Testing Multi Input/Output Transition System with All-Observer*

Zhongjie Li, Jianping Wu, and Xia Yin

Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, P.R.China
{lzj, yxia}@csnet1.cs.tsinghua.edu.cn
jianping@cernet.edu.cn

Abstract. Multi input/output transition system (MIOTS) models the interface distribution of a system by partitioning its inputs and outputs into channels. The MIOTS refusal testing theory has been based on singular observers. Such an observer is useful for eliminating nondeterminism in the testing process, but also contributes to the large size of the test suites. We propose an alternative type of observers: all-observer, which can observe all the output channels simultaneously, and help to reduce a test suite effectively. An algorithm is presented to generate an all-observer test suite from the specification. The derived test suite is sound for all MIOTS systems, and complete for a class of MIOTS systems that are common in practice. We also discuss the problem of factorized all-observer test generation. Our work complements the MIOTS refusal testing with singular observers.

1 Introduction

Conformance testing is an operational way to check the correctness of a system implementation by means of experimenting. Tests are applied to the implementation, and based on observations made during the execution of the tests, a verdict about the correctness of the implementation is given. In formal conformance testing it is assumed that we have a formal specification, and implementations whose behavior is also formally modeled but not apriori known. Labeled Transition System (LTS) is a well-known model [1,2] for describing processes. LTS does not distinguish between the initiative of actions, which is not very realistic. So input/output transition system (IOTS) was proposed to model the more common communication via actions that are initiated by the system (output), and initiated by the environment (input) [3]. The IOTS model has no consideration for the distribution of communication interfaces between the system and the environment. To overcome this deficiency, [4] proposed a new model called multi input/output transition system (MIOTS), which partitions the inputs and outputs into different channels reflecting the locations where actions occur.

* This work was supported by the 973 project No.2003CB314801 and NSFC project No.90104002, No.60102009

For MIOTS implementations, the conformance to an LTS specification is defined as the implementation relation *multi input/output refusal preorder* $\leq_{mior}$. $\mathbf{mioco}_{\mathcal{F}}$ is a generalization of $\leq_{mior}$, which requires that all responses an implementation can perform after every trace in the set of traces $\mathcal{F}$ are allowed by the specification. Testing for $\mathbf{mioco}_{\mathcal{F}}$ consists of serial compositions of providing a single input action at some input channel and detection of its acceptance or rejection, and observing some output channel and detection of the occurrence or absence of outputs produced at this channel. Such tests are modeled by *singular observers*. After each trace in $\mathcal{F}$, an individual test is needed for checking each output channel to see if the implementation behaves correctly at this channel. This method avoids the nondeterministic outputs of the implementation at different channels, and also allows for stronger testing power in general. However, it also contributes to the large size of the derived test suite, which means big time expense in test generation as well as in test execution.

Just like singular observers, all-observer is a special class of MIOTS that can observe all the output channels simultaneously. We present in this paper a test generation algorithm. For a specification with respect to $\mathbf{mioco}_{\mathcal{F}}$, the algorithm generates an all-observer test suite, which is smaller than the singular observer test suite, sound for all MIOTS systems, and complete for special MIOTS systems (which are common in practice, e.g. queue systems).

Factorized test generation [5] is a technique that aims to avoid the generation of tests for a complicated correctness criterion directly from a large specification. In [5], for a specification s and $\mathbf{mioco}_{\mathcal{F}}$, $\mathbf{mioco}_{\mathcal{F}}$ is decomposed into $\mathbf{mioco}_{\{\sigma\}}(\forall\sigma \in \mathcal{F})$. Then for each trace, a selection process is applied to obtain a reduced-size specification. This decomposition is very inefficient, especially when $\mathcal{F}$ contains a lot of traces. So we propose an improved $\mathcal{F}$-partition method, which groups all the traces having the same selection process in one set, and thus makes the test generation more efficient. It is shown that this optimization is necessary for the factorized all-observer test generation.

This paper is organized as follows. Sect. 2 reviews the preliminaries of the refusal testing theory for MIOTS. Sect. 3 describes our work on the all-observer test generation. Sect. 4 discusses the factorized test generation in all-observer based testing. Concluding remarks and future works are presented in sect. 5.

2 Refusal Testing for MIOTS

Definition 1. *A (labeled) transition system over L is a quadruple $\langle S, L, \rightarrow, s_0\rangle$ where S is a (countable) set of states, L is a (countable) set of observable actions, $\rightarrow \subseteq S \times L \times S$ is a set of transitions, and $s_0 \in S$ is the initial state.*

We denote the class of all transition systems over L by $\mathcal{LTS}(L)$. The observable behavior of a transition system is expressed using sequences consisting of actions and sets of refused actions, i.e., sequences in $(L \cup P(L))^*$ ($P(L)$ is the power-set of L). Such sequences are called *failure traces*, comparable to *traces* (those in L^*). *A refusal transition* is defined as a self-loop transition in the form

$s \xrightarrow{A} s'$ where $A \subseteq L$ is called *a refusal* of s, meaning that the system is unable to perform any action in A from state s.

Definition 2. *Let $p \in \mathcal{LTS}(L)$, then*

1. *$init(p) =_{def} \{\alpha \in L | \exists p' : p \xrightarrow{\alpha} p'\}$*
2. *$der(p) =_{def} \{p' | \exists \sigma \in (L \cup P(L))^* : p \xrightarrow{\sigma} p'\}$*
3. *$ftraces(p) =_{def} \{\sigma \in (L \cup P(L))^* : p \xrightarrow{\sigma}\}$*
4. *$pref(\sigma_2) = \{\sigma_1 | \exists \sigma' : \sigma_1 \cdot \sigma' = \sigma_2$ and $\sigma_1, \sigma_2, \sigma' \in (L \cup \mathcal{P}(L))^*\}$*
5. *P **after** $\sigma =_{def} \{p' | \exists p \in P : p \xrightarrow{\sigma} p'\}$*
6. *p is deterministic iff $\forall \sigma \in L^* : |\{p\}$ **after** $\sigma| \leq 1$*
7. *p is output-finite if there is a natural number N s.t. $\forall p' \in der(p)$, the set $X = \{\sigma_u \in (L_U)^* | p' \xrightarrow{\sigma_u}\}$ is finite and $\forall \sigma_u \in X : |\sigma_u| \leq N$, where L_U is the set of output actions.*

Definition 3. *A multi input/output transition system p over partitioning $\mathcal{L}_I = \{L_I^1, \ldots, L_I^n\}$ of L_I and partitioning $\mathcal{L}_U = \{L_U^1, \ldots, L_U^m\}$ of L_U is a transition system with inputs and outputs, $p \in \mathcal{LTS}(L_I \cup L_U)$, such that for all $L_I^j \in \mathcal{L}_I$, $\forall p' \in der(p)$, if $\exists a \in L_I^j : p' \xrightarrow{a}$ then $\forall b \in L_I^j : p' \xrightarrow{b}$. The universe of multi input/output transition systems over $\mathcal{L}_I$ and $\mathcal{L}_U$ is denoted by $\mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U)$.*

Refusal testing [6] is a kind of such implementation relation where experiments are not only able to detect whether actions can occur, but also able to detect whether actions can fail, i.e. refused by the system. In MIOTS refusal testing [4], special action labels $\theta_i^j (j = 1, \ldots, n)$ are added to observe the inability of the implementation to accept an input action in channel L_I^j (input suspension, denoted by ξ^j), and $\theta_u^k (k = 1, ..., m)$ are added to observe the inability to produce outputs in channel L_U^k (output suspension, denoted by δ^k). Let $\Theta = \{\theta_i^1, \ldots, \theta_i^n, \theta_u^1, \ldots, \theta_u^m\}$ denote all the suspension detection labels. Then, implementations that are modeled as members of $\mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U)$ are observed by observers modeled in $\mathcal{MIOTS}(\mathcal{L}_U^\theta, \mathcal{L}_I^\theta)$ where $\mathcal{L}_I^\theta = \{L_I^1 \cup \{\theta_i^1\}, \ldots, L_I^n \cup \{\theta_i^n\}\}$, $\mathcal{L}_U^\theta = \{L_U^1 \cup \{\theta_u^1\}, \ldots, L_U^m \cup \{\theta_u^m\}\}$. Communication between observer and system is modeled by the parallel composition operator $\|$. Observations that can be made by an observer u interacting with p by means of $\|$ now may consist these suspension detection actions: $obs(u, p) =_{def} \{\sigma \in (L \cup \Theta)^* | (u \| p) \xrightarrow{\sigma}\}$.

Singular observers are a special class of MIOTS observers. They consist of finite, serial compositions of providing a single input action at some channel L_I^j and detection of its acceptance or rejection, and observing some channel L_U^k and detection of the occurrence or absence of outputs produced at this channel. The set of all singular observers over $\mathcal{L}_I$ and $\mathcal{L}_U$ is denoted by $\mathcal{SOBS}(\mathcal{L}_U^\theta, \mathcal{L}_I^\theta)$.

In correspondence with the observations defined on $(L_I \cup L_U \cup \Theta)^*$, we define the suspension traces of p to be its failure traces restricted to $(L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^*$: $straces(p) =_{def} ftraces(p) \cap (L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^*$. Responses of the implementation after a specific suspension trace that can be observed by singular observers are collected into the set *out*.

$$\begin{aligned} out(p \textbf{ after } \sigma) =_{def} & \{x \in L_U \mid \exists p' : p \xrightarrow{\sigma} p' \xrightarrow{x}\} \\ \cup \quad & \{\xi^j \mid 1 \leq j \leq n,\ \exists p' : p \xrightarrow{\sigma} p' \text{ and } init(p') \cap L_I^j = \emptyset\} \\ \cup \quad & \{\delta^k \mid 1 \leq k \leq m, \exists p' : p \xrightarrow{\sigma} p' \text{ and } init(p') \cap L_U^k = \emptyset\} \end{aligned}$$

Definition 4 (multi input/output refusal preorder). *$i \in \mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U)$ and $s \in \mathcal{LTS}(L_I \cup L_U)$, then*

$$i \leq_{mior} s =_{def} \forall u \in \mathcal{SOBS}(\mathcal{L}_U^\theta, \mathcal{L}_I^\theta) : obs(u, i) \subseteq obs(u, s).$$

[4] has proved that $i \leq_{mior} s$ iff $\forall \sigma \in (L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^* : out(i \textbf{ after } \sigma) \subseteq out(s \textbf{ after } \sigma)$. Checking this *out* inclusion condition for all the suspension traces is too time consuming in practice. Therefore, [4] further generalizes this condition to an arbitrary (and possible finite) set $\mathcal{F} \subseteq (L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^*$, and define a corresponding implementation relation $\textbf{mioco}_\mathcal{F}$:

Definition 5. *$i \textbf{ mioco}_\mathcal{F}\ s =_{def} \forall \sigma \in \mathcal{F} : out(i \textbf{ after } \sigma) \subseteq out(s \textbf{ after } \sigma)$.*

3 All-Observer Testing for MIOTS

3.1 All-Observer

The all-observer is also a special class of MIOTS observers. Besides providing input actions and observing single-channel outputs or suspensions, they are additionally equipped with an all-output-channel observing mode: observing all the m output channels ($L_U^k, k = 1, \ldots, m$) and detection of an output at some channel, or no output at all. This collective output suspension is denoted by a special label δ (called all-channel output suspension, meaning the refusal of L_U), which can be detected by the all-observer using the label $\theta_u : \theta_u =_{def} \langle \theta_u^1, \ldots, \theta_u^m \rangle$. The set Θ denotes the set of all the suspension detection labels: $\Theta = \{\theta_i^1, \ldots, \theta_i^n, \theta_u^1, \ldots, \theta_u^m, \theta_u\}$, and let $\Psi = \{\xi^1, \ldots, \xi^n, \delta_1, \ldots, \delta^m, \delta\}$ denote the counterpart suspension actions of the system. Other notations follow those in Sect. 2.

Definition 6. *An all-observer u over $\mathcal{L}_I$ and $\mathcal{L}_U$ is a finite, deterministic MIOTS $u \in \mathcal{MIOTS}(\mathcal{L}_U^\theta, \mathcal{L}_I^\theta)$ such that*

$$\begin{aligned} & \forall u' \in der(u) : init(u') = \emptyset \text{ or } init(u') = L_U \cup \{\theta_u\} \\ & \text{or } init(u') = \{a, \theta_i^j\} \text{ for some } j \in \{1, \ldots, n\} \text{ and } a \in L_I^j \\ & \text{or } init(u') = L_U^k \cup \{\theta_u^k\} \text{ for some } k \in \{1, \ldots, m\} \end{aligned}$$

the set of all-observer over $\mathcal{L}_I$ and $\mathcal{L}_U$ is denoted by $\mathcal{AOBS}(\mathcal{L}_U^\theta, \mathcal{L}_I^\theta)$.

Definition 7. *Communication between all-observer and system is modeled by the operator $\| : \mathcal{MIOTS}(\mathcal{L}_U^\theta, \mathcal{L}_I^\theta) \times \mathcal{LTS}(L_I \cup L_U) \rightarrow \mathcal{LTS}(L_I \cup L_U \cup \Theta)$, defined by the following inference rules:*

$$\frac{u \xrightarrow{a} u', p \xrightarrow{a} p'}{u \| p \xrightarrow{a} u' \| p'} (a \in L_I \cup L_U) \qquad \frac{u \xrightarrow{\theta_i^j} u', init(p) \cap L_I^j = \emptyset}{u \| p \xrightarrow{\theta_i^j} u' \| p} (j \in \{1, \ldots, n\})$$

$$\frac{u \xrightarrow{\theta_u} u', init(p) \cap L_U = \emptyset}{u \| p \xrightarrow{\theta_u} u' \| p} \qquad \frac{u \xrightarrow{\theta_u^k} u', init(p) \cap L_U^k = \emptyset}{u \| p \xrightarrow{\theta_u^k} u' \| p} (k \in \{1, \ldots, m\})$$

Definition 8. *An all-observer test* $t \in \mathcal{AOBS}(\mathcal{L}_U^\theta, \mathcal{L}_I^\theta)$ *such that* $\forall t' \in der(t)$*:*

$$init(t') = \emptyset \qquad \textit{iff} \qquad t' = \textbf{pass} \textit{ or } t' = \textbf{fail}$$

where **pass** *and* **fail** *are verdicts that indicate the (in)correctness of implementation* i *when running* t *against* i*.*

A test suite is sound if it never rejects correct implementations, and a test suite is exhaustive if each incorrect implementation always fails this test suite. In practice test suites are required to be sound, but not necessarily exhaustive. A test suite is called complete if it is both sound and exhaustive. [4] presents a test generation algorithm that produces complete $\mathcal{SOBS}$ test suites for specifications with respect to $\textbf{mioco}_\mathcal{F}$. The $\mathcal{SOBS}$ test suite generated by the algorithm tends to be very large because the necessity to check each output channel after each $\sigma \in \mathcal{F}$ using a separate test. Our goal is: for given $\textbf{mioco}_\mathcal{F}$ and specification s, to replace the complete $\mathcal{SOBS}$ test suite with an $\mathcal{AOBS}$ test suite, which has a smaller size, and preserves the soundness unconditionally, and preserves the completeness conditionally. We presents an all-observer test generation algorithm in the next section. It is modified from the singular-observer algorithm with two changes: one, merging test purposes by checking more traces in one test case; two, observing all the output channels simultaneously after performing each trace in $\mathcal{F}$ using one test, instead of checking the output channels one by one using m tests. The two changes all help to reduce the size of a test suite.

3.2 Test Generation

Let $p \in \mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U)$, abbreviate "$p$ **after** σ" to "$p - \sigma$", and define $out_2(p - \sigma)$ to be the union of $out(p - \sigma)$ and the possible all-channel output suspension δ: $out_2(p - \sigma) =_{def} out(p - \sigma) \cup \{\delta | \exists p' \in p - \sigma : init(p') \cap L_U = \emptyset\}$. In addition, we define the out set of a state set P to be: $out(P) =_{def} \{out(p - \epsilon) | p \in P\}$. We denote with $\bar{\sigma}$ the trace σ where each occurrence of a refusal action ξ^j, δ or δ^k is replaced by its detection label θ_i^j, θ_u or θ_u^k, and vice versa.

The $\mathcal{AOBS}$ test generation algorithm is shown as follows. The rationale behind is that it constructs tests that check the condition set forth in Def. 5: $out(i - \sigma) \subseteq out(s - \sigma)$ for each $\sigma \in \mathcal{F}$.

Algorithm 1
input: $s = \langle S, L_I \cup L_U, \rightarrow, s_0 \rangle, \mathcal{F} \subseteq (L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^*$
output: test case $t_{\mathcal{F},\mathcal{S}}$
initial value: $\mathcal{S} = \{s_0\}$ **after** ε
Apply one of the following non-deterministic choices recursively.

1. if $\mathcal{F} = \emptyset$ then $t_{\mathcal{F},\mathcal{S}} := \textbf{pass}$

2. take some $L_I^j \in \mathcal{L}_I$ and $a \in L_I^j$, such that $\mathcal{F}' = \{\sigma | a \cdot \sigma \in \mathcal{F}\} \neq \emptyset$ and $\mathcal{S}' = \mathcal{S}$ **after** a, then

$$t_{\mathcal{F},\mathcal{S}} := a; t_{\mathcal{F}',\mathcal{S}'} + \begin{cases} \theta_i^j; \textbf{fail} & \xi^j \notin out(\mathcal{S}) \textit{ and } \epsilon \in \mathcal{F} \\ \theta_i^j; \textbf{pass} & \textit{otherwise} \end{cases}$$

3. take some $L_I^j \in \mathcal{L}_I$ such that $\mathcal{F}' = \{\sigma | \xi^j \cdot \sigma \in \mathcal{F}\} \neq \emptyset, \mathcal{S}' = (\mathcal{S}\ \mathbf{after}\ \xi^j)$, then

$$t_{\mathcal{F},\mathcal{S}} := a; \mathbf{pass} + \theta_i^j; t_{\mathcal{F}',\mathcal{S}'} \ (a \text{ is any input in } L_I^j)$$

4. observe on all the output channels

$$\begin{aligned} t_{\mathcal{F},\mathcal{S}} := & \sum\{x; \mathbf{fail} \mid x \notin out(\mathcal{S}), \epsilon \in \mathcal{F}\} \\ & + \sum\{x; \mathbf{pass} \mid x \notin out(\mathcal{S}), \epsilon \notin \mathcal{F}\} \\ & + \sum\{x; t_{\mathcal{F}',\mathcal{S}'} \mid x \in out(\mathcal{S}), \mathcal{F}' = \{\sigma | x \cdot \sigma \in \mathcal{F}\}, \mathcal{S}' = \mathcal{S}\ \mathbf{after}\ x\} \\ & + \begin{cases} \theta_u; \mathbf{fail} & \textit{if } \epsilon \in \mathcal{F} \textit{ and } \exists k \in \{1,\dots,m\}, \delta^k \notin out(\mathcal{S}) \\ \theta_u; \mathbf{pass} & \textit{otherwise} \end{cases} \end{aligned}$$

where $x \in L_U$

5. take some $L_U^k \in \mathcal{L}_U$, if $\mathcal{F}' = \{\sigma | \delta^k \cdot \sigma \in \mathcal{F}\} \neq \emptyset$ and $\mathcal{S}' = \mathcal{S}\ \mathbf{after}\ \delta^k$, then

$$t_{\mathcal{F},\mathcal{S}} := \sum\{x; \mathbf{fail} \mid x \notin out(\mathcal{S}), \epsilon \in \mathcal{F}\} + \sum\{x; \mathbf{pass} \mid \textit{otherwise}\} + \theta_u^k; t_{\mathcal{F}',\mathcal{S}'}$$

where $x \in L_U^k$ □

Step 1 assigns **pass** in case no trace in $\mathcal{F}$ is performed. Step 2 and 3 each supplies an input to the implementation at some channel L_I^j and continues if the implementation is able to accept or refuse this input, respectively. Step 4 awaits an output action at any output channel or observes an all-channel output suspension. Step 5 awaits a single-channel output suspension to test deeper. For each response that is unspecified by the specification, a **fail** verdict is given. In particular, when θ_u is observed but there exists an output channel that should not suspend at the current states (i.e., $\delta^k \notin out(\mathcal{S})$), a **fail** verdict should be made. With this strategy, it is apparent that an all-observer test cannot detect unspecified single-channel output suspension, because the suspension will be screened by an output at another channel that is also under observation in the all-output-channel observing mode. This limitation makes Algorithm 1 unable to generate an exhaustive test suite. We will give a demonstration and discuss this problem in Sect. 3.4. Fig. 1 compares $\mathcal{AOBS}$ with $\mathcal{SOBS}$ tests in the way they check the output behavior of the implementation after performing σ. An $\mathcal{SOBS}$ test suite uses m tests: $t_1, \dots, t_m$, each for one output channel. These tests are substituted in the $\mathcal{AOBS}$ test suite by only one test t as shown in Fig. 1b, where x_k generally refers to any $x \in L_U^k$.

3.3 Soundness

Proposition 1. *Let $p \in \mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U)$, then $\delta \in out_2(p - \sigma)$ implies $\delta^k \in out(p - \sigma), k = 1, \dots, m$.*

Tests generated by Algorithm 1 in Sect. 3.2 check all and only the traces in $\mathcal{F}$ to see if the *out*-inclusion condition holds. There are five types of observations: (1) input action $a \in L_I^j$, (2) input suspension ξ^j, (3) output action $y \in L_U^k$, (4) single-channel output suspension δ^k, (5) all-channel output suspension δ. Only three of them (2, 3, 5) may be associated with verdicts. **fail** verdicts are only given for unspecified output or input and output suspension, which all mean non-conformance, so the $\mathcal{AOBS}$ tests generated by Algorithm 1 must be sound.

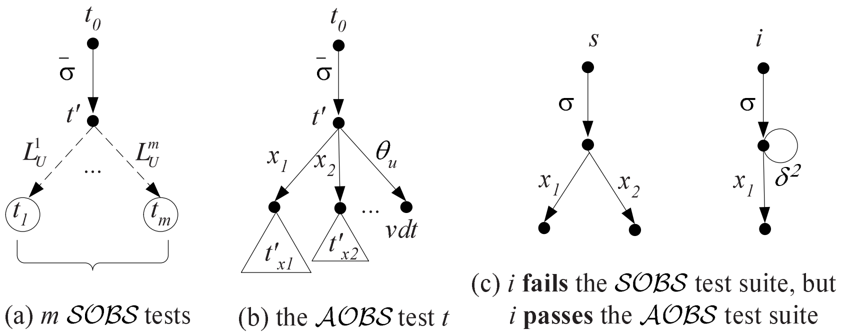

Fig. 1. testing σ for output: $\mathcal{SOBS}$ vs. $\mathcal{AOBS}$ tests

Proposition 2. *Let $s \in \mathcal{LTS}(L_I, L_U)$, $\mathcal{F} \subseteq (L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^*$, then the $\mathcal{AOBS}$ test suite generated by Algorithm 1 is sound for s w.r.t.* **mioco**$_\mathcal{F}$.

Algorithm 1 generates sound but not exhaustive test suites. See Fig. 1c for an example, suppose x_1 and x_2 respectively belong to L_U^1 and L_U^2 of a specification s and $\mathcal{F} = \{\sigma\}$, the implementation i will fail the $\mathcal{SOBS}$ tests because it has an unspecified output suspension (δ^2, as shown by the self-loop transition) after performing σ. This error, however, will not be disclosed by the $\mathcal{AOBS}$ tests generated by Algorithm 1, because the output-checking test always stops with a **pass** verdict after the action x_1. This example shows that the all-observer generally have a weaker testing power than singular observers. The exhaustiveness can only be preserved conditionally, as is shown in the next section.

3.4 Completeness

In this section, we discuss the problem of generating complete all-observer test suites. We show that this is possible for a special class of MIOTS in case $\mathcal{F}$ satisfies a given condition.

Definition 9. *Let $s \in \mathcal{LTS}(L_I, L_U)$, $L_\theta = L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U$, strace-reordering is a relation defined on $(L_\theta)^*$, $\sim \subseteq (L_\theta)^* \times (L_\theta)^*$:*

$\forall \sigma_1, \sigma_2 \in (L_\theta)^*, \sigma_1 \sim \sigma_2 =_{def} \sigma_1 \lceil (L_I \cup \mathcal{L}_I \cup L_U^k \cup \{L_U^k\}) = \sigma_2 \lceil (L_I \cup \mathcal{L}_I \cup L_U^k \cup \{L_U^k\}), k = 1, \ldots, m$

where $\sigma \lceil A$ is the projection of σ on the action set A, resulting in a trace consisting of only actions in A with their original order.

If $\sigma_1 \sim \sigma_2$, we say σ_2 is an *strace-reordering* of σ_1. For example, use a, b for inputs, and x, y, z for outputs at different channels, then $(a \cdot x_1 \cdot x_2 \cdot y \cdot \delta^2 \cdot z \cdot b \cdot x \cdot y) \sim (a \cdot y \cdot x_1 \cdot \delta^2 \cdot z \cdot x_2 \cdot b \cdot y \cdot x)$. $\sim$ is an equivalence relation on $(L_\theta)^*$. A real background of this relation is the queue systems [7].

Definition 10. *Let $p \in \mathcal{LTS}(L_I \cup L_U)$,$\mathcal{L}_I = \{L_I^1, \dots, L_I^n\}$ and $\mathcal{L}_U = \{L_U^1, \dots, L_U^m\}$, we say p is strace-reorderable, if*

$$\forall \sigma_1, \sigma_2 \in (L_\theta)^* \textit{ and } \sigma_1 \sim \sigma_2 \textit{: } \sigma_1 \in straces(p) \textit{ iff } \sigma_2 \in straces(p)$$

Queue systems is an intuitionistic example of strace-reorderable systems, but note that not all the strace-reorderable systems are queue systems. Strace-reorderable systems have two important properties.

Proposition 3. *Suppose $p \in \mathcal{LTS}(L_I \cup L_U)$ is strace-reorderable, $\sigma \in (L_\theta)^*$, $x \in L_U^{k'}$, $k \neq k'$, then*

1. $\delta^k \notin out(p - \sigma)$ implies $\delta^k \notin out(p - \sigma \cdot x)$

2. $\forall p', p \xrightarrow{\sigma} p' : (p' \xrightarrow{\delta^k}$ and $p' \xrightarrow{x})$ implies $\delta^k \in out(p - \sigma \cdot x)$

Proof.

1. (by contradiction) $\delta^k \in out(p - \sigma \cdot x)$ implies $p \xrightarrow{\sigma \cdot x \cdot \delta^k}$, in turn implies $p \xrightarrow{\sigma \cdot \delta^k \cdot x}$ because p is strace-reorderable and $\sigma \cdot x \cdot \delta^k \sim \sigma \cdot \delta^k \cdot x$. Then we have $\delta^k \in out(p - \sigma)$, so a contradiction.

2. We first show that $p \xrightarrow{\sigma} p' : (p' \xrightarrow{\delta^k}$ *and* $p' \xrightarrow{x} p'')$ *implies* $p'' \xrightarrow{\delta^k}$. Otherwise, suppose $\exists y \in L_U^k : p'' \xrightarrow{y}$, then $p \xrightarrow{\sigma \cdot \delta^k \cdot x \cdot y}$. Because p is strace-reorderable and also $\sigma \cdot \delta^k \cdot x \cdot y \sim \sigma \cdot \delta^k \cdot y \cdot x$, we will have $p \xrightarrow{\sigma \cdot \delta^k \cdot y \cdot x}$, which is impossible by the definition of δ^k. From $p'' \xrightarrow{\delta^k}$ we have $\delta^k \in out(p - \sigma \cdot x)$. □

The first statement means that, if p does not suspend on the channel L_U^k after a trace σ, it will not either after the trace $\sigma \cdot x$ where x belongs to a different output channel. The second statement comes from the fact that, if p cannot produce any output at the channel L_U^k in a state p', it cannot either after a further output action x at a different channel. These properties characterize the independence of the output behaviors occurring at different channels of a strace-reorderable system.

Definition 11. *$s \in \mathcal{LTS}(L_I, L_U)$, $\mathcal{F} \subseteq (L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^*$, for $\forall k \in \{1, \dots, m\}$, the boolean predicate $keep_{\mathcal{F}}(\delta^k)$ is true if*

$$\forall \sigma \in \mathcal{F} \textit{ and } \delta^k \notin out(s - \sigma), \forall y \in L_U \backslash L_U^k : y \in out(s - \sigma) \textit{ implies } \sigma \cdot y \in \mathcal{F}$$

"$L_U \backslash L_U^k$" is the difference between L_U and L_U^k. In case s is output-finite (cf. Def. 2.7), from a finite set $\mathcal{F}$, we can always derive a finite set $\mathcal{F}'$ that satisfies $\forall k \in \{1, \dots, m\} : keep_{\mathcal{F}'}(\delta^k)$. One of such sets is $\mathcal{F}_0 = \mathcal{F} \cup \{\sigma \cdot \sigma_u \in straces(s) | \sigma \in \mathcal{F}, \sigma_u \in (L_U)^*\}$. This expansion is necessary for the purpose of detecting unspecified single-channel output suspension δ^k, as explained later.

On the contrary, a system that is not output-finite may produce infinite output sequences (and possible with infinite length) in some state. We exclude such systems in the discussion of completeness and assume that specifications are output-finite. However, we do not require this for implementations; an implementation may produce endless outputs (e.g. on entering an error state). Some straightforward properties of $keep_{\mathcal{F}}(\delta^k)$ are summarized below.

Proposition 4. *Let $s \in \mathcal{LTS}(L_I \cup L_U), \sigma \in (L_\theta)^*, \forall k(k = 1, \dots, m)$*

1. $\delta^k \in out(s - \sigma)$ implies $keep_{\{\sigma\}}(\delta^k)$

2. if $\delta^k \notin out(s - \sigma)$ and $\neg\exists y \in L_U \backslash L_U^k$ s.t. $y \in out(s - \sigma)$, then $keep_{\{\sigma\}}(\delta^k)$

3. $keep_{\{\sigma\}}(\delta^k)(\forall \sigma \in \mathcal{F})$ implies $keep_{\mathcal{F}}(\delta^k)$

Proposition 5. *Let $s \in \mathcal{LTS}(L_I \cup L_U)$, $i \in \mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U)$, s and i are all strace-reorderable, and s is ouput-finite, $\mathcal{F} \subseteq (L_\theta)^*$ and satisfies $keep_{\mathcal{F}}(\delta^k)(k = 1, \dots, m)$, T is the $\mathcal{AOBS}$ test suite generated by Algorithm 1, then T is exhaustive for s w.r.t.* **mioco**$_{\mathcal{F}}$.

Proof. We have to prove: $\forall i \in \mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U) : \neg(i\ \textbf{mioco}_{\mathcal{F}}\ s)$ implies (i **fails** T), this equals to proving:

$$\text{“}\exists \sigma \in \mathcal{F} : out(i - \sigma) \nsubseteq out(s - \sigma)\text{” implies “}\exists t \in T : i\ \textbf{fails}\ t\text{”} \qquad (\#)$$

If $\exists \sigma \in \mathcal{F}$ *s.t.* $out(i - \sigma) \nsubseteq out(s - \sigma)$, at least one of the following three cases must hold according to the definition of *out*:

case1. $\exists \xi^j : \xi^j \in out(i - \sigma)$ but $\xi^j \notin out(s - \sigma)$. Let t be the test checking the input at L_I^j after σ, then $\xi^j \in out(i - \sigma)$ implies $\exists i' : i \xrightarrow{\sigma} i' \xrightarrow{\xi^j} i'$. According to Algorithm 1 step 2, since $\xi^j \notin out(s - \sigma)$, we have ($t \xrightarrow{\bar{\sigma}} t' \xrightarrow{\theta_i^j}$ **fail**). So $\exists i' : t \| i \xrightarrow{\bar{\sigma}} t' \| i' \xrightarrow{\theta_i^j}$ **fail**$\| i'$, this means i **fails** t. (#) holds.

case2. $\exists y \in L_U^k : y \in out(i - \sigma)$ but $y \notin out(s - \sigma)$. Similar reasoning leads to (#).

case3. $\exists \delta^k : \delta^k \in out(i - \sigma)$ but $\delta^k \notin out(s - \sigma)$. Let t be the test checking the output in L_U after σ, then we have three facts (as are illustrated in Fig. 2):

I. $\delta^k \notin out(s - \sigma)$; II. $\delta^k \in out(i - \sigma)$; III. the test t: $t \xrightarrow{\bar{\sigma}} t' \xrightarrow{\theta_u}$ **fail**

Next we consider two cases about i and prove that “i **fails** t” always holds.

(1) The implementation i can suspend simultaneously at all the output channels, i.e., $\delta \in out_2(i - \sigma)$, then $\exists i' : i \xrightarrow{\sigma} i' \xrightarrow{\delta} i'$. So from III: $\exists i'$ and t' s.t. $t \| i \xrightarrow{\bar{\sigma}} t' \| i' \xrightarrow{\theta_u}$ **fail**$\| i'$, then i **fails** t.

(2) i cannot suspend at all the output channels, i.e., i must produce an output at some channel after performing σ:

$\forall i', (i \xrightarrow{\sigma} i' \xrightarrow{\delta^k})$ implies $(\exists k' \neq k$ *and* $y \in L_U^{k'} : i' \xrightarrow{y})$

From II, we know $\exists i' : i \xrightarrow{\sigma} i' \xrightarrow{\delta^k} i'$, so $\exists k' \neq k$ *and* $y \in L_U^{k'} : i' \xrightarrow{y}$ ....IV.

(2.1) If $y \notin out(s - \sigma)$, it can be proved i **fails** t, similar to above case2.

(2.2) Else $y \in out(s - \sigma)$, then from the fact I and the assumption that $\mathcal{F}$ satisfies $keep_{\mathcal{F}}(\delta^k)(k = 1, \dots, m)$, we have $\sigma \cdot y \in \mathcal{F}$.

By Algorithm 1, the $\mathcal{AOBS}$ test t must have a y transition after $\bar{\sigma} : t' \xrightarrow{y}$, as shown in Fig. 2.

Let $\sigma' = \sigma\cdot y$, we have the following facts resembling I, II and III:

I'. $\delta^k \notin out(s - \sigma')$ (By I and Proposition 3.1)

II'. $\delta^k \in out(i - \sigma')$ (By $i \xrightarrow{\sigma} i' \xrightarrow{\delta^k} i'$, IV and Proposition 3.2)

III'. the $\mathcal{AOBS}$ test t: $t \xrightarrow{\bar{\sigma}\cdot y} t'' \xrightarrow{\theta_u}$ **fail** (By I' and Algorithm 1)

Similar to the proof in (1) and (2) we may further grow the $\mathcal{AOBS}$ test t as: $t \xrightarrow{\bar{\sigma'}} t'' \xrightarrow{y'}$ where $k'' \neq k$ *and* $y' \in L_U^{k''}$, as shown in Fig. 2c.

For $\sigma'' = \sigma'\cdot y'$, continue with the proof like that for (1) and (2). Under the assumption that s is output-finite, the process must end at a trace $\sigma^0 \in \mathcal{F}$: $\sigma^0 = \sigma\cdot y\cdot y'\cdot\ldots$, where $y, y', \ldots$ are outputs at channels other than L_U^k. Then we have:

-I- $\delta^k \notin out(s - \sigma^0)$; -II- $\delta^k \in out(i - \sigma^0)$; -III- $t_0 \xrightarrow{\bar{\sigma^0}} t^0 \xrightarrow{\theta_u}$ **fail**

And one of the following two conditions must be true:

end-condition1: $\delta \in out_2(i - \sigma^0)$. Then by -III-: i **fails** t.

end-condition2: $\exists k^0 \neq k$ and $y^0 \in L_U^{k^0}$: $y^0 \in out(i - \sigma^0)$ but $y^0 \notin out(s - \sigma^0)$. Then t: $t^0 \xrightarrow{y^0}$ **fail**, and so i **fails** t.

Now, it can be concluded by case1 to case3 that: $\forall\sigma \in \mathcal{F}$, $out(i - \sigma) \not\subseteq out(s - \sigma)$ implies $\exists t \in T : i$ **fails** t, (#) always holds. □

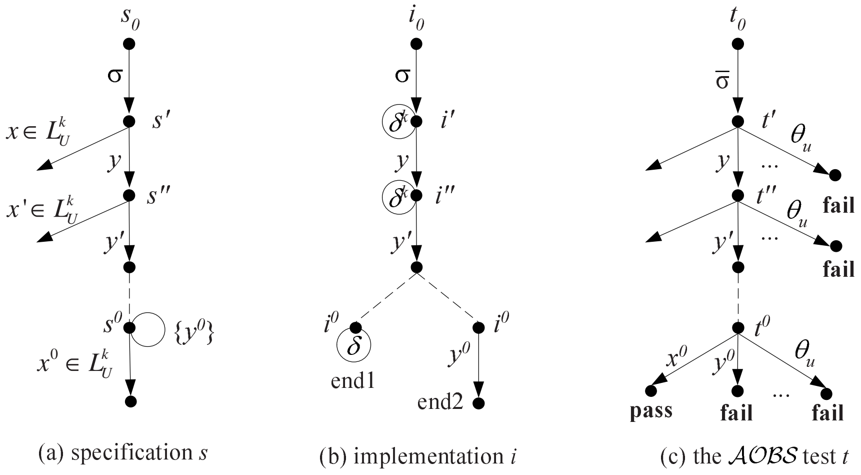

Fig. 2. growing of the $\mathcal{AOBS}$ test

The key in the proof is that, an unspecified single-channel output suspension, screened by a series of outputs at other channels, will manifest itself eventually when all the allowed outputs (they are finite, because the specification s is output-finite) at the interfering channels of the implementation have been produced and then only "unspecified" output or all-output-channel suspension can be produced by the implementation (these two cases both result in a **fail** verdict). Proposition 5 gives the sufficient condition to achieve this effect: to preserve single-channel output suspension through interfering outputs, the specification and implementation must be strace-reorderable; to detect the suspension eventually, all the allowed outputs at other channels must be further checked, which is guaranteed if $\mathcal{F}$ satisfies the *keep* condition.

Theorem 1. *Let $s \in \mathcal{LTS}(L_I \cup L_U)$, $i \in \mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U)$, s and i are all strace-reorderable, and s is ouput-finite, $\mathcal{F} \subseteq (L_\theta)^*$ and satisfies $keep_{\mathcal{F}}(\delta^k)(k = 1, \ldots, m)$,$T$ is the $\mathcal{AOBS}$ test suite generated by Algorithm 1, then T is complete for s w.r.t.* $\mathbf{mioco}_{\mathcal{F}}$.

This theorem is obvious from Propositions 2 and 5. By now, we have achieved the second goal set in Sect. 3.1. The exhaustiveness of all-observer tests is accomplished by making stronger test assumptions about the models of both specifications and implementations. This technique has often been used for test selection, or test-suite size reduction [8]. In practice, queue systems are strace-reorderable and often output-finite. Therefore, the all-observer test generation has promising application values.

3.5 Examples

In this section, we use some examples to illustrate the all-observer test generation algorithm.

Example 1. Figure 3a shows an strace-reorderable and output-finite specification $q \in \mathcal{MIOTS}(\mathcal{L}_I^\theta = \{\{a, \xi\}\}, \mathcal{L}_U^\theta = \{\{x, \delta^1\}, \{y, \delta^2\}\})$. After the input action a, q either produces x, or produces y, or produces x and y in an arbitrary order. τ denotes an internal action. Let $\mathcal{F} = \{a, a{\cdot}x, a{\cdot}y, a{\cdot}x \cdot y, a{\cdot}y{\cdot}x\}$. The $\mathcal{AOBS}$ and $\mathcal{SOBS}$ test suites generated by Algorithm 1 and the algorithm in [4] (after merging relevant tests) respectively are shown in Fig. 3b and Fig. 3c. Only output-checking tests are listed for comparison. The $\mathcal{SOBS}$ test suite contains six tests $t_1 \sim t_6$ (e.g. t_3 checks the output at the channel L_U^1 both after the trace a and after $a{\cdot}x{\cdot}y$, and also the output at the channel L_U^2 after $a{\cdot}x$). Since $\forall \sigma \in \mathcal{F} : \delta^k \in out(s{-}\sigma), keep_{\mathcal{F}}(\delta^k)(k = 1, \ldots, m)$ holds (by Propositions 4.1 and 4.3), by Theorem 1 we know that the $\mathcal{AOBS}$ test suite is complete. It contains only one test, but is testingly equivalent to the six $\mathcal{SOBS}$ tests.

Example 2. Also for q in Example 1, now let $\mathcal{F} = \{a, a{\cdot}\,\delta^1, a{\cdot}\delta^2\}$, the $\mathcal{SOBS}$ and $\mathcal{AOBS}$ test suites are shown in Fig. 3d. It can be verified that $\mathcal{F}$ satisfies $keep_{\mathcal{F}}(\delta^k)$ for each k in $\{1, \ldots, m\}$. First, we have $\delta^k \in out(s - a)$ for $k = 1, 2$, so $keep_{\{a\}}(\delta^k)$. Then, although $\delta^2 \notin out(s - a{\cdot}\delta^1)$, there isn't any $z \in L_U \backslash L_U^2$: $z \in out(s - a{\cdot}\delta^1)$; by Proposition 4.2 we have $keep_{\{a{\cdot}\delta^1\}}(\delta^2)$. Also, by the fact

$\delta^1 \in out(s - a \cdot \delta^1)$ and Proposition 4.1 we know $keep_{\{a \cdot \delta^1\}}(\delta^1)$. Similar arguments hold for $keep_{\{a \cdot \delta^2\}}(\delta^1)$ and $keep_{\{a \cdot \delta^2\}}(\delta^2)$. Therefore by Theorem 1 the $\mathcal{AOBS}$ test suite is complete. It contains three tests with the same testing power as the four $\mathcal{SOBS}$ tests (which are left to readers and not shown in Fig. 3). Furthermore, execution of t_1' always results in **pass** verdict and thereby can be removed.

We have illustrated Algorithm 1 by two examples where each $\mathcal{F}$ satisfies the exhaustiveness-preserving condition. In case $\mathcal{F}$ does not satisfy this condition, we must first expand it to get one that does. At the same time, we will obtain a stronger conformance relation than the original $\mathbf{mioco}_{\mathcal{F}}$.

4 Factorized All-Observer Test Generation

4.1 Factorized Test Generation

Definition 12. *The universe of selection processes over L_I is defined by*
$\mathcal{SLTS}(L_I) =_{def} \{p \in \mathcal{LTS}(L_I) | p \text{ is deterministic}\}$
Let $s \in \mathcal{LTS}(L_I \cup L_U)$ and $q \in \mathcal{SLTS}(L_I)$, then the transition system $s\|_{L_I} q \in \mathcal{SLTS}(L_I \cup L_U)$ is defined by the following inference rules.

$$\frac{s \xrightarrow{a} s', q \xrightarrow{a} q'}{s\|_{L_I} q \xrightarrow{a} s'\|_{L_I} q'} (a \in L_I) \qquad \frac{s \xrightarrow{x} s'}{s\|_{L_I} q \xrightarrow{x} s'\|_{L_I} q} (x \in L_U)$$

For a very large specification s, selection process q can be used to isolate a smaller specification $s\|_{L_I} q$ that contains only the responses to the input sequences specified in q, but discards all responses to other input sequences. The operator $\|_{L_I}$ imitates the synchronous communication operator $\|$ but only synchronizes input actions. $s\|_{L_I} q$ can be seen as the projected specification of s onto q.

[5] gives the soundness-preserving condition for the factorized test generation. For $s \in \mathcal{LTS}(L_I \cup L_U), q \in \mathcal{LTS}(L_I)$ and $\mathcal{F} \subseteq (L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^*$, if q satisfies the condition "$\mathcal{F}\lceil L_I \subseteq traces(q)$"($\mathcal{F}\lceil L_I =_{def} \{\sigma\lceil L_I | \sigma \in \mathcal{F}\}$), then a sound test suite of $s\|_{L_I} q$ with respect to $\mathbf{mioco}_{\mathcal{F}}$ is also sound for s with respect to $\mathbf{mioco}_{\mathcal{F}}$. [5] also gives the completeness-preserving condition for the factorized test generation. A boolean predicate is defined as $accept_q(\sigma) =_{def} \forall \sigma' \in pref(\sigma), \exists q'(q \xrightarrow{\sigma'} q') : q' \xrightarrow{a} (\forall a \in L_I)$. If q satisfies the condition "$\forall \sigma \in \mathcal{F} : accept_q(\sigma\lceil L_I)$", then a complete test suite of $s\|_{L_I} q$ with respect to $\mathbf{mioco}_{\mathcal{F}}$ is also complete for s with respect to $\mathbf{mioco}_{\mathcal{F}}$.

Another problem in test generation for $\mathbf{mioco}_{\mathcal{F}}$ is that $\mathcal{F}$ may contain many traces, which means both time and space challenging to test generation tools. A feasible way is to decompose the correct criterion $\mathbf{mioco}_{\mathcal{F}}$ into smaller ones.

Proposition 6.
1. $\mathbf{mioco}_{\mathcal{F}} = \bigcap_{\sigma \in \mathcal{F}} \mathbf{mioco}_{\{\sigma\}}$
2. $\mathbf{mioco}_{\mathcal{F}} = \mathbf{mioco}_{\mathcal{F}_1} \bigcap \mathbf{mioco}_{\mathcal{F}_2} \bigcap \ldots \bigcap \mathbf{mioco}_{\mathcal{F}_n}$ $(\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_n)$

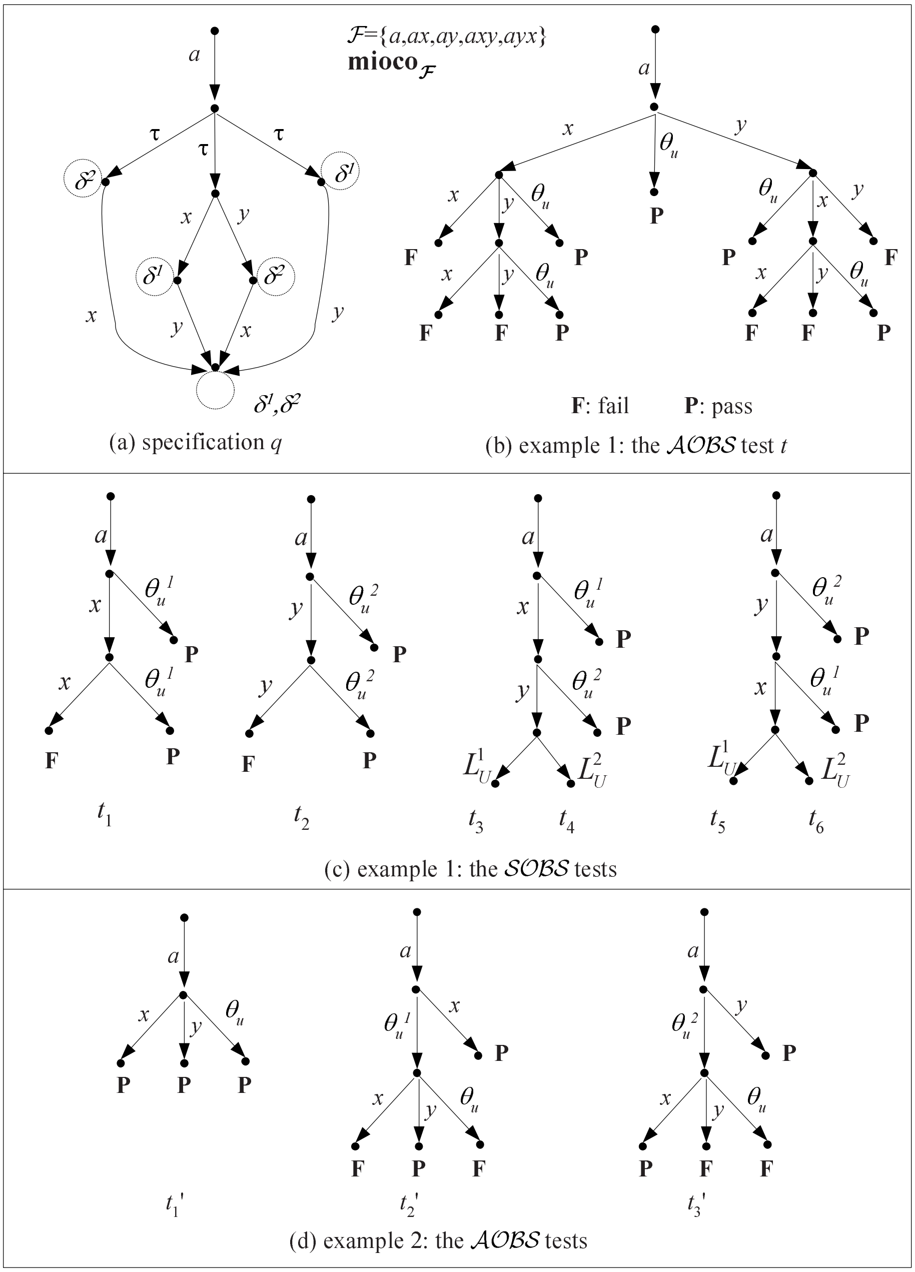
$\mathcal{F}=\{a,ax,ay,axy,ayx\}$
$\mathbf{mioco}_{\mathcal{F}}$
(a) specification q
(b) example 1: the $\mathcal{AOBS}$ test t
F: fail P: pass
(c) example 1: the $\mathcal{SOBS}$ tests
(d) example 2: the $\mathcal{AOBS}$ tests

Fig. 3. All-observer test generation

By Proposition 6.1, $\mathbf{mioco}_{\mathcal{F}}$ can be splitted into a set of smaller relations $\mathbf{mioco}_{\{\sigma\}}(\sigma \in \mathcal{F})$ for which the correctness check may be performed independently. For each of these small criteria, it suffices to take a selection process q that satisfies either the soundness-preserving or the completeness-preserving condition to generate a (sound or complete) test suite for $s\|_{L_I}q$ and $\mathbf{mioco}_{\{\sigma\}}$, respectively. These test suites are then combined into one test suite, which is sound or complete in testing implementations for its $\mathbf{mioco}_{\mathcal{F}}$-relation with s. This method was proposed in [5] and called factorized test generation. The test generation algorithm is the one in [4] and it generates $\mathcal{SOBS}$ tests.

To partition $\mathcal{F}$ into a set of singletons reduces the calculation complexities at furthest, but it is very inefficient due to the large number of traces contained in $\mathcal{F}$: an individual application of the selection process is needed for each σ in $\mathcal{F}$, even for the traces sharing the same selection process and the same projected specification. To overcome this problem, we need a coarser partitioning of $\mathcal{F}$.

Definition 13. *Let $\sigma, \sigma' \in (L_I \cup L_U \cup \mathcal{L}_I \cup \mathcal{L}_U)^*$, we say σ is affinal to σ' if: $\sigma\lceil L_I = \sigma'\lceil L_I$, denoted by: $\sigma \perp \sigma'$.*

affinal ($\perp$) is an equivalence relation and $\mathcal{F}$ forms a partition over $\perp$: $\mathcal{F}_\perp = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h\}$. $\mathcal{F}_i\lceil L_I$ contains only one trace, denoted as σ_i. By Proposition 6.2, we know that $\mathbf{mioco}_{\mathcal{F}}$ can be decomposed into a set of criteria $\mathbf{mioco}_{\mathcal{F}_i}(i = 1, \ldots, h)$. Traces in $\mathcal{F}_i$ have the same projection on L_I; thus they share the same selection process (e.g. $stick(\sigma_i)$ or $fan(\sigma_i)$, two kinds of selection processes defined in [5]). Now we generate a test suite for each $s\|_{L_I}q_i$ and $\mathbf{mioco}_{\mathcal{F}_i}$ for $i = 1, \ldots, h$, where q_i is constructed in line with σ_i. This improvement can greatly reduce the frequency of calculating projected specifications. Furthermore, such trace grouping helps to merge test cases that check the failure traces having common prefixes, and to reduce the size of the final composed test suite. This improvement is just an optimization for the factorized $\mathcal{SOBS}$ test generation, but for factorized $\mathcal{AOBS}$ test generation using Algorithm 1, it is indeed necessary, as we show in the next section.

4.2 Factorized All-Observer Test Generation

Using Algorithm 1 for factorized test generation we get $\mathcal{AOBS}$ test suites. Will they still be sound or complete under the respective condition established in Sect. 4.1? Proposition 2 shows that Algorithm 1 generates sound $\mathcal{AOBS}$ test suites for general MIOTS systems; so does it for projected specifications, which are certainly MIOTS systems. Therefore, Algorithm 1 can be used to generate a sound $\mathcal{AOBS}$ test suite for each $\mathbf{mioco}_{\mathcal{F}_i}$ and $s\|_{L_I}q_i$ ($\mathcal{F}_i \in \mathcal{F}_\perp, i = 1, \ldots, h$), where q_i satisfies $\sigma_i \in traces(q_i)$. All these test suites are united into one test suite, which is sound for s with respect to $\mathbf{mioco}_{\mathcal{F}}$.

Proposition 5 specifies the requirements imposed on the specification s, the implementation i and $\mathcal{F}$ for Algorithm 1 to generate an exhaustive $\mathcal{AOBS}$ test suite for s with respect to $\mathbf{mioco}_{\mathcal{F}}$. To apply this proposition to factorized $\mathcal{AOBS}$ test generation, we must prove that these requirements are still satisfied by both the projected specifications and $\mathcal{F}_i$ in the smaller criterion $\mathbf{mioco}_{\mathcal{F}_i}$. Ideally, all of the following three guesses should true:

guess1. the projected specifications $s\|_{L_I}q_i$ are output-finite

guess2. the projected specifications $s\|_{L_I}q_i$ are strace-reorderable

guess3. for each $\mathcal{F}_i \in \mathcal{F}_\perp$ and the projected specification $s\|_{L_I}q_i$: $keep_{\mathcal{F}_i}(\delta^k)(k = 1, \ldots, m)$

It is obvious that guess1 is true if the specification s is output-finite, because any selection process only "copies" the original outputs but never adds. However, guess2 is not hinted by the assumption that s is strace-reorderable. Nevertheless, if q satisfies $accept_q(\sigma\lceil L_I)$, $s\|_{L_I}q_i$ will have properties similar to Proposition 3.

Proposition 7. *Let* $s \in \mathcal{LTS}(L_I \cup L_U)$ *be strace-reorderable,* $q \in \mathcal{SLTS}(L_I)$, $p \equiv s\|_{L_I}q$, $x \in L_U^{k'}$, $k \neq k'$. *For* $\forall\sigma \in (L_\theta)^*$, *if* $accept_q(\sigma\lceil L_I)$, *then*

1. $\delta^k \notin out(p - \sigma)$ *implies* $\delta^k \notin out(p - \sigma{\cdot}x)$

2. $\forall p', p \xrightarrow{\sigma} p' : (p' \xrightarrow{\delta^k}$ *and* $p' \xrightarrow{x})$ *implies* $\delta^k \in out(p - \sigma{\cdot}x)$

This proposition can be proved using the fact that $accept_q(\sigma\lceil L_I)$ implies $out(s - \sigma) = out(p - \sigma)$ and $out(s - \sigma{\cdot}x) = out(p - \sigma{\cdot}x)$ (note that $\sigma \perp \sigma{\cdot}x$, so $accept_q(\sigma \cdot x\lceil L_I)$. Then $\delta^k \notin out(p - \sigma)$ implies $\delta^k \notin out(s - \sigma)$. From Proposition 3 and the assumption s is strace-reorderable, we have $\delta^k \notin out(s - \sigma{\cdot} x)$, so also $\delta^k \notin out(p - \sigma{\cdot}x)$. Similar reasoning leads to part 2 of the proposition. These properties of $s\|_{L_I}q$ expressed by Proposition 7 are really what matter in the proof of exhaustiveness of the $\mathcal{AOBS}$ test suites in Proposition 5. It doesn't matter that $s\|_{L_I}q$ is not strace-reorderable.

Proposition 8. *Let* $s \in \mathcal{LTS}(L_I \cup L_U), q \in \mathcal{SLTS}(L_I), \mathcal{F} \subseteq (L_\theta)^*$ *and* $\forall\sigma \in \mathcal{F}$: $accept_q(\sigma\lceil L_I), p \equiv s\|_{L_I}q$. *Then for each* k *in* $\{1, \ldots, m\}$*: "*$keep_{\mathcal{F}}(\delta^k)$ *holds for* s*" iff "*$keep_{\mathcal{F}}(\delta^k)$ *holds for* p*".*

Proposition 9. *Let* $s \in \mathcal{LTS}(L_I \cup L_U)$*, and* $\mathcal{F}_\perp = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h\}$. *If* $\mathcal{F}$ *satisfies* $keep_{\mathcal{F}}(\delta^k)$ *for* s, $\mathcal{F}_i$ *will also satisfy* $keep_{\mathcal{F}_i}(\delta^k)$ *for* s*:*

$$\forall\delta^k(k = 1, \ldots, m) : keep_{\mathcal{F}}(\delta^k) \textit{ implies } keep_{\mathcal{F}_i}(\delta^k)(i = 1, \ldots, h)$$

Proposition 9 can be proved easily using Def. 11 and the fact $\sigma \perp \sigma{\cdot}y$. It means that the predicate $keep_{\mathcal{F}_i}(\delta^k)$ is preserved in the affinal partition of $\mathcal{F}$. Proposition 8 can be proved using Def. 11 and the equation $out(s - \sigma) = out(p - \sigma)$. It means that the predicate is preserved in the projected specification. From these two propositions, it can be concluded: given that $\mathcal{F}$ satisfies $keep_{\mathcal{F}}(\delta^k)$ for s, we have $\forall\mathcal{F}_i \in \mathcal{F}_\perp : keep_{\mathcal{F}_i}(\delta^k)(k = 1, \ldots, m)$ hold for each projected specification $s\|_{L_I}q_i$. q_i is the selection process satisfying $accept_{q_i}(\sigma_i\lceil L_I)$ where $\mathcal{F}_i\lceil L_I = \sigma_i$. This is guess3. Now, completely analogous to the process of proving Proposition 5, it can be shown that Algorithm 1 generates a complete $\mathcal{AOBS}$ test suite for each of the projected specification $s\|_{L_I}q_i$ with respect to the corresponding decomposed criterion $\mathbf{mioco}_{\mathcal{F}_i}$, if only the original requirements imposed on the specifications s, the implementation i and the set of failure traces $\mathcal{F}$ are satisfied, and also a selection process q_i complying with the completeness-preserving condition is used. Combining this result with the fact that $\mathbf{mioco}_{\mathcal{F}}$ can be decomposed according to the affinal partition of $\mathcal{F}$, we get Theorem 2.

Theorem 2. *Let $s \in \mathcal{LTS}(L_I \cup L_U)$, $i \in \mathcal{MIOTS}(\mathcal{L}_I, \mathcal{L}_U)$, s and i are all strace-reorderable, and s is ouput-finite, $\mathcal{F} \subseteq (L_\theta)^*$ and satisfies $keep_{\mathcal{F}}(\delta^k)(k = 1, \ldots, m)$. $\mathcal{F}_\perp = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h\}, \mathcal{F}_i\lceil L_I = \{\sigma_i\}$. For $i = 1, \ldots, h$, let $q_i \in \mathcal{SLTS}(L_I)$ and satisfies $accept_{q_i}(\sigma_i)$, let $p_i \equiv s\|_{L_I} q_i$, and $T_{\mathcal{F}_i}(p_i)$ is the $\mathcal{AOBS}$ test suite generated by Algorithm 1 for p_i w.r.t. $\mathbf{mioco}_{\mathcal{F}_i}$, Then*

$\bigcup_{\mathcal{F}_i \in \mathcal{F}_\perp} T_{\mathcal{F}_i}(p_i)$ is complete for s w.r.t. $\mathbf{mioco}_{\mathcal{F}}$.

Consequently, for strace-reorderable specifications and implementations, if the specification is output-finite, Algorithm 1 can be used for the factorized generation of complete $\mathcal{AOBS}$ test suites, following the same rule that governs the factorized generation of $\mathcal{SOBS}$ tests. Thus a single framework exists for the factorized generation of both singular-observer and all-observer tests.

5 Conclusions and Future Work

In this paper we present a new method of testing MIOTS, viz. with a kind of observers called all-observer. The all-observer is superior to singular observers in the sense that they can observe the responses of the implementation at all the output channels simultaneously in a testing process. As a result, a test suite consisting of all-observer tests is often much smaller than the one consisting of singular observer tests. This means time-savings in both test generation and test execution. However, the discriminating power of all-observer is generally weaker than that of singular observers, because it may fail to capture the unspecified single-channel output suspensions. We give a test generation algorithm to derive all-observer tests for a specification s with respect to the relation $\mathbf{mioco}_{\mathcal{F}}$. The algorithm can generate a sound test suite for any MIOTS systems, but can only generate a complete test suite for a subset of the MIOTS systems: *strace-reorderable*, which characterizes the independence of the output behaviors occurring at different channels, and *output-finite*, which is only required for the specification but not for the implementation.

We then studied the problem of factorized all-observer test generation. For the decomposition of the correctness criterion, an optimized $\mathcal{F}$-partition technique is proposed to take advantage of the fact that many traces in $\mathcal{F}$ may share the same selection process. For the reduction of the specification, it is proved that the soundness and completeness conditions in factorized singular-observer test generation are also valid for the factorized all-observer test generation.

In theory, the all-observer complements the singular observers in testing MIOTS. In practice, queue systems is a possible domain applying the all-observer testing. Finally, in case an MIOTS is strace-reorderable, the trace set $\mathcal{F}$ may be reduced according to the strace-reordering relation without weakening the correct criterion. This is to be studied in future.

References

1. R. De Nicola and M.C.B. Hennessy. Testing equivalences for processes. Theoretical Computer Science, 34:83–133, 1984.
2. R. De Nicola. Extensional equivalences for transition systems. Acta Informatica, 24:211–237, 1987.
3. J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. Software — Concepts and Tools, 17(3):103–120, 1996.
4. L. Heerink and J. Tretmans. Refusal testing for classes of transition systems with inputs and outputs. In FORTE X/PSTV XVII'97, pp. 23–38. 1997.
5. E. Brinksma, L. Heerink, and J. Tretmans. Factorized test generation for multi-input/output transition systems. TestCom'98. pp. 67–82. 1998.
6. I. Phillips. Refusal testing. Theoretical Computer Science, 50(2):241–284, 1987.
7. J. Tretmans, L. Verhaard. A queue model relating synchronous and asynchronous communication. In PSTV'92, pp. 131–145. 1992.
8. J. Tretmans. A formal approach to conformance testing. In Sixth International Workshop on Protocol Test Systems, pp. 257–276. 1994.