wedge p_5 \parallel \gamma(p'_5 \parallel s'_5) \leftrightarrows p_3 \parallel s_4 \Big]$. Note that $p_2 = p_5 = p'_5$ because the process p is deterministic, $e \in E_s$, and $E_p = \emptyset$. And from Equations 2 and 3 we have

$$p_2 \parallel \gamma(p_2 \parallel s'_5) \leftrightarrows p_3 \parallel s_4 \leftrightarrows p_3 \parallel s_3 \leftrightarrows p_2 \parallel \gamma(p_2 \parallel s_2). \qquad \square$$

Lemma 13. *Let p be a RW-plant and s be a concrete process. If $p \parallel s$ satisfies the diamond property modulo $\leftrightarrows$ then $p \parallel \gamma(p \parallel s)$ also satisfies the diamond property modulo $\leftrightarrows$.*

Proof. Assume the following transitions in the synchronous system $p \parallel \gamma(p \parallel s)$:

$$p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{m} p_2 \parallel \gamma(p'_2 \parallel s_2) \text{ and } p_1 \parallel \gamma(p_1 \parallel s_1) \xrightarrow{n} p_3 \parallel \gamma(p'_3 \parallel s_3),$$

for some $m \in M_p$, $n \in M_s$, and $p_1 \parallel \gamma(p_1 \parallel s_1) \in \Re(p \parallel \gamma(p \parallel s))$.

By Lemma 10 we know that $p_1 \parallel s_1 \leftrightarrows p_1 \parallel \gamma(p_1 \parallel s_1)$. From the transfer conditions of strong bisimulation we get the following transition in the synchronous system $p \parallel s$: $p_1 \parallel s_1 \xrightarrow{m} p_4 \parallel s_4 \wedge p_4 \parallel s_4 \leftrightarrows p_2 \parallel \gamma(p'_2 \parallel s_2)$ and $p_1 \parallel s_1 \xrightarrow{n} p_5 \parallel s_5 \wedge p_5 \parallel s_5 \leftrightarrows p_3 \parallel \gamma(p'_3 \parallel s_3)$, for some p_4, s_4, p_5, s_5. But the process $p \parallel s$ satisfies the diamond property modulo $\leftrightarrows$. Thus, we get there exists p'_4, s'_4, p'_5, s'_5 such that $p_4 \parallel s_4 \xrightarrow{n} p'_4 \parallel s'_4$, $p_5 \parallel s_5 \xrightarrow{m} p'_5 \parallel s'_5$, and $p'_4 \parallel s'_4 \leftrightarrows p'_5 \parallel s'_5$. Clearly by reflecting back these transitions in the synchronous system $p \parallel \gamma(p \parallel s)$ using the transfer conditions of strong bisimulation we get the desired result.

I.e., there exists p_6, p'_6, s_6 such that $p_2 \parallel \gamma(p'_2 \parallel s_2) \xrightarrow{n} p_6 \parallel \gamma(p'_6 \parallel s_6)$ and $p'_4 \parallel s'_4 \leftrightarrows p_6 \parallel \gamma(p'_6 \parallel s_6)$. Likewise, there exists p_7, p'_7, s_7 such that $p_3 \parallel \gamma(p'_3 \parallel s_3) \xrightarrow{m} p_7 \parallel \gamma(p'_7 \parallel s_7)$ and $p'_5 \parallel s'_5 \leftrightarrows p_7 \parallel \gamma(p'_7 \parallel s_7)$. Finally, using the equation $p'_4 \parallel s'_4 \leftrightarrows p'_5 \parallel s'_5$ and by transitivity we conclude that $p_6 \parallel \gamma(p'_6 \parallel s_6) \leftrightarrows p_7 \parallel \gamma(p'_7 \parallel s_7)$. $\square$

Now we are ready to prove the main result of this subsection that establishes half-duplex desynchronisability of the synchronous system $p \parallel \gamma(p \parallel s)$.

Theorem 10. *Let p be a RW-plant and let s be a concrete process. If p is controllable by s, $p \parallel s$ is input-deterministic modulo $\leftrightarrows$, $p \parallel s$ is independent of external actions modulo $\leftrightarrows$, and $p \parallel s$ satisfies the diamond property modulo $\leftrightarrows$, then*

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

Proof. Lemmas 7, 11, 12, and 13 ensures that the synchronous system $p \parallel \gamma(p \parallel s)$ is well-posed, input-deterministic modulo $\leftrightarrows$, independent of external actions modulo $\leftrightarrows$, and satisfies the diamond property modulo $\leftrightarrows$, respectively. Finally, from Theorem 6 we conclude that the synchronous system $p \parallel \gamma(p \parallel s)$ is desynchronisable. $\square$

Corollary 2. *Let p be a RW-plant and let s be a concrete process. If p is controllable by s, $p \parallel s$ is input-deterministic modulo $\leftrightarrows$, and $p \parallel s$ is independent of external actions modulo $\leftrightarrows$, then*

$$p \parallel \gamma(p \parallel s) \leftrightarrows_{\mathbf{b}} p \,|[\epsilon, \epsilon]|_{\mathrm{h}}\, \gamma(p \parallel s).$$

The following corollary states that every synchronous system synthesised using supervisory control theory of Ramadge and Wonham [6] is half-duplex desynchronisable by construction.

Corollary 3. *Let p be a RW-plant and s be its RW-supervisor. Then, $p \parallel s \leftrightarrows_{\mathbf{b}} p \parallel \gamma(p \parallel s) \leftrightarrows_{\mathbf{b}} p\,|[\epsilon, \epsilon]|_{\mathrm{h}}\,\gamma(p \parallel s)$.*

Proof. Recall that every RW-plant p is controllable by its RW-supervisor s which guarantees well-posedness of $p \parallel \gamma(p \parallel s)$. Furthermore, the synchronous system $p \parallel s$ is input-deterministic and independent of external actions because a RW-plant p and its RW-supervisor s are deterministic and do not contain external actions. Thus, from Theorem 10 we have $p \parallel \gamma(p \parallel s) \leftrightarrows_{\mathbf{b}} p\,|[\epsilon, \epsilon]|_{\mathrm{h}}\,\gamma(p \parallel s)$. Since a RW-plant and a RW-supervisor are concrete processes, applying Lemma 8 give us the desired equation $p \parallel s \leftrightarrows_{\mathbf{b}} p \parallel \gamma(p \parallel s) \leftrightarrows_{\mathbf{b}} p\,|[\epsilon, \epsilon]|_{\mathrm{h}}\,\gamma(p \parallel s)$. □

In hindsight, the absence of external actions and nondeterminism in the plant model allowed us to fulfil the conditions for half-duplex desynchronisability almost for free. The only thing required was the well-posedness between a given RW-plant p and its supervisor s. To this end, we demonstrated that the process $\gamma(p \parallel s)$ – if considered as a new supervisor – will always be well-posed with the given RW-plant p.

Unfortunately, the above approach to achieve desynchronisability fails in general when the plant contains external actions and nondeterminism. The following are two immediate problems in this new setting that needs to be addressed in the future.

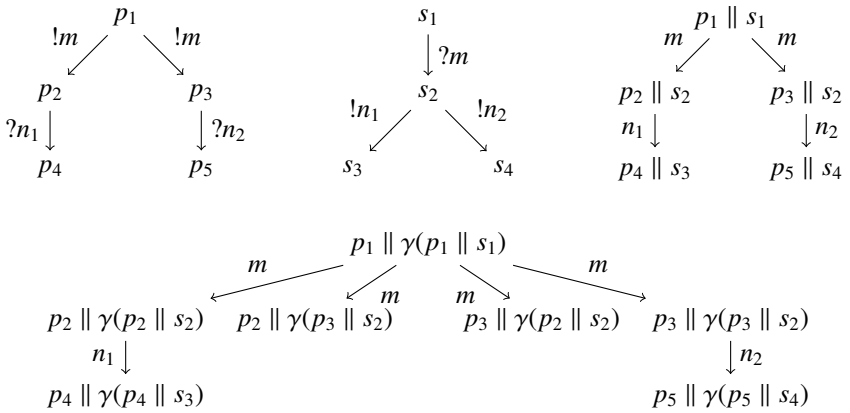


Figure 9: Difficulty in establishing bisimilarity between $p_1 \parallel s_1$ and $p_1 \parallel \gamma(p_1 \parallel s_1)$ when p_1 is nondeterministic.

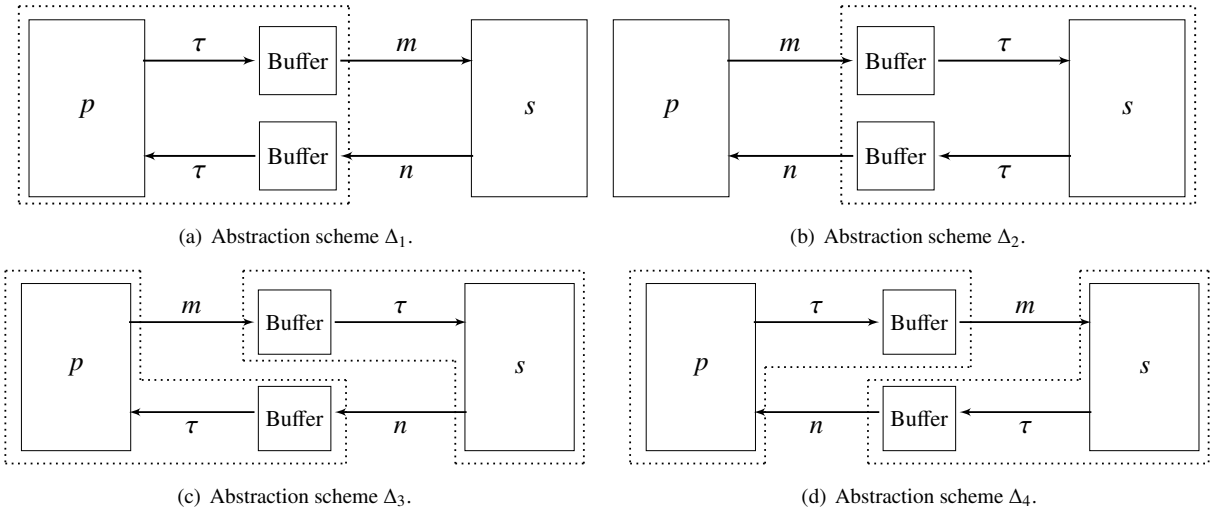
1. First, the issue of ensuring $p \parallel s \leftrightarrows p \parallel \gamma(p \parallel s)$ is more involved when the plant process p contains nondeterminism. Consider, for instance, the plant p_1 and its supervisor s_1 as shown in Figure 9. The problem here is that a supervisor state is trying to control two different plant states, where the interaction of the supervisor with one local plant state disables the interaction with the other. For example, the interaction of p_3 with s_2 disables the sending of message n_1; even though, s_2 can send the message n_1. As a result, the processes $p_1 \parallel s_1$ and $\gamma(p_1 \parallel \gamma(p_1 \parallel s_1))$ are not strongly bisimilar because the states $p_2 \parallel \gamma(p_3 \parallel s_2)$ and $p_3 \parallel \gamma(p_2 \parallel s_2)$ cannot be related to any state in the process $p_1 \parallel s_1$ by a strong bisimulation relation.
2. Second, the construction of new supervisor as the parallel composition of the given plant and its old supervisor is unsound in the presence of external actions from the plant. This is because the new supervisor $\gamma(p \parallel s)$ can now perform the external actions of the plant p, whereas the old supervisor s cannot. Thus, the equality $p \parallel s \leftrightarrows p \parallel \gamma(p \parallel s)$ (Lemma 8) does not hold when we drop the assumption $E_p = \emptyset$.

5. Discussion

In this section, we discuss the effects of using other abstraction schemes on desynchronisation and also consider the issue of desynchronising synchronous systems with invisible actions.

5.1. Why abstraction scheme Δ(_)?

Throughout this paper, we developed a theory of desynchronisation around a fixed abstraction scheme Δ() in which the interactions between the communicating processes and their respective input queues were made invisible. At this juncture, it is interesting to find out whether is it possible to further weaken the obtained conditions of desynchronisability by adopting the other abstraction schemes of [10].



(a) Abstraction scheme Δ_1.

(b) Abstraction scheme Δ_2.

(c) Abstraction scheme Δ_3.

(d) Abstraction scheme Δ_4.

Figure 10: Abstraction schemes [8].

The motivation for our work in this paper originates from supervisory control theory, although its application may be far more general. This origin is reflected to some extent in the choice of the abstraction scheme Δ(), because in supervisory control theory, the specification of what the controlled system should do is usually given in terms of the observed communications. Hence, it is for example undesirable to choose an abstraction scheme in which all communication is considered unobservable (as in some CCS-like formalisms).

Given that communications should remain visible, we have to make a choice whether they become visible when a message is sent or when a message is received. In our setting, this leads to four possible abstraction schemes, shown in Figure 10. The dashed boxes in Figure 10 denotes which interactions are to be made hidden in an asynchronous system. For instance, the abstraction scheme Δ_1 states that introduce buffers between the processes p and s such that *the interactions between the process p and the buffers are* hidden (see Figure 10(a)). Similar explanations can be given to the other abstraction schemes Δ_2, Δ_3[4], and Δ_4. For the sake of completeness, the abstraction schemes Δ_1, Δ_2, and Δ_4 are formally defined in Table 7.

Of these four abstraction schemes, it is difficult to say which one is in fact more adequate. One might call an abstraction scheme adequate if it preserves all desired properties. But the only difference between the abstraction schemes is the interpretation of a 'communication' as something that is visible upon sending or upon receiving. The properties of interest to us are assumed to be definable for synchronous models as well. And in synchronous models this distinction is not yet made. Hence, the properties that are relevant for us, when properly interpreted, are preserved by each of the four abstraction schemes. So the only reason to pick one or another, is the restrictions it puts on our desynchronisation theorem. Next, we will argue that these conditions are indeed most convenient for abstraction scheme Δ_3.

One of the results of [10] is that the choice of the abstraction scheme is crucial in obtaining useful conditions for desynchronisation. For instance, if we choose to construct an asynchronous system using the abstraction scheme Δ_4 instead of Δ_3, then we need an additional condition for desynchronisation saying that at any reachable state of $p \parallel s$ only one send-transition is allowed. This condition is known as *X-singularity* in [10], where $X \subseteq A$.

[4]For the sake of uniformity, we denote the abstraction operator Δ by Δ_3 in this subsection.

Table 7: SOS rules for the abstraction schemes Δ_1, Δ_2, and Δ_4, where $\square \in \{!, ?\}$.

$$\frac{q_1 \xrightarrow{\square m} q_2,\ \square m \in ?M_s \cup !M_s}{\Delta_1(q_1) \xrightarrow{m} \Delta_1(q_2)} \qquad \frac{q_1 \xrightarrow{e} q_2,\ e \in E_p \cup E_s}{\Delta_1(q_1) \xrightarrow{e} \Delta_1(q_2)} \qquad \frac{q_1 \xrightarrow{\square m} q_2,\ \square m \in ?M_p \cup !M_p}{\Delta_1(q_1) \xrightarrow{\tau} \Delta_1(q_2)} \qquad \frac{q\sqcup}{\Delta_1(q)\sqcup}$$

$$\frac{q_1 \xrightarrow{\square m} q_2,\ \square m \in ?M_p \cup !M_p}{\Delta_2(q_1) \xrightarrow{m} \Delta_2(q_2)} \qquad \frac{q_1 \xrightarrow{e} q_2,\ e \in E_p \cup E_s}{\Delta_2(q_1) \xrightarrow{e} \Delta_2(q_2)} \qquad \frac{q_1 \xrightarrow{\square m} q_2,\ \square m \in ?M_s \cup !M_s}{\Delta_2(q_1) \xrightarrow{\tau} \Delta_2(q_2)} \qquad \frac{q\sqcup}{\Delta_2(q)\sqcup}$$

$$\frac{q_1 \xrightarrow{\square m} q_2,\ \square m \in ?M_p \cup ?M_s}{\Delta_4(q_1) \xrightarrow{m} \Delta_4(q_2)} \qquad \frac{q_1 \xrightarrow{e} q_2,\ e \in E_p \cup E_s}{\Delta_4(q_1) \xrightarrow{e} \Delta_4(q_2)} \qquad \frac{q_1 \xrightarrow{\square m} q_2,\ \square m \in !M_p \cup !M_s}{\Delta_4(q_1) \xrightarrow{\tau} \Delta_4(q_2)} \qquad \frac{q\sqcup}{\Delta_4(q)\sqcup}$$

Definition 15. Let $X \subseteq A$. A process $q \in \mathbb{P}$ is *X-singular* if every reachable state $q_1 \in \Re(q)$ satisfies the following condition:

$$\forall q_2, q_3, \alpha_1, \alpha_2 \in X. \left[q_1 \xrightarrow{\alpha_1} q_2 \wedge q_1 \xrightarrow{\alpha_2} q_3 \Rightarrow \alpha_1 = \alpha_2 \right].$$

Example. *To see the role of M_p-singular in desynchronisation, consider the following transition systems of p_1, s_1:*

$$\left(\{p_1, p_2, p_3\}, \{p_1 \xrightarrow{!m_1} p_2, p_1 \xrightarrow{!m_2} p_3\} \right), \quad \textit{and}$$
$$\left(\{s_1, s_2, s_3\}, \{s_1 \xrightarrow{?m_1} s_2, s_1 \xrightarrow{?m_2} s_3\} \right).$$

The transition system of the synchronous system $p_1 \parallel s_1$ and the asynchronous system $\Delta_4(p_1 \,|[\epsilon, \epsilon]|\, s_1)$ constructed using the abstraction scheme Δ_4 are shown in Figure 11. An attempt to construct a branching bisimulation relation between the two systems will fail to relate the states $p_1 \parallel s_1$ and $\Delta_4(p_2 \,|[\epsilon, m_1]|\, s_1)$. This is because the state $\Delta_4(p_2 \,|[\epsilon, m_1]|\, s_1)$ cannot simulate the transition $p_1 \parallel s_1 \xrightarrow{m_2} p_3 \parallel s_3$. Thus, the external deterministic choice of

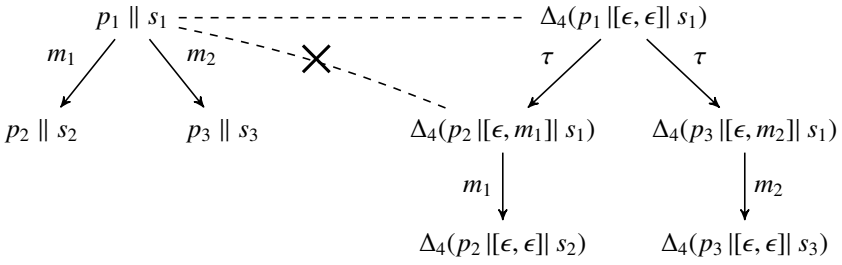


Figure 11: Non-inert τ transitions caused by the abstraction scheme Δ_4.

messages from the p-side is transformed into an internal choice by the abstraction scheme Δ_4 that cannot be resolved.

In contrast, suppose $p_1 \parallel s_1 \leftrightarrows_{\mathbf{b}} \Delta_4(p_2 \,|[\epsilon, m_1]|\, s_1)$. Then, from the transfer conditions of branching bisimulation we have there exists q_1, q_2 such that

$$\Delta_4(p_2 \,|[\epsilon, m_1]|\, s_1) \longrightarrow\!\!\!\!\!\rightarrow q_1 \xrightarrow{m_2} q_2 \wedge q_1 \leftrightarrows_{\mathbf{b}} p_1 \parallel s_1 \wedge q_2 \leftrightarrows_{\mathbf{b}} p_3 \parallel s_3.$$

Since, concrete processes p, s are used to construct the asynchronous system we have $q_1 = \Delta_4(p_4 \,|[\epsilon, m_1.m_2]|\, s_2)$, for some $p_4 \in \mathbb{P}$ such that $\Delta_4(p_2 \,|[\epsilon, m_1]|\, s_1) \xrightarrow{\tau} \Delta_4(p_4 \,|[\epsilon, m_1.m_2]|\, s_2)$. But, there cannot exists a process q_2 with $\Delta_4(p_4 \,|[\epsilon, m_1.m_2]|\, s_2) \xrightarrow{m_2} q_2$ because queues are used as buffers and the message m_2 can only be read by s_2 after it has read the message m_1. Thus, this suggests that the condition M_p-singular is also a candidate for the necessary conditions of desynchronisation modulo $\leftrightarrow_{\mathbf{b}}$ when queues and the abstraction scheme Δ_4 are used.

Notice that if we had deterministic choice between the messages from the s-side in the previous example, we would have required the condition M_s-singularity for desynchronisation. Furthermore, in the case of the abstraction scheme Δ_1 (Δ_2) we would have required exclusively the condition M_p-singularity (M_s-singularity) because only the outputs from p-side (s-side) are made invisible in the abstraction scheme Δ_1 (Δ_2). However, in the case of the abstraction scheme Δ_3, the send messages of both the processes p, s remains visible; thus, the choice present in the synchronous system is preserved when it is made asynchronous. In other words, the abstraction scheme Δ_3 prevents an external choice from being transformed into an internal choice, which was the purpose of imposing the conditions M_p-singularity or M_s-singularity.

Next, we argue that in the context of half-duplex mechanism, the abstraction scheme Δ_3 again yields a less restrictive condition for desynchronisation in comparison to the other abstraction schemes. This is essentially due to the abstraction scheme Δ_3 that preserves the external choices between the messages of distinct local processes present at a state in a synchronous system upon adding buffers. The abstraction schemes Δ_1, Δ_2, and Δ_4 fail to preserve these nondeterministic choices, thus disallowing the two systems to be related by a branching bisimulation relation. Notice that here we are highlighting the preservation of an external choice between the messages of *distinct* local processes; whereas, in the previous paragraph, we focused on the preservation of an external choice between the messages of *a* local process.

Consider the following transition system of a synchronous system $p_1 \parallel s_1$:

$$\Big(\{p_1 \parallel s_1, p_2 \parallel s_2, p_3 \parallel s_3\}, \{p_1 \parallel s_1 \xrightarrow{n} p_2 \parallel s_2, p_1 \parallel s_1 \xrightarrow{m} p_3 \parallel s_3\}\Big),$$

where $n \in M_s$ and $m \in M_p$. In Figure 12, the transition systems induced by different abstraction schemes are drawn. The dotted lines show an attempt to construct a branching bisimulation relation between the synchronous system and the asynchronous systems with the respective abstraction schemes. Observe that a branching bisimulation relation only exists in the case of abstraction scheme Δ_3. For instance, consider the case of Δ_1, where the states $p_1 \parallel s_1$ and $\Delta_1(p_3 \,|[\epsilon, m]|_{\mathrm{h}}\, s_1)$ are not branching bisimilar. This is due to the half-duplex mechanism which prevents the process s_1 to send the message $!n$ and thus, disallowing the state $\Delta_1(p_3 \,|[\epsilon, m]|_{\mathrm{h}}\, s_1)$ to simulate the matching message n. Similar reasons can be given for the abstraction schemes Δ_2 and Δ_4.

Thus, in the context of (half-duplex) queues, the abstraction scheme Δ_3 leads to less restrictive conditions for desynchronisability than the abstraction schemes Δ_1, Δ_2, or Δ_4. This is because the abstraction scheme Δ_3 is the only abstraction scheme that preserves the external choice between the messages sent by the communicating processes in a synchronous system. On the contrary, the other abstraction schemes Δ_1, Δ_2, or Δ_4 fail to preserve these choices by transforming them into the internal choices in the corresponding asynchronous system and thus, an additional constraint like every reachable state of the synchronous system has only one send-transition is required for desynchronisability.

5.2. Desynchronisation of non-concrete synchronous systems

In this subsection, we focus on the desynchronisation 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 either the process p or 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 possibility to circumvent this problem is by minimising the communicating processes p, s modulo branching bisimulation into p', s', respectively. In case, there are no τ-transitions in the transition systems of the processes p', s', then (half-duplex) desynchronisability of the original synchronous system $p \parallel s$ follows directly from (Corollary 4) Lemma 14.

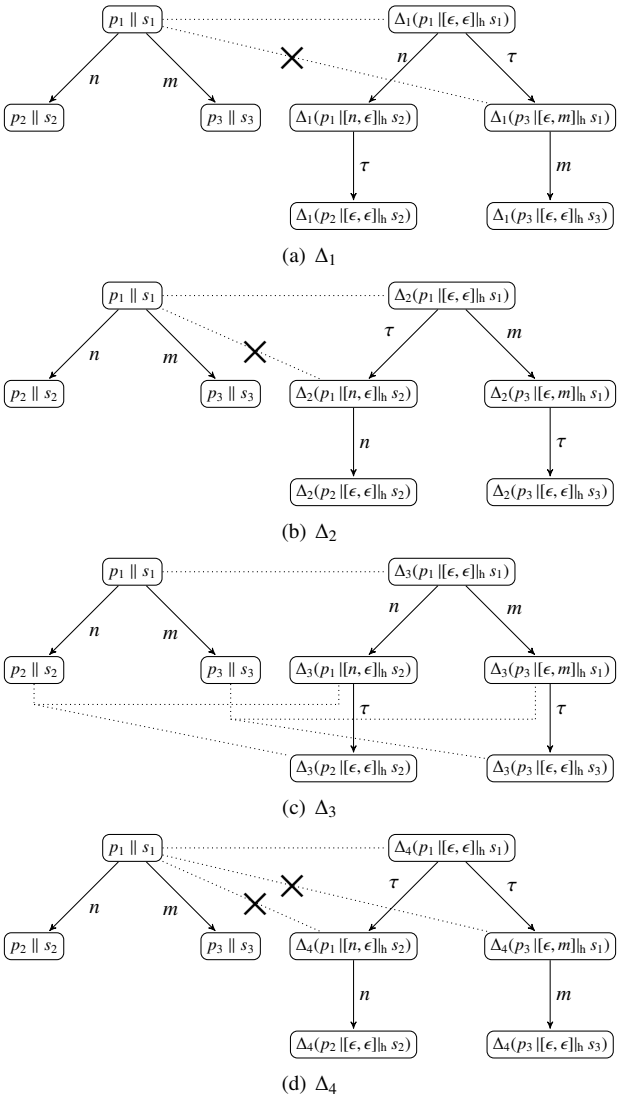


Figure 12: Different behaviour induced by the abstractions schemes $\Delta_1, \Delta_2, \Delta_3$, and Δ_4 in the presence of half-duplex mechanism.

Lemma 14. *Let* $p_1 \leftrightarrow_{\mathbf{b}} p_2$ *and* $s_1 \leftrightarrow_{\mathbf{b}} s_2$*. Then,* $p_1 \,|[\epsilon,\epsilon]|\, s_1 \leftrightarrow_{\mathbf{b}} p_2 \,|[\epsilon,\epsilon]|\, s_2$*.*

Proof. Define a relation $\mathcal{B} = \{(p_1 \,|[\mu,\nu]|\, s_1, p_2 \,|[\mu,\nu]|\, s_2) \mid p_1 \leftrightarrow_{\mathbf{b}} p_2 \wedge s_1 \leftrightarrow_{\mathbf{b}} s_2\}$. It is routine to verify that $\mathcal{B}$ is a branching bisimulation relation. □

Corollary 4. *Let* p_1, s_1, p_2, s_2 *be any four processes such that* $p_1 \leftrightarrow_{\mathbf{b}} p_2$ *and* $s_1 \leftrightarrow_{\mathbf{b}} s_2$*. Then,* $p_1 \,|[\epsilon,\epsilon]|_{\mathrm{h}}\, s_1 \leftrightarrow_{\mathbf{b}} p_2 \,|[\epsilon,\epsilon]|_{\mathrm{h}}\, s_2$*.*

Theorem 11. *Let* p, p', s, s' *be any four processes such that* $p \leftrightarrow_{\mathbf{b}} p'$ *and* $s \leftrightarrow_{\mathbf{b}} s'$*. If* $p' \parallel s'$ *is (half-duplex) desynchronisable, then* $p \parallel s$ *is (half-duplex) desynchronisable.*

Proof. Direct from (Corollary 4) Lemma 14. □

However, if the τ-transitions are present in the communicating processes even after branching bisimulation minimisation, it is still possible to desynchronise the synchronous systems in the following way. First, rename the label τ of every τ-transitions present in the processes p and s by the labels τ_p and τ_s, respectively. Second, 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 9) can still be used to assert whether a non-concrete synchronous system is desynchronisable (half-duplex desynchronisable) or not. Nevertheless, despite this soundness result, more research is required in order to examine to what extent are these conditions necessary in the absence of concreteness assumption.

6. Conclusions

In this paper, we studied necessary and sufficient conditions for desynchronisability 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 characterisation of desynchronisability modulo branching bisimulation; moreover, the previous works (cf. [3, 7–9]) on weaker equivalences focused only on giving sufficient conditions for desynchronisability. The study of necessity of those conditions is new and only obtained by assuming observability of empty communication channels.

Our results indicate that the study of desynchronisability 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 desynchronisability results can be obtained even for the finest equivalence in the van Glabbeek spectrum. Perhaps some of the necessary conditions can be relaxed by resorting to any equivalence weaker than contra-simulation [17]. For example, we know that we can eliminate the input determinism property by studying desynchronisability modulo contra-simulation. 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 (see Subsection 5.1). On the other hand, we obtained interesting results in [8] 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 a deterministic and concrete plant without external actions, we showed that it is possible to synthesise a controller that satisfies the well-posedness property by construction. Note that this technique even works for supervisors that contain nondeterminism and external actions in their specifications (see Lemma 7). Furthermore, we also showed that the remaining conditions of desynchronisability are preserved by bisimulation equivalence (Lemmas 11, 12, and 13). However, for a nondeterministic plant with external actions, such correct-by-construction results may be difficult to obtain. Therefore, it would be beneficial if tools for model checking asynchronous systems, like mCRL2 [23] and CADP [24], could be optimised to check the conditions of desynchronisability as well.

As another topic for future research, we would be interested to see if a different choice of abstraction scheme (for example hiding all communication messages in the synchronous and asynchronous system, only leaving the external actions) gives us the possibility to drop the condition of independence of external actions. So far we did not do this, because in supervisory control theory the messages of $p \parallel s$ are often part of the requirement r. Still, we might abstract from those messages that are not used in r, so that their reception does not need to be independent of external behavior anymore. To the best of our knowledge this scenario has not yet been systematically treated in the literature, and at this point it is hard to tell what the necessary and sufficient conditions for such a desynchronisation would be.

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, 25]. Basically, such choreographies serve to restrict the occurrence of diamonds in an asynchronous system, which means that it becomes synchronisable [5]. Perhaps it is also possible to use this idea in the other direction, i.e., to desynchronise a system using a choreography on the communication buffer. It would be interesting to see if, for example, the proposed semi-duplex buffering strategy discussed in Subsection 3.7 can be implemented using a choreography.

Acknowledgement

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] C. A. R. Hoare, Communicating sequential processes, Commun. ACM 21 (8) (1978) 666–677.
[2] D. Brand, P. Zafiropulo, On Communicating Finite-State Machines, J. ACM 30 (1983) 323–342, ISSN 0004-5411.
[3] C. Fischer, W. Janssen, Synchronous Development of Asynchronous Systems, in: U. Montanari, V. Sassone (Eds.), Proceedings of CONCUR'96, vol. 1119 of *Lecture Notes in Computer Science*, Springer-Verlag, 735–750, 1996.
[4] S. Basu, T. Bultan, Choreography conformance via synchronizability, in: Proceedings of the 20th international conference on World wide web, WWW '11, ACM, New York, NY, USA, 795–804, 2011.
[5] S. Basu, T. Bultan, M. Ouederni, Synchronizability for Verification of Asynchronously Communicating Systems, in: V. Kuncak, A. Rybalchenko (Eds.), Verification, Model Checking, and Abstract Interpretation, vol. 7148 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 56–71, 2012.
[6] P. J. Ramadge, W. M. Wonham, Supervisory Control of a Class of Discrete Event Processes, SIAM Journal on Control and Optimization 25 (1) (1987) 206–230.
[7] S. Balemi, Control of Discrete Event Systems: Theory And Application, Ph.D. thesis, Swiss Federal Institute of Technology, Automatic Control Laboratory, ETH Zurich, 1992.
[8] H. Beohar, P. J. L. Cuijpers, Desynchronizability of (Partial) Synchronous Closed Loop Systems, Scientific Annals of Computer Science 21 (2011) 5–38.
[9] J. Udding, Classification and Composition of Delay-Insensitive Circuits, Ph.D. thesis, Eindhoven University of Technology, Eindhoven, 1984.
[10] H. Beohar, Refinement of communication and states in models of embedded systems, Ph.D. thesis, Eindhoven university of technology, Eindhoven, The Netherlands, 2013.
[11] S. T. J. Forschelen, Supervisory control of theme park vehicles, Master's thesis, Eindhoven University of Technology, System Engineering Group, Dept. of Mechanical Engineering, 2010.
[12] A. Tanenbaum, Computer Networks, Prentice Hall Professional Technical Reference, 4th edn., 2002.
[13] G. Cécé, A. Finkel, Verification of programs with half-duplex communication, Inf. Comput. 202 (2005) 166–190.
[14] M. Fabian, A. Hellgren, PLC-based implementation of supervisory control for discrete event systems, Proceedings of the 37th IEEE Conference on Decision and Control, 1998 3 (1998) 3305–3310.
[15] R. Hennicker, S. Janisch, A. Knapp, Refinement of Components in Connection-Safe Assemblies with Synchronous and Asynchronous Communication, in: C. Choppy, O. Sokolsky (Eds.), Foundations of Computer Software. Future Trends and Techniques for Development, vol. 6028 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 154–180, 2010.
[16] J. C. M. Baeten, T. Basten, M. A. Reniers, Process Algebra: Equational Theories of Communicating Processes, Cambridge University Press, New York, NY, USA, 1st edn., 2009.
[17] R. J. v. Glabbeek, The Linear Time - Branching Time Spectrum II, in: Proceedings of the 4th International Conference on Concurrency Theory, CONCUR '93, Springer-Verlag, London, UK, 66–81, 1993.
[18] H. Beohar, P. J. L. Cuijpers, Avoiding Diamonds in Desynchronisation, in: C. Pasareanu, G. Salaün (Eds.), 9th International Symposium on Formal Aspects of Component Software, Springer Link, Mountain View, California, USA, 2012.
[19] G. D. Plotkin, A Structural Approach to Operational Semantics, Tech. Rep. DAIMI FN-19, University of Aarhus, 1981.
[20] L. Alfaro, T. Henzinger, Interface-Based Design, in: M. Broy, J. Grnbauer, D. Harel, C. A. R. Hoare (Eds.), Engineering Theories of Software Intensive Systems, vol. 195 of *NATO Science Series*, Springer Netherlands, 83–104, 2005.
[21] K. Peters, J.-W. Schicke, U. Nestmann, Synchrony vs Causality in Asynchronous Pi-Calculus, in: B. Luttik, F. Valencia (Eds.), 18th International Workshop on Expressiveness in Concurrency, EXPRESS, vol. 64 of *EPTCS*, 89–103, 2011.
[22] M. Fabian, B. Lennartson, On non-deterministic supervisory control, in: Decision and Control, 1996., Proceedings of the 35th IEEE Conference on, vol. 2, 2213–2218, 1996.
[23] J. F. Groote, A. Mathijssen, M. Reniers, Y. Usenko, M. van Weerdenburg, The Formal Specification Language mCRL2, in: E. Brinksma, D. Harel, A. Mader, P. Stevens, R. Wieringa (Eds.), Methods for Modelling Software Systems, Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany, 2007.
[24] R. Mateescu, Modeling and Verification of Real-Time Systems, chap. Specification and Analysis of Asynchronous Systems using CADP, ISTE, ISBN 9780470611012, 141–169, 2010.
[25] G. Salaün, T. Bultan, Realizability of Choreographies Using Process Algebra Encodings, in: Proceedings of the 7th International Conference on Integrated Formal Methods, IFM '09, Springer-Verlag, Berlin, Heidelberg, 167–182, 2009.