Synchronizing Asynchronous IOCO

Neda Noroozi[1,2], Ramtin Khosravi[3],
MohammadReza Mousavi[1], and Tim 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

Abstract. We present three theorems and their proofs which enable using synchronous testing techniques such 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 system 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 with the implementation under test (IUT). This fact has been noted by the pioneering pieces work in the area of formal conformance testing, e.g., see [6, Chapter 5], [9] and [10], and has been addressed extensively by several researchers in this field ever since [2–5, 11, 12].

A widely accepted model of asynchronous testing assumes a FIFO channel (or a number of them) as the asynchronous communication medium with the implementation under test and generates test-cases using the algorithm given for input-output conformance testing (IOCO) [7, 8], or a variant thereof. It is well-known that not all specifications are amenable to asynchronous testing since they may feature phenomena (e.g., an internal choice between accepting input and generating output) that cannot be reliably observed in the asynchronous setting (e.g., due to unknown delays in the asynchronous setting). In other words, in order to make sure that test-cases generated from the original specification can test the IUT by asynchronous interactions and reach verdicts that are meaningful for the original system, either the style of specification, or the test-case generation (or both) has to be adopted.

Related work In [11, Chapter 8] and [12], for example, both the class of specifications 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 **??** and report a slight adjustment to this result, without which a counter-example is shown to violate the property.

In [5] a method is presented for generating test-cases from the synchronous specification that are sound for the asynchronous implementation. The main idea is to take an IOCO test-case and delay observing output actions before the next observable quiescent and this will account for the delay caused by the asynchronous channel. The adjustment proposed in the our paper is inspired by a restriction imposed on asynchronous specifications in [5, Theorem 1].

In [3, 4] the asynchronous test framework is extended to the setting where separate test-processes can observer 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 [3, 4] since our ultimate goal is to compare asynchronous testing with the standard IOCO framework and the framework of [3, 4] 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 and non-theorem that show when it is (im)possible to synchronize asynchronous testing, i.e., use the (synchronous) IOCO test-case generation algorithm in order to test systems through asynchronous interaction. This guarantees that the verdict obtained from using IOCO on the synchronous specification coincides with the verdict obtained from using IOCO (or its asynchronous variant, in case of [11]), on the asynchronous version of the specification, which is equipped with input and output queues.

Structure of the paper To this end, after presenting some preliminaries in Section 2, we give a full proof of the main result of [11, Chapter 8] and [12] (with a slight modification) in Section 3. Then, in Section 4, we show that the results of re-formulate and strengthen the same results in the pure IOCO setting and show that even under weaker assumptions on the specification, one can recast the results in the IOCO setting. Finally, in Section 5, we show that the restriction imposed on the specification in Section 4 are not only sufficient to obtain the results but also necessary and hence characterize the specifications for which asynchronous testing can be reduced to synchronous testing. The paper is concluded in Section 6.

2 Preliminaries

In this section, we review some common formal definitions from the literature of labeled transition systems and input-output conformance testing [8].

Specifications, actions and traces Specifications in our approach to model-based testing are in the form of labeled transition systems (LTSs), defined below.

Definition 1 (LTS). *A labeled transition system (LTS) is a 4-tuple* $M = (Q, L \cup \{\tau\}, \rightarrow, q_0)$*, where* Q *is a set of states,* L *is a finite alphabet,* $\tau \notin L$ *is an unobservable action,* $\rightarrow \subseteq Q \times (L \cup \{\tau\}) \times Q$ *is the transition relation, and* $q_0 \in Q$ *is the initial state.*

In LTSs labels are treated uniformly while for testing, it is essential to distinguish input and output actions; this is achieved in the following definition of input-output labeled transition systems (IOLTSs).

Definition 2 (IOLTS). *An input-output labeled transition system (IOLTS) is an LTS* $M = (Q, L \cup \{\tau\}, \rightarrow, q_0)$*, where the alphabet* L *is partitioned into two sets* L_I *and* L_U*, representing the input and output alphabets, respectively. The class of IOLTSs with* L_I *and* L_U*, respectively, as the set of input- and output alphabets is denoted by* $IOLTS(L_I, L_U)$*.*

We write $q \xrightarrow{a} q'$ rather than $(q, a, q') \in\rightarrow$; moreover, we write $q \xrightarrow{a}$ when $q \xrightarrow{a} q'$ for some q, and $q \not\xrightarrow{a}$ when not $q \xrightarrow{a}$. The transition relation is generalized as follows:

Definition 3 ((Weak) Transitions and Traces). *Given an IOLTS* $(Q, L \cup \{\tau\}, \rightarrow, q_0)$ *and* $q, q', q_i \in Q, a, a_i \in L \cup \{\tau\}$ *and* $\sigma \in L^*$*, the notions of (weak) transition and trace are defined as follows.*

1. $q \overset{\epsilon}{\Longrightarrow} q' =_{def} (q = q') \vee \exists q_0, ..., q_n \bullet (q = q_0 \xrightarrow{\tau} q_1 \wedge ... \wedge q_{n-1} \xrightarrow{\tau} q_n = q')$
2. $q \overset{a}{\Longrightarrow} q' =_{def} \exists q_1, q_2 \bullet q \overset{\epsilon}{\Longrightarrow} q_1 \xrightarrow{a} q_2 \overset{\epsilon}{\Longrightarrow} q'$
3. $q \overset{a_1 ... a_n}{\Longrightarrow} =_{def} \exists q_0, ..., q_n \bullet q = q_0 \overset{a_1}{\Longrightarrow} q_1 ... q_{n-1} \overset{a_n}{\Longrightarrow} q_n = q'$

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

Definition 4 (Initial and After-States). *Given an IOLTS* $(Q, L \cup \{\tau\}, \longrightarrow, q_0)$*, some* $q \in Q$*,* $S \subseteq Q$*, we define:*

1. $\mathbf{init}(q) =_{def} \{a \mid q \xrightarrow{a}\}$*, and we set* $\mathbf{init}(S) =_{def} \bigcup_{q \in S} \mathbf{init}(q)$
2. $\mathbf{Sinit}(q) =_{def} \{a | q \overset{a}{\Longrightarrow}\}$*, and we define* $\mathbf{Sinit}(S) =_{def} \bigcup_{q \in S} \mathbf{Sinit}(q)$

A state in an LTS is said to *diverge* if it is the start of an infinite sequence of τ-labeled transitions.

Quiescence, defined below, is an essential notion for conformance testing; it characterizes a system state that cannot produce outputs and is stable, i.e., it cannot evolve to another state by performing a silent action.

Definition 5 (Quiescence (under queue context)).
A state $q \in Q$ *is called* quiescent*, denoted by* $\delta(q)$*, when* $\mathbf{init}(q) \subseteq L_I$*; it is called* quiescent under queue context*, denoted by* $\delta_q(q)$*, iff* $\mathbf{Sinit}(q) \subseteq L_I$*.*

The notion of quiescence under queue context refers to the asynchronous setting where quiescence cannot be observed directly (i.e., the tester cannot observe whether the system under test is engaged in some internal transitions or has come to a standstill) and hence the system is considered quiescent when it cannot show any observable output even after performing some internal transitions. In practice, this is implemented by a timeout mechanism which is set to the maximum waiting time before producing outputs. 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.

Next, we define the notion of test-case, which is a tree-shaped IOLTS prescribing when an input should be fed to the implementation under test and when its possible outputs should be observed leading to a leaf of the tree labeled with the pass- or the fail verdict. In a test case, the *observation* of quiescence is modeled using a θ symbol.

Definition 6 (Test case). *Given an input alphabet L_I and an output alphabet L_U, a test case t is an IOLTS $t = \langle Q, L_U \cup L_I \cup \{\theta\}, T, q_0 \rangle$, where Q is a finite set of states reachable from $q_0 \in Q$ containing two distinct terminal states* pass *and* fail, *θ is a fresh label ($\theta \notin L_I \cup L_U \cup \{\tau\}$), T is an acyclic and deterministic transition relation where pass and fail states appear only as targets of transitions labeled by an element of $L_U \cup \{\theta\}$ and for each non-terminal state $q \in Q$, it holds that $\mathbf{init}(q) = L_U \cup \{a\}$ for some $a \in L_I \cup \{\theta\}$.*

TW: this definition does not sound right to me; why omit θ in one case? after all, it is an observed output

The class of test cases for the sets of L_I of input labels and L_U of output labels, is denoted by $TTS(L_U, L_I)$ (note that the order of input and output are reversed: outputs of the system are inputs of the test-case and vice versa). A test suite T is defined a set of a test cases, i.e. $T \subseteq TTS(L_U, L_I)$.

Definition 7 (Synchronous execution). *Given a test case $t \in TTS(L_U, L_I)$, an IUT $i \in IOLTS(L_I, L_U)$, and let $a \in L$, then the synchronous test case execution operator $\rceil\!|$ is defined by the following inference rules:*

$$\frac{i \xrightarrow{\tau} i'}{t \rceil\!| i \xrightarrow{\tau} t \rceil\!| i'}\ \textit{(R1)} \qquad \frac{t \xrightarrow{a} t',\ i \xrightarrow{a} i'}{t \rceil\!| i \xrightarrow{a} t \rceil\!| i'}\ \textit{(R2)} \qquad \frac{t \xrightarrow{\theta} t',\ \delta(i)}{t \rceil\!| i \xrightarrow{\theta} t' \rceil\!| i}\ \textit{(R3)}$$

Definition 8 (passes). *Given an implementation $i \in IOLTS(L_I, L_U)$ and a test case $t \in TTS(L_U, L_I)$, then i* **passes** *$t \Leftrightarrow_{def} \forall \sigma \in (L \cup \{\theta\})^*, \forall i' \bullet t \rceil\!| i \overset{\sigma}{\not\Longrightarrow} fail \rceil\!| i'$.*

3 AIOCO Settings

In order to perform conformance testing in the asynchronous setting, in [11] and [12], both the class of specifications and test cases have been restricted to the so-called *internal choice* specifications. 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 [11] and [12], give full proof of their main result and point out a slight imprecision in it.

3.1 Internal Choice Specifications

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. Hence, if the specification produces different outputs based on the exact point of time when an input is consumed (so-called external-choice), the verdict of a synchronous tester may be different from that of an asynchronous one. Hence, in [11, 12], the class of specifications is restricted to those in which the choice about the exact moment of input is not determined by the tester but by the specification itself, leading to *internal-choice* specifications.

Example 1. Figure 1 shows the difference between internal and external choice specifications. In the IOLTS m_0, there is a race between input and output. Thus the tester controls the test execution and based on the exact moment of performing the input transition, can decide whether the appropriate output is produced or not (i.e., it rejects an implementation which produces a 1! output after feeding an a? to the system). This exact moment of time is not visible in asynchronous testing and hence, asynchronous testing may accept an implementation which is rejected by the synchronous testing (i.e., the tester may provide an a? to the system, but before its consumption, the system may produce a 1! output). Although IOLTS n_0 does not feature an immediate race between input and output actions, the tester can rule out the possibility of output 1! by providing input a?. Again this kind of control is not available to an asynchronous tester and hence this kind of specification falls beyond the internal-choice category. All other IOLTS's depicted in Figure 1 fall into the internal-choice category. For example, in the IOLTS starting from i_0, the tester can rule out the possibility of being in the initial state by observing quiescence; in that case the user can make sure that by providing an input to the system it will be consumed by the system and no output is allowed to be produced.

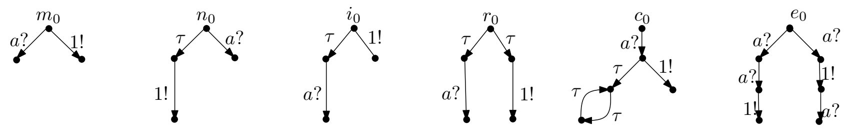

Fig. 1. Input-output labeled transition systems with different choice

These observations has led into the the following definition of internal-choice specification in [11, 12].

Definition 9. *A given LTS $M = (Q, L_I \cup L_U \cup \{\tau\}, \to, q_0)$ is an internal choice $LTS^{\sqcap}(L_I, L_U)$, if input actions are enabled only in quiescent states, i.e. $\forall q \in Q, (\exists a \in L_I \bullet a \in init(q))$ implies $\delta(q)$.*

By observing quiescence before any input, the tester will provide an input for the implementation only when it is ready to accept it. Hence implementation doesn't need to be assumed input enabled at any state. It is sufficient that implementation accept all inputs at states which the quiescence is observable.

we haven't said anything about input enabledness yet! So this paragraph had better to be rephrased, maybe

Definition 10. *An internal choice input output transition system ($IOTS^{\sqcap}(L_I, L_U)$) is an internal choice $IOLTS^{\sqcap}(L_I, L_U)$ where all input actions are enabled (possibly preceded by τ transitions) in quiescent states, $\forall a \in L_I, \forall q \in Q \bullet (\delta(q)$ implies $q \xrightarrow{a})$.*

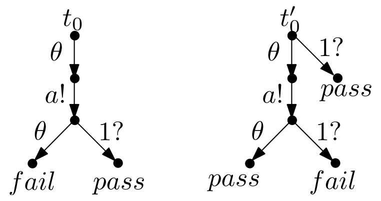

Fig. 2. An example of internal choice test case

Definition 11 (Internal choice test case). *An internal choice test case ($TTS^{\sqcap}$) is an TTS $M = \langle Q, L_U \cup L_I \cup \{\theta\}, T, q_0\rangle$, where any state $q \in Q$ with $init(q) \subseteq L_I$ is only reachable by θ-labeled transitions.*

Example 2. The left and right IOLTS's in figure 2 show internal test cases of $IOLTS^{\sqcap}(L_I, L_U)$ c_0 and i_0 in figure 1 respectively. Outputs (inputs of implementation) are enabled only in states reachable by θ-transitions.

Property 1. In an internal choice test execution, following properties always hold:

1. $t \xrightarrow{\sigma.a} t'$ and $a \in L_I$ implies that $\exists \sigma' \bullet \sigma = \sigma'.\theta$.
2. $t \rceil\!| i \xrightarrow{\sigma.a} t' \rceil\!| i'$ and $a \in L_I$ implies that $\exists \sigma' \bullet \sigma = \sigma'.\theta$.

3.2 Asynchronous Specifications

Asynchronous communication, as mentioned in [6, Chapter 5], can be simulated by composition of two FIFO channels with implementation, one for input and one for output. Tester puts its stimulus in the input queue and get the outputs of the implementation from the output queue. Also, communication between tester and implementation equipped with FIFO queues is done synchronously. In the following definition, the behavior of FIFO channels are formally defined.

Definition 12 (Queue operator). *Let $\sigma_i \in L_I^*$ and $\sigma_u \in L_U^*$, then the unary queue operator ${}_{[\sigma_u \ll}\cdot_{\ll \sigma_i]}$: $LTS(L_I, L_U) \rightarrow LTS(L_I, L_U)$ is defined by the following two axioms*

${}_{[\sigma_u \ll}S_{\ll \sigma_i]} \xrightarrow{a} {}_{[\sigma_u \ll}S_{\ll \sigma_i^\wedge a]}, a \in L_I$ *(A1)*

${}_{[x^\wedge \sigma_u \ll}S_{\ll \sigma_i]} \xrightarrow{x} {}_{[\sigma_u \ll}S_{\ll \sigma_i]}, x \in L_U$ *(A2)*

and by the following inference rules:

$$\frac{S \xrightarrow{\tau} S'}{{}_{[\sigma_u \ll}S_{\ll \sigma_i]} \xrightarrow{\tau} {}_{[\sigma_u \ll}S'_{\ll \sigma_i]}} \; (I1)$$

$$\frac{S \xrightarrow{a} S', a \in L_I}{{}_{[\sigma_u \ll}S_{\ll a^\wedge \sigma_i]} \xrightarrow{\tau} {}_{[\sigma_u \ll}S'_{\ll \sigma_i]}} \; (I2)$$

$$\frac{S \xrightarrow{x} S', x \in L_U}{{}_{[\sigma_u \ll}S_{\ll \sigma_i]} \xrightarrow{\tau} {}_{[\sigma_u^\wedge x \ll}S'_{\ll \sigma_i]}} \; (I3)$$

The initial state of a queue context containing an LTS S is given by $Q(S) =_{def} {}_{[\langle\rangle \ll}S_{\ll \langle\rangle]}$.

Property 2. Let $i, i' \in IOTS(L_I, L_U)$, $t, t' \in TTS(L_U, L_I)$ and $\sigma \in (L \cup \{\theta\})^*$ then,

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

Proposition 1. *For each $i, i' \in IOTS(L_I, L_U)$, $t \in TTS(L_U, L_I)$, $\sigma_i, \sigma_i' \in L_I$ and $\sigma_u, \sigma_u' \in L_U$ the following statements hold:*

1. $i \overset{\epsilon}{\Longrightarrow} i'$ *iff* $t \rceil\!| i \overset{\epsilon}{\Longrightarrow} t \rceil\!| i'$ *(R_1^*)*
2. ${}_{[\sigma_u \ll}S_{\ll \sigma_i]} \overset{\sigma_i'}{\Longrightarrow} {}_{[\sigma_u \ll}S_{\ll \sigma_i . \sigma_i']}, \sigma_i' \in L_I^*$ *(A_1^*).*
3. ${}_{[\sigma_u' . \sigma_u \ll}S_{\ll \sigma_i]} \overset{\sigma_u'}{\Longrightarrow} {}_{[\sigma_u \ll}S_{\ll \sigma_i]}, \sigma_u' \in L_U$ *(A_2^*).*
4. ${}_{[\sigma_u \ll}i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll}i'_{\ll \sigma_i]}$ *iff* $i \overset{\epsilon}{\Longrightarrow} i'$ *(I_1^*).*
5. ${}_{[\sigma_u \ll}i_{\ll \sigma_i . \sigma_i']} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll}i'_{\ll \sigma_i']}$ *iff* $i \overset{\sigma_i}{\Longrightarrow} i'$ *(I_2^*).*
6. ${}_{[\sigma_u \ll}i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u . \sigma_u' \ll}i'_{\ll \sigma_i]}$ *iff* $i \overset{\sigma_u'}{\Longrightarrow} i'$ *(I_3^*).*

Corollary 1. *For each $i, i' \in IOTS(L_I, L_U)$, $t, t' \in TTS(L_U, L_I)$, and $x \in \{L_U \cup \theta\}$, if $t \rceil\!| Q(i) \overset{x}{\Longrightarrow} t' \rceil\!| Q(i')$, then the following two statements hold:*

1. $t \xrightarrow{x} t'$ *and*
2. $Q(i) \overset{x}{\Longrightarrow} Q(i')$*. Moreover, if $x = \theta$, then $\delta_q(Q(i))$ and $\delta_q(Q(i'))$ are concluded.*

Corollary 2. *For each $i, i' \in IOTS(L_I, L_U)$ and $t, t' \in TTS(L_U, L_I)$, $t \rceil\!| Q(i) \overset{a}{\Longrightarrow} t' \rceil\!| Q(i') \wedge a \in L_I$ concludes the following:*

1. $t \xrightarrow{a} t'$
2. $Q(i) \overset{a}{\Longrightarrow} Q(i')$*. Moreover, if $i \in IOTS^{\sqcap}(L_I, L_U)$, then $\exists i'' \in IOTS^{\sqcap}(L_I, L_U) \bullet i \overset{\epsilon}{\Longrightarrow} i'' \overset{a}{\Longrightarrow} i' \wedge \delta(i'')$.*

3.3 Synchronizing Theorem for AIOCO

It is argued in [12, 5], if an input is provided to the IUT only after observing quiescence (i.e., in a stable state), the queue cannot distort the order of observations anymore. Hence, a subset of synchronous test-cases, namely those which only provide an input after observing quiescence, are sufficient for testing asynchronous systems. This is summarized in the following theorem from [12, 11] (and with a slightly different formulation in [5]):

Theorem 1 (non-theorem). *Let* $i \in IOTS^{\sqcap}(L_I, L_U)$ *and* $t \in TTS^{\sqcap}(L_U, L_I)$, *then* i **passes** $t \iff Q(i)$ **passes** t.

The theorem however, does not hold in its full generality as illustrated by the following example.

Example 3. Running test case t in figure 2 asynchronously with c= $IOTS^{\sqcap}(L_I, L_U)$ in figure 1 may result in fail, though c **passes** t. In fact $IOTS^{\sqcap}(L_I, L_U)$ c has a $\tau-$ loop which is considered quiescence in queue context($\delta_q(c)$).

By omitting diverging IOLTS from the internal choice specifications, we restrict the domain of the internal choice specifications to be able to prove the theorem given in [12, 11].

Theorem 2. *Let* $i \in IOTS^{\sqcap}(L_I, L_U)$ *and* $t \in TTS^{\sqcap}(L_U, L_I)$ *and* i doesn't diverge *then* i **passes** $t \iff Q(i)$ **passes** t.

Unfortunately, only a proof sketch is provided in [12, 11] for Theorem 1 and our order of business in this section is to give a full proof for its corrected version, Theorem 4. (In [5] only the theorem is mentioned without a formal proof.) To this end, we need a number of auxiliary lemmata and corollaries, given below.

In the following lemma, we show that if $\delta_q(Q(i))$, then either i is a quiescent state or i can reach a quiescent state by taking finite τ-steps. As we mentioned in Example 3, if i can diverge, then $Q(i)$ is considered quiescent, though i neither is quiescent or reach a quiescent state. This may lead to wrong verdict in the test execution.

Lemma 1. *Let* $i \in IOTS^{\sqcap}(L_I, L_U)$, *then* $\delta_q(Q(i))$ *implies that* $\exists i' \in IOTS^{\sqcap}(L_I, L_U)$ *such that* $i \overset{\epsilon}{\Longrightarrow} i' \wedge \delta(i')$

Proposition 2. *Let* $s, s' \in IOTS^{\sqcap}(L_I, L_U)$, $TTS^{\sqcap}$ $T^{\sqcap} = \langle Q, L_U \cup L_I \cup \{\theta\}, T, q_0\rangle, t, t' \in Q$ *and* $\sigma \in (L \cup \{\theta\})^*$. *Then* $t \rceil\!\rceil Q(s) \overset{\sigma}{\Longrightarrow} t' \rceil\!\rceil_{[\sigma_u \ll} s'_{\ll \sigma_i]}$ *implies that there exists a* $s'' \in IOTS^{\sqcap}(L_I, L_U)$ *such that* $t \rceil\!\rceil Q(s) \overset{\sigma}{\Longrightarrow} t' \rceil\!\rceil Q(s'')$

Corollary 3. *Let* $s, s' \in IOTS^{\sqcap}(L_I, L_U)$, $TTS^{\sqcap}$ $T^{\sqcap} = \langle Q, L_U \cup L_I \cup \{\theta\}, T, q_0\rangle, t, t' \in Q$ *and* $\sigma \in (L \cup \{\theta\})^*$. *Then* $t \rceil\!\rceil Q(s) \overset{\sigma}{\Longrightarrow} t' \rceil\!\rceil Q(s')$ *and* $\sigma = \sigma'.x$ *with* $\sigma \in (L \cup \{\theta\})^*$ *and* $x \in (L \cup \{\theta\})$ *implies that there exist a* $s'' \in IOTS^{\sqcap}(L_I, L_U)$ *and* $t'' \in Q$ *such that* $t \rceil\!\rceil Q(s) \overset{\sigma'}{\Longrightarrow} t'' \rceil\!\rceil Q(s'') \overset{x}{\Longrightarrow} t' \rceil\!\rceil Q(s')$.

Lemma 2. *Let* $s, s' \in IOTS^{\sqcap}(L_I, L_U)$, $TTS^{\sqcap}$ $T^{\sqcap} = \langle Q, L_U \cup L_I \cup \{\theta\}, T, q_0\rangle, t, t' \in Q$ *and* $\sigma \in (L \cup \{\theta\})^*$. *Then* $t \rceil\!\rceil Q(s) \overset{\sigma}{\Longrightarrow} t' \rceil\!\rceil Q(s')$ *implies that there exists a non-empty subset of* $(s' \,\mathbf{after}\, \epsilon)$ *like* $\mathbb{S}$ *such that* $\forall q \in \mathbb{S} \bullet t \rceil\!\rceil s \overset{\sigma}{\Longrightarrow} t' \rceil\!\rceil q \ \wedge \ \mathbf{Sinit}(t' \rceil\!\rceil Q(s')) = \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rceil\!\rceil q)$.

Proof. We prove this lemma by induction on the length of σ (excluding *tau*-transition). Assume for the induction basis that the length of σ is 0; thus it follows from the item 4 in Proposition 1 that $s \overset{\epsilon}{\Longrightarrow} s'$. We claim $\mathbb{S} = \{q | s' \overset{\epsilon}{\Longrightarrow} q\}$. It follows from the item 1 in Proposition 1 that $\forall q \in \mathbb{S}$ we have $t \rceil\!\rceil s \overset{\epsilon}{\Longrightarrow} t \rceil\!\rceil s'$ and $t \rceil\!\rceil s' \overset{\epsilon}{\Longrightarrow} t \rceil\!\rceil q$. Combination of the two transition culminate in $t \rceil\!\rceil s \overset{\epsilon}{\Longrightarrow} t \rceil\!\rceil q$ which is the first property of $\mathbb{S}$ and $\mathbf{Sinit}(t \rceil\!\rceil Q(s')) = \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t \rceil\!\rceil q)$ is concluded from item 2 in Definition 4 as well. Thus $\mathbb{S}$ holds the two properties which was to be shown. For the induction step, assume that the thesis holds for all σ with length $n-1$ or less and length of σ is n. It follows from the item 3 in Definition 3 and Corollary 3 that there exist $i_{n-1} \in IOTS^{\sqcap}(L_I, L_U)$, a $t'_{n-1} \in TTS^{\sqcap}(L_U, L_I)$ and $\sigma_{n-1} \in (L \cup \theta)^*$ and $x \in (L \cup \theta)$ such that $\sigma_n = \sigma_{n-1}.x$ and $t \rceil\!\rceil Q(s) \overset{\sigma_{n-1}}{\Longrightarrow} t'_{n-1} \rceil\!\rceil Q(i_{n-1}) \overset{x}{\Longrightarrow} t'_n \rceil\!\rceil Q(s')$. Induction hypothesis follows that $\exists \mathbb{S}_{n-1} \subseteq (i'_{n-1} \,\mathbf{after}\, \epsilon) \bullet (\forall q \in \mathbb{S}_{n-1} \bullet t \rceil\!\rceil s \overset{\sigma_{n-1}}{\Longrightarrow} t'_{n-1} \rceil\!\rceil q_{n-1})$ and $\mathbf{Sinit}(t'_{n-1} \rceil\!\rceil Q(i'_{n-1})) = \bigcup_{q \in \mathbb{S}_{n-1}} \mathbf{Sinit}(t'_{n-1} \rceil\!\rceil q)$. We distinguish three cases, based on the type of x: either it is θ, an input action or an output action.

$x = \theta$ Corollary 1 and $t'_{n-1} \rceil\!\rceil Q(i'_{n-1}) \overset{\theta}{\Longrightarrow}$ and Lemma 1 conclude that there exists an $i'' \in IOTS^{\sqcap}(L_I, L_U)$ such that $i'_{n-1} \overset{\epsilon}{\Longrightarrow} s' \overset{\epsilon}{\Longrightarrow} i''$ and i'' is quiescent. We claim that $\mathbb{S}_n = \{q \in (\mathbb{S}_{n-1} \,\mathbf{after}\, \epsilon) | s' \overset{\epsilon}{\Longrightarrow} q \wedge \delta(q)\}$. Definition of $\mathbf{Sinit}()$ (item 2 in Definition 4) results all member of $\mathbb{S}_{n-1}$ are either quiescent or leading to a quiescent state after some τ steps, thus $\mathbb{S}_n$ is not empty and also $\mathbb{S}_n \subseteq \mathbb{S}_{n-1}$. It is concluded from induction hypothesis that $\forall q \in \mathbb{S}_n \exists q' \in \mathbb{S}_{n-1} \bullet t \rceil\!\rceil s \overset{\sigma_{n-1}}{\Longrightarrow} t'_{n-1} \rceil\!\rceil q' \overset{\epsilon}{\Longrightarrow} t'_{n-1} \rceil\!\rceil q \xrightarrow{\theta} t' \rceil\!\rceil q$ is concluded. Combination of the two transition shows that $\forall q \in \mathbb{S}_n \bullet t \rceil\!\rceil s \overset{\sigma_n}{\Longrightarrow} t'_n \rceil\!\rceil q$. Hence the first property is held by $\mathbb{S}$. $Q(s')$ is quiescent $(\delta_q(Q(s')))$, thus $\mathbf{Sinit}(t' \rceil\!\rceil Q(s')) = L_I \cup \{\theta\}$. On the other hand, definition of $\mathbf{Sinit}()$ culminates in $\mathbf{Sinit}(t' \rceil\!\rceil q) = L_I \cup \{\theta\}$ for all member of $\mathbb{S}$. Thus $\mathbf{init}(t' \rceil\!\rceil Q(s')) = \bigcup_{q \in \mathbb{S}_n} \mathbf{init}(t' \rceil\!\rceil q)$ and the second property is held by $\mathbb{S}$ as well.

$x \in L_I$ It follows from induction hypothesis and Definition 10 that all member of $\mathbb{S}_{n-1}$ are quiescent, thus they can perform any input action. We claim that $\mathbb{S}_n = \{q | \ q' \overset{x}{\Longrightarrow} q \wedge \ q' \in \mathbb{S}_{n-1} \wedge s' \overset{\epsilon}{\Longrightarrow} q\}$. It follows from Corollary 1 that $\exists i'' \in IOTS^{\sqcap}(L_I, L_U) \bullet i_{n-1} \overset{\epsilon}{\Longrightarrow} i'' \overset{x}{\Longrightarrow} s' \wedge \delta(i'')$. According to the definitions of $\mathbb{S}_{n-1}$ and $\mathbb{S}_n$, it is clear that $i'' \in \mathbb{S}_{n-1}$ and $i'_n \in \mathbb{S}_n$, thus $\mathbb{S}_n$ cannot be empty and it has at least one member. Induction hypothesis results that $\forall q \in \mathbb{S}_n \exists q' \in \mathbb{S}_{n-1} \bullet t \rceil\!\rceil s \overset{\sigma_{n-1}}{\Longrightarrow} t'_{n-1} \rceil\!\rceil q' \overset{x}{\Longrightarrow} t' \rceil\!\rceil q$, thus the first property is held by $\mathbb{S}_n$. Since $s' \in \mathbb{S}_n$, it is clear that $\mathbf{Sinit}(t' \rceil\!\rceil s') \subseteq \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rceil\!\rceil q)$. For each member of $\mathbb{S}$ such as $q \in \mathbb{S}_n$, we have $\mathbf{Sinit}(t' \rceil\!\rceil q) \subseteq \mathbf{Sinit}(t' \rceil\!\rceil s')$, thus $\bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rceil\!\rceil q) \subseteq \mathbf{Sinit}(t' \rceil\!\rceil s')$ as well. These observations lead to $\mathbf{Sinit}(t' \rceil\!\rceil s') = \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rceil\!\rceil q)$. Property 2 leads to $\mathbf{Sinit}(t' \rrbracket Q(s')) = \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rrbracket q)$. Thus $\mathbb{S}_n$ holds the two required properties which was to be shown.

$x \in L_U$ We claim that $\mathbb{S}_n = \{q |\ q' \xrightarrow{x} q \wedge\ q' \in \mathbb{S}_{n-1} \wedge s' \xRightarrow{\epsilon} q\}$. It follows from the induction hypothesis that $\forall q \in \mathbb{S}_n \exists q' \in \mathbb{S}_{n-1} \bullet t \| s \xRightarrow{\sigma_{n-1}} t'_{n-1} \rrbracket q' \xRightarrow{x} t' \rrbracket q$. Hence $\mathbb{S}_n$ holds the first property. By Corollary 1 we know that $\exists i'' \in IOTS^{\sqcap}(L_I, L_U) \bullet i'_{n-1} \xRightarrow{\epsilon} i'' \xRightarrow{x} i'_n$, thus $i'' \in \mathbb{S}_{n-1}$ and subsequently $s' \in \mathbb{S}_n$. These results show that $\mathbb{S}$ is not empty and it has at least one member. Since $s' \in \mathbb{S}_n$, it is clear that $\mathbf{Sinit}(t' \rrbracket s') \subseteq \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rrbracket q)$. For each member of $\mathbb{S}$ such as $q \in$, we have $\mathbf{Sinit}(t' \rrbracket q) \subseteq \mathbf{Sinit}(t' \rrbracket s')$, thus $\bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rrbracket q) \subseteq \mathbf{Sinit}(t' \rrbracket s')$ as well. These observations lead to $\mathbf{Sinit}(t' \rrbracket s') = \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rrbracket q)$. Property 2 leads to $\mathbf{Sinit}(t' \rrbracket Q(s')) = \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rrbracket q)$. Thus $\mathbb{S}_n$ holds the two required properties which was to be shown.

Using the lemmas given above we are able to provide the proof of Theorem 4 as follows.

Proof of Theorem 4. We prove each implication by contradiction method.

$\Rightarrow$ Assume, towards a contradiction, that i **passes** t doesn't imply that $Q(i)$ **passes** t. It follows from Definition 8 that $\forall \sigma \in (L \cup \{\theta\})^* \forall i' \in IOTS^{\sqcap}(L_I, L_U) \bullet t \rrbracket i \overset{\sigma}{\not\Longrightarrow} \mathbf{fail} \rrbracket i'$ and $\exists \sigma' \in (L \cup \theta)^* \exists i'' \in IOTS^{\sqcap}(L_I, L_U) \bullet t \rrbracket Q(i) \xRightarrow{\sigma'} \mathbf{fail} \rrbracket i''$. Corollary 3 follows $t \rrbracket Q(i) \xRightarrow{\sigma''} t' \rrbracket Q(i') \xRightarrow{x} \mathbf{fail} \rrbracket i''$, with $\sigma'' \in (L \cup \{\theta\})^*$ and $\sigma' = \sigma''.x$. According to Definition 6, we know that a test case only by observing either an output action or θ reaches fail state, thus x is either an output or θ. Corollary 1 results in $t' \xrightarrow{x} \mathbf{fail}$. Lemma 2 concludes that $\exists \mathbb{S} \subseteq (i' \, \mathbf{after} \, \epsilon) \bullet \forall q \in \mathbb{S} \bullet t \rrbracket i \xRightarrow{\sigma''} t' \rrbracket q \wedge\ \mathbf{Sinit}(t' \rrbracket Q(i')) \subseteq \bigcup_{q \in \mathbb{S}} \mathbf{Sinit}(t' \rrbracket q)$. Thus there must exist an $s \in \mathbb{S}$ such that $t' \rrbracket s \xRightarrow{x}$. According to the previous observation $t \xrightarrow{x} \mathbf{fail}$. Hence there exists a path leading $t \rrbracket i$ to $\mathbf{fail} \rrbracket i'$ state and this is in contradictory to our assumption that i **passes** t.

$\Leftarrow$ proof of the left implication is almost identical to the right implication. Assume, towards a contradiction that $Q(i)$ **passes** t doesn't imply that i **passes** t. It follows from Definition 8 $\forall \sigma \in (L \cup \theta)^* \forall i' \in IOTS^{\sqcap}(L_I, L_U) \bullet t \rrbracket Q(i) \overset{\sigma}{\not\Longrightarrow} \mathbf{fail} \rrbracket i'\ \wedge\ \exists \sigma' \in (L \cup \theta)^* \exists i'' \in IOTS^{\sqcap}(L_I, L_U) \bullet t \rrbracket i \xRightarrow{\sigma'} \mathbf{fail} \rrbracket i''$. Property 2 shows the behavior of asynchronous context includes the behavior of synchronous context, thus $t \rrbracket i \xRightarrow{\sigma'} \mathbf{fail} \rrbracket i''$ results in $t \rrbracket Q(i) \xRightarrow{\sigma'} \mathbf{fail} \rrbracket Q(i'')$. Hence there is a path leading $t \rrbracket Q(i)$ to $\mathbf{fail} \rrbracket i'$ and this contradicts our assumption that $Q(i)$ **passes** t.

$\boxtimes$

4 IOCO Settings

In this section, we aim at re-casting the results of the previous section to the setting with the original IOCO test-case generation algorithm. We first define

IOCO and its test-case generation algorithm below and then show that the results of the previous section cannot be trivially generalized to the IOCO-setting. Then using an approach inspired by [6, Chapter 5] and [5], we show how to re-formulate Theorem in this setting.

4.1 Specifications and Test Cases

Input output labeled transition systems (IOLTS) are allowed to under-specify the behavior of a system; this is achieved by selectively omitting input actions in the behavioral model. The testing hypothesis underlying the IOCO theory, however, states that implementations are always *input enabled.* Formally, implementations range over *input-output transition systems*, formally defined below.

Definition 13 (IOTS). *An input-output transition system (IOTS) is an IOLTS in which all input actions are enabled (possibly preceded by τ-transitions) in all states. The IOTS subset of $IOLTS(L_I, L_U)$ is denoted by $IOTS(L_I, L_U)$.*

Informally, the *ioco* relation states that an implementation shown by an IOTS conforms a given specification modeled in IOLTS iff the observable behaviors of an implementation are also valid observable behaviors of the specification. In the context of IOCO, the observable behaviors are essentially traces, called *suspension traces*, consisting of inputs, outputs and observations of quiescence. For a given set of states Q and a transition relation $\rightarrow \subseteq Q \times (L \cup \{\tau\}) \times Q$, suspension traces are defined through an auxiliary transition relation $\Longrightarrow_\delta \subseteq Q \times (L \cup \{\delta\}) * \times Q$, specified through the following deduction rules:

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

Definition 14 (Suspension traces). *Assuming an IOLTS $(Q, L \cup \{\tau\}, \rightarrow, q_0)$, the suspension traces of state $q \in Q$ are $\mathsf{Straces}(q) =_{def} \{\sigma \in L_\delta^* | q \overset{\sigma}{\Longrightarrow}_\delta\}$.*

Definition 15. *Given an IOLTS $(Q, L \cup \{\tau\}, \longrightarrow, q_0)$, some $q \in Q$, $S \subseteq Q$, $a \in L$ and $\sigma \in (L \cup \{\delta\})^*$, we define:*

1. $\mathbf{out}(q) =_{def} \{a \in L_U | q \overset{a}{\longrightarrow}\} \cup \{\delta | \delta(q)\}$*, and we set* $\mathbf{out}(S) =_{def} \bigcup_{q \in S} \mathbf{out}(q)$
2. $q \,\mathbf{after}\, \sigma =_{def} \{q' | q \overset{\sigma}{\Longrightarrow}_\delta q'\}$*, and we set* $S \,\mathbf{after}\, \sigma =_{def} \bigcup_{q \in S} q \,\mathbf{after}\, \sigma$

Definition 16 (IOCO). *Let $i \in IOTS(L_I, L_U)$ and $s \in LTS(L_I, L_U)$ then $i \,\mathbf{ioco}\, s \Longleftrightarrow_{def} \forall \sigma \in \mathsf{Straces}(s) \bullet \mathbf{out}(i \,\mathbf{after}\, \sigma) \subseteq \mathbf{out}(s \,\mathbf{after}\, \sigma)$.*

Algorithm 17 (IOCO test case generation) *Let $s \in LTS(L_I, L_U)$ be a specification, and let S initially be $S = s \,\mathbf{after}\, \epsilon$.*
A test case $t \in TTS(L_U, L_I)$ is obtained from a non-empty set of states S by a finite number of recursive applications of one of the following three non-deterministic choices:

1. $t :=$***pass***

2. $t := a; t_a$
$\Box \Sigma\{x_j;\boldsymbol{fail} | x_j \in L_U, x_j \notin out(S)\}$
$\Box \Sigma\{x_i; t_{x_i} | x_i \in L_U, x_i \in out(S)\}$
where $a \in L_I$ *such that* $S\,\mathbf{after}\,a \neq \emptyset$, t_a *is obtained by recursively applying the algorithm for the set of states* $S\,\mathbf{after}\,a$, *and for each* $x_i \in out(S)$, t_{x_i} *is obtained by recursively applying the algorithm for the set of states* $S\,\mathbf{after}\,x_i$.

3. $t := \Sigma\{x_j;\boldsymbol{fail} | x_j \in L_U, x_j \notin out(S)\}$
$\Box \Sigma\{\theta;\boldsymbol{fail} | \delta \notin out(S)\}$
$\Box \Sigma\{x_i; t_{x_i} | x_i \in L_U, x_i \in out(S)\}$
$\Box \Sigma\{\theta; t_\theta | \delta \in out(S)\}$
where for each $x_i \in out(S)$, t_{x_i} *is obtained by recursively applying the algorithm for the set of states* $S\,\mathbf{after}\,x_i$ *and* t_θ *is obtained by recursively applying the algorithm for the set of states* $S\,\mathbf{after}\,\delta$.

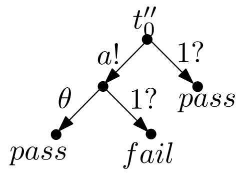

Fig. 3. An example of IOCO test case

Example 4. Figure 3 shows a test case for IOLTS i_0 in Figure 1 which is generated according to ioco test case generation algorithm(Algorithm 17). Although IOLTS i_0 is internal choice, test case t_0 in Figure 3 feeds input a to the implementation without observing quiescence despite of test case t_0' in Figure 2. Consider sequence $a?.1!$ which leads t_0 to $fail$ state. In queue context, the execution $t_0 \rrbracket Q(i_0) \xRightarrow{\epsilon} t_0 \rrbracket_{[1 \ll i_1 \ll \epsilon]} \xrightarrow{a?} t_1 \rrbracket_{[1 \ll i_1 \ll a]} \xrightarrow{1!} fail \rrbracket_{[\epsilon \ll i_1 \ll a]}$ is possible which leads to $fail$ state but $a?.1! \notin \mathsf{Straces}(t_0 \rrbracket e_0)$. Hence as we can see in this example, Theorem 4 cannot be generalized to the IOCO-setting.

4.2 Synchronizing Theorem for IOCO

In this section, we investigate specifications whose synchronous IOCO test cases are sound for asynchronous execution as well. To this end, we first consider the relation between traces of system and its queue context's. In fact, traces in queue context are reordered in respect to their original one by preceding input actions to output actions.

Definition 18 (Delay relation).

1. The relation $@ \subseteq L^* \times L^*$ *is defined as the smallest relation such that:*

(a) if $\sigma_1, \sigma_2 \in L_I^*$, then $\sigma_1 @ \sigma_2 =_{def} \sigma_1 \preceq \sigma_2$, where $\preceq$ denotes the trace-prefix pre-order,

(b) if $\sigma_1 = \rho_1.x_1.\sigma_1', \sigma_2 = \rho_2.x_2.\sigma_2'$, with $\rho_1, \rho_2 \in L_I^*, x_1, x_2 \in L_U$, and $\sigma_1', \sigma_2' \in L^*$, then $\sigma_1 @ \sigma_2 =_{def} \rho_1 \preceq \rho_2$ and $x_1 = x_2$ and $\sigma_1' @ (\rho_2 \setminus \rho_1).\sigma_2'$

2. if $\sigma_1 @ \sigma_2$, then the operation $\setminus\setminus$ is defined by

(a) if $\sigma_1, \sigma_2 \in L_I^*$, then $\sigma_2 \setminus\setminus \sigma_1 =_{def} \sigma_2 \setminus \sigma_1$

(b) if $\sigma_1 = \rho_1.x_1.\sigma_1', \sigma_2 = \rho_2.x_2.\sigma_2'$, with $\rho_1, \rho_2 \in L_I^*, x_1, x_2 \in L_U$, and $\sigma_1', \sigma_2' \in L^*$, then $\sigma_2 \setminus\setminus \sigma_1 =_{def} (\rho_2 \setminus \rho_1).\sigma_2' \setminus\setminus \sigma_1'$

Definition 19 (Fixed Delay relation). *The relation $|@| \subseteq L^* \times L^*$ is defined as if $\sigma_1, \sigma_2 \in L^*$ and $\sigma_1 |@| \sigma_2$ then $\sigma_1 @ \sigma_2$ and $|\sigma_1| = |\sigma_2|$.*

Corollary 4. *Let $S \in IOLTS(L_I, L_U)$ and $\sigma_1, \sigma_2 \in (L_I \cup L_U)^*$, then*

1. *$\sigma_1 @ \sigma_2$ and $\sigma_1 \in \mathsf{traces}(S)$ imply $\sigma_2 \in \mathsf{traces}(Q(S))$*
2. *$\sigma_1 @ \sigma_2$ and $\sigma_1 \in \mathsf{traces}(Q(S))$ imply $\sigma_2 \in \mathsf{traces}(Q(S))$*
3. *$\sigma_2 \in \mathsf{traces}(Q(S))$ implies that $\exists \sigma_1 \in \mathsf{traces}(S) \bullet \sigma_1 @ \sigma_2$*

Corollary 4.2 expresses that $\mathsf{traces}(Q(S))$ is right-closed with respect to relation @.

Definition 20 ((Fixed) Delay right-closure). *Set s is (fixed) delayed right-closed if $\sigma_1 @ \sigma_2$ ($\sigma_1 |@| \sigma_2$), with $\sigma_1, \sigma_2 \in L^*$ then $\sigma_1 \in S$ implies $\sigma_2 \in S$.*

Definition 21 ((Fixed) Delay right-closed IOLTS). *A given LTS $M = (Q, L_I \bigcup L_U \cup \{\tau, \delta\})$ is a (fixed) delay-right-closed $(LTS^{|@|}(L_I, L_U))LTS^{@}(L_I, L_U)$, if $\forall q \in Q \forall \sigma \in \mathsf{Straces}(q), (\exists x \in L_U, a \in L_I \bullet x \in \mathbf{init}(q\,\mathbf{after}\,\sigma) \wedge a \in \mathbf{init}(q\,\mathbf{after}\,\sigma))$ implies $\mathsf{Straces}((q\,\mathbf{after}\,\sigma))$ is (fixed) delay right-closed.*

Definition 22 ((Fixed) Delay right-closed IOTS). *A delay-right-closed input output transition system $IOTS^{@}(L_I, L_U)$, is a (fixed) delay-right-closed $(IOLTS^{|@|}(L_I, L_U))$ $IOLTS^{@}(L_I, L_U)$ where all input action is enabled (possibly preceded by τ-transition) in all states.*

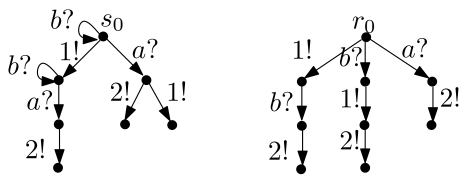

Fig. 4. Examples of delayed right-closed IOLTS

Example 5. Figure 4 shows two $IOLTS$ for $L_I = \{a, b\}$ and $L_U = \{1, 2\}$ as input and output alphabet respectively. In IOLTS s_0, $\{a, b\} \subseteq \mathbf{init}(s_0 \, \mathbf{after} \, \epsilon)$ and $x \in \mathbf{init}(s_0 \, \mathbf{after} \, \epsilon)$ as well. Also, $\mathsf{Straces}(s_0 \, \mathbf{after} \, \epsilon) = \{(b?)^*1!(b?)^*a?2!, (b?)^*a?1!, (b?)^*a?2!\}$ and due to Definition 20, $\mathsf{Straces}(s_0 \, \mathbf{after} \, \epsilon)$ is fixed delayed right-closed. Thus IOLTS s is fixed delay right-closed $IOLTS^{|@|}$. Similarly, IOLTS r is $IOLTS^{|@|}$ too.

As stated in the following theorem, the results of test execution of an IOCO test case with an $IOTS^{@}$ implementation are same in both synchronous and asynchronous environment.

Theorem 3. *$i \in IOTS^{@}, t \in TTS_{ioco}$ then i* **passes** *$t \Longleftrightarrow Q(i)$* **passes** *$t$.*

To prove this theorem we first show, as expressed in the following lemma, that i and its queue context, $Q(i)$ have a unique suspension traces set.

Lemma 3. *Let LTS $M = (Q, L_I \cup L_U \cup \{\tau\}) \subseteq IOTS^{@}(L_I, L_U)$ and $i \in Q$ and $\sigma \in (L_I \cup L_U)^*$, then $\sigma \in (StracesQ(i))$ implies $\sigma \in \mathsf{Straces}(i)$.*

Proof. The proof is given by induction on the number of output actions in σ. Assume, for the induction basis, $\sigma \in L_I^*$. It follows from Corollary 4.3 that there exists a $\sigma_1 \in \mathsf{Straces}(i)$ such that $\sigma_1 @ \sigma$. Due to Definition 18, $\sigma = \sigma_1.\rho_i$ with $\rho_i \in L_I^*$. Since i is input-enabled, $\rho_i \in \mathsf{Straces}(i \, \mathbf{after} \, \sigma_1)$. Thus $\sigma \in \mathsf{Straces}(i)$. For the induction step, assume that the thesis holds for all σ with $n - 1$ or less output actions. Suppose that $\sigma = \rho.x.\sigma'$ with $\rho \in L_I^*, x \in L_U, \sigma' \in L^*$ and the number of output actions in σ is equal to n. It follows from Corollary 4.3 there exists a $\sigma_1 \in \mathsf{Straces}(i)$ such that $\sigma_1 @ \sigma$. From Definition 18 we know that $\sigma_1 = \rho_1.x.\sigma_1'$ with $\rho_1 \in L_I^*, \sigma_1 \in L^*$ and $\sigma_1' @ (\rho \setminus \rho_1)\sigma'$. Distinguish between $\rho \setminus \rho_1 = \epsilon$ and $\rho \setminus \rho_1 \neq \epsilon$.

$\rho \setminus \rho_1 = \epsilon$ In this case, no input is preceded to output action x, thus there exists an i' such that $Q(i) \overset{\rho.x}{\Longrightarrow} Q(i') \overset{\sigma'}{\Longrightarrow}$. Thus $\sigma' \in \mathsf{Straces}(Q(i'))$ with one output action less than σ. It follows from induction hypothesis that $\sigma' \in \mathsf{Straces}(i')$. Also, it is concluded from the first observation($\rho.x$ is not a delayed sequence) that $i \overset{\rho.x}{\Longrightarrow} i'$. Thus $i \overset{\rho.x}{\Longrightarrow} i' \overset{\sigma'}{\Longrightarrow}$, or in other words $(\sigma = \rho.x\sigma') \in \mathsf{Straces}(i)$ which was to be shown.

$\rho \setminus \rho_1 \neq \epsilon$ The sequence σ can be written as $\rho_1(\rho \setminus \rho_1).x.\sigma'$. Since $\rho \setminus \rho_1 \neq \epsilon$, $\exists a \in L_I$ such that $\rho \setminus \rho_1 = a.\sigma_i^*$ with $\sigma_i^* \in L_I$. Thus $a \in \mathbf{init}(i \, \mathbf{after} \, \rho_1)$. It is concluded from sequence σ_1 that $x \in \mathbf{init}(i \, \mathbf{after} \, \rho_1)$. According to Definition 22, $(i \, \mathbf{after} \, \rho_1)$ is delay-closed-right. Thus $\sigma_1' \in \mathsf{Straces}(i \, \mathbf{after} \, \rho_1)$ implies if $\sigma_1' @ \sigma_d$ then $\sigma_d \in \mathsf{Straces}(i \, \mathbf{after} \, \rho_1)$. It follows from $\sigma_1' @ (\rho \setminus \rho_1)\sigma'$ that $x.\sigma_1' @ x.(\rho \setminus \rho_1)\sigma'$. From Definition 18 we know that $x.(\rho \setminus \rho_1)\sigma' @ (\rho \setminus \rho_1)x.\sigma'$ as well. Thus the two last observations and transitivity property of relation @, result $x.\sigma_1' @ (\rho \setminus \rho_1)x.\sigma'$. Hence $(\rho \setminus \rho_1)x.\sigma' \in \mathsf{Straces}(i \, \mathbf{after} \, \rho_1)$ which culminates in $\rho_1(\rho \setminus \rho_1)x.\sigma' \in \mathsf{Straces}(i)$. Thus $\sigma \in \mathsf{Straces}(i)$.

Proof of Theorem 3. Using the lemma given above, the proof of theorem becomes straightforward. We prove each implication separately.

- $\Rightarrow$ It follows from Lemma 3 that $\sigma \in \mathsf{Straces}(i)$ implies that $\sigma \in \mathsf{Straces}(Q(i))$. Thus if $\exists \sigma, q' \bullet t]\!|Q(i) \overset{\sigma}{\Longrightarrow} \mathsf{fail}]\!|q'$, it implies that $\exists i' \bullet t]\!|i \overset{\sigma}{\Longrightarrow} \mathsf{fail}]\!|i'$ and subsequently it results in i **passes** $t \Longrightarrow Q(i)$ **passes** t.
- $\Leftarrow$ Since $\mathsf{Straces}(i) \subseteq \mathsf{Straces}(Q(i))$, the left-to-right implication is very clear. If $\exists \sigma, i' \bullet t]\!|i \overset{\sigma}{\Longrightarrow} \mathsf{fail}]\!|i'$, it implies that $\exists q' \bullet t]\!|Q(i) \overset{\sigma}{\Longrightarrow} \mathsf{fail}]\!|q'$. Subsequently it results in $Q(i)$ **passes** $t \Longrightarrow i$ **passes** t.

$\boxtimes$

It is argued in [7, 8] that ioco test case generation is sound and exhaustive. Soundness means generated test cases can detect errors and exhaustiveness means at least theoretically they can detect all non-conforming implementations.

Definition 23 (Soundness and exhaustiveness). *Let $s \in IOLTS(L_I, L_U)$ and $T \subseteq TTS(L_U, L_I)$ then,*

1. *T is sound $\Longleftrightarrow_{def} \forall i \in IOTS(L_I, L_U) \bullet i$ **ioco** s implies i **passes** T.*
2. *T is sound $\Longleftrightarrow_{def} \forall i \in IOTS(L_I, L_U) \bullet i$ **ioco** s if i **passes** T.*

Theorem 4. *$s \in IOLTS^{@}, i \in IOTS^{@}$ then i **ioco** $s \Longleftrightarrow Q(i)$ **ioco** s.*

Proof. From Definition 23, we know that i **ioco** s iff i **passes** T with T is a test suite generated by Algorithm 17 from specification s. Also, it is concluded from Theorem 3 that i **passes** T implies that $Q(i)$ **passes** T. Thus due to Definition 23, the last observation results in $Q(i)$ **ioco** s.

Remark 1. It is noteworthy that i **ioco** s with $s \in IOLTS^{@}, i \in IOTS$ doesn't imply necessarily that $i \in IOTS^{@}$ and subsequently $Q(i)$ **ioco** s. But, if our specification is completely specified for input actions then the above implication will hold. In other words, i **ioco** s with $s \in IOTS^{@}, i \in IOTS$ does imply that $i \in IOTS^{@}$ and subsequently $Q(i)$ **ioco** s.

5 Necessary and Sufficient Condition

In the previous section, we have presented a class of implementation so-called delay right-closed whose synchronous and asynchronous test executions lead to a same verdict. We now show that being delay right-closed IOTS is necessary condition to have the same verdict simultaneously in synchronous and asynchronous execution, too.

Theorem 5. *Let $i \in IOTS(L_I, L_U)$ and $t \in TTS_{ioco}(L_I, L_U)$, then i **passes** $t \Longleftrightarrow Q(i)$ **passes** t implies $i \in IOTS^{@}(L_I, L_U)$.*

Proof. To prove the theorem given above, we prove $i \notin IOTS^{@}(L_I, L_U)$ implies i **passes** $t \not\Longrightarrow Q(i)$ **passes** t. Without loss of generality we assume that there exists a $\sigma \in (L \cup \{\tau, \delta\})^*$ such that i **after** σ is both input and output enabled. In general we can formulate it as $\exists q_1, q_2 \in (i\, \textbf{after}\, \sigma), q'_1, q'_2, a \in L_i, x \in L_U \bullet q_1 \xrightarrow{a} q'_1 \wedge q_2 \xrightarrow{x} q'_2$. It is concluded from $i \notin IOTS^{@}(L_I, L_U)$ that $\mathsf{Straces}(i\, \textbf{after}\, \sigma)$ is not delay-right-closed. Thus without loss of generality we can assume that $x.a \in \mathsf{Straces}(i\,\mathbf{after}\,\sigma)$ as its delayed sequence $a.x \notin \mathsf{Straces}(i\,\mathbf{after}\,\sigma)$. According to Algorithm 17, there exists a state t' at which the tester can either provide the input a or observe the output x nondeterministically($\exists t' \bullet t' \xrightarrow{a} \wedge t' \xrightarrow{x}$) and from the previous assumption we know observing the output x after providing a leads to $fail$ state. Consider a situation in which the sequence of σ is executed during the test execution and t reaches t'. Then the tester provides an input a being put in the input queue as the implementation under the test, produces an output x being put in the output queue, according to rule $A1$ and $I3$ in Definition 12 respectively. Thus the implementation during this execution reaches the state ${}_{[x\ll}q'_{2\ll a]}$. According to rule $A2$ in Definition 12, the output transition of x can happen which leads the tester to $fail$ state which was to be shown.

6 Conclusions

In this paper, 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 semantical in nature. We intend to formulate syntactic conditions that imply the semantical conditions presented 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 improve our practical results.

References

1. HamidReza Asadi, Ramtin Khosravi, MohammadReza Mousavi, and Neda Noroozi. Towards model-based testing of electronic funds transfer systems. In *Proceedings of the 4th International on Fundamentals of Software Engineering (FSEN 2011)*, Lecture Notes in Computer Science. Springer, 2011.
2. Claude Jard, Thierry Jéron, Lénaick Tanguy, and César Viho. Remote testing can be as powerful as local testing. In *IFIP Joint Conferences: FORTE XII) and (PSTV XIX)*, volume 156 of *IFIP Conference Proceedings*, pages 25–40. Kluwer, 1999.
3. Alexandre Petrenko and Nina Yevtushenko. Queued testing of transition systems with inputs and outputs. In *Proceedings of the Workshop on Formal Approaches to Testing of Software FATES 2002*, pages 79–93, 2002.
4. Alexandre Petrenko, Nina Yevtushenko, and Jiale Huo. Testing transition systems with input and output testers. volume 2644 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2003.
5. Adenilso Simao and Alexandre Petrenko. From test purposes to asynchronous test cases. In *Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW 2010)*, pages 1–10. IEEE CS, 2010.

6. Jan Tretmans. *A formal Approach to conformance testing.* PhD thesis, University of Twente, The Netherlands, 1992.
7. Jan Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software—Concepts and Tools*, 3:103–120, 1996.
8. Jan Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008.
9. Jan Tretmans and Louis Verhaard. A queue model relating synchronous and asynchronous communication. In *Proceedings of the IFIP Symposium on Protocol Specification, Testing and Verification*, volume C-8 of *IFIP Transactions*, pages 131–145. North-Holland, 1992.
10. Louis Verhaard, Jan Tretmans, Pim Kars, and Ed Brinksma. On asynchronous testing. In *Proceedings of the IFIP Workshop on Protocol Test Systems*, volume C-11 of *IFIP Transactions*, pages 55–66. North-Holland, 1993.
11. Martin Weiglhofer. *Automated Software Conformance Testing.* PhD thesis, Graz University of Technology, Austria, 2009.
12. Martin Weiglhofer and Franz Wotawa. Asynchronous input-output conformance testing. In *Proceedings of the International Computer Software and Applications Conference (COMPSAC'09)*, pages 154–159. IEEE Computer Society, 2009.

Appendix

Proof of Proposition 3.1.

1. $i \xRightarrow{\epsilon} i'$ iff $t \rceil\!\rceil i \xRightarrow{\epsilon} t \rceil\!\rceil i'$ (R_1^*)
   We prove the two implications by a straightforward induction on the length of the τ-traces leading to $\xRightarrow{\epsilon}$, see item 1 in Definition 3:
   - ⇒ Assume, for the induction basis, that $i \xRightarrow{\epsilon} i'$ is due to a τ-trace of length 0; thus, it follows from the first disjunct of item 1 in Definition 3 that $i = i'$. It then follows from the same item that $t \rceil\!\rceil i \xRightarrow{\epsilon} t \rceil\!\rceil i$ and since $i = i'$, we have that $t \rceil\!\rceil i \xRightarrow{\epsilon} t \rceil\!\rceil i'$, which was to be shown.
     For the induction step, assume that the thesis holds for all $\xRightarrow{\epsilon}$ 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 $t \rceil\!\rceil i \xRightarrow{\epsilon} t \rceil\!\rceil i_{n-1}$. Also from $i_{n-1} \xrightarrow{\tau} i'$ and deduction rule *R1* in Definition 7, we have that $t \rceil\!\rceil i_{n-1} \xRightarrow{\epsilon} t \rceil\!\rceil i'$. Hence, it follows from item 1 in Definition 3 that $t \rceil\!\rceil i \xRightarrow{\epsilon} t \rceil\!\rceil i'$, which was to be shown.
   - ⇐ 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 $t \rceil\!\rceil i_{n-1} \xRightarrow{\epsilon} t \rceil\!\rceil i'$ can only be due to deduction rule *R1* and hence we have $i_{n-1} \xRightarrow{\epsilon} i'$, which in turn implies, by using item 1 in Definition 3, that $i \xRightarrow{\epsilon} i'$.
2. ${}_{[\sigma_u \ll}S_{\ll \sigma_i]} \xRightarrow{\sigma_i'} {}_{[\sigma_u \ll}S_{\ll \sigma_i . \sigma_i']}, \sigma_i' \in L_I^*$ (A_1^*).
3. ${}_{[\sigma_u' . \sigma_u \ll}S_{\ll \sigma_i]} \xRightarrow{\sigma_u'} {}_{[\sigma_u \ll}S_{\ll \sigma_i]}, \sigma_u' \in L_U$ (A_2^*).

4. ${}_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll} i'_{\ll \sigma_i]}$ iff $i \overset{\epsilon}{\Longrightarrow} i' (I_1^*)$. Almost identical to the previous item: we prove the two implications by induction on the length of the τ-trace for leading to $\overset{\epsilon}{\Longrightarrow}$:
   - $\Rightarrow$ Assume, for the induction basis, that $i \overset{\epsilon}{\Longrightarrow} i'$ is due to a τ-trace of length 0; thus, it follows from the first disjunct of item 1 in Definition 3 that $i = i'$. It then follows from deduction the same item that ${}_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll} i_{\ll \sigma_i]}$ and since $i = i'$, we have that ${}_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll} i'_{\ll \sigma_i]}$, which was to be shown.
     For the induction step, assume that the thesis holds for all $\overset{\epsilon}{\Longrightarrow}$ 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 ${}_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll} i_{n-1 \ll \sigma_i]}$. Also from $i_{n-1} \xrightarrow{\tau} i'$ and deduction rule *I1* in Definition 12, we have that ${}_{[\sigma_u \ll} i_{n-1 \ll \sigma_i]} \xrightarrow{\tau} {}_{[\sigma_u \ll} i'_{\ll \sigma_i]}$.Hence, it follows from item 1 in Definition 3 that ${}_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll} i'_{\ll \sigma_i]}$, which was to be shown.
   - $\Rightarrow$ Similar to the above item. The induction basis is identical. The induction step follows from the same reasoning. Note that ${}_{[\sigma_u \ll} i_{n-1 \ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll} i'_{\ll \sigma_i]}$ can only be proven using deduction rule *I1* in Definition 12, 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, $i_{n-1} \xrightarrow{\tau} i'$. Hence, using the induction hypothesis we obtain that $i \overset{\epsilon}{\Longrightarrow} i'$.
5. ${}_{[\sigma_u \ll} i_{\ll \sigma_i . \sigma_i']} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll} i'_{\ll \sigma_i']}$ iff $i \overset{\sigma_i}{\Longrightarrow} i' (I_2^*)$.
6. ${}_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u . \sigma_u' \ll} i'_{\ll \sigma_i]}$ iff $i \overset{\sigma_u'}{\Longrightarrow} i' (I_3^*)$.

$\boxtimes$

Lemma 4. *For each $i, i' \in IOTS^{\sqcap}(L_I, L_U)$ and $TTS^{\sqcap}$ $T^{\sqcap} = \langle Q, L_U \cup L_I \cup \{\theta\}, T, q_0 \rangle, t, t' \in Q$ and $\sigma_u \in L_U^*$ and $\sigma_i \in L_I^*$, if $t \rceil\!\rceil_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{a}{\Longrightarrow} t' \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i \hat{} a]}$, then ${}_{[\sigma_u \ll} i'_{\ll \sigma_i]}$ is quiescent($\delta_q({}_{[\sigma_u \ll} i'_{\ll \sigma_i]})$).*

Proof of Lemma 4. Assume $a \in L_I$ and $t \rceil\!\rceil_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{a}{\Longrightarrow} t' \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i \hat{} a]}$, from Definition 3 (item 2), we know there exists an $i'' \in IOTS^{\sqcap}(L_I, L_U)$ such that $t \rceil\!\rceil_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} t \rceil\!\rceil_{[\sigma_u \ll} i''_{\ll \sigma_i]} \xrightarrow{a} t' \rceil\!\rceil_{[\sigma_u \ll} i''_{\ll \sigma_i \hat{} a]} \overset{\epsilon}{\Longrightarrow} t' \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i \hat{} a]}$. It follows from item 4 in Proposition 1 and $t \rceil\!\rceil_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} t \rceil\!\rceil_{[\sigma_u \ll} i''_{\ll \sigma_i]}$ that $i \overset{\epsilon}{\Longrightarrow} i''$. Also it is concluded from Proposition 1(item 4) and $t' \rceil\!\rceil_{[\sigma_u \ll} i''_{\ll \sigma_i \hat{} a]} \overset{\epsilon}{\Longrightarrow} t' \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i \hat{} a]}$ that $i'' \overset{\epsilon}{\Longrightarrow} i'$. Thus, according to item 1 in Definition 3, $i \overset{\epsilon}{\Longrightarrow} i'$ and subsequently according to Proposition 1(item 4), ${}_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma_u \ll} i'_{\ll \sigma_i]}$. The former observation and item 1 in Proposition 1 lead to $t \rceil\!\rceil_{[\sigma_u \ll} i_{\ll \sigma_i]} \overset{\epsilon}{\Longrightarrow} t \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i]}$. Using deduction rule *A1* in Definition 12 and applying deduction rule *R2* in Definition 7 result in $t \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i]} \overset{a}{\Longrightarrow} t' \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i \hat{} a]}$. Hence, there is a trace starting from $t \rceil\!\rceil_{[\sigma_u \ll} i_{\ll \sigma_i]}$ to $t \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i]} \overset{a}{\Longrightarrow} t' \rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i \hat{} a]}$. It follows then from Definition 11 that $\delta_q({}_{[\sigma_u \ll} i'_{\ll \sigma_i]})$ (since test case t can only provide an input if it has observed quiescent, by looking into all future traces), which was to be shown. $\boxtimes$

Lemma 5. *For each $i, i' \in IOTS^{\sqcap}(L_I, L_U)$ and $TTS^{\sqcap}$ $T^{\sqcap} = \langle Q, L_U \cup L_I \cup \{\theta\}, T, q_0\rangle, t, t' \in Q$, there is no trace $\sigma \in (L_I \cup L_U \cup \{\theta\})^*$ such that $t\rceil\!\rceil Q(i) \overset{\sigma}{\Longrightarrow} t'\rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i]}$ and $(\sigma_i \neq \epsilon \wedge \sigma_u \neq \epsilon)$.*

Proof of Lemma 5. Assume, towards a contradiction, that the following items hold:

1. $t\rceil\!\rceil Q(i) \overset{\sigma}{\Longrightarrow} t'\rceil\!\rceil_{[\sigma_u \ll} i'_{\ll \sigma_i]}$
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 $\sigma', \sigma'' \in (L_I \cup L_U \cup \theta)^*$ and $y \in (L_I \cup L_U \cup \{\tau, \theta\})$ and a postfix σ'' of σ such that:

1. $\sigma = \sigma'.y.\sigma''$
2. there exist $\sigma'_i \in (L_I)^*, \sigma'_u \in (L_U)^*$ such that $t\rceil\!\rceil Q(i) \overset{\sigma'}{\Longrightarrow} t_1\rceil\!\rceil_{[\sigma'_u \ll} i_1{}_{\ll \sigma'_i]} \wedge ((\sigma'_u = \epsilon \wedge \sigma'_i \neq \epsilon) \vee (\sigma'_i = \epsilon \wedge \sigma'_u \neq \epsilon))$
3. there exist $\sigma''_i \in (L_I)^*, \sigma''_u \in (L_U)^*$ such that $t_1\rceil\!\rceil_{[\sigma'_u \ll} i_1{}_{\ll \sigma'_i]} \xrightarrow{y} t_2\rceil\!\rceil_{[\sigma''_u \ll} i_2{}_{\ll \sigma''_i]} \wedge ((\sigma'_u = \epsilon \wedge \sigma'_i \neq \epsilon \wedge \sigma''_u \neq \epsilon \wedge \sigma''_i = \sigma'_i) \vee (\sigma'_i = \epsilon \wedge \sigma'_u \neq \epsilon \wedge \sigma''_i \neq \epsilon \wedge \sigma''_u = \sigma'_u))$
4. $t_2\rceil\!\rceil_{[\sigma''_u \ll} i_2{}_{\ll \sigma''_i]} \overset{\sigma''}{\Longrightarrow} t\rceil\!\rceil_{[\sigma_u \ll} i_{\ll \sigma_i]}$

Note that after σ' both input and output queues cannot be empty, since a single transition y can at most increment the size of one of the two queues (see rules *A1* and *I3* in Definition 12). Below, we distinguish two cases based on the status after performing the trace σ': either the input queue is empty (and the output queue is not), or the other way around.

$\sigma'_u = \epsilon$ The only possible transition that can fill an output queue is due to the application of deduction rule *I3* in Definition 12. Hence, there must exists some $i_2 \in LTS^{\sqcap}(L_I, L_U)$ and $x \in L_U$ such that ${}_{[\epsilon \ll} i_1{}_{\ll \sigma'_i]} \xrightarrow{\tau} {}_{[x \ll} i_2{}_{\ll \sigma'_i]}$ and subsequently, $(t_1\rceil\!\rceil_{[\epsilon \ll} i_1{}_{\ll \sigma'_i]} \xrightarrow{\tau} t_2\rceil\!\rceil_{[x \ll} i_2{}_{\ll \sigma'_i]})$ (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 12 and hence, we have $i_1 \xrightarrow{x} i_2$. However, it follows from $\sigma'_i \neq \epsilon$ that there exit an $a \in L_I$, $i_p \in IOTS^{\sqcap}(L_I, L_U)$, a prefix of σ' like σ'_p and $\rho_i \in L_I^*$ such that $\sigma'_i = \rho_i.a$ and $t\rceil\!\rceil Q(i) \overset{\sigma'_p}{\Longrightarrow} t'_1\rceil\!\rceil_{[\epsilon \ll} i_p{}_{\ll \rho_i]} \overset{a}{\Longrightarrow} t_1\rceil\!\rceil_{[\epsilon \ll} i_1{}_{\ll \sigma'_i]}$. Since, $i \in IOTS^{\sqcap}(L_I, L_U)$, we have from Lemma 4 that $\delta_q({}_{[\epsilon \ll} i_1{}_{\ll \rho_i]})$. Using deduction rule *A2* on $i_1 \xrightarrow{x} i_2$, we obtain that ${}_{[\epsilon \ll} i_1{}_{\ll \rho_i]} \overset{\epsilon}{\Longrightarrow} {}_{[x \ll} i_2{}_{\ll \rho_i]}$. Hence according to Definition 5, state ${}_{[\epsilon \ll} i_1{}_{\ll \rho_i]}$ is not quiescent, which is contradictory to our earlier observation about $\delta_q({}_{[\epsilon \ll} i_1{}_{\ll \rho_i]})$.

$\sigma'_i = \epsilon$ The only transition which allows for filling the input queue is due the subsequent application of deduction rules R2 and A1. Hence, there exists an $a \in L_I$, such that $t_1\rceil\!\rceil_{[\sigma'_u \ll} i_1{}_{\ll \epsilon]} \xrightarrow{a} t_2\rceil\!\rceil_{[\sigma'_u \ll} i_2{}_{\ll a]})$ and ${}_{[\sigma'_u \ll} i_1{}_{\ll \epsilon]} \xrightarrow{a} {}_{[\sigma'_u \ll} i_2{}_{\ll a]}$

(where the former satisfies the third item by taking $\sigma_i' = \epsilon$ and $\sigma_i'' = a$); It follows from $i \in IOTS^{\sqcap}(L_I, L_U)$, and Lemma 4 that $\delta_q({}_{[\sigma_u' \ll} i_2 {}_{\ll \epsilon]})$. 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 ${}_{[\sigma_u' \ll} i_2 {}_{\ll \epsilon]} \xrightarrow{x}$ and thus, ${}_{[\sigma_u' \ll} i_2 {}_{\ll \epsilon]}$ is not quiescent, which is contradictory to our earlier observation.

⊠

Lemma 6. *For each $i, i' \in IOTS^{\sqcap}(L_I, L_U)$, $TTS^{\sqcap}$ $T^{\sqcap} = \langle Q, L_U \cup L_I \cup \{\theta\}, T, q_0 \rangle, t, t' \in Q$ and $\sigma, \sigma_u, \sigma_i \in \{L_I \cup L_U \cup \theta\}^*$ and $\sigma_i \neq \epsilon$. If $t \rrbracket Q(i) \overset{\sigma}{\Longrightarrow} t' \rrbracket_{[\sigma_u \ll} i' {}_{\ll \sigma_i]}$ then $\delta_q(i')$ and $\sigma_u = \epsilon$.*

Proof of Lemma 6. By lemma 5, we have that $\sigma_u = \epsilon$. Assume, towards a contradiction that there exists an $x \in L_U$ such that $x \in \textbf{Sinit}(i')$. Since $x \in \textbf{Sinit}(i')$, it follows from item 2 in Definition 4 that there exists an $i'' \in IOTS^{\sqcap}(L_I, L_U)$ such that $i' \overset{x}{\Longrightarrow} i''$. Since $\sigma_i \neq \epsilon$ there exist $\sigma' \in \{L_I \cup L_U \cup \theta\}^*$, $i_p \in IOTS^{\sqcap}(L_I, L_U)$, $t_p \in TTS^{\sqcap}(L_U, L_I)$, $a \in L_I$, and $\rho_i \in L_I^*$ such that $\sigma_i = \rho_i.a$ and $t \rrbracket Q(i) \overset{\sigma'}{\Longrightarrow} t_p \rrbracket_{[\epsilon \ll} i_p {}_{\ll \rho_i]} \overset{a}{\Longrightarrow} t' \rrbracket_{[\epsilon \ll} i' {}_{\ll \sigma_i]}$. Hence by Lemma 4, ${}_{[\epsilon \ll} i' {}_{\ll \rho_i]}$ is quiescent($\delta_q({}_{[\epsilon \ll} i' {}_{\ll \rho_i]})$).
It follows from the assumption and deduction rule I_3^* in Proposition 1, ${}_{[\epsilon \ll} i' {}_{\ll \rho_i]} \overset{\tau}{\Longrightarrow} {}_{[x \ll} i'' {}_{\ll \rho_i]}$. Since the output queue is non-empty we can apply deduction rule *A2* on the target state and obtain ${}_{[x \ll} i'' {}_{\ll \rho_i]} \xrightarrow{x} {}_{[\epsilon \ll} i'' {}_{\ll \rho_i]}$. Combining the two transitions, we obtain ${}_{[\epsilon \ll} i' {}_{\ll \rho_i]} \overset{y}{\Longrightarrow} {}_{[\epsilon \ll} i'' {}_{\ll \rho_i]}$. From the latter transition, we conclude that ${}_{[\epsilon \ll} i' {}_{\ll \rho_i]}$ is not quiescent which is contradictory to the former observation. ⊠

Proof of Lemma 3.1. Assume, towards a contradiction, that for all i' such that $i \overset{\epsilon}{\Longrightarrow} i'$, it doesn't hold $\delta(i')$. Take the i' with the largest empty trace (by counting the numbers of τ-labeled transitions). Such i' must exist since otherwise, there must be a loop of τ-labeled transition which is opposed to the assumption that i doesn't diverge. Since i' is not quiescent, according to Definition 5, there exists an $x \in L_u$ such that $i' \xrightarrow{x}$. From item **??** in Definition 3, we know that there must exist an i'' such that $i' \xrightarrow{x} i''$. It follows from item 4 in Proposition 1 and deduction rule *I3* in Definition 12 that $Q(i) \overset{\epsilon}{\Longrightarrow} {}_{[x \ll} i'' {}_{\ll \epsilon]}$ and since the output queue is non-empty we can apply the deduction rule *A2* on the target state and obtain ${}_{[x \ll} i'' {}_{\ll \epsilon]} \xrightarrow{x} Q(i'')$. Combining the two transition we obtain $Q(i) \overset{x}{\Longrightarrow} Q(i'')$. From the latter transition we can conclude that $Q(i)$ is not quiescent which is contradictory to the thesis. ⊠

Proof of Proposition 3.2. We distinguish four cases based on the status of input and output queues.

($\sigma_i = \epsilon$, $\sigma_u = \epsilon$) By assuming $s' = s$, the thesis is proved.
($\sigma_i \neq \epsilon$, $\sigma_u \neq \epsilon$) According to Lemma 5, no trace leads to this situation.
($\sigma_i \neq \epsilon$, $\sigma_u = \epsilon$) 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 6, we know that $\forall x \in L_U, x \notin \mathbf{Sinit}(s')$ and since s' doesn't diverge, it must reach eventually a state such as i which performs a transition other than an internal one, hence the only possible choice is an input transition. From Definition 9 we know that $\delta(i)$ and according to Definition 10, state i is input-enabled as well. Thus $\exists i' \bullet i \xrightarrow{x} i'$. Due to the subsequent application of deduction rules of *I1*, *I2* in Definition 12 and *R1* in Definition 7, transition $t' \rrbracket_{[\epsilon \ll s' \ll x]} \overset{\epsilon}{\Longrightarrow} t' \rrbracket Q(i')$ is possible. By assuming $s'' = i'$ and combination of the latter transition and the assumption, we have $t \rrbracket Q(s) \overset{\sigma}{\Longrightarrow} t' \rrbracket Q(i')$ which was to be shown. For the induction step, assume that the thesis holds for all non-empty input queues with length $n-1$ or less and length of σ_i is n. It follows from $\sigma_i \neq \epsilon$ that there exists an $a \in L_I$ and $\sigma'_i \in L_I*$ and $\sigma' \in (L \cup \{\theta\})^*$ such that $\sigma_i = \sigma'_i.a$ and $t \rrbracket Q(s) \overset{\sigma'}{\Longrightarrow} t_p \rrbracket_{[\epsilon \ll i' \ll \sigma'_i]} \overset{a}{\Longrightarrow} t' \rrbracket_{[\epsilon \ll s' \ll \sigma_i]}$. It follows from the induction hypothesis that $\exists i \bullet t \rrbracket Q(s) \overset{\sigma'}{\Longrightarrow} t_p \rrbracket Q(i)$. Due to the application of deduction rule *R2* in Definition 7 and *A1* in Definition12, we have $t_p \rrbracket Q(i) \overset{a}{\Longrightarrow} t' \rrbracket_{[\epsilon \ll i \ll a]}$. It follows from the induction basis that $\exists s'' \bullet t_p \rrbracket Q(i) \overset{a}{\Longrightarrow} t' \rrbracket Q(s'')$. Combination of the two transitions leads to $\exists s'' \bullet t \rrbracket Q(s) \overset{\sigma}{\Longrightarrow} t' \rrbracket Q(s'')$ which was to be shown.

$(\sigma_i = \epsilon, \sigma_u \neq \epsilon)$ 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 12. Hence, there must exist some $s'', q'' \in IOTS^{\sqcap}(L_I, L_U)$ such that $_{[\sigma'_u \ll} s''_{\ll \sigma'_i]} \xrightarrow{\tau} {}_{[\sigma'_u.x \ll} q''_{\ll \sigma'_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma'_u.x \ll} s'_{\ll \sigma'_i]}$. Combination of the two transition concludes that $_{[\sigma'_u \ll} s''_{\ll \sigma'_i]} \overset{\epsilon}{\Longrightarrow} {}_{[\sigma'_u.x \ll} q''_{\ll \sigma'_i]}$. It follows from the application of deduction rule $R1^*$ in Proposition 1 that the input queue at state $_{[\sigma'_u \ll} s''_{\ll \sigma'_i]}$ must be empty since otherwise according to Lemma 6, s'' would be quiescent and could not produce any output. Thus there exist $\sigma' \in (L \cup \{\theta\})^*$, $\sigma'_u \in L_U^*$ and $t'_p \in TTS^{\sqcap}(L_U, L_I)$ such that $t \rrbracket Q(s) \overset{\sigma'}{\Longrightarrow} t'_p \rrbracket_{[\sigma'_u \ll s'' \ll \epsilon]} \overset{\epsilon}{\Longrightarrow} t'_p \rrbracket_{[\sigma'_u.x \ll s' \ll \epsilon]} \overset{\sigma'_u}{\Longrightarrow} t' \rrbracket_{[x \ll s' \ll \epsilon]}$ and $\sigma = \sigma'.\sigma'_u$. Due to the application of deduction rules *R2* in Definition 7 and *A2* in Definition 12, it concludes that $t'_p \rrbracket_{[\sigma'_u \ll s'' \ll \epsilon]} \overset{\sigma'_u}{\Longrightarrow} t' \rrbracket Q(s'')$ and subsequently we have $t \rrbracket Q(s) \overset{\sigma'}{\Longrightarrow} t'_p \rrbracket_{[\sigma'_u \ll s'' \ll \epsilon]} \overset{\sigma'_u}{\Longrightarrow} t' \rrbracket Q(s'')$ 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 $x \in L_U$ and $\sigma'_u \in L_U^*$ and $\sigma' \in (L \cup \{\theta\})^*$ such that $\sigma_u = \sigma'_u.x$ and $t \rrbracket Q(s) \overset{\sigma'}{\Longrightarrow} t_p \rrbracket_{[\sigma''_u.\sigma'_u \ll q \ll \epsilon]} \xrightarrow{\tau} t_p \rrbracket_{[\sigma''_u.\sigma'_u.x \ll q' \ll \epsilon]} \overset{\sigma''_u}{\Longrightarrow} t' \rrbracket_{[\sigma'_u.x \ll s' \ll \epsilon]}$ and $\sigma = \sigma'.\sigma''_u$. Due the application of deduction rule *R2* in Definition 7 and *A2* in Definition 12, we have $t_p \rrbracket_{[\sigma''_u.\sigma'_u \ll q \ll \epsilon]} \overset{\sigma''_u}{\Longrightarrow} t' \rrbracket_{[\sigma'_u \ll q \ll \epsilon]}$. Thus we can run the previous execution in a new order such

as $t]\!\|Q(s) \overset{\sigma'}{\Longrightarrow} t_p]\!\|_{[\sigma''_u.\sigma'_u \ll q \ll \epsilon]} \overset{\sigma''_u}{\Longrightarrow} t']\!\|_{[\sigma'_u \ll q \ll \epsilon]} \xrightarrow{\tau} t']\!\|_{[\sigma'_u.x \ll s' \ll \epsilon]}$. 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 $\exists s'' \bullet t]\!\|Q(s) \overset{\sigma}{\Longrightarrow} t']\!\|Q(s'')$ which was to be shown.

$\boxtimes$