Noname manuscript No.
(will be inserted by the editor)

Synchrony and Asynchrony in Conformance Testing

Neda Noroozi[1,2], **Ramtin Khosravi**[3],
Mohammad Reza Mousavi[1], **Tim A.C. Willemse**[1]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Fanap Corporation (IT Subsidiary of Pasargad Bank), Tehran, Iran
[3] University of Tehran, Tehran, Iran

The date of receipt and acceptance will be inserted by the editor

Abstract We present and compare different notions of conformance testing based on labeled transition systems. We formulate and prove several theorems which enable using synchronous conformance testing techniques such as input output conformance testing (**ioco**) in order to test implementations only accessible through asynchronous communication channels. These theorems define when the synchronous test cases are sufficient for checking all aspects of conformance that are observable by asynchronous interaction with the implementation under test.

1 Introduction

Due to the ubiquitous presence of distributed systems (ranging from distributed embedded systems to the Internet), it becomes increasingly important to establish rigorous model-based testing techniques with an asynchronous model of communication in mind. This fact has been noted by the pioneering pieces of work in the area of formal conformance testing, e.g., see [8, Chapter 5], [11] and [12], and has been addressed extensively by several researchers in this field ever since [2,5–7,13,14].

We stumbled upon this problem in our attempt to apply input-output conformance testing (**ioco**) [9,10] to an industrial embedded system from the banking domain [1]. A schematic view of the implementation under test (IUT) and its environment is given in Figure 1.(a). The IUT is an Electronic Funds Transfer (EFT) switch (henceforth referred to as *the switch*), which provides a communication mechanism among different components of a card-based financial system. On one side of the IUT, there are components that the end-user deals with, such as Automated Teller Machines (ATMs), Point-of-Sale (POS) devices and e-Payment applications. On the other side, there are Core-Banking systems and the inter-bank network connecting the switches of different financial institutions.

To test the switch, an automated on-line test-case generator is connected to it; the tester communicates (using an adapter) via a network with the IUT. This communication is inherently asynchronous and hence subtleties concerning asynchronous testing arise naturally in our context. A simplified specification of the switch, in which these subtleties appear, is depicted in Figure 1.(b). In this figure, the switch sends a purchase request to the core banking system and either receives a response, or after an internal step (e.g., an internal time-out, denoted by τ) sends a reversal request to the POS. In the synchronous setting, after sending a purchase request and receiving a response, observing a reversal request will lead to the fail verdict. This is justified by the fact that receiving a response should force the system to take the topmost transition at the moment of choice in the specification depicted in Figure 1.(b). However, in the asynchronous

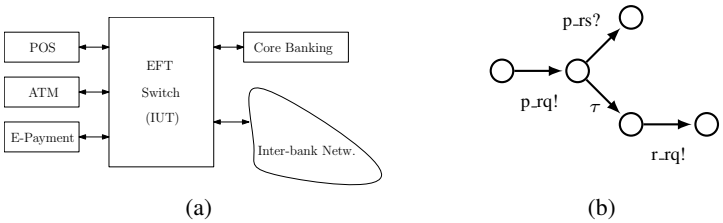


Fig. 1 The EFT Switch and a simplified specification

setting, a response is put on a channel and is yet to be communicated to the IUT. It is unclear to the remote observer when the response is actually consumed by the IUT. Hence, even when a response is sent to the system the observer should still expect to receive a reversal request.

The problems encountered in our practical case study have been encountered by other researchers. It is well-known that not all systems are amenable to asynchronous testing since they may feature phenomena (e.g., a choice between accepting input and generating output) that cannot be reliably observed in the asynchronous setting (e.g., due to unknown delays). In other words, to make sure that test-cases generated from the specification can test the IUT by asynchronous interactions and reach verdicts that are meaningful for the original IUT, either the class of IUTs, or the class of specifications, or the test-case generation algorithm (or a combination thereof) has to be adapted.

Related work. In [13, Chapter 8] and [14], both the class of IUTs has been restricted (to the so-called *internal choice* specifications) and further the test-case generation algorithm is adapted to generate a restricted set of test-cases. Then, it is argued (with a proof sketch) that in this setting, the verdict obtained through asynchronous interaction with the system coincides with the verdict (using the same set of restricted test-cases) in the synchronous setting. We give a full proof of this result in Section 5 and report a slight adjustment to it, without which a counter-example is shown to violate the property.

In [7] a method is presented for generating test-cases from the synchronous specification that are sound for the asynchronous implementation. The main idea is to saturate a test-case with observation delays caused by asynchronous interactions. In this paper, we adopt a restriction imposed on the implementation inspired by [7, Theorem 1] (dating back to [8]) and prove that in the setting of **ioco** testing this is sufficient for using synchronous test-case for the asynchronous implementation.

In [5,6] the asynchronous test framework is extended to the setting where separate test-processes can observe input and output events and relative distinguishing power of these settings are compared. Although this framework may be natural in practice, we avoid following the framework of [5,6] since our ultimate goal is to compare asynchronous testing with the standard **ioco** framework and the framework of [5,6] is notationally very different. For the same reason, we do not consider the approach of [2], which uses a stamping mechanism attached to the IUT, thus observing the actual order input and output before being distorted by the queues.

To summarize, the present paper re-visits the much studied issue of asynchronous testing and formulates and proves some theorems that show when it is (im)possible to synchronize asynchronous testing, i.e., interaction with an IUT through asynchronous channels and still obtain verdicts that coincide with that of testing the IUT using the synchronous interaction mechanisms.

This paper substantially extends the results we reported in [3,4]. Most importantly, we present a novel intensional representation of the conformance testing relation presented [13, 14] in this paper. (This was mentioned as future work in [3,4].) Using this representation, we compare the testing power of different conformance relations in [10,13,14]. Moreover, we give external representations of the studied notions by providing a generic test-case generation algo-

rithm and show that the test case generation algorithm is sound and exhaustive with respect to our intensional representation. (The novel parts, compared to [3,4], include the results presented in Sections 3 and 4.)

Structure of the paper We present in Section 2 preliminary definitions regarding labeled transition systems and different variants thereof. In Section 3, we present a unifying intensional definition of input output conformance testing, from which the different conformance relations presented in [13,14] and [10] can be obtained as special cases. In the same section, we define a notion of testing power and using that compare several notions of conformance relation obtained from different hypotheses assumed in [13,14] and [10]. In Section 4, we present corresponding extensional notions of conformance testing using test cases and show that they are indeed sound and exhaustive with respect to their intensional counterparts. We give a full proof of the main result of [13, Chapter 8] and [14] (with a slight modification) in Section 5. Then, in Section 6, we re-formulate the same results in the pure **ioco** setting and show that our constraints precisely characterize the implementations for which asynchronous testing can be reduced to synchronous testing. The paper is concluded in Section 7.

2 Preliminaries

In model-based testing theory, the two prevailing ways for modeling reactive systems are by using *finite state machines* (FSMs) [15] or *labeled transition systems* (LTSs) [10]. We are mainly concerned with the latter. In this section, we give a brief account of the concepts, relevant to LTS-based testing theory explored in this paper.

LTS models consist of states and transitions. The latter are decorated with *actions*, modeling events that trigger state changes. Events that are internal to a system, i.e., unobservable to a tester or observer of the system, are modeled by the constant action τ.

Definition 1 (LTS) *A labeled transition system (LTS) is a 4-tuple $\langle S, L, \rightarrow, s_0\rangle$, where S is a set of states, L is a finite alphabet of actions that does not contain the internal action τ, $\rightarrow \subseteq S \times (L \cup \{\tau\}) \times S$ is the transition relation, and $s_0 \in S$ is the initial state. We shall often refer to the LTS by referring to its initial state s_0.*

Fix an arbitrary LTS $\langle S, L, \rightarrow, s_0\rangle$. Let $s, s' \in S$ and $x \in L \cup \{\tau\}$. We use the standard notational conventions, *i.e.*, we write $s \xrightarrow{x} s'$ rather than $(s, x, s') \in \rightarrow$, we write $s \xrightarrow{x}$ when $s \xrightarrow{x} s'$ for some s' and we write $s \not\xrightarrow{x}$ when not $s \xrightarrow{x}$. The transition relation is generalized to (weak) traces by the following deduction rules:

$$\frac{}{s \overset{\epsilon}{\Longrightarrow} s} \qquad \frac{s \overset{\sigma}{\Longrightarrow} s'' \qquad s'' \xrightarrow{x} s' \qquad x \neq \tau}{s \overset{\sigma x}{\Longrightarrow} s'} \qquad \frac{s \overset{\sigma}{\Longrightarrow} s'' \qquad s'' \xrightarrow{\tau} s'}{s \overset{\sigma}{\Longrightarrow} s'}$$

In line with our notation for transitions, we write $s \overset{\sigma}{\Longrightarrow}$ if there is a s' such that $s \overset{\sigma}{\Longrightarrow} s'$, and $s \overset{\sigma}{\not\Longrightarrow}$ when no s' exists such that $s \overset{\sigma}{\Longrightarrow} s'$.

Definition 2 (Traces and Enabled Actions) *Let $s \in S$ and $S' \subseteq S$. We define:*

1. $\mathsf{traces}(s) =_{def} \{\sigma \in L^* \mid s \overset{\sigma}{\Longrightarrow}\}$, *and we define* $\mathsf{traces}(S') =_{def} \bigcup_{s \in S'} \mathsf{traces}(s)$
2. $\mathbf{init}(s) =_{def} \{a \in L \cup \{\tau\} \mid s \xrightarrow{a}\}$, *and we define* $\mathbf{init}(S') =_{def} \bigcup_{s \in S'} \mathbf{init}(s)$,
3. $\mathbf{Sinit}(s) =_{def} \{a \in L \mid s \overset{a}{\Longrightarrow}\}$, *and we define* $\mathbf{Sinit}(S') =_{def} \bigcup_{s \in S'} \mathbf{Sinit}(s)$.

A state in an LTS is said to *diverge* if it is the source of an infinite sequence of τ-labeled transitions. An LTS is *divergent* if one of its reachable states diverges.

Inputs, Outputs and Quiescence. In LTSs labels are treated uniformly. When engaging in an interaction with an actual system, the initiative to communicate is often not fully symmetric: the system is stimulated and observed. We therefore refine the LTS model to incorporate this distinction.

Definition 3 (IOLTS) *An input-output labeled transition system (IOLTS) is an LTS* $\langle S, L, \rightarrow, s_0 \rangle$*, where the alphabet* L *is partitioned into a set* L_I *of inputs and a set* L_U *of outputs.*

Throughout this paper, whenever we are dealing with an IOLTS (or one of its refinements), we tacitly assume that the given alphabet L for the IOLTS is partitioned in sets L_I and L_U. In our examples we distinguish inputs from outputs by annotating them with a question- (?) and exclamation-mark (!), respectively. Note that these annotations are *not* part of action names.

Observations of output, and the absence thereof, are essential ingredients in the conformance testing theories we consider. A system state that does not produce outputs is called *quiescent*. In its traditional phrasing, quiescence characterizes system states that do not produce outputs and which are stable, i.e., those that cannot evolve to another state by performing a silent action.

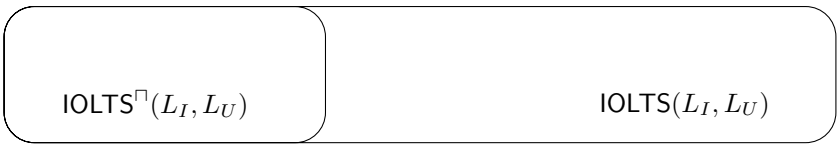
Definition 4 (Quiescence and Outputs) *State* $s \in S$ *is called* quiescent*, denoted by* $\delta(s)$*, iff* $\mathbf{init}(s) \subseteq L_I$*. We say* s *is* weakly quiescent*, denoted by* $\delta_q(s)$*, iff* $\mathbf{Sinit}(s) \subseteq L_I$*. The* outputs *of* s*, denoted* $\mathbf{out}(s)$ *is the set* $\{x \in L_U \mid s \xrightarrow{x}\} \cup \{\delta \mid \delta(s)\}$*; we set* $\mathbf{out}(S') = \bigcup_{s' \in S'} \mathbf{out}(s')$

The notion of weak quiescence is appropriate in the asynchronous setting, where the lags in the communication media interfere with the observation of quiescence: an observer cannot tell whether a system is engaged in some internal transitions or has come to a standstill. By the same token, in an asynchronous setting it becomes impossible to distinguish divergence from quiescence; we re-visit this issue in our proofs of synchronizing asynchronous conformance testing.

We next recall the specialization of IOLTSs, introduced by Weiglhofer and Wotawa [13, 14].

Definition 5 (Internal choice IOLTS) *An IOLTS* $\langle S, L, \rightarrow, s_0 \rangle$ *is an* internal choice input output labeled transition system *(IOLTS*$^{\sqcap}$*), if only quiescent states may accept inputs, i.e., for all* $s \in S$*, if* $\mathbf{init}(s) \cap L_I \neq \emptyset$ *then* $\delta(s)$*.*

We denote the class of IOLTS$^{\sqcap}$ models ranging over L_I and L_U by $\mathsf{IOLTS}^{\sqcap}(L_I, L_U)$. The Venn diagram below (which we extend in the next section) illustrates the relation between IOLTS$^{\sqcap}$ and IOLTS.



Example 1 The LTS depicted in Figure 1.(b) is an IOLTS, but it is *not* in the IOLTS$^{\sqcap}$ subset. Namely, the only input action, i.e., p_rs, is enabled at a state where the internal action τ is also enabled and is hence, not quiescent.

We finish this section with a generalization of the extended transition relation $\Longrightarrow$ to also include observations of quiescence, and we use this to define the notion of *suspension traces*. For a given set of states S of an arbitrary IOLTS with transition relation $\rightarrow \subseteq S \times (L \cup \{\tau\}) \times S$, we define $\Longrightarrow_{\delta} \subseteq S \times (L \cup \{\delta\})^* \times S$, through the following set of deduction rules:

$$\frac{}{s \overset{\epsilon}{\Longrightarrow}_{\delta} s} \qquad \frac{s \overset{\sigma}{\Longrightarrow}_{\delta} s' \qquad \delta(s')}{s \overset{\sigma\delta}{\Longrightarrow}_{\delta} s'} \qquad \frac{s \overset{\sigma}{\Longrightarrow}_{\delta} s'' \qquad s'' \overset{x}{\Longrightarrow} s'}{s \overset{\sigma x}{\Longrightarrow}_{\delta} s'}$$

Henceforth, given an alphabet L, we write L_{δ} to denote the set $L \cup \{\delta\}$.

Definition 6 (Suspension traces and After) *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOLTS. Let $s \in S$ be an arbitrary state, $S' \subseteq S$ and $\sigma \in L_\delta^*$.*

1. *The set of* suspension traces *of s, denoted $\mathsf{Straces}(s)$ is the set $\{\sigma \in L_\delta^* \mid s \stackrel{\sigma}{\Longrightarrow}_\delta\}$; we set $\mathsf{Straces}(S') = \bigcup_{s' \in S'} \mathsf{Straces}(s')$*
2. *The σ-*reachable states *of s, denoted $s\,\mathbf{after}\,\sigma$ is the set $\{s' \in S \mid s \stackrel{\sigma}{\Longrightarrow}_\delta s'\}$; we set $S'\,\mathbf{after}\,\sigma = \bigcup_{s' \in S'} s'\,\mathbf{after}\,\sigma$.*

3 Implementation Relations

Several formal testing theories build on the assumption that the implementations can be modeled by a particular IOLTS; this assumption is part of the so-called *testing hypothesis* underlying the testing theory. Not all theories rely on the same assumptions. We introduce two models, viz., the *input output transition systems*, used in Tretmans' testing theory [10] and the *internal choice input output transition systems*, introduced by Weiglhofer and Wotawa [13, 14].

Tretmans' input-output transition systems, formally defined below, are basically plain IOLTSs with the additional assumption that inputs can always be accepted.

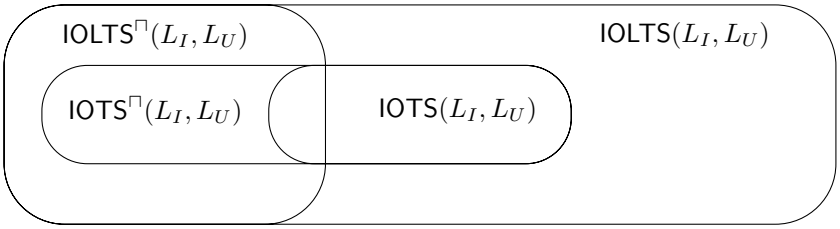
Definition 7 (IOTS) *A state $s \in S$ in an IOLTS $\langle S, L, \rightarrow, s_0 \rangle$ is* input-enabled*, iff $L_I \subseteq \mathbf{Sinit}(s)$. The IOLTS s_0 is an* input output transition system *(IOTS), iff every state $s \in S$ is input-enabled.*

The class of input output transition systems ranging over L_I and L_U is denoted by $\mathsf{IOTS}(L_I, L_U)$.

Weiglhofer and Wotawa's internal choice input output transition systems relax Tretmans' input-enabledness requirement; at the same time, however, they impose an additional restriction on the presence of inputs, which stems from the fact that their class of implementations specialize the IOLTS$^\sqcap$ class.

Definition 8 (Internal choice IOTS) *An IOLTS$^\sqcap$ $\langle S, L, \rightarrow, s_0 \rangle$ is an* internal choice input output transition system *(IOTS$^\sqcap$), iff every quiescent state is input-enabled, i.e., for all $s \in S$, if $\delta(s)$, then $L_I \subseteq \mathbf{Sinit}(s)$.*

We denote the class of IOTS$^\sqcap$ models ranging over L_I and L_U by $\mathsf{IOTS}^\sqcap(L_I, L_U)$. The following Venn-diagram depicts the relation between the IOLTS, IOLTS$^\sqcap$, IOTS and IOTS$^\sqcap$ models.



Example 2 Consider four IOLTSs c_0, e_0, o_0 and i_0 in Figure 2. All of them model a coffee machine which, after receiving money (m), either refunds it (r), or after that the coffee button is pressed (b), produces coffee (c). In IOLTS c_0, after receiving money, there is a choice between input and output; the exact behavior modeled by the transition system is, arguably, awkward, as by pressing a button the refund of the money can be prevented. Although IOLTS e_0 does not feature an immediate race between input and output actions, the possibility of output r can still be ruled out by providing input b. IOLTS o_0 in Figure 2 models a malfunctioning coffee machine which, after pressing the coffee button, may or may not deliver coffee. IOLTS i_0 does not contain this fault and can be considered a reasonable specification of a coffee machine.

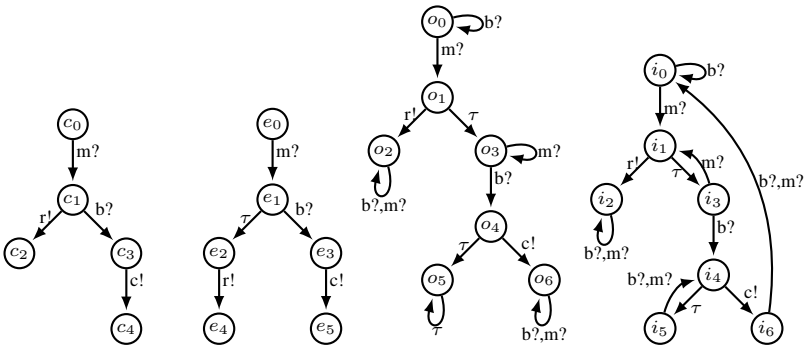


Fig. 2 IOLTSs with different moments of choice

IOLTS c_0 is not input enabled, and neither is e_0: for example after input m, neither of the two allow for input m any more. IOLTS o_0 is not input-enabled either, because for example at state o_5 it refuses to accept any input. The aforementioned IOLTSs can be made IOTSs by adding self-loops for all absent input transitions at each and every state. IOLTS i_0 is input-enabled, however, and is thus an IOTS.

Neither c_0, nor e_0 belong to the class IOLTS$^{\sqcap}$, whereas o_0 and i_0 do. Namely, in the two IOLTSs o_0 and i_0, input actions are only enabled in states where no output or internal action is enabled. Additionally, both o_0 and i_0 belong to the class IOTS$^{\sqcap}$. IOLTS$^{\sqcap}$ i_0 is input-enabled and hence is also an IOTS$^{\sqcap}$. IOLTS$^{\sqcap}$ o_0 is input-enabled in all states but o_4 and o_5 and since these two states are not quiescent, it follows from Definition 8 that o_0 is indeed an IOTS$^{\sqcap}$.

In formal testing, an implementation is said to be correct when its executions are as prescribed by its formal specification. By the testing hypothesis, we can assume that implementations (and their behaviors) can be modeled by a matching IOTS (or IOTS$^{\sqcap}$). This assumption allows one to formalize the notion of *conformance*. Tretmans formalized in [10] a family of conformance relations by parameterizing a single behavioral relation with a set of decorated traces. We generalize this conformance relation by parameterizing it with the behavioral models it assumes as implementations and specifications, leading to a family of conformance relations.

Definition 9 (ioco$_{\mathcal{F}}^{a,b}$) *Let $a, b \in \{\sqcap, \llcorner\}$ and let i_0 be an $IOTS^a$, s_0 an $IOLTS^b$, and $\mathcal{F} \subseteq L_\delta^*$. We say that implementation i_0 is* input-output conforming *to specification s_0, denoted by $i_0 \,\mathbf{ioco}_{\mathcal{F}}^{\mathbf{a,b}}\, s_0$, iff*

$$\forall \sigma \in \mathcal{F} : \mathbf{out}(i_0 \,\mathbf{after}\, \sigma) \subseteq \mathbf{out}(s_0 \,\mathbf{after}\, \sigma)$$

Remark 1 Note that $\llcorner$ depicts the space character (*i.e.*, a blank). That is, for $a = \llcorner$ we have $IOTS^a = IOTS$.

If we assume that our implementations can be modeled as IOTSs, the family of conformance relations $\mathbf{ioco}_{\mathcal{F}}^{\llcorner,\llcorner}$ reduces to the family of conformance relations $\mathbf{ioco}_{\mathcal{F}}$, studied by Tretmans [10]. By assigning $\mathcal{F}$ to $\mathsf{Straces}(s_0)$ for a given specification s_0, the conformance relation **ioco** [10] is obtained.

In the remainder of this section, we investigate several instances of the $\mathbf{ioco}_{\mathcal{F}}^{a,b}$ testing theory. First, we study whether restricting the class of specifications in the $\mathbf{ioco}_{\mathcal{F}}^{a,b}$ relation affects the testing power. Then, we consider how, for fixed specifications, the testing power of $\mathbf{ioco}_{\mathcal{F}}^{a,b}$ is affected by considering different instances for $\mathcal{F}$.

We start by defining what it means for two classes of specifications to have equal testing power.

Definition 10 *Let* $\mathsf{MOD_i}$ *be a class of implementations and let* $\mathsf{MOD_s}$ *be a class of specifications. Let* $\mathsf{MOD'_s}$ *be a subset of the class of specifications* $\mathsf{MOD_s}$*. Then* $\mathsf{MOD_s}$ *and* $\mathsf{MOD'_s}$ *have the same testing power with respect to a given implementation relation* $\boldsymbol{imp} : \mathsf{MOD} \times \mathsf{MOD_i}$*, iff*

$$\forall s \in \mathsf{MOD_s} : \exists s' \in \mathsf{MOD'_s} \; : \forall i \in \mathsf{MOD_i} : i \; \boldsymbol{impl} \; s \text{ iff } i \; \boldsymbol{impl} \; s'$$

Informally, given a class of specifications $\mathsf{MOD_s}$, a subclass $\mathsf{MOD'_s}$ has equivalent testing power when for every specification from $\mathsf{MOD_s}$, we can find an alternative specification from $\mathsf{MOD'_s}$ that identifies exactly the same set of correct and the same set of incorrect implementations. Note that we do not require such an alternative specification to be obtained constructively.

The theorem below states that restricting specifications from IOLTS to IOLTS$^{\sqcap}$ does influence the testing power with respect to implementation relation $\mathbf{ioco}^{\cup,\cup}_{\mathsf{Straces}(s)}$, i.e., **ioco**.

Theorem 1 *The testing power of IOLTS*$^{\sqcap}$ *is not equal to the testing power of IOLTS with respect to implementation relation* $\boldsymbol{ioco}^{\cup,\cup}_{\mathsf{Straces}(s)}$.

Proof Formally, we must show that the following statement does not hold:

$$\begin{array}{l} \forall s \in \mathsf{IOLTS}(L_I, L_U) : \exists s' \in \mathsf{IOLTS}^{\sqcap}(L_I, L_U) : \\ \forall i \in \mathsf{IOTS}(L_I, L_U) : i \, \mathbf{ioco}^{\cup,\cup}_{\mathsf{Straces(s)}} \, s \text{ iff } i \, \mathbf{ioco}^{\cup,\cup}_{\mathsf{Straces(s)}} \, s' \end{array}$$

We will disprove this statement by showing that there is a specification in IOLTS whose testing power cannot be mimicked by any specification in IOLTS$^{\sqcap}$. More specifically, we will show that there is a set of implementations on which the IOLTS specification's verdict will always differ from any candidate alternative IOLTS$^{\sqcap}$ specification.

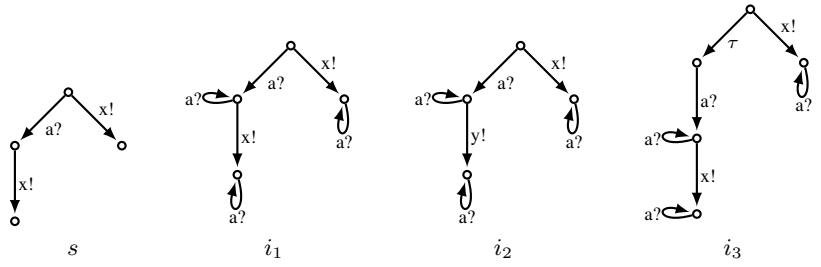


Fig. 3 An input-output labeled transition system specification and three implementations that together show that conformance testing using internal-choice input-output labeled transition system specifications does not have the same testing power as conformance testing using input-output labeled transition systems (Theorem 1).

Consider the specification $s \in \mathsf{IOLTS}(\{a\}, \{x, y\})$, depicted in Figure 3. Observe that $\mathsf{Straces}(s) = \{\epsilon, x\delta^*, a\delta^*, ax\delta^*\}$. Next, consider the three implementations i_1, i_2 and i_3, also depicted in Figure 3. We have:

- $i_1 \; \mathbf{ioco} \; s$, as for all $\sigma \in \mathsf{Straces}(s)$, $\mathbf{out}(i_1 \; \mathbf{after} \; \sigma) \subseteq \mathbf{out}(s \; \mathbf{after} \; \sigma)$.
- $i_2 \; \mathbf{ioco} \; s$, as we have $\mathbf{out}(i_2 \; \mathbf{after} \; a) = \{y\}$, whereas $\mathbf{out}(s \; \mathbf{after} \; a) = \{x\}$.
- $i_3 \; \mathbf{ioco} \; s$, as we have $\mathbf{out}(i_3 \; \mathbf{after} \; \epsilon) = \{x, \delta\}$, whereas $\mathbf{out}(s \; \mathbf{after} \; a) = \{x\}$.

We next show that no IOLTS$^{\sqcap}$ specification leads to the same partitioning on the set of implementations $\{i_1, i_2, i_3\}$, and, therefore, also not on the entire set of implementations $IOTS$. We first show that any IOLTS$^{\sqcap}$ specification s' that satisfies $i_1 \; \mathbf{ioco} \; s'$ must necessarily also satisfy either $i_2 \; \mathbf{ioco} \; s'$ or $i_3 \; \mathbf{ioco} \; s'$. More formally, we show that:

$$\forall s' \in \mathsf{IOLTS}^{\sqcap}(L_I, L_U) : i_1 \; \mathbf{ioco} \; s' \text{ implies } (i_2 \; \mathbf{ioco} \; s') \text{ or } (i_3 \; \mathbf{ioco} \; s') \qquad (*)$$

Let s' be an arbitrary IOLTS$^{\sqcap}$ specification such that $i_1 \,\mathbf{ioco}\, s'$. Now, assume that $i_2 \,\mathbf{ioco}\!\!\!\!/\; s'$. Towards a contradiction, assume that $i_3 \,\mathbf{ioco}\!\!\!\!/\; s'$. We then have $z \in \mathbf{out}(i_3 \,\mathbf{after}\, \sigma)$ and $z \notin \mathbf{out}(s' \,\mathbf{after}\, \sigma)$ for some z and some $\sigma \in \mathsf{Straces}(s')$. Observe that for all $\sigma' \in \mathsf{Straces}(s') \setminus \mathsf{Straces}(i_3)$, we have $\mathbf{out}(i_3 \,\mathbf{after}\, \sigma') = \emptyset \subseteq \mathbf{out}(s' \,\mathbf{after}\, \sigma')$, so, necessarily, $\sigma \in \mathsf{Straces}(s') \cap \mathsf{Straces}(i_3)$. We have

$$\mathsf{Straces}(i_3) = \{\epsilon\} \cup \delta^+ \cup \delta^* a^+ \cup \delta^* a^+ x\{\delta, a\}^* \cup x\{\delta, a\}^*$$

We next analyze each of these possibilities.

- Case $\sigma = \epsilon$. Since $i_1 \,\mathbf{ioco}\, s'$, we have $x \in \mathbf{out}(s' \,\mathbf{after}\, \epsilon)$. As $\mathbf{out}(i_3 \,\mathbf{after}\, \epsilon) = \{\delta, x\}$ and $x \in \mathbf{out}(s' \,\mathbf{after}\, \epsilon)$, we have $\delta \notin \mathbf{out}(s' \,\mathbf{after}\, \epsilon)$. But then $a \notin \mathbf{Sinit}(s')$, since in s', inputs are only allowed in quiescent states. This means that s' cannot distinguish between i_1 and i_2, contradicting $i_1 \,\mathbf{ioco}\, s'$ and $i_2 \,\mathbf{ioco}\!\!\!\!/\; s'$. So $\sigma \neq \epsilon$.
- Case $\sigma \in \delta^+$. Since after observing quiescence, we are necessarily in a quiescent state, we find that $\mathbf{out}(i_3 \,\mathbf{after}\, \sigma) = \{\delta\} = \mathbf{out}(s' \,\mathbf{after}\, \sigma)$. So $\sigma \notin \delta^+$.
- Case $\sigma \in \delta^* a^+$. Observe that since s' is an IOLTS$^{\sqcap}$, we have $\mathbf{out}(s' \,\mathbf{after}\, \rho\, \delta\, a'\, \rho') = \mathbf{out}(s' \,\mathbf{after}\, \rho\, a'\, \rho')$ for all inputs a'. This means that we have $\mathbf{out}(s' \,\mathbf{after}\, \sigma) = \mathbf{out}(s' \,\mathbf{after}\, \sigma')$, where $\sigma' \in a^+$ is obtained from σ by removing all observations of δ. Since $\mathbf{out}(i_3 \,\mathbf{after}\, \sigma) = \{x\}$, we must have $x \notin \mathbf{out}(s' \,\mathbf{after}\, \sigma)$. Since $\mathbf{out}(s' \,\mathbf{after}\, \sigma) = \mathbf{out}(s' \,\mathbf{after}\, \sigma')$, we find that $x \notin \mathbf{out}(s' \,\mathbf{after}\, \sigma')$. But that contradicts $i_1 \,\mathbf{ioco}\, s'$. So $\sigma \notin \delta^*\, a^+$.
- Case $\sigma \in \delta^* a^+ x\{\delta, a\}^*$. We have $\mathbf{out}(i_3 \,\mathbf{after}\, \sigma) = \{\delta\}$, so, necessarily, $\delta \notin \mathbf{out}(s' \,\mathbf{after}\, \sigma)$. Again, since s' is an IOLTS$^{\sqcap}$, we have $\mathbf{out}(s' \,\mathbf{after}\, \sigma) = \mathbf{out}(s' \,\mathbf{after}\, \sigma')$, where $\sigma' \in a^+ x a^+$ is obtained from σ by removing all observations of δ. That means that $\delta \notin \mathbf{out}(s' \,\mathbf{after}\, \sigma')$, which contradicts $i_1 \,\mathbf{ioco}\, s'$. So $\sigma \notin \delta^* a^+ x\{\delta, a\}^*$.
- Case $\sigma \in x\{\delta, a\}^*$. Since $\mathbf{out}(i_3 \,\mathbf{after}\, \sigma) = \{\delta\}$, we must have $\delta \notin \mathbf{out}(s' \,\mathbf{after}\, \sigma)$. Following the same reasoning as in the previous cases, we find that this contradicts $i_1 \,\mathbf{ioco}\, s'$. So $\sigma \notin x\{\delta, a\}^*$.

Since none of the possible traces $\sigma \in \mathsf{Straces}(i_3) \cap \mathsf{Straces}(s')$ can lead to $\mathbf{out}(i_3 \,\mathbf{after}\, \sigma) \not\subseteq \mathbf{out}(s' \,\mathbf{after}\, \sigma)$, we find that $i_3 \,\mathbf{ioco}\, s'$.

Summarizing, this means that there is no IOLTS$^{\sqcap}$ specification s' that has the same testing power as the IOLTS specification s, proving Theorem 1.

In the remainder of this section, we investigate the effect of varying the set of observations $\mathcal{F}$ on the testing power of the resulting conformance relations. Note that the question here is orthogonal to the one that we asked above: here we fix the specifications and ask whether by considering a subset of the set of observations $\mathcal{F}$, we obtain conformance relations that retain the testing power of the full set of observations $\mathcal{F}$. The proposition below states that the testing power of $\mathbf{ioco}_{\mathcal{F}}^{a,b}$ is monotonic in the set of observations $\mathcal{F}$; from this, it follows that testing power may be affected by considering different sets $\mathcal{F}$.

Proposition 1 *Let $\mathcal{F}, \mathcal{F}' \subseteq L_\delta^*$. Then $\mathcal{F}' \subseteq \mathcal{F}$ implies $\boldsymbol{ioco}_{\mathcal{F}'}^{a,b} \subseteq \boldsymbol{ioco}_{\mathcal{F}}^{a,b}$.*

We are, in particular, interested in suspension traces that naturally capture the observations that we can make of an IOTS$^{\sqcap}$ implementation. The crucial difference between IOTS$^{\sqcap}$ implementations and IOTS implementations is that the latter are always willing to accept inputs, whereas the former only accepts inputs when we can also observe quiescence. Providing inputs in any other situation is undesirable, and, hence, reasoning about traces that would attempt to do so in our conformance relation would be equally undesirable. We therefore introduce a new class of traces, called *internal choice traces*, which naturally characterize the observable behaviors of IOTS$^{\sqcap}$ implementations.

Definition 11 (Internal choice traces) *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOLTS. Let $s \in S$ be an arbitrary state and $\sigma \in L_\delta^*$. The set of* internal choice traces *of s, denoted $\mathsf{ICtraces}(s)$ is a*

subset of suspension traces in which quiescence is observed before every input action, i.e. $\mathsf{ICtraces}(s) = \mathsf{Straces}(s) \cap (L_U \cup (\{\delta\}^+ L_I) \cup \{\delta\})^*$*; we set* $\mathsf{ICtraces}(S') = \bigcup_{s' \in S'} \mathsf{ICtraces}(s')$ *for* $S' \subseteq S$.

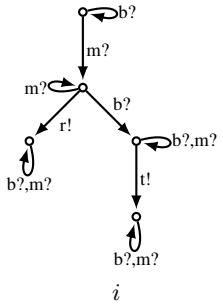


Fig. 4 An implementation illustrating that the testing power of internal choice traces is strictly less than the testing power of suspension traces in the family of conformance relations $\mathbf{ioco}_{\mathcal{F}}$.

Note that, as a result of Proposition 1, using internal choice traces instead of suspension traces leads to a weaker testing relation. It is not, however, immediate that the inclusion of Proposition 1 is strict. The following example shows that the inclusion is indeed strict in the standard **ioco** testing theory.

Example 3 Let c_0 be the specification depicted in Figure 2 and let i in Figure 4 be its implementation. Following Definition 9, $i \,\mathbf{ioco}\, c_0$ because the observed output t in the implementation after execution of trace mb is not allowed by specification c_0 after that trace. The set $\mathsf{ICtraces}(s) = \{\epsilon, \delta\sigma m, \delta\sigma m r \mid \sigma \in \delta^*\}$. Clearly, for all $\sigma \in \mathsf{ICtraces}(c_0)$, we have $\mathbf{out}(i \,\mathbf{after}\, \sigma) \subseteq \mathbf{out}(c_0 \,\mathbf{after}\, \sigma)$. Hence, $i \,\mathbf{ioco}_{\mathsf{ICtraces}(c_0)}\, c_0$.

We next consider restricting the set of observations $\mathcal{F}$ to internal choice traces in the conformance family $\mathbf{ioco}_{\mathcal{F}}^{\cdot,\sqcap}$ and compare the resulting testing power to the one obtained using suspension traces. As illustrated by example below, restricting the set of specifications to internal choice labeled transition systems is not a sufficient condition to retain the testing power of the full set of suspension traces.

Example 4 Consider again Figure 2. Take IOLTS$^{\sqcap}$ o_0 as specification and again consider i in Figure 4 as its implementation. Clearly, we have $i \,\mathbf{ioco}\, o_0$. For instance, considering trace mb, we find that $\mathbf{out}(i \,\mathbf{after}\, mb) = \{t\}$, whereas $\mathbf{out}(o_0 \,\mathbf{after}\, mb) = \{c\}$. In conformance testing with respect to $\mathbf{ioco}_{\mathsf{ICtraces}(\mathbf{o_0})}$, trace $\delta m \delta b$ is examined instead of trace mb. We find that $\mathbf{out}(i \,\mathbf{after}\, \delta m \delta b) = \emptyset \subseteq \mathbf{out}(o_0 \,\mathbf{after}\, \delta m \delta b)$. It is obtained by checking all other traces in $\mathsf{ICtraces}(o_0)$ that $i \,\mathbf{ioco}^{\cdot,\sqcap}_{\mathsf{ICtraces}(\mathbf{o_0})}\, o_0$.

We next investigate whether switching to a different model of implementations will change these results: we henceforth assume that implementations can be modeled using IOTS$^{\sqcap}$s. The example below shows that, assuming that specifications can still be arbitrary $IOLTSs$, the testing power of using internal choice traces is inferior to using suspension traces.

Example 5 Consider IOLTS s in Figure 5. Analogous to the IOLTSs in Figure 2, it models a coffee machine which after receiving money, either refunds or accepts it; if accepted, coffee is produced after pressing a coffee button (in this case, c_b), and, similarly, tea is produced after pressing a tea button (t_b). The transition system i is input-enabled only at quiescent states, *i.e.*, it is an IOTS$^{\sqcap}$. Take IOTS$^{\sqcap}$ i, also in Figure 5, as a potential implementation. Regarding Definition 9, we find that $i \,\mathbf{ioco}^{\sqcap}\, s$, because specification s after executing trace mc_b allows only output c, whereas i after the same trace produces t. The set

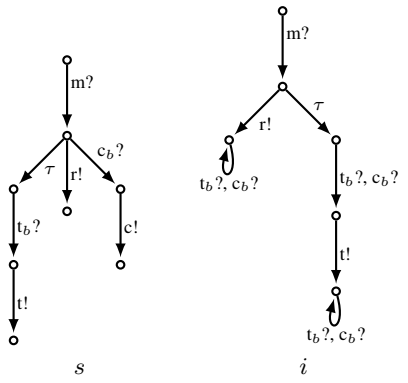


Fig. 5 A specification and an implementation illustrating that the testing power of internal choice traces is strictly less than the testing power of suspension traces in the family of conformance relations $\mathbf{ioco}^{\sqcap,\smile}_{\mathcal{F}}$.

$\mathsf{ICtraces}(s) = \{\epsilon, \sigma\delta m, \sigma\delta m r\sigma, \sigma\delta m c_b c\sigma, \sigma\delta m\sigma\delta t_b, \sigma\delta m\sigma\delta t_b t\sigma \mid \sigma \in \delta^*\}$. Obviously, we have $\mathbf{out}(i\,\mathbf{after}\,\sigma) \subseteq \mathbf{out}(s\,\mathbf{after}\,\sigma)$. Hence, $i\,\mathbf{ioco}^{\sqcap,\smile}_{\mathsf{ICtraces(s)}}\,s$.

Finally, we investigate the case that specifications are assumed to be internal choice IOLTSs. The result below shows that, contrary to the previous cases we analyzed, the resulting conformance relations for internal choice traces and suspension traces coincide.

Theorem 2 *Let $s \in \mathsf{IOLTS}^{\sqcap}(L_I, L_U)$ be a specification and $i \in \mathsf{IOTS}^{\sqcap}(L_I, L_U)$ be an implementation. Then $i\,\mathbf{ioco}^{\sqcap,\sqcap}_{\mathsf{ICtraces(s)}}\,s$ iff $i\,\mathbf{ioco}^{\sqcap,\sqcap}_{\mathsf{Straces(s)}}\,s$.*

Proof The implication from right to left is an instance of Proposition 1. We therefore focus on the implication from left to right.

We first show that for every $\sigma \in \mathsf{Straces}(s)$, there is some $\sigma' \in \mathsf{ICtraces}(s)$ such that both $s\,\mathbf{after}\,\sigma = s\,\mathbf{after}\,\sigma'$ and $i\,\mathbf{after}\,\sigma = i\,\mathbf{after}\,\sigma'$. We do this by induction on the number of input actions in σ.

- *Base case.* For the induction basis assume that $\sigma \in (L_U \cup \{\delta\})^*$. Following Definition 11, $\sigma \in \mathsf{ICtraces}(s)$. Hence, $\sigma' = \sigma$ satisfies the required condition.
- *Induction step.* Assume for the induction step that the given claim holds for all sequences with $n-1$ input actions. Suppose that we have a sequence σ with n input actions; that is, $\sigma = \sigma_1 a \sigma_2$ with $\sigma_1 \in L_\delta^*$, $\sigma_2 \in (L_U \cup \{\delta\})^*$ and $a \in L_I$. Thus, σ_1 has $n-1$ input actions. Following the induction hypothesis, there exists a $\sigma_1' \in \mathsf{ICtraces}(s)$ such that $s\,\mathbf{after}\,\sigma_1 = s\,\mathbf{after}\,\sigma_1'$ and $i\,\mathbf{after}\,\sigma_1 = i\,\mathbf{after}\,\sigma_1'$ hold. We conclude from $s \in \mathsf{IOLTS}^{\sqcap}(L_I, L_U)$ along with $\sigma_1 a \in \mathsf{Straces}(s)$ that there exists a non-empty subset of states in $s\,\mathbf{after}\,\sigma_1$ consisting of quiescent states. Suppose S' is the largest possible set of quiescent states in $s\,\mathbf{after}\,\sigma_1$. We know from Definition 5 that $s\,\mathbf{after}\,\sigma_1 a\sigma_2 = S'\,\mathbf{after}\,a\sigma_2$. Consequently, by substituting S' with $s\,\mathbf{after}\,\sigma_1'\delta$ we have $s\,\mathbf{after}\,\sigma = s\,\mathbf{after}\,\sigma_1'\delta a\sigma_2$.
  It follows from Definition 11 that $\sigma_1'\delta a\sigma_2 \in \mathsf{ICtraces}(s)$. Therefore, $s\,\mathbf{after}\,\sigma = s\,\mathbf{after}\,\sigma_1'\delta a\sigma_2$ holds. Along the same lines of reasoning, we can show that for the same internal choice trace we have $i\,\mathbf{after}\,\sigma = i\,\mathbf{after}\,\sigma_1'\delta a\sigma_2$.

We next prove the property by contraposition. Suppose that $i\,\not\mathbf{ioco}^{\sqcap,\sqcap}_{\mathsf{Straces}(s)}\,s$. Then for some $\sigma \in \mathsf{Straces}(s)$, $\mathbf{out}(i\,\mathbf{after}\,\sigma) \not\subseteq \mathbf{out}(s\,\mathbf{after}\,\sigma)$. By the above result, we find that there must be some $\sigma' \in \mathsf{ICtraces}(s)$, such that $i\,\mathbf{after}\,\sigma = i\,\mathbf{after}\,\sigma'$ and $s\,\mathbf{after}\,\sigma = s\,\mathbf{after}\,\sigma'$. But then also $\mathbf{out}(i\,\mathbf{after}\,\sigma') \not\subseteq \mathbf{out}(s\,\mathbf{after}\,\sigma')$. So, it also must also hold that $i\,\not\mathbf{ioco}^{\sqcap,\sqcap}_{\mathsf{ICtraces}(s)}\,s$.

As an immediate consequence of Theorem 2, for implementations in the intersection of $\mathsf{IOTS}^{\sqcap}$ and IOTS, the testing power of $\mathbf{ioco}^{\sqcap,\sqcap}_{\mathsf{ICtraces(s)}}$ and that of the standard **ioco** coincide, as stated by the proposition below.

Proposition 2 *Let* $s \in \mathsf{IOLTS}^{\sqcap}(L_I, L_U)$ *be a specification and* $i \in \mathsf{IOTS}^{\sqcap}(L_I, L_U) \cap \mathsf{IOTS}(L_I, L_U)$ *be an implementation. Then* $i \,\mathbf{ioco}^{\sqcap,\sqcap}_{\mathsf{ICtraces}(\mathbf{s})}\, s$ *iff* $i \,\mathbf{ioco}\, s$.

4 Test Case Generation

The definition of the family of conformance relations introduced and studied in the previous section assumes that we can reason about implementations as if these were transition systems we can inspect. Since this is in practice not the case (we only know that a model exists that underlies such an implementation), the definition cannot be used to check whether an implementation conforms to a given specification.

This problem can be sidestepped if there is a set of test cases that can be run against an actual implementation, and which has exactly the same discriminating power as the specification. In this section, we study the test cases that are needed to test for the family of conformance relations introduced in the previous section.

A test case can, in the most general case, be described by a tree-shaped IOLTS. Such a test case prescribes when to stimulate an implementation-under-test by sending an input, and when to observe outputs emitted by the implementation-under-test. In general, the inputs to a test case are the outputs of the implementation-under-test, whereas the outputs of a test case are the inputs of the implementation-under-test. In order to formally distinguish between observing quiescence and "being" quiescent, we introduce a special action label θ, which stands for the former. Since we sometimes reason about the behaviors σ of an implementation from the viewpoint of a tester, we interpret δ labels as θ labels; formally, we then write $\overline{\sigma}$ to denote the sequence σ in which all δ labels have been replaced by θ labels.

Definition 12 (Test case) *A test case is an IOLTS* $\langle S, L, \rightarrow, s_0 \rangle$*, in which:*

1. S *is a finite set of states reachable from* s_0*,*
2. *terminal nodes of* S *are called* $\mathbf{pass}$ *or* $\mathbf{fail}$*,*
3. *the quiescence observation* θ *belongs to* L_I*,*
4. *the transition relation* $\rightarrow$ *is acyclic, self-loop free and deterministic.*
5. $\mathbf{pass}$ *and* $\mathbf{fail}$ *states appear only as targets of transitions labeled by an element of* L_I*, and*
6. *for all non-terminal states* s*, either* $\mathbf{init}(s) = L_I \cup \{\theta\}$ *or* $\mathbf{init}(s) = L_I \cup \{x\}$ *for some* $x \in L_U$.

We denote the class of test cases ranging over inputs L_I and outputs L_U by $\mathsf{TTS}(L_U, L_I)$. Note that due to the determinism of a test case, none of the transitions of a test case are labeled with the silent action τ.

In [14, 13] a subclass of $\mathsf{TTS}(L_U, L_I)$ is introduced; test cases in this subclass are called *internal choice* test cases. Such test cases stimulate an implementation-under-test only when quiescence has been observed. Intuitively, this will ensure that the test case is actually executable for implementations that behave as internal choice transition systems.

Definition 13 (Internal choice test case) *A test case* $\langle S, L, \rightarrow, s_0 \rangle$ *is an* internal choice test case*, denoted TTS*$^{\sqcap}$*, if for all* $s \in S$*,* $x \in L_U$ *and* $\sigma \in L^*$*, if* $\sigma x \in \mathsf{traces}(s)$ *then* $\sigma = \sigma' \theta$.

We denote the class of internal choice test cases ranging over inputs L_I and outputs L_U by $\mathsf{TTS}^{\sqcap}(L_U, L_I)$.

The property below provides us with an alternative characterization of an internal choice test case.

Property 1 Let t be a test case. t is an internal choice test case iff $\mathsf{traces}(t) \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$.

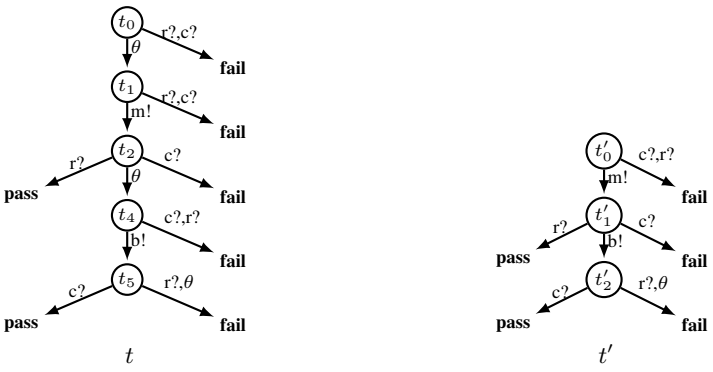


Fig. 6 Two test cases for IOTS$^{\sqcap}$ o_0 in Figure 2

Example 6 IOLTSs t and t' in Figure 6 show two test cases for IOTS$^{\sqcap}$ o_0 in Figure 2. IOLTS t' is an internal choice test case. In this test case, inputs for the implementation are enabled only in states reached by a θ-transition.

We next formalize what it means to execute a test case on an implementation-under-test. The intuition is that whenever a test case stimulates the implementation-under-test by sending an input, the latter consumes the input and responds by moving to a (possibly new) next state. In the same vein, whenever the implementation issues an output, the output is consumed by the test case, upon which the test case moves to a next state. Observe that the communication between the test case and the implementation-under-test can be instantaneous (*i.e.*, *synchronous*), or through some underlying infrastructure that may introduce delays in the communication (*i.e.*, communication is *asynchronous*). The latter form of communication is addressed in the next sections. In the remainder of this section, we assume that communication between implementations and test cases is *synchronous*.

Definition 14 (Synchronous execution) *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOLTS, and let $\langle T, L', \rightarrow, t_0\rangle$ be a test case, such that $L_I = L'_U$ and $L_U = L'_I \setminus \{\theta\}$. Let $s, s' \in S$ and $t, t' \in T$. Then the synchronous execution of the test case and s_0 is defined through the following inference rules:*

$$\frac{s \xrightarrow{\tau} s'}{t \rceil\!\rceil s \xrightarrow{\tau} t \rceil\!\rceil s'}\ (R1) \qquad \frac{t \xrightarrow{x} t' \qquad s \xrightarrow{x} s'}{t \rceil\!\rceil s \xrightarrow{x} t' \rceil\!\rceil s'}\ (R2) \qquad \frac{t \xrightarrow{\theta} t' \qquad \delta(s)}{t \rceil\!\rceil s \xrightarrow{\theta} t' \rceil\!\rceil s}\ (R3)$$

The terminal state(s) **pass** or **fail** of a test case can be used to formalize what it means for an implementation to *pass* or *fail* a test case.

Definition 15 (Verdict) *Let $T \subseteq \mathsf{TTS}(L_I, L_U)$ be a set of test cases for some IOLTS implementation $\langle S, L', \rightarrow, s_0\rangle$ and let $t_0 \in T$ be a test case. We say that state $s \in S$* passes *the test case t_0, denoted s $\mathbf{passes}$ t_0 iff there is no $\sigma \in L^*$ and no state $s' \in S$, such that $t_0 \rceil\!\rceil s_0 \stackrel{\sigma}{\Longrightarrow} \mathbf{fail} \rceil\!\rceil s'$. We also say that state $s \in S$* passes *the set of test cases T, denoted s $\mathbf{passes}$ T iff s* passes *all test cases in T.*

We next introduce a test case generation algorithm, based on Tretmans' original algorithm [9], that is suited for testing against a conformance relation $\mathbf{ioco}^{a,b}_{\mathcal{F}}$. The set of test cases generated by this algorithm is both *sound* and *exhaustive*. Soundness basically means that, for a given specification, executing the test case on an implementation-under-test will not lead to a test failure if the implementation conforms to the specification. Exhaustiveness boils down to the ability of the algorithm to generate a test case that has the potential to detect a non-conforming implementation.

Definition 16 (Soundness and exhaustiveness) *Let $T \subseteq \mathsf{TTS}(L_I, L_U)$ be a set of test cases for IOLTS specification s_0. Then for an implementation relation* $\mathbf{imp}$*, we say that*

$$\begin{array}{rl} T \textit{ is sound} & =_{def} \forall i : i \;\mathbf{imp}\; s_0 \;\; \textit{implies} \;\; i \;\mathbf{passes}\; T \\ T \textit{ is exhaustive} & =_{def} \forall i : i \;\mathbf{imp}\; s_0 \quad\;\; \textit{if} \quad\;\; i \;\mathbf{passes}\; T \end{array}$$

Note that Tretmans' original test case generation algorithm did not produce test cases that were input-enabled. However, this issue was addressed fairly recently in [10], in which the algorithm for (plain) **ioco** was made to generate test cases that, in all non-terminal states, are willing to accept all the outputs produced by an implementation. We have used the ideas of the latter algorithm and incorporated them in Tretmans' original algorithm.
In order to concisely describe the algorithm, we borrow Tretmans' notation (see for instance [10]) for behavioral expressions using the operators $;, \Box$ and Σ. Such behavioral expressions represent transition systems. Informally, for an action label a (taken from some set of actions), and a behavioral expression B, the behavioral expression $a;B$ denotes the transition system that starts with executing the a action, leading to a state that behaves as B. For a countable set of behavioral expressions $\mathcal{B}$, the *choice expression* $\Sigma\mathcal{B}$ denotes the transition system that, from its initial state, can nondeterministically choose between all behaviors described by the expressions in $\mathcal{B}$. The expression $B_1 \Box B_2$, for behavioral expressions B_1 and B_2, is used as an abbreviation for $\Sigma\{B_1, B_2\}$, i.e., it behaves either as B_1 or B_2.

Algorithm 1 *Let IOLTS $\langle S, L, \rightarrow, s_0\rangle$ be a specification, let $S' \subseteq S$, and let $F \subseteq \mathsf{Straces}(S')$; then a test case $t \in \mathsf{TTS}(L_U, L_I \cup \{\theta\})$ is obtained by a finite number of recursive application of one of the following nondeterministic choices:*

$t :=$ $\mathbf{pass}$
$t :=$ $\Sigma\{\bar{x}; \mathbf{fail} \mid x \in L_U, x \notin \mathbf{out}(S'), \epsilon \in F\}$
$\Box$ $\Sigma\{\bar{x}; \mathbf{pass} \mid x \in L_U, x \notin \mathbf{out}(S'), \epsilon \notin F\}$
$\Box$ $\Sigma\{\bar{x}; t_x \mid x \in L_U, x \in \mathbf{out}(S')\}$,
where t_x is obtained by recursively applying the algorithm for $\{\sigma \in L_\delta^* \mid x\sigma \in F\}$ and S' $\mathbf{after}\, x$
$\Box$ $a; t_a$,
where $a \in L_I$, such that $F' = \{\sigma \in L_\delta^* \mid a\sigma \in F\} \neq \emptyset$ and t_a is obtained by recursively applying the algorithm for F' and S' $\mathbf{after}\, a$
$t :=$ $\Sigma\{\bar{x}; \mathbf{fail} \mid x \in L_U \cup \{\delta\}, x \notin \mathbf{out}(S'), \epsilon \in F\}$
$\Box$ $\Sigma\{\bar{x}; \mathbf{pass} \mid x \in L_U \cup \{\delta\}, x \notin \mathbf{out}(S'), \epsilon \notin F\}$
$\Box$ $\Sigma\{\bar{x}; t_x \mid x \in L_U \cup \{\delta\}, x \in \mathbf{out}(S')\}$,
where t_x is obtained by recursively applying the algorithm for $\{\sigma \in L_\delta^* \mid x\sigma \in F\}$ and S' $\mathbf{after}\, x$

Upon termination, algorithm 1 generates a test case for a set of states S' and a subset of its suspension traces F of a given specification $s_0 \in \mathsf{IOLTS}(L_I, L_u)$. The parameters S' and F are typically initialized as s_0 $\mathbf{after}\, \epsilon$ and $\mathsf{Straces}(s_0$ $\mathbf{after}\, \epsilon)$, respectively.

The proposition below establishes a formal connection between a subset of the suspension traces of a given specification, and the traces of the test cases generated with Algorithm 1 for that specification. The proposition is essential in establishing the exhaustiveness of the test case generation algorithm.

Proposition 3 *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOLTS. Let $F \subseteq \mathsf{Straces}(S')$ with $S' \subseteq S$, let $\sigma \in F$. Define $t_{[\sigma,F,S']}$ by:*

$$
\begin{array}{lll}
t_{[\epsilon,F,S']} & =_{def} & \Sigma\{\bar{x}; \mathbf{fail} \mid x \in L_U \cup \{\delta\}, x \notin \mathbf{out}(S')\} \\
 & & \Box\ \Sigma\{\bar{x}; \mathbf{pass} \mid x \in L_U \cup \{\delta\}, x \in \mathbf{out}(S')\} \\
 & & \\
t_{[\bar{a}\sigma,F,S']}\ (\bar{a} \in L_I) & =_{def} & \Sigma\{\bar{x}; \mathbf{fail} \mid x \in L_U, x \notin \mathbf{out}(S'), \epsilon \in F\} \\
 & & \Box\ \Sigma\{\bar{x}; \mathbf{pass} \mid x \in L_U, x \notin \mathbf{out}(S'), \epsilon \notin F\} \\
 & & \Box\ \Sigma\{\bar{x}; \mathbf{pass} \mid x \in L_U, x \in \mathbf{out}(S')\} \\
 & & \Box\ \bar{a}; t_{[\sigma,F',S'']} \\
 & & (\textit{where } F' = \{\sigma' \in L_\delta^* \mid a\sigma' \in F\} \textit{ and } S'' = S' \mathbf{after}\, a) \\
 & & \\
t_{[\bar{y}\sigma,F,S']}\ (\bar{y} \in L_U \cup \{\delta\}) & =_{def} & \Sigma\{\bar{x}; \mathbf{fail} \mid x \in L_U \cup \{\delta\}, x \notin \mathbf{out}(S'), \epsilon \in F\} \\
 & & \Box\ \Sigma\{\bar{x}; \mathbf{pass} \mid x \in L_U \cup \{\delta\}, x \notin \mathbf{out}(S'), \epsilon \notin F\} \\
 & & \Box\ \Sigma\{\bar{x}; \mathbf{pass} \mid x \in L_U \cup \{\delta\}, x \in \mathbf{out}(S'), x \neq y\} \\
 & & \Box\ \bar{y}; t_{[\sigma,F',S'']} \\
 & & (\textit{where } F' = \{\sigma' \in L_\delta^* \mid y\sigma' \in F\} \textit{ and } S'' = S' \mathbf{after}\, y)
\end{array}
$$

then

1. $t_{[\sigma,F,S']}$ *can be obtained from* F *and* S' *with Algorithm 1*
2. $x \notin \mathbf{out}(S' \mathbf{after} \sigma)$ *implies* $t_{[\sigma,F,S']} \stackrel{\overline{\sigma x}}{\Longrightarrow} \mathbf{fail}$

Proof The proof is identical to the proof of Lemma $A.25$ in [10].

Theorem 3 *Let IOLTS* $\langle S, L, \rightarrow, s_0 \rangle$ *be a specification. Then*

1. *a test case obtained with Algorithm 1 from* $s_0 \mathbf{after} \epsilon$ *and* $\mathcal{F} \subseteq \mathsf{Straces}(s_0)$ *is sound for* s_0 *with respect to* $\mathbf{ioco}_{\mathcal{F}}^{a,b}$ *for* $a, b \in \{\sqcap, \llcorner\}$.
2. *the set of all possible test cases that can be obtained from Algorithm 1 from* $s_0 \mathbf{after} \epsilon$ *and* $\mathcal{F} \subseteq \mathsf{Straces}(s_0)$ *is exhaustive for* s_0 *with respect to* $\mathbf{ioco}_{\mathcal{F}}^{a,b}$ *for* $a, b \in \{\sqcap, \llcorner\}$.

Proof The proof is similar to the proof of Theorem 6.3 in [10]; the exhaustiveness of the algorithm follows from Proposition 3.

Observe that the above theorem does not imply that the test cases derived by Algorithm 1 can be executed successfully on both classes of implementations that we discussed in the previous sections. Whereas for Tretmans' implementations behaving as IOTSs, successful test case execution is no issue, this is not the case for Weiglhofer and Wotawa's implementations behaving as IOTS$^{\sqcap}$s. For the latter class of implementations it is possible that the test case is forced to observe outputs, since the implementation is unwilling to accept stimuli from the test case. It thus makes no sense to consider such test cases, as the example below illustrates.

Example 7 Consider again Figure 6. Take IOLTS t' as the test case generated with Algorithm 1 from IOTS o_0 and sequence $m\ b$ and take IOTS d_0, depicted in Figure 7 as a potential implementation. Consider the execution $t'_0 \rceil d_0 \stackrel{m}{\longrightarrow} t'_1 \rceil d_1$. At state t'_1, test case t' can try to provide the input b to the implementation-under-test while IOTS d_0 is not willing to accept any inputs. Therefore, the test case is prevented from executing the sequence $m\ b$.

To cope with the issue of successful executability of test cases, we next investigate when our test case generation algorithm can be made to produce only executable test cases, while still guaranteeing soundness and exhaustiveness. Our studies of the $\mathbf{ioco}_{\mathcal{F}}^{a,b}$ family of conformance relations in the previous section are essential in establishing the latter results.

First, we have the following technical lemma and proposition which state that traces of a test case can be chopped up into individual traces.

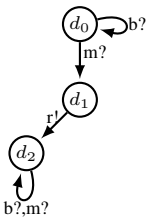


Fig. 7 An internal choice implementation of a malfunctioning coffee machine

Lemma 1 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOLTS. Let $S' \subseteq S$ be a set of states and $F \subseteq \mathsf{Straces}(S')$. Then for all $y\sigma \in F$ we have:*

$$\begin{array}{l} \mathsf{traces}(t_{[y\sigma,F,S']}) \\ = \{\epsilon\} \cup L_U \cup \{\theta \mid y \notin L_I\} \cup (\{y\} \cap L_I) \\ \cup\ \{\overline{y}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma,\{\rho \mid y\rho \in F\},S' \,\mathbf{after}\, y]})\} \end{array}$$

Proof Follows immediately from the definition of $\mathsf{traces}(t_{\sigma,F,S'})$.

Proposition 4 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOLTS. Let $S' \subseteq S$ be a set of states and $F \subseteq \mathsf{Straces}(S')$. Then for all $\sigma_1\sigma_2 \in F$ satisfying $\sigma_1 \neq \epsilon$, we have:*

$$\begin{array}{l} \mathsf{traces}(t_{[\sigma_1\sigma_2,F,S']}) \\ = \mathsf{traces}(t_{[\sigma_1,F,S']}) \cup \{\overline{\sigma_1}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma_2,\{\rho \mid \sigma_1\rho \in F\},S' \,\mathbf{after}\, \sigma_1]})\} \end{array}$$

Proof The proof proceeds by induction on the length of σ_1.

- *Base case.* Follows immediately from Lemma 1.
- *Induction step.* Assume for the induction step that the above statement holds for all sequences of length $n-1$ and the length of σ_1 is n. Suppose $\sigma_1 = x\sigma_1'$ with $\sigma_1' \in L_\delta^*$ and $x \in L_\delta$. Therefore, the length of σ_1' is $n-1$.
  From our base case we know that $\mathsf{traces}(t_{[x\sigma_1'\sigma_2,F,S']}) = \mathsf{traces}(t_{[x,F,S']}) \cup \{\overline{x}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma_1'\sigma_2,F_0,S_0]})\}$ where $S_0 = S' \,\mathbf{after}\, x$ and $F_0 = \{\sigma' \mid x\sigma' \in F\}$. Clearly, $\sigma_1'\sigma_2 \in F_0$. Following our induction hypothesis, $\mathsf{traces}(t_{[\sigma_1'\sigma_2,F_0,S_0]}) = \mathsf{traces}(t_{[\sigma_1',F_0,S_0]}) \cup \{\overline{\sigma_1'}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma_2,F_1,S_1]})\}$ where $S_1 = S_0 \,\mathbf{after}\, \sigma_1'$, $F_1 = \{\rho \mid \sigma_1'\rho \in F_0\}$ and $\sigma_2 \in F_1$. Combining these two observations results in:

$$\begin{array}{l} \mathsf{traces}(t_{[x\sigma_1'\sigma_2,F,S']}) \\ = \mathsf{traces}(t_{[x,F,S']}) \cup \{\overline{x}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma_1',F_0,S_0]})\} \cup \{\overline{\sigma_1'}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma_2,F_1,S_1]})\} \end{array} \quad (*)$$

  From our base case, we know that:

$$\mathsf{traces}(t_{[x\sigma_1',F,S']}) = \mathsf{traces}(t_{[x,F,S']}) \cup \{\overline{x}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma_1',F_0,S_0]})\} \quad (**)$$

  Together, $*$ and $**$ yield the desired equivalence.

The proposition given below formalizes that, indeed, the interaction between an internal choice test case and an IOLTS proceeds in an orchestrated fashion: the IOLTS is only provided a stimulus whenever it has reached a stable situation, and is thus capable of consuming the stimulus.

Proposition 5 *Let s_0 be an arbitrary IOLTS and t_0 be an internal choice test case. Let $x \in L_I$. Then for all $\sigma \in L_\delta^*$, we have:*

$$t \| s \stackrel{\overline{\sigma x}}{\Longrightarrow} \textit{ implies } \exists \sigma' \in L^* : \overline{\sigma} = \overline{\sigma'}\theta$$

Due to the above results, we can thus guarantee that test cases are successfully executable on implementations that behave as IOTS$^{\sqcap}$s. It thus suffices to investigate whether the test case generation algorithm can be made to generate internal choice test cases only. The proposition below confirms that this is indeed possible. This proposition relies on Property 1.

Proposition 6 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOLTS. Then for all $S' \subseteq S$, all $F \subseteq \mathsf{ICtraces}(S')$ and all $\sigma \in F$, the test case $t_{[\sigma,F,S']}$ is an internal choice test case.*

Proof Because of Property 1, it suffices to show that $\mathsf{traces}(t_{[\sigma,F,S']}) \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$. We prove it by induction on the number of input actions in σ.

- *Base case.* Assume for the basis of the induction that $\sigma \in (L_U \cup \{\delta\})^*$. We proceed by a second induction on the length of σ.
  - *Base case.* Suppose $\sigma = \epsilon$ for the basis of the second induction. From Proposition 3, we can deduce that $\mathsf{traces}(t_{[\epsilon,F,S']}) = \{\epsilon\} \cup L_U \cup \{\theta\}$: $t_{[\epsilon,F,S']}$ has an x-labeled transition to the **pass** state for $x \in \mathbf{out}(S')$, and to the **fail** state for $x \notin \mathbf{out}(S')$. Clearly, $L_U \cup \{\theta\} \in (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$. Hence, $t_{[\epsilon,F,S']}$ is an internal choice test case.
  - *Induction step.* Assume for the induction step of the second induction that the above statement holds for all sequences of length $n-1$ and that the length of σ is n. Take $\sigma = y\sigma'$ with $\sigma \in (L_U \cup \{\delta\})^+$. Following Proposition 4, $\mathsf{traces}(t_{[y\sigma',F,S']}) = \{\epsilon\} \cup (L_U \cup \{\theta\}) \cup \{\bar{y}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma',F',S'']})\}$ with $F' = \{\rho \mid y\rho \in F\}$ and $S'' = S' \mathbf{after} y$. We know from our induction hypothesis that $\mathsf{traces}(t_{[\sigma',F',S'']}) \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$. Consequently, we find that $\{\bar{y}\rho \mid \rho \in \mathsf{traces}(t_{[x\sigma',F',S'']})\} \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$. Combined with our previous observations, we find that $\mathsf{traces}(t_{[\sigma,F,S']}) \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$.
- *Induction step.* Assume for the induction step that our statement holds for all sequences with $n-1$ input actions. Let $\sigma \in F$ be a sequence containing n input actions, where $F \subseteq \mathsf{ICtraces}(S')$; assume $\sigma = \sigma_1\ \delta\ a\ \sigma_2$, where $\sigma_1 \in L_\delta^*$, $a \in L_I$ and $\sigma_2 \in L_U^*$. From Proposition 4, we find that $\mathsf{traces}(t_{[\sigma,F,S']}) = \mathsf{traces}(t_{[\sigma_1,F,S']}) \cup \{\overline{\sigma_1}\rho \mid \rho \in \mathsf{traces}(t_{[\delta a\sigma_2,F',S'']})\}$ where $F' = \{\rho \mid \sigma_1\rho \in F\}$ and $S'' = S' \mathbf{after} \sigma_1$.
  From our induction hypothesis, we know that $\mathsf{traces}(t_{[\sigma_1,F,S']}) \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$. Therefore, it suffices to show that $\{\overline{\sigma_1}\rho \mid \rho \in \mathsf{traces}(t_{[\delta a\sigma_2,F',S'']})\} \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$.
  We know from $\sigma_1 \in \mathsf{ICtraces}(S')$ that $\overline{\sigma_1} \subseteq (L_U \cup ((\theta)^+ L_I) \cup \{\theta\})^*$. Therefore, $\{\overline{\sigma_1}\rho \mid \rho \in \mathsf{traces}(t_{[\delta a\sigma_2,F',S'']})\} \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$ follows if we can prove that $\mathsf{traces}(t_{[\delta a\sigma_2,F',S'']}) \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$ holds.
  By applying Proposition 4 twice, we find that

$$\begin{aligned} &\mathsf{traces}(t_{[\delta a\sigma_2,F',S'']}) \\ = &\mathsf{traces}(t_{[\delta,F',S'']}) \cup \mathsf{traces}(t_{[a,F'_a,S''_a]}) \cup \{\overline{\delta a}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma_2,F'',S''']})\} \end{aligned}$$

  with $F'_a = \{\rho \mid \delta\rho \in F'\}$, $F'' = \{\rho \mid \delta a\rho \in F'\}$, $S''_a = S'' \mathbf{after} \delta$ and $S''' = S''_a \mathbf{after} a$. We know from Lemma 1 that $\mathsf{traces}(t_{[\delta,F',S'']}) = \{\epsilon\} \cup L_U \cup \{\theta\}$ and also $\mathsf{traces}(t_{[a,F'_a,S'_a]}) = \{\epsilon\} \cup L_U \cup \{a\}$. Following the base case of our induction, we find that $\mathsf{traces}(t_{[\sigma_2,F'',S''']}) \subseteq (L_U \cup \{\theta\})^*$ as well. Combining all observations, we find that

$$\mathsf{traces}(t_{[\delta a\sigma_2,F',S'']}) = \{\epsilon\} \cup L_U \cup \{\theta\} \cup \{\overline{\delta x} \mid x \in L_U\} \cup \{\overline{\delta a}\} \cup \{\overline{\delta a}\rho \mid \rho \in (L_U \cup \{\theta\})^*\}$$

  From this, we obtain $\mathsf{traces}(t_{[\delta a\sigma_2,F',S'']}) \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$, which was to be shown.

Proposition 7 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOLTS, let $F \subseteq \mathsf{Straces}(S')$ with $S' \subseteq S$, and let T be a set of test cases obtained with Algorithm 1 from S' and F. We have $\mathsf{traces}(T) \subseteq \bigcup_{\sigma \in F} \mathsf{traces}(t_{[\sigma,F,S]})$.*

Proof The proof is given by induction on the number of recursions of Algorithm 1 in generating a test case $t \in T$.

- *Base case.* We assume for the induction basis that test case t is generated by one time application of the algorithm. It is obvious that $t := \mathbf{pass}$. It follows from $\mathsf{traces}(\mathbf{pass}) = \epsilon$ that $\mathsf{traces}(\mathbf{pass}) \subseteq \bigcup_{\sigma \in F} \mathsf{traces}(t_{[\sigma,F,S]})$.
- *Induction step.* For the induction basis assume that the above thesis holds for all test cases obtained from $n-1$ times or less recursive application of the algorithm and test case t is generated from n times recursion. We distinguish two cases.
  - We suppose the second choice of the algorithm is selected at the first round of the application of the algorithm. Following the algorithm, $\mathsf{traces}(t) = \{\bar{x} \mid x \notin \mathbf{out}(S)\} \bigcup_{x \in \mathbf{out}(S)} \{\bar{x}\rho \mid x \in \mathbf{out}(S), \rho \in \mathsf{traces}(t_x)\} \cup \{\bar{a}\rho \mid a \in L_I, \rho \in \mathsf{traces}(t_a)\}$. We consider three cases.
    - We consider $x \notin \mathbf{out}(S)$. Upon observing $x \notin \mathbf{out}(S)$, t goes to terminal states and the algorithm terminates. Therefore, t is obtained by one time application of the algorithm. Following the induction hypothesis, $\{\bar{x} \mid x \notin \mathbf{out}(S)\} \subseteq \bigcup_{\sigma \in F} t_{[\sigma,F,S]}$.
    - We suppose that $t := x; t_x$ for some $x \in \mathbf{out}(S)$. We know that t_x is obtained by recursively applying the algorithm for $F' = \{\sigma \mid x\sigma \in F\}$ and $S' = S\,\mathbf{after}\,x$. Clearly, t_x is obtained by at most $n-1$ times of application of the algorithm. It follows from the induction hypothesis that $\mathsf{traces}(t_x) \subseteq \bigcup_{\sigma \in F'} \mathsf{traces}(t_{[\sigma,F',S']})$. We know from Lemma 1 that for every $\sigma \in F'$, $\{\bar{x}\rho \mid \rho \in \mathsf{traces}(t_{[\sigma,F',S']})\} \subseteq \{\mathsf{traces}(t_{[x\sigma,F,S]})\}$ (Note that $\forall \sigma \in F'$ we know that $x\sigma \in F$). Therefore, the previous observation along with $\{\bar{x}\rho \mid \rho \in \mathsf{traces}(t_x)\} \subseteq \bigcup_{\sigma \in F'} \{x\rho \mid \mathsf{traces}(t_{[\sigma,F',S']})\}$ leads to $\{\bar{x}\rho \mid \rho \in \mathsf{traces}(t_x)\} \subseteq \bigcup_{\sigma \in F} \mathsf{traces}(t_{[\sigma,F,S]})$. Consequently, $\bigcup_{x \in \mathbf{out}(S)} \{x\rho \mid \rho \in \mathsf{traces}(t_x)\} \subseteq \bigcup_{\sigma \in F} \mathsf{traces}(t_{[\sigma,F,S]})$ is resulted.
    - We suppose that $t := a; t_a$ for some $a \in L_I$ where $F' = \{\sigma \mid a\sigma \in F\} \neq \emptyset$ and t_a is obtained recursively by applying the algorithm for F' and $S' = S\,\mathbf{after}\,a$. With the same lines of reasoning in the previous item, we conclude that $\{a\rho \mid \rho \in \mathsf{traces}(t_a)\} \subseteq \bigcup_{\sigma \in F} \mathsf{traces}(t_{[\sigma,F,S]})$.

    Therefore, we show that all three sets $\{\bar{x} \mid x \notin \mathbf{out}(S)\}$, $\bigcup_{x \in \mathbf{out}(S)} \{\bar{x}\rho \mid x \in \mathbf{out}(S), \rho \in \mathsf{traces}(t_x)\}$ and $\{\bar{a}\rho \mid a \in L_I, \rho \in \mathsf{traces}(t_a)\}$ are a subset of $\bigcup_{\sigma \in F} \mathsf{traces}(t_{[\sigma,F,S]})$. Hence, $\mathsf{traces}(t) \subseteq \bigcup_{\sigma \in F} \mathsf{traces}(t_{[\sigma,F,S]})$.
  - We suppose the third choice of the algorithm is selected at the first round of the application of the algorithm. Following the algorithm, $\mathsf{traces}(t) = \{\bar{x} \mid x \notin \mathbf{out}(S)\} \bigcup_{x \in \mathbf{out}(S)} \{\bar{x}\rho \mid x \in \mathbf{out}(S), \rho \in \mathsf{traces}(t_x)\}$. The remainder of the proof is identical to the previous one.

Proposition 8 *Let IOLTS s be a specification, let IOTS$^{\sqcap}$ i be an implementation, and let t be a test case generated with Algorithm 1 from $s\,\mathbf{after}\,\epsilon$ and $\mathsf{ICtraces}(s)$. Then t is an internal choice test case and hence, it is successfully executable against i.*

Proof We know from Propositions 6 and 7 that $\mathsf{traces}(t) \subseteq (L_U \cup (\{\theta\}^+ L_I) \cup \{\theta\})^*$. Therefore, test case t is an internal choice test case. Following Proposition 5 i reaches a quiescent state before an input is provided by t; this input can be accepted by the implementation, which is input enabled in quiescent states. Therefore, t is executable against i.

By combining Theorem 3 with the above proposition, we get the following corollary. It states that our test case generation algorithm is sound and exhaustive for the internal choice setting.

Corollary 1 *Let IOLTS$^{\sqcap}$ $\langle S, L, \rightarrow, s_0 \rangle$ be a specification. Then*

1. *a test case obtained with Algorithm 1 from $s_0\,\mathbf{after}\,\epsilon$ and $\mathsf{ICtraces}(s_0)$ is sound for s_0 with respect to $\boldsymbol{ioco}^{\sqcap,\sqcap}_{\mathsf{ICtraces}(s_0)}$.*
2. *the set of all possible test cases that can be obtained from Algorithm 1 from $s_0\,\mathbf{after}\,\epsilon$ and $\mathsf{ICtraces}(s_0)$ is exhaustive for s_0 with respect to $\boldsymbol{ioco}^{\sqcap,\sqcap}_{\mathsf{ICtraces}(s_0)}$.*

5 Adapting IOCO to Asynchronous Setting

In order to perform conformance testing in the asynchronous setting in [13] and [14] both the class of implementations and test cases are restricted to internal choice class. Then, it is argued (with a proof sketch) that in this setting, the verdict obtained through asynchronous interaction with the system coincides with the verdict (using the same set of restricted test-cases) in the synchronous setting. In this section, we re-visit the approach of [13] and [14], give full proof of their main result and point out a slight imprecision in it.

5.1 Asynchronous Test Execution

Asynchronous communication delays obscure the observation of the tester; for example, the tester cannot precisely establish when the input sent to the system is actually consumed by it.

Asynchronous communication, as described in [8, Chapter 5], can be simulated by modelling the communications with the implementation through two dedicated FIFO channels. One is used for sending the inputs to the implementation, whereas the other is used to queue the outputs produced by the implementation. We assume that the channels are unbounded. By adding channels to an implementation, its visible behavior changes. This is formalized below.

Definition 17 (Queue operator) *Let $\langle S, L, \rightarrow, s_0\rangle$ be an arbitrary IOLTS, $\sigma_i \in L_I^*$, $\sigma_u \in L_U^*$ and $s, s' \in S$. The unary queue operator $_{[\sigma_u \ll - \ll \sigma_i]}$ is then defined by the following axioms and inference rules:*

$$[\sigma_u \ll s \ll \sigma_i] \xrightarrow{a} [\sigma_u \ll s \ll \sigma_i a], \qquad a \in L_I \qquad \textit{(A1)}$$

$$[x\sigma_u \ll s \ll \sigma_i] \xrightarrow{x} [\sigma_u \ll s \ll \sigma_i], \qquad x \in L_U \qquad \textit{(A2)}$$

$$\frac{s \xrightarrow{\tau} s'}{[\sigma_u \ll s \ll \sigma_i] \xrightarrow{\tau} [\sigma_u \ll s' \ll \sigma_i]} \qquad \textit{(I1)}$$

$$\frac{s \xrightarrow{a} s' \quad a \in L_I}{[\sigma_u \ll s \ll a\sigma_i] \xrightarrow{\tau} [\sigma_u \ll s' \ll \sigma_i]} \qquad \textit{(I2)}$$

$$\frac{s \xrightarrow{x} s' \quad x \in L_U}{[\sigma_u \ll s \ll \sigma_i] \xrightarrow{\tau} [\sigma_u x \ll s' \ll \sigma_i]} \qquad \textit{(I3)}$$

We abbreviate $_{[\epsilon \ll s \ll \epsilon]}$ to $Q(s)$. Given an IOLTS s_0, the initial state of s_0 in queue context is given by $Q(s_0)$.

Observe that for an arbitrary IOLTS s_0, $Q(s_0)$ is again an IOLTS. We have the following property, relating the traces of an IOLTS to the traces it has in the queued context.

Property 2 Let $\langle S, L, \rightarrow, s_0\rangle$ be an arbitrary IOLTS. Then for all $s, s' \in S$, we have $s \stackrel{\sigma}{\Longrightarrow} s'$ implies $Q(s) \stackrel{\sigma}{\Longrightarrow} Q(s')$.

The possibility of internal transitions is not observable to the remote asynchronous observer and hence, in [13, 14], weak quiescence is adopted to denote quiescence in the queue context.

Definition 18 (Synchronous execution in the queue context) *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOLTS, and let $\langle T, L', \rightarrow, t_0\rangle$ be a test case, such that $L_I = L'_U$ and $L_U = L'_I \setminus \{\theta\}$. Let $s, s' \in S$ and $t, t' \in T$. Then the synchronous execution of the test case and $Q(s_0)$ is defined through the following inference rules:*

$$\frac{[\sigma_u \ll s \ll \sigma_i] \xrightarrow{\tau} [\sigma'_u \ll s' \ll \sigma'_i]}{t\rceil\!\|_{[\sigma_u \ll s \ll \sigma_i]} \xrightarrow{\tau} t\rceil\!\|_{[\sigma'_u \ll s' \ll \sigma'_i]}} \text{(R1')}$$

$$\frac{t \xrightarrow{x} t' \qquad [\sigma_u \ll s \ll \sigma_i] \xrightarrow{x} [\sigma'_u \ll s' \ll \sigma'_i]}{t\rceil\!\|_{[\sigma_u \ll s \ll \sigma_i]} \xrightarrow{x} t'\rceil\!\|_{[\sigma'_u \ll s' \ll \sigma'_i]}} \text{(R2')}$$

$$\frac{t \xrightarrow{\theta} t' \qquad \delta_q([\sigma_u \ll s \ll \sigma_i])}{t\rceil\!\|_{[\sigma_u \ll s \ll \sigma_i]} \xrightarrow{\theta} t'\rceil\!\|_{[\sigma_u \ll s \ll \sigma_i]}} \text{(R3')}$$

The property below characterizes the relation between the test runs obtained by executing an internal choice test case in the synchronous setting and by executing a test case in the queued setting.

Property 3 Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOLTS and let $\langle T, L', \rightarrow, t_0\rangle$ be a TTS$^{\sqcap}$. Consider arbitrary states $s, s' \in S$ and $t, t' \in T$ and an arbitrary test run $\sigma \in L'^*$. We have the following properties:

1. $t\rceil\!\|s \stackrel{\sigma}{\Longrightarrow} t'\rceil\!\|s'$ implies $t\rceil\!\|Q(s) \stackrel{\sigma}{\Longrightarrow} t'\rceil\!\|Q(s')$
2. $\mathbf{Sinit}(t\rceil\!\|s) = \mathbf{Sinit}(t\rceil\!\|Q(s))$.

The proposition below proves to be essential in establishing the correctness of our main results in the remainder of Section 5. It essentially establishes the links between the internal behaviors of an implementation in the synchronous and the asynchronous settings.

Proposition 9 *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOLTS and let $\langle T, L', \rightarrow, t_0\rangle$ be a TTS$^{\sqcap}$. For all states $t \in T$, $s, s' \in S$, all $\sigma_i \in L_I^*$ and $\sigma_u \in L_U^*$, we have:*

1. $s \stackrel{\epsilon}{\Longrightarrow} s'$ *iff* $t\rceil\!\|s \stackrel{\epsilon}{\Longrightarrow} t\rceil\!\|s'$ *(R1*)*
2. $_{[\sigma_u \ll s \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll s' \ll \sigma_i]}$ *iff* $s \stackrel{\epsilon}{\Longrightarrow} s'$ *(I1*)*.

Proof

1. $s \stackrel{\epsilon}{\Longrightarrow} s'$ iff $t\rceil\!\|s \stackrel{\epsilon}{\Longrightarrow} t\rceil\!\|s'$ (R_1^*)
   We prove the two implications by induction on the length of the τ-traces leading to $\stackrel{\epsilon}{\Longrightarrow}$:
   - $\Rightarrow$ Assume, for the induction basis, that $i \stackrel{\epsilon}{\Longrightarrow} i'$ is due to a τ-trace of length 0; thus, $i = i'$ and it then follows that $t\rceil\!\|i \stackrel{\epsilon}{\Longrightarrow} t\rceil\!\|i$ and since $i = i'$, we have that $t\rceil\!\|i \stackrel{\epsilon}{\Longrightarrow} t\rceil\!\|i'$, which was to be shown.
     For the induction step, assume that the thesis holds for all $\stackrel{\epsilon}{\Longrightarrow}$ resulting from a τ-trace of length $n-1$ or less and that $i \xrightarrow{\tau} \ldots \xrightarrow{\tau} i_{n-1} \xrightarrow{\tau} i'$. It follows from the induction hypothesis that $t\rceil\!\|i \stackrel{\epsilon}{\Longrightarrow} t\rceil\!\|i_{n-1}$. Also from $i_{n-1} \xrightarrow{\tau} i'$ and deduction rule $R1$ in Definition 14, we have that $t\rceil\!\|i_{n-1} \stackrel{\epsilon}{\Longrightarrow} t\rceil\!\|i'$. Hence, that $t\rceil\!\|i \stackrel{\epsilon}{\Longrightarrow} t\rceil\!\|i'$, which was to be shown.
   - $\Leftarrow$ Almost identical to above. The induction basis is identical to the proof of the implication from left to right. For the induction step, note that the last τ-step of $t\rceil\!\|i_{n-1} \stackrel{\epsilon}{\Longrightarrow} t\rceil\!\|i'$ can only be due to deduction rule $R1$ and hence we have $i_{n-1} \stackrel{\epsilon}{\Longrightarrow} i'$, which in turn implies that $i \stackrel{\epsilon}{\Longrightarrow} i'$.
2. $_{[\sigma_u \ll i \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll i' \ll \sigma_i]}$ iff $i \stackrel{\epsilon}{\Longrightarrow} i'$ (I_1^*). Almost identical to the previous item: we prove the two implications by induction on the length of the τ-trace for leading to $\stackrel{\epsilon}{\Longrightarrow}$:
   - $\Rightarrow$ Assume, for the induction basis, that $i \stackrel{\epsilon}{\Longrightarrow} i'$ is due to a τ-trace of length 0; thus, that $i = i'$. It then follows that $_{[\sigma_u \ll i \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll i \ll \sigma_i]}$ and since $i = i'$, we have that $_{[\sigma_u \ll i \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll i' \ll \sigma_i]}$, which was to be shown.

For the induction step, assume that the thesis holds for all $\stackrel{\epsilon}{\Longrightarrow}$ resulting from a τ-trace of length $n-1$ or less and that $i \stackrel{\tau}{\longrightarrow} \ldots \stackrel{\tau}{\longrightarrow} i_{n-1} \stackrel{\tau}{\longrightarrow} i'$. It follows from the induction hypothesis that ${}_{[\sigma_u \ll i \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll i_{n-1} \ll \sigma_i]}$. Also from $i_{n-1} \stackrel{\tau}{\longrightarrow} i'$ and deduction rule *I1* in Definition 17, we have that ${}_{[\sigma_u \ll i_{n-1} \ll \sigma_i]} \stackrel{\tau}{\longrightarrow} {}_{[\sigma_u \ll i' \ll \sigma_i]}$.Hence, that ${}_{[\sigma_u \ll i \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll i' \ll \sigma_i]}$, which was to be shown.

$\Leftarrow$ Similar to the above item. The induction basis is identical. The induction step follows from the same reasoning. Note that ${}_{[\sigma_u \ll i_{n-1} \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll i' \ll \sigma_i]}$ can only be proven using deduction rule *I1* in Definition 17, because deduction rules *I2* and *I3* produce modified queues in their target of the conclusion. Hence, the premise of deduction rule *I1* should hold and thus, $i_{n-1} \stackrel{\tau}{\longrightarrow} i'$. Hence, using the induction hypothesis we obtain that $i \stackrel{\epsilon}{\Longrightarrow} i'$.

5.2 Sound Verdicts of Internal Choice Test Cases

In [14,7], it is argued that providing inputs to an IUT only after observing quiescence (i.e., in a stable state), eliminates the distortions in observable behavior, introduced by communicating to the IUT using queues. Hence, a subset of synchronous test-cases, namely those which only provide an input after observing quiescence, are safe for testing asynchronous systems. This is summarized in the following claim from [14,13] (and paraphrased in [7]):

Claim (Theorem 1 in [14]) Let s_0 be an arbitrary IOTS$^{\sqcap}$, and let $\langle T, L, \rightarrow, t_0 \rangle$ be a TTS$^{\sqcap}$. Then s_0 **passes** t_0 iff $Q(s_0)$ **passes** t_0.

In [7], the claim is taken for granted, and, unfortunately, in [14,13] only a proof sketch is provided for the above claim; the proof sketch is rather informal and leaves some room for interpretation, as illustrated by the following excerpt:

> "...An implementation guarantees that it will not send any output before receiving an input after quiescence is observed..."

As it turns out, the above result does not hold in its full generality, as illustrated by the following example.

Example 8 Consider the internal choice test case with initial state t_0 in Figure 6. Consider the implementation modeled by the IOTS$^{\sqcap}$ depicted in Figure 2, starting in state o_0. Clearly, we find that o_0 **passes** t_0; however, in the asynchronous setting, $Q(o_o)$ **passes** t_0 does not hold. This is due to the divergence in the implementation, which gives rise to an observation of quiescence in the queued context, but not so in the synchronous setting.

The claim *does* hold for non-divergent internal choice implementations. Note that divergence is traditionally also excluded from testing theories such as **ioco**. In this sense, assuming non-divergence is no restriction. Apart from the following theorem, we tacitly assume in all our formal results to follow that the implementation IOLTSs are non-divergent.

Theorem 4 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an arbitrary IOTS$^{\sqcap}$ and let $\langle T, L', \rightarrow, t_0 \rangle$ be a TTS$^{\sqcap}$. If s_0 is non-divergent, then s_0* **passes** *t_0 iff $Q(s_0)$* **passes** *t_0.*

Given the pervasiveness of the original (non-)theorem, a formal correctness proof of our corrections to this theorem (i.e., *our* Theorem 4) is highly desirable. In the remainder of this section, we therefore give the main ingredients for establishing a full proof for Theorem 4. We start by establishing a formal correspondence between observations of quiescence in the synchronous setting and observations of *weak* quiescence in the asynchronous setting.

Lemma 2 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOTS$^{\sqcap}$. Let $s \in S$ be an arbitrary state. Then $\delta_q(Q(s))$ implies $\delta(s')$ for some $s' \in S$ satisfying $s \stackrel{\epsilon}{\Longrightarrow} s'$.*

Proof Assume, towards a contradiction, that for all $s' \in S$ such that $s \stackrel{\epsilon}{\Longrightarrow} s'$, it doesn't hold $\delta(s')$. Take the s' with the largest empty trace (by counting the numbers of τ-labeled transitions). Such s' must exist since otherwise, there must be a loop of τ-labeled transition which is opposed to the assumption that s doesn't diverge. Since s' is not quiescent, according to Definition 4, there exists an $x \in L_u$ such that $s' \stackrel{x}{\longrightarrow}$. Hence, there must exist an $s'' \in S$ such that $s' \stackrel{x}{\longrightarrow} s''$. It follows from Proposition 9 and deduction rule $I3$ in Definition 17 that $Q(s) \stackrel{\epsilon}{\Longrightarrow}_{[x \ll s'' \ll \epsilon]}$ and since the output queue is non-empty we can apply the deduction rule $A2$ on the target state and obtain ${}_{[x \ll s'' \ll \epsilon]} \stackrel{x}{\longrightarrow} Q(s'')$. Combining the two transition we obtain $Q(s) \stackrel{x}{\Longrightarrow} Q(s'')$. From the latter transition we can conclude that $Q(s)$ is not quiescent which is contradictory to the statement.

The above lemma guarantees that all stimuli provided by an TTS$^{\sqcap}$ are accepted by implementations that behave as some IOTS$^{\sqcap}$, even when we adopt the asynchronous communication scheme between testers and the implementation. Following the above lemma, the proposition below states that every asynchronous test case execution can lead to a state in which both communication queues are empty.

Proposition 10 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOTS$^{\sqcap}$, and let $\langle T, L', \rightarrow, t_0 \rangle$ be a TTS$^{\sqcap}$. Assume arbitrary states $t' \in T$ and $s, s' \in S$, and an arbitrary test run $\sigma \in L'^*$. Then for all $\sigma_i \in L_I^*$ and $\sigma_u \in L_U^*$:*

$$t_0 \rceil Q(s) \stackrel{\sigma}{\Longrightarrow} t' \rceil_{[\sigma_u \ll s' \ll \sigma_i]} \textit{ implies } \exists s'' \in S : t_0 \rceil Q(s) \stackrel{\sigma}{\Longrightarrow} t' \rceil Q(s'')$$

Before we address the proof of the above proposition, we first need to show the correctness of some auxiliary lemmata given bellow. The lemma below states that only at weakly quiescent states the input queue can grow.

Lemma 3 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOTS$^{\sqcap}$, and let $\langle T, L', \rightarrow, t_0 \rangle$ be a TTS$^{\sqcap}$. Let $s, s' \in S$, $t, t' \in T$ be arbitrary states and $\sigma_u \in L_U^*$ and $\sigma_i \in L_I^*$ and $a \in L_I$. If $t \rceil_{[\sigma_u \ll s \ll \sigma_i]} \stackrel{a}{\Longrightarrow} t' \rceil_{[\sigma_u \ll s' \ll \sigma_i a]}$, then $\delta_q([\sigma_u \ll s' \ll \sigma_i])$.*

Proof Assume $a \in L_I$ and $t \rceil_{[\sigma_u \ll s \ll \sigma_i]} \stackrel{a}{\Longrightarrow} t' \rceil_{[\sigma_u \ll s' \ll \sigma_i a]}$, we know there exists an $s'' \in S$ such that $t \rceil_{[\sigma_u \ll s \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} t \rceil_{[\sigma_u \ll s'' \ll \sigma_i]} \stackrel{a}{\longrightarrow} t' \rceil_{[\sigma_u \ll s'' \ll \sigma_i a]} \stackrel{\epsilon}{\Longrightarrow} t' \rceil_{[\sigma_u \ll s' \ll \sigma_i a]}$. It follows from Proposition 9(2) that $s \stackrel{\epsilon}{\Longrightarrow} s''$ and also $s'' \stackrel{\epsilon}{\Longrightarrow} s'$. We thus find that $s \stackrel{\epsilon}{\Longrightarrow} s'$ and subsequently according to Proposition 9(2) we have ${}_{[\sigma_u \ll s \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll s' \ll \sigma_i]}$. The former observation and Proposition 9(1) lead to $t \rceil_{[\sigma_u \ll s \ll \sigma_i]} \stackrel{\epsilon}{\Longrightarrow} t \rceil_{[\sigma_u \ll s' \ll \sigma_i]}$. Using deduction rule $A1$ in Definition 17 and applying deduction rule $R2$ in Definition 14 result in $t \rceil_{[\sigma_u \ll s' \ll \sigma_i]} \stackrel{a}{\Longrightarrow} t' \rceil_{[\sigma_u \ll s' \ll \sigma_i a]}$. Hence, there is a trace starting from $t \rceil_{[\sigma_u \ll s \ll \sigma_i]}$ to $t \rceil_{[\sigma_u \ll s' \ll \sigma_i]} \stackrel{a}{\Longrightarrow} t' \rceil_{[\sigma_u \ll s' \ll \sigma_i a]}$. It follows then from Definition 13 that $\delta_q([\sigma_u \ll s' \ll \sigma_i])$ (since test case t only provides an input immediately after if it has observed quiescence), which was to be shown.

We find that in executing an internal choice test case on an implementation behaving as an IOLTS$^{\sqcap}$, the input and output queues cannot be non-empty simultaneously. This is formalized by the lemma below.

Lemma 4 *Let $\langle S, L, \rightarrow, s_0 \rangle$ be an IOTS$^{\sqcap}$, and let $\langle T, L', \rightarrow, t_0 \rangle$ be a TTS$^{\sqcap}$. Let $s, s' \in S$, $t, t' \in T$ be arbitrary states. There is no trace $\sigma_u \in L'^*$ such that $t \rceil Q(s) \stackrel{\sigma}{\Longrightarrow} t' \rceil_{[\sigma_u \ll s' \ll \sigma_i]}$ and the input and output queues are both non-empty at the same time($\sigma_i \neq \epsilon \wedge \sigma_u \neq \epsilon$).*

Proof Assume, towards a contradiction, that the following two statements hold:

1. $t \rceil Q(s) \stackrel{\sigma}{\Longrightarrow} t' \rceil_{[\sigma_u \ll s' \ll \sigma_i]}$
2. $\sigma_i \neq \epsilon \wedge \sigma_u \neq \epsilon$

Since both σ_i and σ_u are non-empty, there must exist the largest prefix σ' of σ during which the two queues are never simultaneously non-empty, i.e., by observing a single action after σ', both queues become non-empty for the first time. Hence, there exists $\sigma', \sigma'' \in L'^*$ as a prefix and postfix of σ respectively and $y \in L'$.

1. $\sigma = \sigma' y \sigma''$
2. there exist $\sigma'_i \in (L_I)^*, \sigma'_u \in (L_U)^*$ such that $t\rceil Q(s) \stackrel{\sigma'}{\Longrightarrow} t_1\rceil_{[\sigma'_u \ll s_1 \ll \sigma'_i]}$ (with $t_1 \in T$ and $s_1 \in S$) and $((\sigma'_u = \epsilon \wedge \sigma'_i \neq \epsilon) \vee (\sigma'_i = \epsilon \wedge \sigma'_u \neq \epsilon))$
3. there exist $\sigma''_i \in (L_I)^*, \sigma''_u \in (L_U)^*$ such that $t_1\rceil_{[\sigma'_u \ll s_1 \ll \sigma'_i]} \stackrel{y}{\longrightarrow} t_2\rceil_{[\sigma''_u \ll s_2 \ll \sigma''_i]}$ (with $t_2 \in T$ and $s_2 \in S$) $\wedge((\sigma'_u = \epsilon \wedge \sigma'_i \neq \epsilon \wedge \sigma''_u \neq \epsilon \wedge \sigma''_i = \sigma'_i) \vee (\sigma'_i = \epsilon \wedge \sigma'_u \neq \epsilon \wedge \sigma''_i \neq \epsilon \wedge \sigma''_u = \sigma'_u))$
4. $t_2\rceil_{[\sigma''_u \ll s_2 \ll \sigma''_i]} \stackrel{\sigma''}{\Longrightarrow} t'\rceil_{[\sigma_u \ll s' \ll \sigma_i]}$

Note that after σ' both input and output queues cannot be empty, since a single transition y only increases the size of one of the two queues (see rules *A1* and *I3* in Definition 17). Below, we distinguish two cases based on the status of the input queue after executing the trace σ': either the input queue is empty (and the output queue is not), or the other way around.

1. *Case* $\sigma'_u = \epsilon$. The only possible transition that can fill an output queue is due to the application of deduction rule *I3* in Definition 17. Hence, there must exists some s_2 and $x \in L_U$ such that ${}_{[\epsilon \ll s_1 \ll \sigma'_i]} \stackrel{\tau}{\longrightarrow} {}_{[x \ll s_2 \ll \sigma'_i]}$ and subsequently, $(t_1\rceil_{[\epsilon \ll s_1 \ll \sigma'_i]} \stackrel{\tau}{\longrightarrow} t_2\rceil_{[x \ll s_2 \ll \sigma'_i]})$ (thereby satisfying the third item with $\sigma'_u = \epsilon$ and $\sigma''_u = x$). The former x-labeled transition can only be due to deduction rule *I3* in Definition 17 and hence, we have $s_1 \stackrel{x}{\longrightarrow} s_2$. However, it follows from $\sigma'_i \neq \epsilon$ that there exists an $a \in L_I$, $s_p \in S$, a prefix σ'_p of σ' and $\rho_i \in L_I^*$ such that $\sigma'_i = \rho_i a$ and $t\rceil Q(s) \stackrel{\sigma'_p}{\Longrightarrow} t'_1\rceil_{[\epsilon \ll s_p \ll \rho_i]} \stackrel{a}{\Longrightarrow} t_1\rceil_{[\epsilon \ll s_1 \ll \sigma'_i]}$. We have from Lemma 3 that $\delta_q({}_{[\epsilon \ll s_1 \ll \rho_i]})$. Using deduction rule *A2* on $s_1 \stackrel{x}{\longrightarrow} s_2$, we obtain that ${}_{[\epsilon \ll s_1 \ll \rho_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[x \ll s_2 \ll \rho_i]}$. Hence according to Definition 4, state ${}_{[\epsilon \ll s_1 \ll \rho_i]}$ is not quiescent, which contradicts our observation that $\delta_q({}_{[\epsilon \ll s_1 \ll \rho_i]})$.
2. *Case* $\sigma'_i = \epsilon$. The only transition which allows for filling the input queue is due to the subsequent application of deduction rules R2 and A1. Hence, there exists an $a \in L_I$, such that $t_1\rceil_{[\sigma'_u \ll s_1 \ll \epsilon]} \stackrel{a}{\longrightarrow} t_2\rceil_{[\sigma'_u \ll s_2 \ll a]})$ and ${}_{[\sigma'_u \ll s_1 \ll \epsilon]} \stackrel{a}{\longrightarrow} {}_{[\sigma'_u \ll s_2 \ll a]}$ (where the former satisfies the third item by taking $\sigma'_i = \epsilon$ and $\sigma''_i = a$). It follows from Lemma 3 that $\delta_q({}_{[\sigma'_u \ll s_2 \ll \epsilon]})$. However since $\sigma'_u \neq \epsilon$, there exists a $y \in L_U$ and $\rho_u \in L_U^*$, such that $\sigma'_u = y\rho_u$ and using deduction rule A2, we obtain that that ${}_{[\sigma'_u \ll s_2 \ll \epsilon]} \stackrel{y}{\longrightarrow}$ and thus, ${}_{[\sigma'_u \ll s_2 \ll \epsilon]}$ is not quiescent, which contradicts our earlier observation.

Finally, the lemma given below states that in a queue context, implementations that have a non-empty input queue are weakly quiescent. The correctness of the lemma follows from the two preceding lemmata.

Lemma 5 *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOTS$^\sqcap$, and let $\langle T, L', \rightarrow, t_0\rangle$ be a TTS$^\sqcap$. Let $s, s' \in S$, $t, t' \in T$ be arbitrary states, $\sigma \in L'^*$, $\sigma_i \in L_I^*$ and $\sigma_u \in L_U^*$. If $t\rceil Q(s) \stackrel{\sigma}{\Longrightarrow} t'\rceil_{[\sigma_u \ll s' \ll \sigma_i]}$ and $\sigma_i \neq \epsilon$ then $\delta_q(s')$ and $\sigma_u = \epsilon$.*

Proof By lemma 4, we have that $\sigma_u = \epsilon$. Assume, towards a contradiction that there exists an $x \in L_U$ such that $x \in \mathbf{Sinit}(s')$. Since $x \in \mathbf{Sinit}(s')$, it follows from Definition 2(3) that there exists an $s'' \in S$ such that $s' \stackrel{x}{\Longrightarrow} s''$. Since $\sigma_i \neq \epsilon$ there exist $\sigma' \in L'^*$, $s_p \in S, t_p \in T, a \in L_I$, and $\rho_i \in L_I^*$ such that $\sigma_i = \rho_i a$ and $t\rceil Q(s) \stackrel{\sigma'}{\Longrightarrow} t_p\rceil_{[\epsilon \ll s_p \ll \rho_i]} \stackrel{a}{\Longrightarrow} t'\rceil_{[\epsilon \ll s' \ll \sigma_i]}$. Hence by Lemma 3, ${}_{[\epsilon \ll s' \ll \rho_i]}$ is quiescent, *i.e.*, $\delta_q({}_{[\epsilon \ll s' \ll \rho_i]})$.

It follows from the assumption that ${}_{[\epsilon \ll s' \ll \rho_i]} \stackrel{\epsilon}{\Longrightarrow} {}_{[x \ll s'' \ll \rho_i]}$. Since the output queue is non-empty we can apply deduction rule *A2* on the target state and obtain ${}_{[x \ll s'' \ll \rho_i]} \stackrel{x}{\longrightarrow}$

$_{[\epsilon\ll s''\ll\rho_i]}$. Combining the two transitions, we obtain $_{[\epsilon\ll s'\ll\rho_i]} \overset{x}{\Longrightarrow} {}_{[\epsilon\ll s''\ll\rho_i]}$. From the latter transition, we conclude that $_{[\epsilon\ll s'\ll\rho_i]}$ is not quiescent which is a contradiction.

We now are in a position to formally establish the correctness of Proposition 10.

Proof (Proposition 10). We distinguish four cases based on the status of input and output queues.

1. *Case* $\sigma_i = \epsilon$, $\sigma_u = \epsilon$. By assuming $s' = s$, the statement holds.
2. *Case* $\sigma_i \neq \epsilon$, $\sigma_u \neq \epsilon$. According to Lemma 4, no trace leads to this situation.
3. *Case* $\sigma_i \neq \epsilon$, $\sigma_u = \epsilon$. We prove this case by an induction on the length of σ_i.
   Since $\sigma_i \neq \epsilon$, for the induction basis, the smallest possible length of σ_i is one. Thus there must be an $x \in L_I$ such that $\sigma_i = x$. From Lemma 5, we know that $\forall y \in L_U, y \notin \mathbf{Sinit}(s')$ and since s' doesn't diverge, it must reach eventually a state such as $i \in S$ which performs a transition other than an internal one, hence the only possible choice is an input transition. From Definition 8 we know that $\delta(i)$ and state i is input-enabled as well. Thus $\exists i' \in S : i \xrightarrow{x} i'$. Due to the subsequent application of deduction rules of *I1*, *I2* in Definition 17 and *R1* in Definition 14, transition $t'\rceil\!\|_{[\epsilon\ll s'\ll x]} \overset{\epsilon}{\Longrightarrow} t'\rceil\!\|Q(i')$ is possible. By assuming $s'' = i'$ and combination of the latter transition and the assumption, we have $t\rceil\!\|Q(s) \overset{\sigma}{\Longrightarrow} t'\rceil\!\|Q(s'')$ which was to be shown.
   For the induction step, assume that the statement holds for all non-empty input queues of length $n-1$ or less and length n for σ_i. It follows from $\sigma_i \neq \epsilon$ that there exists an $a \in L_I$, $\sigma_i' \in L_I*$, $\sigma' \in L'^*$ and $i' \in S$ and $t_p \in T$ such that $\sigma_i = \sigma_i' a$ and $t\rceil\!\|Q(s) \overset{\sigma'}{\Longrightarrow} t_p\rceil\!\|_{[\epsilon\ll i'\ll\sigma_i']} \overset{a}{\Longrightarrow} t'\rceil\!\|_{[\epsilon\ll s'\ll\sigma_i]}$. It follows from the induction hypothesis that $\exists i \in S : t\rceil\!\|Q(s) \overset{\sigma'}{\Longrightarrow} t_p\rceil\!\|Q(i)$. Due to the application of deduction rule *R2* in Definition 14 and *A1* in Definition17, we have $t_p\rceil\!\|Q(i) \overset{a}{\Longrightarrow} t'\rceil\!\|_{[\epsilon\ll i\ll a]}$. It follows from the induction basis that $\exists s'' \in S : t_p\rceil\!\|Q(i) \overset{a}{\Longrightarrow} t'\rceil\!\|Q(s'')$. Combining both transitions leads to $\exists s'' \in S : t\rceil\!\|Q(s) \overset{\sigma}{\Longrightarrow} t'\rceil\!\|Q(s'')$ which was to be shown.
4. *Case* $\sigma_i = \epsilon$, $\sigma_u \neq \epsilon$. We prove this case by an induction on the length of σ_u.
   Since $\sigma_u \neq \epsilon$, for the induction basis, the smallest possible length of σ_u is one. Thus, assume, for the induction basis, that there exists an $x \in L_U$ such that $\sigma_u = x$. The only possible transition that can fill the output queue is due to the application of deduction rule *I3* in Definition 17. Hence, there must exist some $s'', q'' \in S$ such that $_{[\sigma_u'\ll s''\ll\sigma_i']} \xrightarrow{\tau} {}_{[\sigma_u'x\ll q''\ll\sigma_i']} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u'x\ll s'\ll\sigma_i']}$. Combining both transitions, we find $_{[\sigma_u'\ll s''\ll\sigma_i']} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u'x\ll s'\ll\sigma_i']}$. It follows from the application of deduction rule $R1^*$ in Proposition 9 that the input queue at state $_{[\sigma_u'\ll s''\ll\sigma_i']}$ must be empty since otherwise according to Lemma 5, s'' would be quiescent and could not produce any output. Thus there exist $\sigma' \in L'^*$, $\sigma_u' \in L_U^*$ and $t_p' \in T$ such that $t\rceil\!\|Q(s) \overset{\sigma'}{\Longrightarrow} t_p'\rceil\!\|_{[\sigma_u'\ll s''\ll\epsilon]} \overset{\epsilon}{\Longrightarrow} t_p'\rceil\!\|_{[\sigma_u'x\ll s'\ll\epsilon]} \overset{\sigma_u'}{\Longrightarrow} t'\rceil\!\|_{[x\ll s'\ll\epsilon]}$ and $\sigma = \sigma'\sigma_u'$. Applying deduction rules *R2* in Definition 14 and *A2* in Definition 17, we find $t_p'\rceil\!\|_{[\sigma_u'\ll s''\ll\epsilon]} \overset{\sigma_u'}{\Longrightarrow} t'\rceil\!\|Q(s'')$ and subsequently we have $t\rceil\!\|Q(s) \overset{\sigma'}{\Longrightarrow} t_p'\rceil\!\|_{[\sigma_u'\ll s''\ll\epsilon]} \overset{\sigma_u'}{\Longrightarrow} t'\rceil\!\|Q(s'')$ which was to be shown.
   For the induction step, assume that the thesis holds for all non-empty output queues with length $n-1$ or less and length of σ_u is n. It follows from $\sigma_u \neq \epsilon$ that there exist an $x \in L_U$, $\sigma_u' \in L_U^*$, $\sigma' \in L'^*$ and $t_p \in T$ and $q, q' \in S$ such that $\sigma_u = \sigma_u' x$ and $t\rceil\!\|Q(s) \overset{\sigma'}{\Longrightarrow} t_p\rceil\!\|_{[\sigma_u''\sigma_u'\ll q\ll\epsilon]} \xrightarrow{\tau} t_p\rceil\!\|_{[\sigma_u''\sigma_u'x\ll q'\ll\epsilon]} \overset{\sigma_u''}{\Longrightarrow} t'\rceil\!\|_{[\sigma_u'x\ll s'\ll\epsilon]}$ and $\sigma = \sigma'\sigma_u''$. Applying deduction rule *R2* in Definition 14 and *A2* in Definition 17, we have $t_p\rceil\!\|_{[\sigma_u''\sigma_u'\ll q\ll\epsilon]} \overset{\sigma_u''}{\Longrightarrow} t'\rceil\!\|_{[\sigma_u'\ll q\ll\epsilon]}$. Thus we can run the previous execution in a new order such as $t\rceil\!\|Q(s) \overset{\sigma'}{\Longrightarrow} t_p\rceil\!\|_{[\sigma_u''\sigma_u'\ll q\ll\epsilon]} \overset{\sigma_u''}{\Longrightarrow} t'\rceil\!\|_{[\sigma_u'\ll q\ll\epsilon]} \xrightarrow{\tau} t'\rceil\!\|_{[\sigma_u'x\ll s'\ll\epsilon]}$. Hence we can reach a new state with the output length less than the length of σ_u by running the same

execution and it follows from the induction hypothesis that $\exists s'' \in S : t\rceil\!\|Q(s) \stackrel{\sigma}{\Longrightarrow} t'\rceil\!\|Q(s'')$ which was to be shown.

$\boxtimes$

As a consequence of the above proposition, we find the following corollary. It states that each asynchronous test execution can be chopped into individual observations such that before and after each observation the communication queue is empty.

Corollary 2 *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOTS$^{\sqcap}$, and let $\langle T, L', \rightarrow, t_0\rangle$ be a TTS$^{\sqcap}$. Assume arbitrary states $t' \in T$ and $s, s' \in S$, and an arbitrary test run $\sigma \in L'^*$ and $x \in L'$. Then $t_0\rceil\!\|Q(s) \stackrel{\sigma x}{\Longrightarrow} t'\rceil\!\|Q(s')$ implies $\exists t'' \in T, s'' \in S : t_0\rceil\!\|Q(s) \stackrel{\sigma}{\Longrightarrow} t''\rceil\!\|Q(s'') \stackrel{x}{\Longrightarrow} t'\rceil\!\|Q(s')$. Moreover, if $x = \theta$ then $\delta_q(Q(s'))$.*

The lemma below establishes a correspondence between the test runs that can be executed in the asynchronous setting and those runs one would obtain in the synchronous setting. The lemma is basic to the correctness of our main results in this section.

Lemma 6 *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOTS$^{\sqcap}$, and let $\langle T, L', \rightarrow, t_0\rangle$ be a TTS$^{\sqcap}$. Let $s, s' \in S$ and $t' \in T$ be arbitrary states. Then, for all $\sigma \in L'^*$, such that $t_0\rceil\!\|Q(s) \stackrel{\sigma}{\Longrightarrow} t'\rceil\!\|Q(s')$, there is a non-empty set $\mathbb{S} \subseteq \{s'' \in S \mid s' \stackrel{\epsilon}{\Longrightarrow} s''\}$ such that*

1. *$\{s'' \in S \mid \delta(s'') \wedge s' \stackrel{\epsilon}{\Longrightarrow} s''\} \subseteq \mathbb{S}$ if $\exists \sigma' \in L'^* : \sigma = \sigma'\theta$*
2. *$s' \in \mathbb{S}$ if $\nexists \sigma' \in L'^* : \sigma = \sigma'\theta$*
3. *$\forall s'' \in \mathbb{S} : t_0\rceil\!\|s \stackrel{\sigma}{\Longrightarrow} t'\rceil\!\|s''$.*

Proof We prove this lemma by induction on the length of $\sigma \in L'^*$.

- Induction basis. Assume that the length of σ is 0, i.e., $\sigma = \epsilon$. Assume that $t_0\rceil\!\|Q(s) \stackrel{\epsilon}{\Longrightarrow} t_0\rceil\!\|Q(s')$. By Proposition 9(2) we have $s \stackrel{\epsilon}{\Longrightarrow} s'$. Set $\mathbb{S} = \{s'' \mid s' \stackrel{\epsilon}{\Longrightarrow} s''\}$. Let $s'' \in \mathbb{S}$ be an arbitrary state. Proposition 9(1) leads to $t_0\rceil\!\|s \stackrel{\epsilon}{\Longrightarrow} t_0\rceil\!\|s'$ and $t_0\rceil\!\|s' \stackrel{\epsilon}{\Longrightarrow} t_0\rceil\!\|s''$; by transitivity, we have the desired $t_0\rceil\!\|s \stackrel{\epsilon}{\Longrightarrow} t_0\rceil\!\|s''$. It is also clear that $s' \in \mathbb{S}$. We thus find that $\mathbb{S}$ meets the desired conditions.
- Inductive step. Assume that the statement holds for all σ' of length at most $n-1$. Suppose that the length of σ is n. Assume that $t_0\rceil\!\|Q(s) \stackrel{\sigma}{\Longrightarrow} t'\rceil\!\|Q(s')$. By Corollary 2, there is some $s_{n-1} \in S$, a $t_{n-1} \in T$ and $\sigma_{n-1} \in L'^*$ and $x \in L'$, such that $\sigma = \sigma_{n-1}x$ and $t_0\rceil\!\|Q(s) \stackrel{\sigma_{n-1}}{\Longrightarrow} t_{n-1}\rceil\!\|Q(s_{n-1}) \stackrel{x}{\Longrightarrow} t'\rceil\!\|Q(s')$.
  By induction, there must be a set $\mathbb{S}_{n-1} \subseteq \{s'' \in S \mid s_{n-1} \stackrel{\epsilon}{\Longrightarrow} s''\}$, such that
  1. $\{s'' \in S \mid \delta(s'') \wedge s_{n-1} \stackrel{\epsilon}{\Longrightarrow} s''\} \subseteq \mathbb{S}_{n-1}$ if $\exists \sigma' \in L'^* : \sigma = \sigma'\theta$
  2. $s_{n-1} \in \mathbb{S}_{n-1}$ if $\nexists \sigma' \in L'^* : \sigma = \sigma'\theta$
  3. $\forall s'' \in \mathbb{S}_{n-1} : t_0\rceil\!\|s \stackrel{\sigma_{n-1}}{\Longrightarrow} t_{n-1}\rceil\!\|s''$.

  We next distinguish three cases: $x \in L_I$, $x \in L_U$ and $x \notin L_I \cup L_U$.
  1. Case $x = \theta$. We thus find that $t_{n-1}\rceil\!\|Q(s_{n-1}) \stackrel{\theta}{\Longrightarrow} t_n\rceil\!\|Q(s')$. As a result of Corollary 2, we have $\delta_q(s')$. We then find as a result of Lemma 2, there must be some state $s'' \in S$ such that $s_{n-1} \stackrel{\epsilon}{\Longrightarrow} s' \stackrel{\epsilon}{\Longrightarrow} s''$ and $\delta(s'')$. Consider the set $\mathbb{S}_n = \{s'' \in S \mid \delta(s'') \wedge s' \stackrel{\epsilon}{\Longrightarrow} s''\}$.
     Let s'' be an arbitrary state in $\mathbb{S}_n$. Distinguish between cases $s_{n-1} \notin \mathbb{S}_{n-1}$ and $s_{n-1} \in \mathbb{S}_{n-1}$. In the case, $s_{n-1} \notin \mathbb{S}_{n-1}$, we know from the construction of $\mathbb{S}_{n-1}$ that $s'' \in \mathbb{S}_{n-1}$ and $s'' \stackrel{\epsilon}{\Longrightarrow} s''$ always holds. In the case $s_{n-1} \in \mathbb{S}_{n-1}$, we have that $s_{n-1} \stackrel{\epsilon}{\Longrightarrow} s' \stackrel{\epsilon}{\Longrightarrow} s''$. We thus find that $\forall s'' \in \mathbb{S}_n\ \exists \bar{s} \in \mathbb{S}_{n-1} : t_0\rceil\!\|s \stackrel{\sigma_{n-1}}{\Longrightarrow} t_{n-1}\rceil\!\|\bar{s} \stackrel{\epsilon}{\Longrightarrow} t_{n-1}\rceil\!\|s'' \stackrel{\theta}{\longrightarrow} t'\rceil\!\|s''$. Thus $\mathbb{S}_n$ has the desired requirement that $t_0\rceil\!\|s \stackrel{\sigma_{n-1}x}{\Longrightarrow} t'\rceil\!\|s''$ for all $s'' \in \mathbb{S}_n$. Also, $\{s'' \in S \mid \delta(s'') \wedge s' \stackrel{\epsilon}{\Longrightarrow} s''\} \subseteq \mathbb{S}_n$ is concluded from construction of $\mathbb{S}_n$. Hence, $\mathbb{S}_n$ satisfies all desired conditions.

2. Case $x \in L_I$. By Property 5, we find that the last step in σ_{n-1} must be θ. It follows from corollary 2 that $Q(s_{n-1})$ is weakly quiescent and consequently $\delta_q(s_{n-1})$. By induction we have that $\{s'' \in S \mid \delta(s'') \wedge s_{n-1} \stackrel{\epsilon}{\Longrightarrow} s''\} \subseteq \mathbb{S}_{n-1}$. Consider the set $\mathbb{S}_n = \{s'' \in S \mid s' \stackrel{\epsilon}{\Longrightarrow} s''\}$.
   Transition $t_{n-1} \rceil\!\| Q(s_{n-1}) \stackrel{x}{\Longrightarrow} t' \rceil\!\| Q(s')$ implies that $s_{n-1} \stackrel{x}{\Longrightarrow} s'$. By Lemma 2 and Definition 8, we know that $\exists \bar{s} \in S$ such that $s_{n-1} \stackrel{\epsilon}{\Longrightarrow} \bar{s} \stackrel{x}{\Longrightarrow} s'$ and $\delta(\bar{s})$. From construction of $\mathbb{S}_{n-1}$, we know that $\bar{s}$ is in $\mathbb{S}_{n-1}$. We thus have $\forall s'' \in \mathbb{S}_n \; \exists \bar{s} \in \mathbb{S}_{n-1}$: $t_0 \rceil\!\| s \stackrel{\sigma_{n-1}}{\Longrightarrow} t_{n-1} \rceil\!\| \bar{s} \stackrel{x}{\Longrightarrow} t' \rceil\!\| s''$.
   It is clear form construction of $\mathbb{S}_n$ that $s' \in \mathbb{S}_n$ as the required condition that $s' \in \mathbb{S}_n$ if the last step of σ is not θ-labeled transition. We thus find that $\mathbb{S}_n$ fulfills all desired requirements.
3. Case $x \in L_U$. Analogous to the previous case.

We are now in a position to establish the correctness of Theorem 4. We provide the proof below:

Proof (Theorem 4) We prove the theorem by contraposition.

1. Case $\Rightarrow$. Suppose not $Q(s)$ $\mathbf{passes}$ t_0. By Definition 15 and Proposition 10, $t_0 \rceil\!\| Q(s) \stackrel{\sigma'}{\Longrightarrow} \mathbf{fail} \rceil\!\| Q(s')$, for some $\sigma' \in L'^*$ and $s' \in S$. As a result of Lemma 6, there is a non-empty set $\mathbb{S} \subseteq \{s'' \in S \mid s' \stackrel{\epsilon}{\Longrightarrow} s''\}$ such that for all $s'' \in \mathbb{S}$, $t_0 \rceil\!\| s \stackrel{\sigma'}{\Longrightarrow} \mathbf{fail} \rceil\!\| s''$, which was what we needed to prove.
2. Case $\Leftarrow$. Assume, that not s $\mathbf{passes}$ t_0. Then there are $\sigma' \in L'^*$ and $s'' \in S$, $t_0 \rceil\!\| s \stackrel{\sigma'}{\Longrightarrow} \mathbf{fail} \rceil\!\| s''$. Using Property 3 leads to $t_0 \rceil\!\| Q(s) \stackrel{\sigma'}{\Longrightarrow} \mathbf{fail} \rceil\!\| Q(s'')$.

6 Adapting Asynchronous Setting to IOCO

In this section, we re-cast the results of the previous section to the setting with **ioco** test-cases. We first show that the result of Theorem 2 cannot be trivially generalized to the asynchronous setting. Then using an approach inspired by [8, Chapter 5] and [7], we show how to re-formulate Theorem 4 in this setting.

In section 3 it is shown that restricting the set of traces F in implementation relation $\mathbf{ioco}_F^{a,b}$ will lead to a weaker testing power. Yet, we proved in Theorem 1 that discrimination power of $\mathbf{ioco}_{\mathsf{Straces}(s)}^{a,b}$ for a given specification s does not decrease by examining internal choice traces of s instead of suspension traces in setting $a, b \in \{\sqcap\}$. But, in the following example, we motivate that the testing power of $\mathbf{ioco}_{\mathsf{ICtraces}(s)}^{\sqcap,\sqcap}$ and $\mathbf{ioco}_{\mathsf{Straces}(s)}^{\sqcap,\sqcap}$ are different in the asynchronous setting.

Example 9 IOLTS t' in Figure 6 shows a test case for IOLTS o_0 in Figure 2, which is an internal choice IOTS. Assume that at the same time o_0 is also used as the implementation.

For o_0 as specification and implementation, we have that o_0 $\mathbf{ioco}$ o_0. However, we can reach a fail verdict for o_0 under the queue context when using the test case t'_0. Consider the sequence mbr; in the queue context, the execution $t'_0 \rceil\!\| Q(o_0) \stackrel{m}{\longrightarrow} t'_1 \rceil\!\|_{[\epsilon \ll o_0 \ll m]} \stackrel{\epsilon}{\Longrightarrow} t'_1 \rceil\!\| Q(o_1) \stackrel{\epsilon}{\Longrightarrow} t'_1 \rceil\!\|_{[r \ll o_2 \ll \epsilon]} \stackrel{b}{\longrightarrow} t'_2 \rceil\!\|_{[r \ll o_2 \ll b]} \stackrel{r}{\longrightarrow} \mathbf{fail} \rceil\!\|_{[\epsilon \ll o_2 \ll b]}$ is possible, which leads to the **fail** state. Note that the fail verdict is reached even if we omit divergence from the implementation o_0. This shows that Theorem 4 cannot be trivially generalized to the **ioco** setting (even when excluding divergence and allowing for non-input-enabled states).

Our main interest in this section is to investigate implementations for which **ioco** test cases cannot distinguish between synchronous and asynchronous modes of testing. To this end, we consider the relation between traces of a system and those of the system in queue context.

Definition 19 (Delay relation) *Let L be a finite alphabet partitioned in L_I and L_U. The* delay relation $@ \subseteq L_\delta^* \times L_\delta^*$ *is defined by the following deduction rules:*

$$\frac{}{\sigma \mathbin{@} \sigma}\ REF \qquad \frac{\rho_i, \sigma_i \in L_I^* \qquad \sigma_u \in L_U^*}{\rho_i\sigma_u\sigma_i \mathbin{@} \rho_i\sigma_i\sigma_u}\ PUSH \qquad \frac{\sigma \mathbin{@} \sigma' \qquad \rho @ \rho'}{\sigma\rho \mathbin{@} \sigma'\rho'}\ COM$$

Proposition 11 *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOTS. Let $s \in S$ and $\sigma \in L_\delta^*$. Then $\sigma \in \mathsf{Straces}(Q(s))$ implies there is a $\sigma' \in \mathsf{Straces}(s)$ such that $\sigma' \mathbin{@} \sigma$.*

Before we give the proof of the above proposition, we prove the lemmata given below. These allow us to establish links between traces in the synchronous and asynchronous settings.

Lemma 7 *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOTS, $s \in S$ and $\sigma \in L_\delta^*$. Then $\sigma \in \mathsf{Straces}(Q(s))$ implies that there is a $s' \in S$ such that $Q(s) \stackrel{\sigma}{\Longrightarrow}_\delta Q(s')$.*

Proof The proof is given by induction on the number of δ in $\sigma \in L_\delta^*$.

- Induction basis: Assume the number of δ in σ is 0, i.e., $\sigma \in L^*$. We distinguish between two cases based on whether $\sigma \in L_I^*$ and $\sigma \notin L_I^*$.
  1. Case $\sigma \in L_I^*$: Due to deduction rule $A1$ in Definition 17, it always holds that $Q(s) \stackrel{\sigma}{\Longrightarrow} {}_{[\epsilon \ll s \ll \sigma]}$. Since s is input-enabled, there is a state $s' \in S$ such that $s \stackrel{\sigma}{\Longrightarrow} s'$. By applying deduction rule I_2 several times, we have ${}_{[\epsilon \ll s \ll \sigma]} \stackrel{\epsilon}{\Longrightarrow} Q(s')$. We thus find that s' meets the required condition.
  2. Case $\sigma \notin L_I^*$: Let $\sigma = \sigma' x \rho$, with $\sigma' \in L^*$, $x \in L_U$ and $\rho \in L_I^*$. The appearance of x in trace $\sigma' x \rho$ can only be due to deduction rules $I3$ and $A2$ in Definition 17 and hence, we should have
  $$Q(s) \stackrel{\sigma_1}{\Longrightarrow} {}_{[\sigma_u \ll s_1 \ll \sigma_i]} \stackrel{\tau}{\longrightarrow} {}_{[\sigma_u x \ll s_2 \ll \sigma_i]} \stackrel{\sigma_2}{\Longrightarrow}$$
  $$ {}_{[x\sigma_v \ll s_3 \ll \sigma_j]} \stackrel{x}{\longrightarrow} {}_{[\sigma_v \ll s_3 \ll \sigma_j]} \stackrel{\rho}{\Longrightarrow} {}_{[\sigma_w \ll s'' \ll \sigma_k]}$$
  for $\sigma_w, \sigma_u, \sigma_v \in L_U^*$, $\sigma_k, \sigma_i, \sigma_j \in L_I^*$ and $s'', s_1, s_2, s_3 \in S$. We conclude from the last observation and deduction rules $A2$ in Definition 17 that σ_u must be the projection of σ_2 onto L_U^*. It follows from the last observation and deduction rules A_1 and A_2 that also the following derivation is possible, ${}_{[\sigma_u x \ll s_2 \ll \sigma_i]} \stackrel{\sigma_2 x \rho}{\Longrightarrow} {}_{[\epsilon \ll s_2 \ll \sigma_i \sigma_2' \rho]}$, where σ_2' is the projection of σ_2 onto L_I^*. Since, s_2 is input-enabled there is a state $s' \in S$ such that $s_2 \stackrel{\sigma_i\sigma_2'\rho}{\Longrightarrow} s'$. By using deduction rule I_2, we have ${}_{[\epsilon \ll s_2 \ll \sigma_i\sigma_2'\rho]} \stackrel{\sigma_i\sigma_2'\rho}{\Longrightarrow} Q(s')$. Thus s' meets the required condition.
- Inductive step: Assume that the statement holds for all $\sigma' \in L_\delta^*$ with at most $n-1$ occurrences of δ. Suppose the number of occurrences of δ in σ is n. Since $\sigma \in \mathsf{Straces}(Q(s))$, there exists a state $s'' \in S$ such that $Q(s) \stackrel{\sigma}{\Longrightarrow} {}_{[\sigma_u \ll s'' \ll \sigma_i]}$ for some $\sigma_i \in L_I^*$ and $\sigma_u \in L_U^*$. Assume $\sigma = \sigma_1 \delta \bar{\sigma}$ with $\sigma_1 \in L^*$ and $\bar{\sigma} \in L_\delta^*$. Due to Definition 6 the following step has to be taken in the former derivation, $Q(s) \stackrel{\sigma_1\delta}{\Longrightarrow}_\delta {}_{[\sigma_v \ll s_1 \ll \sigma_j]} \stackrel{\bar{\sigma}}{\Longrightarrow}_\delta$, where $\delta_q({}_{[\sigma_v \ll s_1 \ll \sigma_j]})$ for some $s_1 \in S$,$\sigma_v \in L_U^*$ and $\sigma_j \in L_I^*$. Note that σ_v has to be empty since quiescence has been observed beforehand. It follows from Definition 4 that σ_j has to be empty as well, since otherwise, ${}_{[\sigma_v \ll s_1 \ll \sigma_j]}$ can perform an internal transition, hence it cannot be quiescent. We thus find that $Q(s) \stackrel{\sigma_1\delta}{\Longrightarrow}_\delta Q(s_1) \stackrel{\bar{\sigma}}{\Longrightarrow}_\delta$ and s_1 is quiescent. We take the last transition of the previous derivation. It follows from the induction hypothesis that $\exists s' \in S$ such that $Q(s_1) \stackrel{\bar{\sigma}}{\Longrightarrow}_\delta Q(s')$. We thus conclude from the last observation that there is a state $s' \in S$ such that $Q(s) \stackrel{\sigma_1\delta}{\Longrightarrow}_\delta Q(s_1) \stackrel{\bar{\sigma}}{\Longrightarrow}_\delta Q(s')$ which was to be shown.

Lemma 8 *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOTS. Let $s \in S$ and $\sigma \in \mathsf{Straces}(Q(s))$. Then $Q(s) \stackrel{\sigma}{\Longrightarrow} Q(s')$ implies there is a $\sigma' \in \mathsf{Straces}(s)$ such that $s \stackrel{\sigma'}{\Longrightarrow} s'$ and $\sigma' \mathbin{@} \sigma$.*

Proof The proof is given by induction on the number of δ in $\sigma \in L^*_\delta$.

- Induction basis. Assume that there is no occurrence of δ, i.e., $\sigma \in L^*$. Thus, the thesis reduces to $\sigma \in \mathsf{Straces}(Q(s))$ and $\sigma \in L^*$ implies there is a $\sigma' \in \mathsf{traces}(s)$ such that $\sigma' \mathbin{@} \sigma$. We prove the latter by induction on the number of output actions in $\sigma \in L^*$.
  - Induction basis. Assume the number of output actions in σ is 0, i.e., $\sigma \in L^*_I$. By Proposition 7, we have $\sigma \in \mathsf{Straces}(Q(s))$, implying that $\exists s' \in S : Q(s) \stackrel{\sigma}{\Longrightarrow} Q(s')$. This derivation can only be done by applying deduction rules $A1$, $I2$ and maybe $I1$ in Definition 17 some times which result in $s \stackrel{\sigma}{\Longrightarrow} s'$ and subsequently $\sigma \in \mathsf{Straces}(s)$. Using deduction rule REF in Definition 19 results in $\sigma @ \sigma$. By assuming $\sigma' = \sigma$, it fulfills the two desired properties.
  - Inductive step. Assume that the statement holds for all $\sigma'' \in L^*$ with at most $n-1$ output actions. Suppose that the number of output actions of σ is n. Assume that $\sigma = \rho x \bar{\sigma}$ with $\rho \in L^*_I$, $x \in L_U$ and $\bar{\sigma} \in L^*$. We have $Q(s) \stackrel{\rho x \bar{\sigma}}{\Longrightarrow} Q(s')$, implying that somewhere in this derivation the step $s_1 \stackrel{x}{\longrightarrow} s_2$ is taken, for some $s_1, s_2 \in S$. This implies that there are two prefixes ρ_1 and ρ_2 of ρ such that ρ_2 is a prefix of ρ_1 as well and also $Q(s) \stackrel{\rho_1}{\Longrightarrow}_{[\epsilon \ll s_1 \ll \rho_1 \setminus \rho_2]} \stackrel{\tau}{\longrightarrow}_{[x \ll s_2 \ll \rho_1 \setminus \rho_2]} \stackrel{(\rho \setminus \rho_1) x \bar{\sigma}}{\Longrightarrow} Q(s')$. The last step of the previous derivation and deduction rule $A2$ in Definition 17 lead to $\stackrel{(\rho \setminus \rho_1)\bar{\sigma}}{\Longrightarrow}_{[\epsilon \ll s_2 \ll \rho_1 \setminus \rho_2]} Q(s')$. Since the input queue can be filled only under deduction rule $A1$ in Definition 17 that $Q(s_2) \stackrel{(\rho_1 \setminus \rho_2)(\rho \setminus \rho_1)\bar{\sigma}}{\Longrightarrow} Q(s')$. By defining $\sigma_1 = (\rho \setminus \rho_2)\bar{\sigma}$, we have $Q(s_2) \stackrel{\sigma_1}{\Longrightarrow} Q(s')$ with $\sigma_1 \in L^*$ and one output action less than n. It follows from induction hypothesis that $\exists \sigma'_1 \in \mathsf{Straces}(s_2) : s_2 \stackrel{\sigma'_1}{\Longrightarrow} s' \wedge \sigma'_1 @ \sigma_1$. We thus have $s \stackrel{\rho_2}{\Longrightarrow} s_1 \stackrel{x}{\longrightarrow} s_2 \stackrel{\sigma'_1}{\Longrightarrow} s'$ and subsequently, $\rho_2 x \sigma'_1 \in \mathsf{Straces}(s)$. By applying deduction rule REF and COM in Definition 19 respectively, we have $x\sigma'_1 @ x(\rho \setminus \rho_2)\sigma_1$. On the other hand, due to rule REF and COM we know that $x(\rho \setminus \rho_2)\sigma_1 @ (\rho \setminus \rho_2)x\sigma_1$ and consequently, $x\sigma'_1 @ (\rho \setminus \rho_2)x\sigma_1$. Deduction rule COM, the last observation and $\rho_2 @ \rho_2$ lead to $\rho_2 x \sigma'_1 @ \rho_2(\rho \setminus \rho_2)x\sigma_1$. By defining $\sigma' = \rho_2 x \sigma'_1$, we have $\sigma' @ \rho_2(\rho \setminus \rho_2)x\sigma'$ and more clearly, $\sigma' @ \sigma$. We thus find that σ' meets the two desired conditions.
- Inductive step. Assume the statement holds for all σ with at most $n-1$ occurrences of δ. Suppose there are n occurrences of δ in σ. Assume $\sigma = \sigma_1 \delta \bar{\sigma}$ with $\sigma_1 \in L^*$ and $\bar{\sigma} \in L^*_\delta$. By Proposition 7, we know from $\sigma \in \mathsf{Straces}(s)$ that there is a state $s' \in S$ such that $Q(s) \stackrel{\sigma_1 \delta \bar{\sigma}}{\Longrightarrow}_\delta Q(s')$. Due to Definition 4 and Definition 6, there exists a state $s_1 \in S$ such that $Q(s) \stackrel{\sigma_1 \delta}{\Longrightarrow}_\delta Q(s_1) \stackrel{\bar{\sigma}}{\Longrightarrow}_\delta Q(s')$ and $\delta(s_1)$. By taking the first transition of the previous derivation and induction basis, we find that there exists $\sigma'_1 \in \mathsf{Straces}(s)$ such that $s \stackrel{\sigma'_1}{\Longrightarrow} s_1$ and $\sigma'_1 @ \sigma$. From $\delta(s_1)$, we have $s \stackrel{\sigma'_1 \delta}{\Longrightarrow}_{\delta\delta} s_1$ and consequently by applying deduction rule COM in Definition 19, $\sigma'_1 \delta @ \sigma_1 \delta$ is concluded. Take then, the last transition of the first derivation i.e, $Q(s_1) \stackrel{\bar{\sigma}}{\Longrightarrow}_\delta Q(s')$ with $\bar{\sigma} \in L^*_\delta$ and the number of occurrences of δ is $n-1$ (one less than σ). By our induction hypothesis we find that there exists a $\bar{\sigma}' \in \mathsf{Straces}(s_1)$ such that $s_1 \stackrel{\bar{\sigma}'}{\Longrightarrow}_\delta s'$ and $\bar{\sigma}' @ \bar{\sigma}$. We thus have $\exists \sigma'_1 \in \mathsf{Straces}(s), \bar{\sigma}' \in \mathsf{Straces}(s_1) : s \stackrel{\sigma'_1 \delta}{\Longrightarrow}_\delta s_1 \stackrel{\bar{\sigma}'}{\Longrightarrow}_\delta s'$. By applying deduction rule COM to the first and second observation, i.e., $\sigma'_1 \delta @ \sigma_1 \delta$ and $\bar{\sigma}' @ \bar{\sigma}$, we have $\sigma'_1 \delta \bar{\sigma}' @ \sigma_1 \delta \bar{\sigma}$. By defining $\sigma' = \sigma'_1 \delta \bar{\sigma}'$ we find that σ' satisfies the two required properties.

We are now in a position to prove the correctness of the Proposition 11 as given below.

Proof Using the lemmata given above, the proof of the statement follows from the observations below. We have that $\sigma \in \mathsf{Straces}(Q(s))$, implying that $\exists s' \in S : Q(s) \stackrel{\sigma}{\Longrightarrow} Q(s')$, due to Lemma 7. It follows from the previous observation and Lemma 8 that $\exists \sigma' \in \mathsf{Straces}(s) : s \stackrel{\sigma'}{\Longrightarrow} s'$ and $\sigma' @ \sigma$ which was to be shown.

Definition 20 (Delay right-closed IOTS) *Let $\langle S, L, \rightarrow, s_0\rangle$ be an IOTS. A set $L' \subseteq L^*_\delta$ is* delay right-closed *iff for all $\sigma \in L'$ and $\sigma' \in L^*_\delta$, if σ @ σ' then $\sigma' \in L'$. The IOTS s_0 is* delay right-closed *iff* $\mathsf{Straces}(s_0)$ *is delay right-closed.*

We denote the class of delay right-closed IOTSs ranging over L_I and L_U by $\mathsf{IOTS}^{@}(L_I, L_U)$. The property below gives an alternative characterisation of delay right-closed IOTSs.

Property 4 Let $\langle I, L, \rightarrow, i_0\rangle$ be an IOTS. The IOTS i_0 is *delay right-closed* if for all $\sigma \in L^*_\delta$, all $x \in L_U$ and $a \in L_I$, we have:

$$\sigma x a \in \mathsf{Straces}(i_0) \text{ then } \sigma a x \in \mathsf{Straces}(i_0)$$

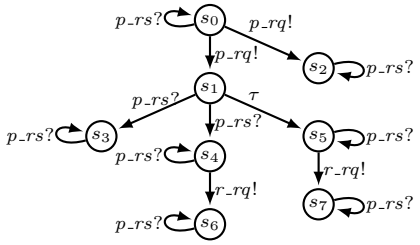


Fig. 8 A delay right-closed IOTS

Example 10 Consider the IOTS s_0 given in Figure 8. It is not hard to check that s_0 is delay right-closed.

As stated in the following theorem, the verdicts obtained by executing an arbitrary test case on a delay right-closed IOTS do not depend on the execution context. That is, the verdict does not change when the communication between the implementation and the test case is synchronous or asynchronous.

Theorem 5 *Let $\langle I, L, \rightarrow, i_0\rangle$ be a delay right-closed IOTS and let $\langle T, L', \rightarrow, t_0\rangle$ be an arbitrary test case. Then i_0* **passes** *t_0 iff $Q(i_0)$* **passes** *t_0.*

Before we address the proof of the above theorem, we first establish the correctness of the lemma below, stating that the suspension traces of a delay right-closed IOTS, as observed in an asynchronous setting are indistinguishable from the set of suspension traces observable in the synchronous setting.

Lemma 9 *Let $\langle S, L, \rightarrow, s_0\rangle$ be a delay right-closed IOTS. Then* $\mathsf{Straces}(Q(s_0)) = \mathsf{Straces}(s_0)$.

Proof We divide the proof obligation into two parts: $\mathsf{Straces}(Q(s_0)) \subseteq \mathsf{Straces}(s_0)$ and $\mathsf{Straces}(s_0) \subseteq \mathsf{Straces}(Q(s_0))$. It is not hard to verify that the latter holds vacuously, even for arbitrary IOTSs.

It therefore remains to show that $\mathsf{Straces}(Q(s_0)) \subseteq \mathsf{Straces}(s_0)$. Consider a $\sigma \in \mathsf{Straces}(Q(s_0))$; by Proposition 11, $\exists \sigma' \in \mathsf{Straces}(s_0) : \sigma' @ \sigma$. As s_0 is delay right-closed, we obtain the required $\sigma \in \mathsf{Straces}(s_0)$.

The above lemma is at the basis of the correctness of Theorem 5.

Proof (Theorem 5) Using the lemma given above, the proof of the theorem follows from the observation that for all test cases $\langle T, L', \rightarrow, t_0\rangle$ and all $\sigma \in L'^*$:

$$\begin{array}{l} \quad \exists i' \in I : t_0 \rceil i_0 \stackrel{\sigma}{\Longrightarrow} \mathbf{fail} \rceil i' \\ \text{iff} \\ \quad \exists i' \in I, \sigma_i \in L^*_I, \sigma_u \in L^*_U : t_0 \rceil Q(i_0) \stackrel{\sigma}{\Longrightarrow} \mathbf{fail} \rceil_{[\sigma_u \ll i' \ll \sigma_i]} \end{array}$$

Theorem 6 *Let $\langle I, L, \rightarrow, i_0 \rangle$ be a delay right-closed IOTS and let IOLTS $\langle S, L, \rightarrow, s_0 \rangle$ be a specification. Then i_0 $\mathbf{ioco}$ s_0 iff $Q(i_0)$ $\mathbf{ioco}$ s_0.*

Proof Follows from the existence of a sound and complete test suite that can test for **ioco**, and the proof of Theorem 5.

We now show that delayed right-closedness of implementations is also a necessary condition to ensure the same verdict in the synchronous and the asynchronous setting.

Theorem 7 *Let $\langle I, L, \rightarrow, i_0 \rangle$ be an IOTS. If for every test case $\langle T, L', \rightarrow, t_0 \rangle$, we have i_0 $\mathbf{passes}$ t_0 $\Leftrightarrow Q(i_0)$ $\mathbf{passes}$ t_0, then i_0 is a delay right-closed IOTS.*

Proof We prove the theorem by contraposition, i.e., we show that if we test a non-delay right-closed IOTS, there is a test case that can detect this by giving a *pass* verdict in the synchronous setting but a *fail* verdict in the asynchronous setting.

Let $\langle I, L, \rightarrow, i_0 \rangle$ be an IOTS that is not delay right-closed. Thus, there is some $x \in L_U$, $a \in L_I$ such that $\sigma x a \in \mathsf{Straces}(i_0)$, but not $\sigma a x \in \mathsf{Straces}(i_0)$. Let $\langle T, L', \rightarrow, t_0 \rangle$ be a test case such that there is a $t' \in T$ satisfying:

1. $t_0 \stackrel{\sigma}{\Longrightarrow} t'$,
2. $t' \stackrel{a}{\longrightarrow} t''$, and $t'' \stackrel{x}{\longrightarrow} \mathbf{fail}$.
3. for all σ' such that $t_0 \stackrel{\sigma'}{\Longrightarrow} \mathbf{fail}$ we have $\sigma' = \sigma a x$.

Observe that the existence of such a test case is immediate. Then there are $\sigma_i \in L_I^*$, $\sigma_u \in L_U^*$ and a state $i \in (i_0 \,\mathbf{after}\, \sigma)$ such that $t_0 \rceil\!\| Q(i_0) \stackrel{\sigma a x}{\Longrightarrow} \mathbf{fail} \rceil\!\|_{[\sigma_u \ll i_{\ll \sigma_i a]}}$, i.e., not $Q(i_0)$ $\mathbf{passes}$ t_0. However, we do not have $t_0 \rceil\!\| i_0 \stackrel{\sigma a x}{\Longrightarrow} \mathbf{fail} \rceil\!\| i$. By construction of the test case, we find that i_0 $\mathbf{passes}$ t_0.

7 Conclusions

In this paper, we studied the theoretical foundations for synchronous and asynchronous conformance testing. To this end, we gave unifying intensional and extensional definitions of conformance testing relations and compared them extensively. Subsequently, we presented theorems which allow for using test-cases generated from ordinary specifications in order to test asynchronous systems. These theorems establish sufficient conditions when the verdict reached by testing the asynchronous system (remotely, through FIFO channels) corresponds with the local testing through synchronous interaction. In the case of **ioco** testing theory, we show that the presented sufficient conditions are also necessary.

The presented conditions for synchronizing **ioco** are semantic in nature and we intend to formulate syntactic conditions that imply the semantic conditions presented in this paper. For example, it is interesting to find out which composition of programming constructs and / or patterns of interaction satisfy the constraints established in this paper. The research reported in this paper is inspired by our practical experience with testing asynchronous systems reported in [1]. We plan to apply the insights obtained from this theoretical study to our practical cases and find out to what extent the constraints of this paper apply to the implementation of our case studies.

Acknowledgments. We would like to thank Sjoerd Cranen (TU/e) and Maciej Gazda (TU/e) for their useful comments and suggestions.

References

1. H.R. Asadi, R. Khosravi, M.R. Mousavi, and N. Noroozi. Towards model-based testing of electronic funds transfer systems. In *Proc. of FSEN 2011*, LNCS. Springer, 2011.
2. C. Jard, T. Jéron, L. Tanguy, and C. Viho. Remote testing can be as powerful as local testing. In *Proc. of FORTE XII*, volume 156 of *IFIP Proc.*, pp. 25–40. Kluwer, 1999.
3. N. Noroozi, R. Khosravi,M.R. Mousavi,T.A.C. Willemse. Synchronizing Asynchronous Conformance Testing. Computer Science Report, No. 11-10, 16 pp. Eindhoven: Technische Universiteit Eindhoven, 2011.
4. N. Noroozi, R. Khosravi,M.R. Mousavi,T.A.C. Willemse. Synchronizing Asynchronous Conformance Testing. In *Proc. of SEFM 2011*, volume 7041 of *LNCS*, pp.334-349, Springer, 2011.
5. A. Petrenko and N. Yevtushenko. Queued testing of transition systems with inputs and outputs. In *Proc. of FATES 2002*, pp. 79–93, 2002.
6. A. Petrenko, N. Yevtushenko, and J. Huo. Testing transition systems with input and output testers. In *Proc. of Testcom 2003*, volume 2644 of *LNCS*, pp. 129–145. Springer, 2003.
7. A. Simao and A. Petrenko. From test purposes to asynchronous test cases. In *Proc. of ICSTW 2010*, pp. 1–10. IEEE CS, 2010.
8. J. Tretmans. *A formal Approach to conformance testing*. PhD thesis, Univ. of Twente, The Netherlands, 1992.
9. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software—Concepts and Tools*, 3:103–120, 1996.
10. J. Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *LNCS*, pp. 1–38. Springer, 2008.
11. J. Tretmans and L. Verhaard. A queue model relating synchronous and asynchronous communication. In *Proc. of PSTV'92*, vol. C-8 of *IFIP Tr.*, pp. 131-145. North-Holland, 1992.
12. L. Verhaard, J. Tretmans, P. Kars, and E. Brinksma. On asynchronous testing. In *Proc. of IWPTS'93*, volume C-11 of *IFIP Tr.*, pp. 55–66. North-Holland, 1993.
13. M. Weiglhofer. *Automated Software Conformance Testing*. PhD thesis, TU Graz, 2009.
14. M. Weiglhofer and F. Wotawa. Asynchronous input-output conformance testing. In *Proc. of COMPSAC'09*, pp. 154–159. IEEE CS, 2009.
15. M. Yannakakis and D. Lee. Testing of Finite State Systems In *Computer Science Logic*, volume 1584 of *LNCS*, pp. 29–44, Springer, 1999.