Science of Computer Programming 239 (2025) 103173

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

# Model independent refusal trace testing

Maciej Gazda [*], Robert M. Hierons

*Department of Computer Science, The University of Sheffield, Sheffield, S1 4DP, UK*

## A B S T R A C T

Software Testing is normally one of the main forms of verification and validation used in software development but it is often manual and so expensive and error prone. One of the proposed solutions to this is to use model-based testing, in which testing is based on a model of how the system should behave. If the model has a formal semantics, then there is potential to automate systematic test generation. In this paper we consider the case where the semantics of the model is a set of refusal traces, also called failure traces. We show how the notions of fundamental refusal and fundamental refusal trace can be used to derive a normalised transition system, which we call an *observation transition system* (OTS), from the semantics. We then show how, if this OTS has finitely many states, and we are given a bound $m$, one can produce a corresponding complete test suite: one that is guaranteed to determine correctness as long as the number of states of the OTS defined by the semantics of the system under test has no more than $m$ states. In practice, the choice of value for $m$ might be based on domain knowledge or a cost-benefit analysis. As far as we are aware, this is the first work to show how a finite complete test suite can be derived when the semantics under consideration is a set of refusal traces.

## 1. Introduction

Software testing is the process of executing the *system under test (SUT)* with test cases and checking that the observed behaviours are allowed by the specification. Software testing is typically one of the main verification and validation approaches used in software development. Even if formal verification is used (and this certainly has merit!), such verification only ever checks that a model of the system behaves correctly. For example, if the source code is verified against a specification then verification assumes that the compiler and hardware behave as expected; one should still test. In addition, where a system is black-box (we do not have access to the code), it typically is not possible to apply formal verification techniques.

Unfortunately, software testing is often a manual process and so expensive and error prone. This issue has been addressed in *model based testing (MBT)*, where testing is based on a model or specification of the required behaviour of the SUT. Ideally, the model used has a formal semantics and then there is potential to reason about test effectiveness [5,12]. Such reasoning requires one to assume that the SUT behaves like an unknown model $I$ and typically one also assumes that $I$ can be described using the same formalism as the specification (the minimum hypothesis [5]). Testing can then be seen as a process of executing the SUT in order to compare two models: the specification $S$ and an unknown model $I$ of the SUT and there is potential to automate systematic test generation (see, for example, [10,23]).

Much of the MBT work concerns state-based systems that have an internal state and so behaviours are sequences (traces). Many systems are state-based, with examples including communications protocols, CPUs, and embedded control systems used in areas such as avionics and the automotive industry. Testing is then normally based on models expressed as either a *finite state machine (FSM)* or *labelled transition system (LTS)*; software development might use a higher-level/more abstract formalism whose semantics is expressed as an LTS or FSM (see, for example, [3,9]).

* Corresponding author.
*E-mail addresses:* m.gazda@sheffield.ac.uk (M. Gazda), r.hierons@sheffield.ac.uk (R.M. Hierons).

https://doi.org/10.1016/j.scico.2024.103173
Received 25 October 2023; Received in revised form 5 July 2024; Accepted 10 July 2024

Available online 23 July 2024

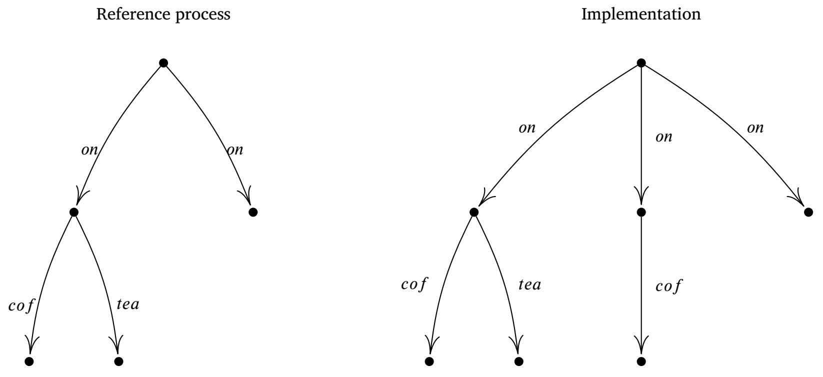


**Fig. 1.** Specification of a simple coffee machine (left) and an incorrect implementation w.r.t. refusal trace semantics (right).

Most work on FSM-based testing has concerned test generation algorithms, with FSM-based testing research going back to work by Moore in the 1950s [17]. The early work concerned deterministic FSMs (DFSMs). The semantics of an FSM is given by the corresponding regular languages (all states are 'final' states and so the language is prefix closed). In order to reason about test effectiveness, most work on testing from an FSM $M$ assumes that the SUT behaves like an unknown FSM $M_I$ that has the same input and output alphabets as $M$. Testing thus involves choosing a set of input sequences and then applying them to the SUT in order to check whether $M$ and $M_I$ agree on these input sequences.

In FSM testing, there has been significant interest in test generation techniques that are *complete* in the sense that the resultant test suites are guaranteed to determine whether the SUT is correct as long as the SUT satisfies a stated test hypothesis. The test hypothesis is usually either that the FSM $M_I$ representing the SUT has no more states than the specification $M_S$ or there is a known upper bound $m$ on the number of states of $M_I$. There are FSM-based test generation algorithms that return finite complete test suites (see, for example, [4,11,16,24,26]). Possibly the best known such technique for testing from a DFSM is the W-method [4,26]. Although this technique was devised for testing from a DFSM, we will see that it can be extended to LTSs.

It is worth briefly explaining why we chose to adapt the W-method, despite there being many other DFSM-based test techniques. First, we wanted to adapt a test generation technique that returns a complete test suite and is general in the sense that it can be used to test from any (complete, minimal) DFSM. There are several such test generation techniques (see, for example, [16,20]). However, the W-method is arguably the simplest of these techniques, with its use avoiding additional complexity. In addition, the core building blocks of the W-method (reaching and separating states) are used in many other FSM-based test generation techniques. Thus, it should be possible to extend the approach taken in this paper to adapt other DFSM-based test generation techniques.

In FSM-based testing, an observation is a trace: a sequence of events. LTS-based work, however, often allows the tester to observe the *refusal* of a set $X$ of events [25,19,23]; a refusal occurs if the LTS is in a (stable) state in which none of the events in $X$ can occur. A tester can observe the refusal of $X$ by offering the events in $X$ and observing a deadlock, with the deadlock normally being observed through a timeout. There is a range of implementation relations, which state how the behaviours (possible observations) of the SUT and specification must relate in order for the SUT to be correct [8]. The implementation relation used in testing will depend upon the types of observations that a tester, or system environment/user, can make. In this paper we assume that an observation is a refusal trace, also called a failure trace: a sequence that includes events and refusals [22]. It has been noted that this provides the finest implementation relation that is consistent with testing [25]; stronger implementation relations require backtracking and normally this is not possible in testing. Refusal traces are particularly important in the context of (discrete) timed systems, where it is essential that refusal information is recorded at every point where time progresses [1,18].

In this paper we consider the problem of testing to determine whether the sets of refusal traces $\mathcal{L}_S$ and $\mathcal{L}_I$, of the specification and SUT respectively, are the same.

A simple example, depicted in Fig. 1, demonstrates the distinguishing capability of refusal trace semantics. The specification on the left is a simple coffee machine that, after being activated with the *on* button, either offers the user a choice between coffee and tea, or no beverage at all (the latter in case when the user is e.g. not authorised or payment has not been made – for simplicity, we abstract from this aspect here). Refusal trace semantics allows us to specify the requirement that a correct machine must always offer (to an authorised user) both coffee and tea; indeed, the implementation on the right that may in some circumstance offer only coffee, is incorrect, since it has a refusal trace $on.\{tea\}.cof$ that does not belong to the specification. We note that the above processes are equivalent w.r.t. coarser semantics such as failures or traces.

We take inspiration from the work of Huang and Peleska [14] that concerns traces-based testing. Their approach maps a set of traces (the semantics of the specification or SUT) $\mathcal{L}$ to a canonical LTS. They use the Nerode-congruence: two traces $\sigma_1$ and $\sigma_2$ reach the same state of the induced LTS if and only if $\sigma_1$ and $\sigma_2$ have the same continuations in $\mathcal{L}$. Testing then involves comparing the corresponding LTSs for $\mathcal{L}_S$ and $\mathcal{L}_I$. They observe that an advantage of this approach is that it is based on the semantics of the specification and not the form of its model; this contrasts with other techniques that would produce different test suites from different specifications even if these specifications had the same semantics. They then show how, if the derived LTS is finite, the classical W-method [4,26], for testing from an FSM, can be extended to derive finite complete test suites.

The aim of this paper is to generalise the approach of Huang and Peleska to the case where observations are refusal traces. An initial challenge was that it is not possible to directly use the Nerode-congruence because the set of continuations of a refusal trace $\sigma$ might not correspond to any possible state: as a result of non-determinism, it might be possible, in language $\mathcal{L}$ under consideration, to follow $\sigma$ by event $a$ and also by the refusal of $a$. In order to address this issue, we draw on the recently introduced notion of a *fundamental refusal*: $X$ is a fundamental refusal after $\sigma$ if $\sigma.X$ is in $\mathcal{L}$ and for every event $a \notin X$ we have that $\sigma.X.a$ is also in $\mathcal{L}$ [6,7]. We used fundamental refusal traces, which are refusal traces in which all refusals are fundamental, as the basis for deriving a special normalised transition structure, called an *observation transition system* $\mathcal{OTS}(\mathcal{L})$, from a language $\mathcal{L}$. Conceptually, OTS is to a certain extent similar to the language LTS of Huang and Peleska [14], however, its structure corresponds more closely to the testing experiment one can perform on the correct/canonical LTS.

Having shown how $\mathcal{OTS}(\mathcal{L})$ can be derived, the next problem is to show how a complete test suite can be generated from $\mathcal{OTS}(\mathcal{L}_S)$, where $\mathcal{L}_S$ is the language (set of refusal traces) of the specification. Similar to FSM-based testing, we use the test hypothesis that the derived OTS of the language $\mathcal{L}_I$ of the SUT has at most $m$ states for some given $m$. We show how the concepts of state cover and characterising set, used in the W-method for testing from an FSM [4,26], can be extended to our setting if $\mathcal{OTS}(\mathcal{L}_S)$ has finitely many states. These are then used in order to construct a complete test suite, with us separately considering two cases: $m = n$, where $n$ is the number of states of $\mathcal{OTS}(\mathcal{L}_S)$, and $m > n$. Finally, we then show how we can considerably reduce the test suite size by restricting it to traces whose prefixes are fundamental refusal traces of the specification.

Similar to FSM-based testing, the tester will be faced with the standard problem of choosing a value of $m$ to use in test generation. Such a choice can be based on a mixture of domain knowledge and also a cost-benefit assessment, with both the effectiveness and cost of testing increasing as $m$ increases.

This paper makes several contributions.

- We develop a theory of refusal traces based solely on healthy languages of refusal traces (sets of refusal traces representing semantics of valid computational models). In particular:
  - we define a local equivalence that identifies refusals in the context of a specific trace (adapting the theory of [6,7] to pure language-based setting) (Section 3.1)
  - we provide a language state space construction for healthy languages of refusal traces (based broadly on the suffix language solution [14] but more involved due to the presence of refusals) (Section 3.2)
- We draw on FSM-based test generation techniques (Section 5).
  - These allow one to produce a finite set $T$ of finite behaviours (refusal traces), with this finite set acting as a complete test suite (one guaranteed to determine correctness as long as the SUT satisfies the stated test hypotheses).
  - FSM-based test generation techniques are defined for traces and so there is the challenge of generalising them to refusal traces.
  - The tests contain both positive test cases (behaviours that the SUT should have) and negative test cases (behaviours that the SUT should not have).
  - Using fundamental refusal traces, one can substantially reduce the size of a test suite.

*Related work* As far as we are aware, this is the first work that directly shows how a *finite* complete test suite can be constructed when testing for *refusal trace equivalence*. Note that [6,7] also provides a (minimal) complete test suite construction, however, it differs in two important aspects: the conformance is refinement rather than equivalence, and it is parameterised by a bound on the trace length (i.e. refinement up-to-$K$ steps) whereas in this work, the conformance relation itself is not bounded, but bounds are assumed on the specification and implementation. While in theory one could decide equivalence by performing two refinement checks, it is undesirable for reasons of efficiency, in particular in the realm of testing where there is asymmetry in access between the specification and implementation. Because of this, test suites in [6,7] can only check whether an implementation refines the specification, but not the other way round.

As mentioned, a closely related work is [14], where the conformance relation is trace equivalence (i.e. based on sequences of events with no refusals). In [19], finite complete test suites are constructed for trace and failures refinement (the latter is based on sequences of traces that potentially end in refusals). Complete test suites for refusal trace refinement with inputs and outputs has been provided in [3]; however, they are not finite. Regarding ioco work, test suites for a relation similar to refusal trace refinement, mioco, have been defined in [10]; they are, however, not finite for the general (unbounded) conformance.

## 2. Preliminaries

Here, we briefly introduce our semantic model (labelled transition systems), and refusal trace semantics. We shall use $\Sigma$ to denote the alphabet of visible actions. A *labelled transition system* (LTS) is a tuple $\mathcal{L} = \langle S, \rightarrow, \Sigma \cup \{\tau\}\rangle$ where $S$ is a finite set of states, $\Sigma$ a finite set of visible/external actions, $\tau$ a special silent/internal action, and $\rightarrow \subseteq S \times \Sigma \cup \{\tau\} \times S$ a transition relation. We use the shorthand notation $s \xrightarrow{\alpha}$ [resp. $s \not\xrightarrow{\alpha}$ ] whenever there exists [does not exist] a state $s'$ such that $s \xrightarrow{\alpha} s'$. A state $s$ is *stable* if $s \not\xrightarrow{\tau}$.

We use the notation $\overset{\epsilon}{\Longrightarrow}$ for the reflexive transitive closure of $\xrightarrow{\tau}$, i.e. $s \overset{\epsilon}{\Longrightarrow} s'$ iff $s = s'$ or there is a chain of states and transitions $s = s_0 \xrightarrow{\tau} \dots \xrightarrow{\tau} s_n = s'$. Moreover, $s \overset{a}{\Longrightarrow} s'$ denotes that $s \overset{\epsilon}{\Longrightarrow} t \xrightarrow{a} t' \overset{\epsilon}{\Longrightarrow} s'$ for some states $t, t'$.

A *process* can be defined by indicating a set of its possible initial states within a certain LTS. Formally, a process is a tuple $P = \langle S_I, \mathcal{L}\rangle$, where $\mathcal{L} = \langle S, \rightarrow, \Sigma \cup \{\tau\}\rangle$ and $S_I \subseteq S$. Typically, the underlying LTS should be clear from the context and we shall identify a process $P$ with the set $S_I$. Moreover, the *internal closure* of $P$, notation $\mathrm{I}_\tau(P)$, is the set of states reachable from initial states of $P$ with internal actions, i.e. $\mathrm{I}_\tau(P) = \{s' \mid \exists s \in P : s \overset{\epsilon}{\Longrightarrow} s'\}$ [here, we identify the process $P$ with its set of initial states].

An important assumption that we make throughout the paper is *divergence-freedom*, that is, in all systems under consideration we assume that there are no infinite paths of $\tau$-labelled transitions. This is a reasonable assumption since: 1) divergence in a specification normally represents a fault; and 2) for a system under test, divergence looks the same[1] (to the tester) as deadlock and so is not needed.

For a stable state $s$, a set $X \subseteq \Sigma$ is a *stable refusal*, or simply *refusal* of $s$, if for all $a \in X$, $s \overset{a}{\not\longrightarrow}$. The set of all refusals of $s$ is denoted with $\mathbf{R}(s)$. Note that $\mathbf{R}(s) = \emptyset$ for unstable states. The *state refusal* of $s$ is the largest refusal of $s$, denoted with $\mathbf{SR}(s)$.

Since all refusals $X \subseteq \Sigma$, including $\emptyset$, can be observed in stable states only, an additional refusal observation is required that can be made in any state (including unstable states). We shall use $\bullet$ to denote such null refusal observation. A *refusal trace (failure trace)* $\sigma$ of a state $s$ is a sequence that is either an empty word $\epsilon$, or a word of the form $X_0 a_1 X_1 a_2 \dots X_{n-1} a_n X_n$, or $X_0 a_1 X_1 a_2 \dots X_{n-1} a_n$, where $a_i$ range over $\Sigma$ and $X_i$ range over $\mathcal{P}(\Sigma) \cup \{\bullet\}$, such that there is a chain of transitions $s \overset{\epsilon}{\Longrightarrow} s_0 \overset{a_1}{\Longrightarrow} s_1 \overset{a_2}{\Longrightarrow} \dots \overset{a_{n-1}}{\Longrightarrow} s_{n-1} \overset{a_n}{\Longrightarrow} s_n \overset{\epsilon}{\Longrightarrow} q$ and for all $k \in \{0, \dots, n-1, [n]\}$, $X_k \in \mathbf{R}(s_k)$ or $X_k = \bullet$. We denote the existence of such a chain of transitions by $s \overset{\sigma}{\Longrightarrow} q$.

We define the *length* of a refusal trace in one of the above forms as *the number of refusals occurring in the trace*, that is

$$\begin{aligned} |\epsilon| &\triangleq 0 \\ |X_0 a_1 \dots X_{n-1} a_n| &\triangleq n \\ |X_0 a_1 \dots X_{n-1} a_n X_n| &\triangleq n+1 \end{aligned}$$

Observe that in particular the length of a trace ending in refusal $X_n$ is $n+1$.

The set of refusal traces originating from a state $s$ [of length $\leq l$] is denoted with $\mathbf{RT}(s)$ [resp. $\mathbf{RT}^l(s)$]; the notation is lifted to processes. The language of *well-formed refusal traces* over alphabet $\Sigma$ is defined as:

$$\begin{aligned} \mathbf{RT}[\Sigma] \triangleq \{\epsilon\} \cup \{&\sigma \in (\mathcal{P}(\Sigma) \cup \{\bullet\}) \times (\Sigma \times (\mathcal{P}(\Sigma) \cup \{\bullet\}))^* \cup ((\mathcal{P}(\Sigma) \cup \{\bullet\}) \times \Sigma)^* \\ &\mid \sigma = \rho.X_i.a_{i+1}.\rho' \Longrightarrow X_i = \bullet \vee a_{i+1} \notin X_i\} \end{aligned}$$

The alphabet annotation will often be dropped when the alphabet is clear from the context. In addition, we define the following subclasses of refusal traces, determined by the type of the terminal symbol.

$$\mathbf{RT}_{\mathsf{R}}[\Sigma] \triangleq \{\pi.X \in \mathbf{RT}[\Sigma]\} \qquad \mathbf{RT}_{\mathsf{A}}[\Sigma] \triangleq \{\epsilon\} \cup \{\pi.a \in \mathbf{RT}[\Sigma]\}$$

Refusal traces in $\mathbf{RT}_{\mathsf{R}}[\Sigma]$ [$\mathbf{RT}_{\mathsf{A}}[\Sigma]$] will also be referred to as *proper* [*partial*] refusal traces. Sometimes we shall also use the following type of suffixes:

$$\mathbf{RT}_{\mathsf{A\text{-}SUF}} \triangleq \{\lambda \mid \exists \pi \in \mathbf{RT}_{\mathsf{R}} \mid \pi.\lambda \in \mathbf{RT}_{\mathsf{R}}\}$$

Note that, since $\tau$ represents an unobservable event, all observations of a process $P$ are precisely those of its internal closure $\mathrm{I}_\tau(P)$ – the two processes are indistinguishable in this observational model. Moreover, we use internal closure to formally lift the notions of refusal and refusal trace from states to processes:

$$\begin{aligned} \mathbf{R}(P) &\triangleq \textstyle\bigcup_{q \in \mathrm{I}_\tau(P)} \mathbf{R}(q) \\ \mathbf{RT}(P) &\triangleq \textstyle\bigcup_{q \in \mathrm{I}_\tau(P)} \mathbf{RT}(q) \end{aligned}$$

A process $Q$ is a *refusal trace refinement* of a process $P$, notation $P \sqsubseteq_{\mathbf{RT}} Q$, iff $\mathbf{RT}(Q) \subseteq \mathbf{RT}(P)$. $P$ and $Q$ are *refusal trace equivalent*, notation $P =_{\mathbf{RT}} Q$ iff $\mathbf{RT}(P) = \mathbf{RT}(Q)$.

For a process $P$ and refusal trace $\sigma$, the process $P$ after $\sigma$, denoted with $P \,|\, \sigma$, is the set $\{q \in S \mid \exists s \in P : s \overset{\sigma}{\Longrightarrow} q\}$. On a formal note, observe that for all $P$ we have $P \,|\, \epsilon = \mathrm{I}_\tau(P)$.

**Example 2.0.1.** Consider the LTS from Fig. 2, representing a process $P$. Refusals of the process $P$ are all subsets of $\{b, c\}$, i.e. $\mathbf{R}(P) = \{\{b, c\}, \{b\}, \{c\}, \emptyset\}$. Refusals of the process $Q = P \,|\, \{b, c\}.a$ are the respective maximal refusals of states comprising $Q$, i.e. $\{b\}$ and $\{c\}$, as well as all of their proper subsets, in this case only the empty set, hence $\mathbf{R}(Q) = \{\{b\}, \{c\}, \emptyset\}$.

## 3. Refusal traces: a language-based approach

In this work, we focus on a more abstract language-based view, in which processes are represented solely by their refusal trace semantics, that is languages (sets) of refusal traces. Hence our primary objects of interest are the so-called *healthy languages* of refusal traces. Healthiness conditions are syntactic constraints on a language that, ideally, provide a sufficient and necessary conditions for a language to represent actual semantics of a process.

[1] Note that although divergence and deadlock look the same to the tester, since there are no visible actions/observations, they would not look the same if the tester was able to determine whether a process is active (divergence involves progress, while deadlock does not).

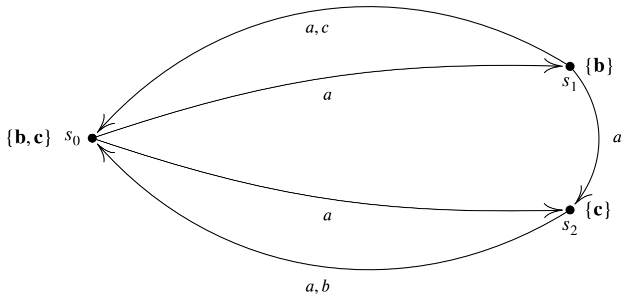


**Fig. 2.** A labelled transition system underlying a process $P$. States are labelled with their maximal refusals (the so-called state refusals). The initial state of $P$ is $s_0$.

We first define a natural preorder on refusal traces, denoted by $\leq_{\mathsf{RT}}$, which combines prefix order and pointwise inclusion:

$$X \leq_{\mathsf{RT}} Y \iff (X = \bullet) \vee (X \neq \bullet \wedge X \subseteq Y)$$

$$X_0 a_1 \ldots a_n [X_n] \leq_{\mathsf{RT}} Y_0 a_1 \ldots a_m [Y_m]$$
$$\overset{def}{\iff} n \leq m \wedge \forall i \in \{0, \ldots, n-1, [n]\}\, X_i \leq_{\mathsf{RT}} Y_i$$

Intuitively, $\pi_1 \leq_{\mathsf{RT}} \pi_2$ indicates that $\pi_1$ contains no more information on system behaviour than $\pi_2$, for instance: $\epsilon \leq_{\mathsf{RT}} \pi$ for any $\pi$, and $\{a\}.b.\emptyset.d \leq_{\mathsf{RT}} \{a, c\}.b.\emptyset.d.\{a, b, c\}$.

The healthiness conditions for a language $\mathcal{L} \subseteq \mathsf{RT}$ can now be defined as follows:

RT0 $\epsilon \in \mathcal{L}$
RT1 $\pi_2 \in \mathcal{L} \wedge \pi_1 \leq_{\mathsf{RT}} \pi_2 \implies \pi_1 \in \mathcal{L}$
RT2 $X, Y \in \mathcal{P}(\Sigma) \wedge \pi_1.X.\pi_2 \in \mathcal{L} \wedge (\forall a \in Y\, \pi_1.X.a \notin \mathcal{L}) \implies \pi_1.(X \cup Y).\pi_2 \in \mathcal{L}$
RT3 $\bullet \in \mathcal{L} \wedge (\pi.a \in \mathcal{L} \implies \pi.a.\bullet \in \mathcal{L})$
RTC $\emptyset \in \mathcal{L} \wedge (\pi.a \in \mathcal{L} \implies \pi.a.\emptyset \in \mathcal{L})$

In particular, the condition RT2 intuitively states that if, in a given context, no action from $Y$ can take place, then this must be reflected in the language by the presence of refusal $Y$ in the same context. We note that the last condition RTC is only required/valid in case we assume *convergence (divergence freedom)* of the underlying model (as we do in this work).

From hereon, by $\mathcal{L} \subseteq \mathsf{RT}$ we shall always denote a *healthy* language, i.e. satisfying syntactic healthiness conditions RT0–RT3 + RTC.

By $\mathsf{RT}^l(\mathcal{L})$ we shall denote a sublanguage of $\mathcal{L}$ consisting of traces of length not exceeding $l$.

## 3.1. Fundamental refusals and equivalence

In this section, we introduce the notions of fundamental refusals and equivalence [6,7]. In [6], fundamental refusals were introduced as intersections of state refusals (i.e. maximal refusals of states) and subsequently shown to have an equivalent characterisation based solely on the syntax of refusal traces. Since in our setting we work with a specification language not tied to an explicit model, we use the latter syntax-based definition as the default one. In fact, we show how one can arrive at the theory from [7] while only having a language of refusal traces at our disposal.

### 3.1.1. Behaviour after a partial trace

We first focus on the behaviour after *partial traces* ($\mathsf{RT}_\mathsf{A}$), in particular the local structure of refusals after such traces: we identify classes of refusals that are equivalent in the context of a partial trace.

**Definition 3.1.** For $\pi \in \mathsf{RT}_\mathsf{A}$, we define:

- the *language $\mathcal{L}$ after $\pi$* as:

$$\mathcal{L} \,|\, \pi \triangleq \{\lambda \in \mathsf{RT} \mid \pi.\lambda \in \mathcal{L}\}$$

- the set of *refusals of $\mathcal{L}$ after $\pi$* as:

$$\mathbf{R}_\mathcal{L}(\pi) = \{X \subseteq \mathcal{P}(\Sigma) \mid \pi.X \in \mathcal{L}\}$$

We also define $\mathbf{R}(\mathcal{L}) \triangleq \mathbf{R}_\mathcal{L}(\epsilon)$.

We sometimes refer to a language after a partial trace as a *language process*.

It is straightforward to observe that suffix languages after partial traces are of the same kind as the language under consideration $\mathcal{L}$ (i.e. healthiness conditions are satisfied in suffix languages).

**Lemma 3.1.** *If $\mathcal{L}$ is a healthy language, then for any $\sigma \in \mathcal{L}$, $\mathcal{L} \,|\, \sigma$ is healthy as well.*

**Proof.** The proof is very straightforward and we shall only provide it for one case (healthiness condition RT2); proofs of the remaining cases follow the same pattern.

Suppose that $X, Y \in \mathcal{P}(\Sigma)$, $\pi_1.X.\pi_2 \in \mathcal{L} \,|\, \sigma$, and for all $a \in Y$ we have $\pi_1.X.a \notin \mathcal{L} \,|\, \sigma$. From the definition of $\mathcal{L} \,|\, \sigma$, this in particular entails $\sigma.\pi_1.X.\pi_2 \in \mathcal{L}$ and $\sigma.\pi_1.X.a \notin \mathcal{L}$ for all $a \in Y$. Since $\mathcal{L}$ is healthy, we have $\sigma.\pi_1.(X \cup Y).\pi_2 \in \mathcal{L}$, from which and the definition of $\mathcal{L} \,|\, \sigma$ follows $\pi_1.(X \cup Y).\pi_2 \in \mathcal{L} \,|\, \sigma$. Hence RT2 holds for $\mathcal{L} \,|\, \sigma$. $\square$

**Definition 3.2.** A refusal observation $X \in \mathbf{R}(\mathcal{L}) \cup \{\bullet\}$ *entails* $Y$ in the context of $\mathcal{L}$ and $\pi \in \mathbf{RT}_\mathsf{A} \cap \mathcal{L}$, notation $X \succeq_{\mathcal{L},\pi} Y$, if either $Y = \bullet$, or $X, Y \subseteq \Sigma$, and

$$\forall a \in Y \;\; \pi.X.a \notin \mathcal{L}$$

In other words, for any $X \in \mathbf{R}_\mathcal{L}(\pi)$, $X$ entails $Y$ if any event in $Y$ is always forbidden by $\mathcal{L}$ after $\pi.X$ (note that this includes the case when $Y$ is a subset of $X$). The interpretation of entailment in model-based testing is as follows: suppose we are testing if an implementation satisfies a specification $\mathcal{L}$, and we have already executed/observed a trace $\pi$ in the implementation. If we observe a subsequent refusal $X$, then the current state of a *correct* implementation must also exhibit any refusal $Y$ such that $X \succeq_{\mathcal{L},\pi} Y$.

We can now define the *fundamental equivalence* of refusals in the context of $\mathcal{L}$ and $\pi \in \mathbf{RT}_\mathsf{A}$ as:

$$X \approx_{\mathcal{L},\pi} Y \overset{def}{\iff} X \succeq_{\mathcal{L},\pi} Y \wedge Y \succeq_{\mathcal{L},\pi} X$$

We proceed to show that refusal entailment is a preorder, and fundamental equivalence an actual equivalence relation. We start with the following simple property.

**Lemma 3.2.** *If $X, Y \in \mathcal{P}(\Sigma)$ and $X \succeq_{\mathcal{L},\pi} Y$, then for any $\lambda \in \mathbf{RT}_{\mathsf{A\text{-}SUF}}$:*

$$\pi.X.\lambda \in \mathcal{L} \implies \pi.X \cup Y.\lambda \in \mathcal{L}$$

**Proof.** In the context described above, suppose $\pi.X.\lambda \in \mathcal{L}$. Since $X \succeq_{\mathcal{L},\pi} Y$, for all $b \in Y \setminus X$ we have $\pi.X.b \notin \mathcal{L}$. This, combined with the healthiness condition RT2, yields $\pi.X \cup Y.\lambda = \pi.X \cup (Y \setminus X).\lambda \in \mathcal{L}$. $\square$

The above lemma has an important corollary – equivalent refusals are interchangeable in the context of a trace.

**Corollary 3.2.1.** *For any $X, Y \in \mathcal{P}(\Sigma) \cup \{\bullet\}$:*

1. $X \succeq_{\mathcal{L},\pi} Y \implies \forall \lambda \in \mathbf{RT}_{\mathsf{A\text{-}SUF}} \; (\pi.X.\lambda \in \mathcal{L} \implies \pi.Y.\lambda \in \mathcal{L})$
2. $X \approx_{\mathcal{L},\pi} Y \implies \forall \lambda \in \mathbf{RT}_{\mathsf{A\text{-}SUF}} \; (\pi.X.\lambda \in \mathcal{L} \iff \pi.Y.\lambda \in \mathcal{L})$

**Proof.** We only need to show the first property, as the latter follows immediately. Take any $X, Y$ such that $X \succeq_{\mathcal{L},\pi} Y$ and suppose that $\pi.X.\lambda \in \mathcal{L}$. The case when $Y = \bullet$ is trivial; otherwise we must have $X, Y \subseteq \Sigma$. From Lemma 3.2 we have $\pi.(X \cup Y).\lambda \in \mathcal{L}$, and from the closure condition (RT1) of $\mathcal{L}$ we obtain $\pi.Y.\lambda \in \mathcal{L}$. $\square$

The above property can be generalised as follows.

**Proposition 3.3.** *Consider two traces $\pi_1$ and $\pi_2$ of the same length, with the same actions appearing at the corresponding positions, and moreover such that refusal observations at the corresponding positions are equivalent. That is, formally we have: $\pi_1 = X_0 a_1 X_1 \ldots a_n [X_n], \pi_2 = Y_0 a_1 Y_1 \ldots a_n [Y_n]$, and for all $i \in \{0, \ldots, n-1, [n]\}$ $X_i \approx_{\mathcal{L}, X_0 a_1 \ldots a_{i-1}} Y_i$. Then*

$$\pi_1 \in \mathcal{L} \iff \pi_2 \in \mathcal{L}$$

**Proof.** Starting with $\pi_1 = X_0 \ldots a_{n-1} [X_n]$, we can modify the trace by iteratively substituting $X_i$ with $Y_i$ for each position $i \in \{0, \ldots, n-1, [n]\}$. Due to Corollary 3.2.1, after each such substitution the modified trace is contained in $\mathcal{L}$ if and only if the original one ($\pi_1$) is. The chain of modified traces ultimately results in the trace $\pi_2$. $\square$

We are now in position to establish that refusal entailment is a preorder.

**Proposition 3.4.** *For any healthy $\mathcal{L}$ and $\pi \in \mathbf{RT}_A \cap \mathcal{L}$, $\succeq_{\mathcal{L},\pi}$ is a preorder. As a consequence, its kernel $\approx_{\mathcal{L},\pi}$ is an equivalence relation.*

**Proof.** Suppose $X \succeq_{\mathcal{L},\pi} Y$ and $Y \succeq_{\mathcal{L},\pi} Z$. Take any $a \in Z \setminus X$; we need to show that $\pi.X.a \notin \mathcal{L}$. There are two cases:

- if $a \in Y$, then $a \in Y \setminus X$, and from $Y \succeq_{\mathcal{L},\pi} X$ follows $\pi.X.a \notin \mathcal{L}$
- if $a \notin Y$, then $a \in Z \setminus Y$, and hence $\pi.Y.a \notin \mathcal{L}$.
  Suppose, towards contradiction, that $\pi.X.a \in \mathcal{L}$. Then from $X \succeq_{\mathcal{L},\pi} Y$ and Lemma 3.2 we would have $\pi.(X \cup Y).a \in \mathcal{L}$, which contradicts $\pi.Y.a \notin \mathcal{L}$ (violation of subset closure of $\mathcal{L}$ – RT1). We have thus shown that $\pi.X.a \notin \mathcal{L}$. $\square$

Refusal entailment and equivalence are lifted to refusal traces in a natural way:
$X_0 a_1 X_1 \dots a_n [X_n] \succeq_{\mathcal{L}} Y_0 a_1 Y_1 \dots a_n [Y_n]$
$\overset{def}{\iff} \forall i \in \{0, \dots, n-1, [n]\}\ X_i \succeq_{\mathcal{L}, X_0 \dots a_{i-1}} Y_i$
The pointwise lifting of refusal equivalence to traces, denoted with $\approx_{\mathcal{L}}$, is defined similarly (just substitute relation symbols in the above definition).

### 3.1.2. Fundamental refusals and fundamental refusal traces

**Definition 3.3.** Given a healthy language $\mathcal{L}$, and a trace $\pi \in \mathbf{RT}_A$, we define the set of *fundamental refusals [of $\mathcal{L}$] after $\pi$* as:

$$\mathbf{FR}_{\mathcal{L}}(\pi) = \{X \in \mathbf{R}_{\mathcal{L}}(\pi) \mid \forall a \in \Sigma \setminus X \ \ \pi.X.a \in \mathcal{L}\}$$

We also define $\mathbf{FR}(\mathcal{L}) \triangleq \mathbf{FR}_{\mathcal{L}}(\epsilon)$.

Intuitively, a refusal $X$ is fundamental after $\pi$ if there is a valid model for $\mathcal{L}$ that, after performing $\pi$, can reach a state with a *state refusal* $X$ i.e. state which refuses precisely the actions in $X$ (and where all actions in $\Sigma \setminus X$ are enabled).

**Example 3.4.1.** Let $\mathcal{L}$ be the refusal semantics of process $P$ from Fig. 2 in the context of $\Sigma = \{a, b, c\}$. Observe that all nonempty traces in $\mathcal{L}$ are of the form $X$ or $X.a.\sigma$, where $X = \bullet$ or $X \subseteq \{b, c\}$. We therefore have $\mathbf{FR}(\mathcal{L}) = \{\{b, c\}\}$, since $\{b, c\}.a \in \mathcal{L}$, and $\{b, c\}$ is the only refusal meeting the condition from the definition of $\mathbf{FR}$.

Consider now the context after the trace $\{b, c\}.a$. Its extensions in $\mathcal{L}$ are of the form:

- $\{b, c\}.a.X.a.\sigma, \{b, c\}.a.X.c.\sigma$ for $X \subseteq \{b\}$
  (traces contributed by paths of the form $s_0.s_1.\dots$)
- $\{b, c\}.a.X.a.\sigma, \{b, c\}.a.X.b.\sigma$ for $X \subseteq \{c\}$ (traces contributed by paths of the form $s_0.s_2.\dots$)

We have thus $\mathbf{FR}_{\mathcal{L}}(\{b, c\}.a) = \{\{b\}, \{c\}, \emptyset\}$. Observe in particular that $\emptyset$ is a fundamental refusal due to $\{b, c\}.a.\emptyset.a, \{b, c\}.a.\emptyset.b$ and $\{b, c\}.a.\emptyset.c$ being present in $\mathcal{L}$.

**Definition 3.4.** The set of *fundamental refusal traces* (or *fundamental traces* for short) of $\mathcal{L}$ is defined as:

$$\begin{aligned}\mathbf{FRT}(\mathcal{L}) \quad \triangleq \quad & \{\epsilon\} \cup \{X_0 a_1 \dots X_n [a_{n+1}] \in \mathcal{L} \mid X_0 = \bullet \vee X_0 \in \mathbf{FR}(\mathcal{L}) \\ & \wedge \forall i : X_i = \bullet \vee X_i \in \mathbf{FR}_{\mathcal{L}}(X_0 a_1 \dots a_i)\}\end{aligned}$$

We also define, for a given $X \in \mathbf{R}_{\mathcal{L}}(\pi)$, the so-called top refusal for $X$:

$$\mathsf{top}_{\mathcal{L},\pi}(X) = \{a \in \Sigma \mid \pi.X.a \notin \mathcal{L}\}$$

Observe that for any $\pi.X \in \mathcal{L}$, we also have $\pi.\mathsf{top}_{\mathcal{L},\pi}(X) \in \mathcal{L}$ – this is due to the healthiness condition RT2.

The top refusal for $X$ is the largest refusal entailed by $X$, as well as the least fundamental refusal that contains $X$, as witnessed by, respectively, Lemma 3.5 and Lemma 3.6.

**Lemma 3.5.** *For any refusal $X \subseteq \mathcal{P}(\Sigma)$, $\mathsf{top}_{\mathcal{L},\pi}(X)$ is the largest refusal entailed by $X$, i.e. $X \succeq_{\mathcal{L},\pi} \mathsf{top}_{\mathcal{L},\pi}(X)$ and*

$$\forall Y \ X \succeq_{\mathcal{L},\pi} Y \implies Y \leq_{\mathbf{RT}} \mathsf{top}_{\mathcal{L},\pi}(X)$$

*Similarly, $\mathsf{top}_{\mathcal{L},\pi}(X)$ is the largest refusal equivalent to $X$, i.e. $\mathsf{top}_{\mathcal{L},\pi}(X) \approx_{\mathcal{L},\pi} X$ and*

$$\forall Y \ X \approx_{\mathcal{L},\pi} Y \implies Y \leq_{\mathbf{RT}} \mathsf{top}_{\mathcal{L},\pi}(X)$$

**Proof.** We first show that $X \succeq_{\mathcal{L},\pi} \mathsf{top}_{\mathcal{L},\pi}(X)$ and $\mathsf{top}_{\mathcal{L},\pi}(X) \approx_{\mathcal{L},\pi} X$.

For any $a \in \mathsf{top}_{\mathcal{L},\pi}(X)$, we have $\pi.X.a \notin \mathcal{L}$ directly from the definition of $\mathsf{top}_{\mathcal{L},\pi}$; hence $X \succeq_{\mathcal{L},\pi} \mathsf{top}_{\mathcal{L},\pi}(X)$. Moreover, from the definition of top and the well-formedness condition for $\mathbf{RT}$, we have $X \subseteq \mathsf{top}_{\mathcal{L},\pi}(X)$. From this and the well-formedness condition for $\mathbf{RT}$, follows $\mathsf{top}_{\mathcal{L},\pi}(X) \succeq_{\mathcal{L},\pi} X$, and hence we obtain $\mathsf{top}_{\mathcal{L},\pi}(X) \approx_{\mathcal{L},\pi} X$.

Take any refusal observation $Y$ such that $X \succeq_{\mathcal{L},\pi} Y$ $[Y \approx_{\mathcal{L},\pi} X]$. If $Y = \bullet$, then $Y \leq_{\mathbf{RT}} \mathsf{top}_{\mathcal{L},\pi}(X)$ holds immediately. We thus assume that $Y \in \mathcal{P}(\Sigma)$.

Take an arbitrary $a \in Y$. From $X \succeq_{\mathcal{L},\pi} Y$ it follows that $\pi.X.a \notin \mathcal{L}$. This entails $a \in \mathsf{top}_{\mathcal{L},\pi}(X)$. □

**Lemma 3.6.** *$\mathsf{top}_{\mathcal{L},\pi}(X)$ is the least refusal in the set $\{\widehat{Y} \in \mathbf{FR}_{\mathcal{L}}(\pi) \mid X \subseteq \widehat{Y}\}$, that is, it is the least fundamental refusal that contains $X$.*

**Proof.** That $\pi.\mathsf{top}_{\mathcal{L},\pi}(X) \in \mathcal{L}$ follows from $\pi.X \in \mathcal{L}$, definition of $\mathsf{top}_{\mathcal{L},\pi}(X)$, and the healthiness condition RT2. Hence $\mathsf{top}_{\mathcal{L},\pi}(X) \in \mathbf{R}_{\mathcal{L}}(\pi)$.

We proceed to prove that $\mathsf{top}_{\mathcal{L},\pi}(X) \in \mathbf{FR}_{\mathcal{L}}(\pi)$. Consider any $a \in \Sigma \setminus \mathsf{top}_{\mathcal{L},\pi}(X)$; from the definition of $\mathsf{top}_{\mathcal{L},\pi}(X)$ we have $\pi.X.a \in \mathcal{L}$. From this, the definition of $\mathsf{top}_{\mathcal{L},\pi}(X)$, and the healthiness condition RT2, we obtain $\pi.\mathsf{top}_{\mathcal{L},\pi}(X).a \in \mathcal{L}$. Hence we have shown that $\mathsf{top}_{\mathcal{L},\pi}(X) \in \mathbf{FR}_{\mathcal{L}}(\pi)$.

Finally, let $\widehat{Y} \in \mathbf{FR}_{\mathcal{L}}(\pi)$ be such that $X \subseteq \widehat{Y}$. Take any $a \in \Sigma$ such that $\pi.X.a \notin \mathcal{L}$. From RT1 we have $\pi.\widehat{Y}.a \notin \mathcal{L}$, and since $\widehat{Y} \in \mathbf{FR}_{\mathcal{L}}(\pi)$, it must be the case that $a \in \widehat{Y}$. Hence we obtain

$$\mathsf{top}_{\mathcal{L},\pi}(X) = \{a \in \Sigma \mid \pi.X.a \notin \mathcal{L}\} \subseteq \widehat{Y} \quad \square$$

For technical convenience, we extend the definition of top operator to null refusals:

$$\mathsf{top}_{\mathcal{L},\pi}(\bullet) \triangleq \bullet$$

Furthermore, we also extend the definition of the top operator to all refusal traces of a given language in a natural way. Namely, given a language $\mathcal{L}$, we define $\mathsf{top}_{\mathcal{L}}(\epsilon) \triangleq \epsilon$, and for all nonempty refusal traces of $\mathcal{L}$:

$$\mathsf{top}_{\mathcal{L}}(X_0 a_1 \dots a_n [X_n]) \triangleq \mathsf{top}_{\mathcal{L}_0}(X_0) a_1 \dots a_n [\mathsf{top}_{\mathcal{L}_n}(X_n)]$$

where: $X_0 a_1 \dots X_{n-1} a_n [X_n] \in \mathcal{L}$, $\mathcal{L}_0 = \mathcal{L}$ and $\mathcal{L}_i = \mathcal{L} \mid X_0 a_1 \dots a_i$ for $i \in \{1, \dots, n\}$.

It is straightforward to see that for any $\pi \in \mathcal{L}$, we have $\mathsf{top}_{\mathcal{L}}(\pi) \in \mathcal{L}$.

**Corollary 3.6.1.** *Given a language $\mathcal{L}$ and refusal trace $\sigma \in \mathcal{L} \cap \mathbf{RT}_A$, we have:*

$$\mathcal{L} \mid \sigma = \mathcal{L} \mid \mathsf{top}_{\mathcal{L}}(\sigma)$$

**Proof.** Follows from $\mathsf{top}_{\mathcal{L},\pi}(X) \approx_{\mathcal{L},\pi} X$ (Lemma 3.5) and Proposition 3.3. □

We have now arrived at the following corollary, which essentially states that fundamental refusal traces determine the refusal trace semantics: two healthy languages are equal whenever their respective restrictions to fundamental refusal traces are equal.

**Corollary 3.6.2.** *For all healthy $\mathcal{L}_1, \mathcal{L}_2 \subseteq \mathbf{RT}$*

$$\mathcal{L}_1 = \mathcal{L}_2 \iff \mathbf{FRT}(\mathcal{L}_1) = \mathbf{FRT}(\mathcal{L}_2)$$

**Proof.** The direction from left to right is immediate, since $\mathbf{FRT}(\mathcal{L})$ is a function of $\mathcal{L}$. For the other direction, assume that $\mathbf{FRT}(\mathcal{L}_1) = \mathbf{FRT}(\mathcal{L}_2)$ and w.l.o.g. take any $\pi \in \mathcal{L}_1$; we need to show that $\pi \in \mathcal{L}_2$. Let $\widehat{\pi} = \mathsf{top}_{\mathcal{L}}(\pi)$; we have $\pi \leq_{\mathbf{RT}} \widehat{\pi}$. From $\mathbf{FRT}(\mathcal{L}_1) = \mathbf{FRT}(\mathcal{L}_2)$ we have $\widehat{\pi} \in \mathbf{FRT}(\mathcal{L}_2) \subseteq \mathcal{L}_2$. That $\pi \in \mathcal{L}_2$ follows now from downward closure property (RT1) of $\mathcal{L}_2$. □

In the final part of this section, we recall the result from [7] stating that the syntactic notion of fundamental refusals coincides with the original, model-dependent notion based on intersections of state refusals, introduced in [7].

Given a state $s$ of an LTS, the *state refusal* of $s$, denoted with $\mathbf{SR}(s)$, is the largest refusal of $s$, i.e. the largest element of $\mathbf{R}(s)$. Note that state refusals are complements of the so-called initial sets (or ready sets) of individual states. State refusals of a process $P$ are all state refusals of states comprising $P$, i.e. $\mathbf{SR}(P) = \bigcup_{s \in P} \{\mathbf{SR}(s)\}$.

Let $P$ be a process and $\pi \in \mathbf{RT}_{\mathsf{A}}(P)$. A refusal $\widehat{X}$ is a *fundamental refusal in the sense of [7]* of $P$ after $\sigma$ iff it is an intersection of some state refusals in $P \mid \pi$, i.e. iff there exist state refusals $X_1, \dots, X_n \in \mathbf{SR}(P \mid \pi)$ such that $\widehat{X} = X_1 \cap \dots \cap X_n$. As has been shown in [7], the two notions of fundamental refusals are equivalent, i.e. the following holds:

**Proposition 3.7.** *Let $P$ be a process, $\mathcal{L} = \mathbf{RT}(P)$ be the refusal trace semantics of $P$, and $\pi \in \mathbf{RT}_A$. For any refusal $\widehat{X} \subseteq \Sigma$ we have:*

$$\widehat{X} \in \boldsymbol{FR}(\mathcal{L} \mid \pi) \iff \widehat{X} \in \{X_1 \cap \dots \cap X_n \mid X_1, \dots, X_n \in \mathbf{SR}(P \mid \pi)\}$$

### 3.2. Observation transition system

In this section, we define a normalised canonical model for a language that is particularly suitable for testing purposes. It is a specific type of labelled transition system whose labels correspond to observations that can be made during a testing experiment, hence its labels contain refusals as well as actions. We found this approach simplest to work with, as it avoids a lot of redundant technical overhead that would otherwise be necessary to manage the structure of a more standard labelled transition system.

We note that the size of an observation transition system is in worst case exponential in the size of the original model (e.g. LTS).

**Definition 3.5.** For a healthy language $\mathcal{L}$, the *observation transition system (OTS) induced by* $\mathcal{L}$ is defined as:

$$\mathcal{OTS}(\mathcal{L}) \triangleq (\mathcal{Q}(\mathcal{L}) \cup \{q_\top\}, q^I(\mathcal{L}), \longrightarrow_{\mathcal{L}})$$

where:

- $\mathcal{Q}(\mathcal{L}) \triangleq \{\mathcal{L} \,|\, \pi \mid \pi \in \mathcal{L} \cap \mathbf{RT}_\mathsf{A}\}$
- $q_\top = \emptyset$ is the terminal state
- $q^I$ is the initial (language) state of $\mathcal{L}$ defined as:

$$q^I(\mathcal{L}) \triangleq \mathcal{L} \,|\, \epsilon = \mathcal{L}$$

- the transition relation $\longrightarrow_{\mathcal{L}}$ is defined as:

$$\begin{aligned} q \xrightarrow{X.a}_{\mathcal{L}} q' \quad &\overset{def}{\iff} \quad X.a \in q \wedge q' = q \,|\, X.a \\ q \xrightarrow{X}_{\mathcal{L}} q_\top \quad &\overset{def}{\iff} \quad X \in q \end{aligned}$$

Note that the state $q_\top$ represents termination of a testing experiment after observing a refusal, and it is considered separate from the language states $\mathcal{Q}(\mathcal{L})$: it is in particular not a healthy language.

Note that $q^I(\mathcal{L}) \neq \emptyset$ due to the healthiness condition RT0 holding for $\mathcal{L}$.

We shall use the notion of *observation traces* of an OTS and its states, which are essentially their standard traces. Formally we define $\mathbf{OT}_{\mathcal{L}}(q_\top) \triangleq \{\epsilon\}$, and for every $q \in \mathcal{Q}(\mathcal{L})$:

$$\begin{aligned} \mathbf{OT}_{\mathcal{L}}(q) \triangleq \{\epsilon\} & \\ &\cup \{X.a.\pi \mid \exists q' : q \xrightarrow{X.a}_{\mathcal{L}} q' \wedge \pi \in \mathbf{OT}_{\mathcal{L}}(q')\} \\ &\cup \{X \mid q \xrightarrow{X}_{\mathcal{L}} q_\top\} \end{aligned}$$

The validity of our construction can be now established with the following result.

**Proposition 3.8.** *Given a language $\mathcal{L}$ and $q \in \mathcal{Q}(\mathcal{L})$, we have:*

$$\mathbf{OT}_{\mathcal{L}}(q) = q$$

**Proof.** We shall prove that for any $\pi \in \mathbf{RT}$, we have

$$\pi \in \mathbf{OT}_{\mathcal{L}}(q) \iff \pi \in q$$

using induction on the structure of $\pi$.

- Base:
  - $\pi = \epsilon$: we have $\epsilon \in \mathbf{OT}_{\mathcal{L}}(q)$ directly from the definition of $\mathbf{OT}_{\mathcal{L}}(q)$. Since $q = \mathcal{L} \,|\, \pi$ for some $\pi \in \mathcal{L}$, from Lemma 3.1 and healthiness condition RT0 we obtain $\epsilon \in q$.
  - $\pi = X$: We have:
    $X \in \mathbf{OT}_{\mathcal{L}}(q) \iff$ (def. of $\mathbf{OT}_{\mathcal{L}}(q)$)
    $q \xrightarrow{X}_{\mathcal{L}} q_\top \iff$ (def. of $\longrightarrow_{\mathcal{L}}$)
    $X \in q$
  - Inductive step:
    Let $\pi = X.a.\rho$. We have:
    $X.a.\rho \in \mathbf{OT}_{\mathcal{L}}(q) \iff$ (def. of $\mathbf{OT}_{\mathcal{L}}(q)$)
    $\exists q' : q \xrightarrow{X.a}_{\mathcal{L}} q' \wedge \rho \in \mathbf{OT}_{\mathcal{L}}(q') \iff$ (def. of $\longrightarrow_{\mathcal{L}}$)
    $\exists q' : X.a \in q \wedge q' = q \,|\, X.a \wedge \rho \in \mathbf{OT}_{\mathcal{L}}(q') \iff$ (IH)
    $\exists q' : X.a \in q \wedge q' = q \,|\, X.a \wedge \rho \in q' \iff$ (def. of $q \,|\, X.a$)
    $\exists q' : X.a \in q \wedge q' = \{\lambda \mid X.a.\lambda \in q\} \wedge \rho \in q' \iff$
    $X.a.\rho \in q$ $\square$

**Corollary 3.8.1.** *For any language of refusal traces $\mathcal{L}$*

$$\boldsymbol{OT}_{\mathcal{L}}(q^I(\mathcal{L})) = \mathcal{L}$$

Next, we formally state a very straightforward and useful property regarding the relationship between (fundamental) traces of a language and (fundamental) traces of its language states.

**Lemma 3.9.** *Given a language $\mathcal{L}$, some $v \in \mathcal{L}$, and a language state $s = \mathcal{L} \,|\, v$, we have:*

1. *For any $\sigma \in \mathbf{RT}$*

$$\sigma \in s \iff v.\sigma \in \mathcal{L}$$

2. *If $v \in \mathbf{FRT}(\mathcal{L})$, then*

$$\sigma \in \mathbf{FRT}(s) \iff v.\sigma \in \mathbf{FRT}(\mathcal{L})$$

**Proof.** 1. Immediate, since $s = \mathcal{L} \,|\, v = \{\lambda \in \mathbf{RT} \mid v.\lambda \in \mathcal{L}\}$

2. First, observe that we can state the definition of fundamental refusal traces in the following way:
$\pi \in \mathbf{FRT}(\mathcal{L}) \iff \forall \pi'.X : \pi = \pi'.X.\lambda \Longrightarrow X \in \mathbf{FR}(\mathcal{L} \,|\, \pi')$
Furthermore, it is straightforward to observe that for any $\pi \in \mathbf{RT}_{\mathsf{A}}, \rho \in \mathbf{RT}$:
(*) $\mathcal{L} \,|\, \pi.\rho = (\mathcal{L} \,|\, \pi) \,|\, \rho$ Since we have assumed that $v \in \mathbf{FRT}(\mathcal{L})$, we have:
$v.\sigma \in \mathbf{FRT}(\mathcal{L}) \iff$ [def. of $\mathbf{FRT}$]
$\forall \pi.X : v.\sigma = \pi.X.\lambda \Longrightarrow X \in \mathbf{FR}(\mathcal{L} \,|\, \pi) \iff [v \in \mathbf{FRT}(\mathcal{L})]$
$\forall v.\pi'.X : v.\sigma = v.\pi'.X.\lambda \Longrightarrow X \in \mathbf{FR}(\mathcal{L} \,|\, v.\pi') \iff$
$\forall \pi'.X : \sigma = \pi'.X.\lambda \Longrightarrow X \in \mathbf{FR}(\mathcal{L} \,|\, v.\pi') \iff$ [(*)]
$\forall \pi'.X : \sigma = \pi'.X.\lambda \Longrightarrow X \in \mathbf{FR}(s \,|\, \pi') \iff$ [def. of $\mathbf{FRT}$]
$\sigma \in \mathbf{FRT}(s)$ □

We can work with transitions of an OTS modulo fundamental equivalence, as witnessed by the following proposition.

**Proposition 3.10.** *Suppose $q = \mathcal{L} \,|\, \pi$. For all $X, Y$ such that $X \approx_{\mathcal{L},\pi} Y$ and $\pi.X, \pi.Y \in \mathcal{L}$, we have:*

$$q \xrightarrow{X.a}_{\mathcal{L}} q' \iff q \xrightarrow{Y.a}_{\mathcal{L}} q'$$
$$q \xrightarrow{X}_{\mathcal{L}} q_{\top} \iff q' \xrightarrow{Y}_{\mathcal{L}} q_{\top}$$

**Proof.** Follows from Corollary 3.2.1. □

**Example 3.10.1.** Let $\mathcal{L}$ be the language consisting of refusal traces of process $P$ from Fig. 2. $\mathcal{OTS}(\mathcal{L})$ is depicted on Fig. 3.

## 4. Testing framework

Here, we describe the testing framework. In Section 4.1 we explain how testing can be seen as comparing two OTSs. Section 4.2 explains how, given refusal trace $\sigma$, one can test to determine whether $\sigma$ is a refusal trace of the SUT. Test generation can then be defined in terms of choosing suitable traces. Sections 4.3 and 4.4 then define what we mean for traces to reach states of the specification and also to identify a state; these definitions will be used in the test generation technique (Section 5).

### *4.1. Basic test hypotheses*

In order to formally reason about test effectiveness one needs to consider formal entities. It is thus normal to assume that the SUT behaves like an unknown model that can be expressed in the language used to write the specification (the *minimum hypothesis* [5]). We therefore have a scenario in which we have a specification, with language $\mathcal{L}_S$, an SUT with unknown language $\mathcal{L}_I$ and we would like to test to determine whether $\mathcal{L}_S = \mathcal{L}_I$. We also assume that $\mathcal{L}_I$ induces an unknown OTS $\mathcal{OTS}(\mathcal{L}_I)$ with the same alphabet as the specification.

When testing from an FSM, an additional test hypothesis is usually made: that the SUT behaves like an unknown FSM with at most *m* states, for some given *m*. This defines a fault domain: the set of FSMs, with the same input and output alphabets as the specification, that have no more than *m* states. We use the corresponding test hypothesis, which is that there is a given *m* such that the unknown OTS $\mathcal{OTS}(\mathcal{L}_I)$ has at most *m* states.

Non-determinism creates a problem in testing since, even if $X.a.\sigma$ is a refusal trace of the SUT, the SUT might have a state that can be reached through $X.a$ but that does not allow $\sigma$: one can then test arbitrarily many times without observing $X.a.\sigma$. This issue

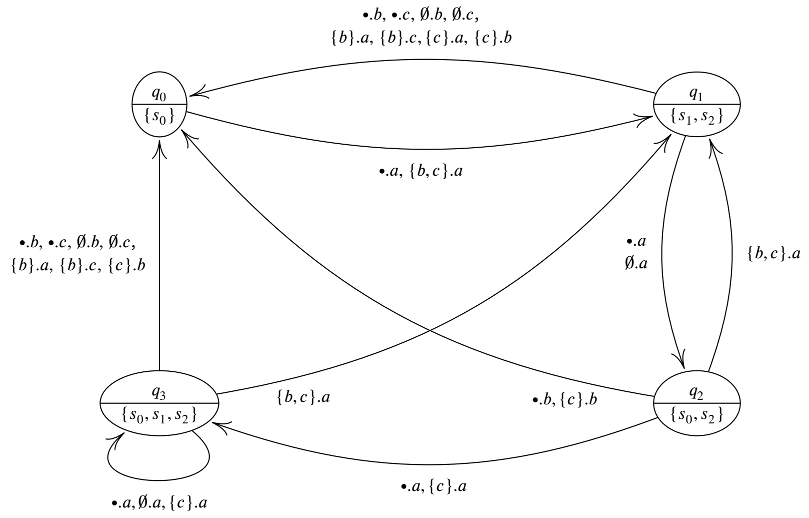

$q_0$
$\{s_0\}$
$q_1$
$\{s_1, s_2\}$
$q_2$
$\{s_0, s_2\}$
$q_3$
$\{s_0, s_1, s_2\}$
$\bullet.b$, $\bullet.c$, $\emptyset.b$, $\emptyset.c$, $\{b\}.a$, $\{b\}.c$, $\{c\}.a$, $\{c\}.b$
$\bullet.a$, $\{b,c\}.a$
$\bullet.b$, $\bullet.c$, $\emptyset.b$, $\emptyset.c$, $\{b\}.a$, $\{b\}.c$, $\{c\}.b$
$\bullet.a$
$\emptyset.a$
$\{b,c\}.a$
$\{b,c\}.a$
$\bullet.b$, $\{c\}.b$
$\bullet.a$, $\{c\}.a$
$\bullet.a$, $\emptyset.a$, $\{c\}.a$

**Fig. 3.** Observation transition system of a language induced by process $P$ from Fig. 2. Only the proper language states (without the terminal state $q_\top$) are shown; moreover, transition labels of the form $X.a$ are only included if either X is fundamental, or $X = \bullet$.

also arises for testing based on traces. Clearly, in the presence of possible non-determinism one cannot make guarantees about test effectiveness without additional assumptions.

The standard solution is to make a *fairness assumption* that, for some given $k$, it is sufficient to apply a test case $k$ times to observe all possible responses of the SUT (see, for example, [16,21]; Huang and Peleska also make a similar assumption [13,14]). The choice of $k$ could depend on criticality/risk or domain knowledge, and there is also the potential to use probabilistic arguments. We therefore assume that a fairness assumption is being made.

To summarise, we assume that language $\mathcal{L}_I$ of the SUT induces an unknown OTS $\mathcal{OTS}(\mathcal{L}_I)$ that has the same alphabet as the specification, that $\mathcal{OTS}(\mathcal{L}_I)$ has at most $m$ states, and that a fairness assumption can be made.

## 4.2. Testing for refusal traces

We will consider test generation as a process of choosing a set of refusal traces and determining whether these are refusal traces of the SUT. One could use refusal traces in $\mathcal{L}_S$ or refusal traces not allowed (not in $\mathcal{L}_S$). Here, we briefly address how one can test whether the language of the SUT includes a refusal trace $\sigma$.

First consider the problem of testing to decide whether $X \in \mathcal{P}(\Sigma)$ is a refusal in the current state of the SUT. The normal process is for the tester to offer the set $X$ of actions to the SUT and check whether there is a deadlock. The tester thus behaves like a process that provides an external choice between the events in $X$. In testing, deadlock is normally observed through the use of a timeout, with the choice of time used being problem specific.

Now let us suppose that we wish to test to determine whether $X.a.\sigma$ is a refusal trace of the SUT, for some refusal trace $\sigma$. As before, the tester provides an external choice between the events in $X$. If deadlock is observed (there is a timeout) then the tester offers event $a$. If $a$ is observed then we recurse, testing to check whether $\sigma$ is a possible refusal trace in the current state of the SUT (that has been reached by $X.a$). In the presence of (possible) non-determinism, if $X.a.\sigma$ is not observed then the overall process/experiment is repeated a sufficient number of times to use the fairness assumption.

## 4.3. State covers

In FSM-based testing, many techniques use the notion of a state cover: a set of input sequences that, between them, reach all of the states of the FSM specification. Based on the material above, we can now adapt this concept to language states.

**Definition 4.1.** Given language $\mathcal{L}$ and language state $q \in \mathcal{Q}(\mathcal{L})$, a refusal trace $\pi \in \mathcal{L} \cap \mathbf{RT}_\mathsf{A}$ *reaches* $q$ if $\mathcal{L} \mid \pi = q$.

It is straightforward to say what it means for a set of refusal traces to be a state cover.

**Definition 4.2.** A set $V \subseteq \mathbf{RT}_\mathsf{A} \cap \mathcal{L}$ of refusal traces is a *state cover* for language $\mathcal{L}$ if the following conditions hold:

1. For all $q \in \mathcal{Q}(\mathcal{L})$, there is some $\pi_q \in V$ that reaches $q$; and
2. $\epsilon \in V$

The second condition is a natural property (we use the shortest refusal trace that reaches the initial state) and will be required by the test generation algorithm.

If $\mathcal{Q}(\mathcal{L})$ is finite and the alphabet is finite then one can use a breadth-first search to generate a state cover.

**Proposition 4.1.** *Given language $\mathcal{L}$, if $\mathcal{Q}(\mathcal{L})$ is finite such that $n = |\mathcal{Q}(\mathcal{L})|$, then there is a state cover $V$ such that:*

- *$V$ contains $n$ refusal traces*
- *$V$ contains only fundamental refusal traces of $\mathcal{L}$*
- *each refusal trace has length at most $n$*
- *$V$ is prefix-closed w.r.t. $\mathbf{RT}_A$*
- *each refusal trace $v \in V$ has minimal length among all refusal traces in $\mathbf{RT}_A \cap \mathcal{L}$ that reach $\mathcal{L} \mid v$.*

*We shall use the term* normal state cover *for any state cover with the above properties.*

**Proof.** That a state cover has $|\mathcal{Q}(\mathcal{L})|$ refusal traces follows immediately from its definition, requiring one refusal trace for each language state. Moreover, from every state cover $V'$ we can obtain a state cover $V$ with the same relevant properties and in addition consisting of fundamental refusals only by taking $V = \{\mathsf{top}_{\mathcal{L}}(\sigma) \mid \sigma \in V'\}$.

The existence of a state cover with the remaining properties can be shown constructively using a breadth-first search approach.

Let $T$ be an arbitrary breadth-first search tree rooted in $q^I(\mathcal{L})$. Observe that, for a given state $q \in \mathcal{Q}(\mathcal{L})$, the length of the path from the root $q^I(\mathcal{L})$ to $q$ in $T$ is equal to the minimal length of a refusal trace that reaches $q$.

Thus let $V$ be a collection of refusal traces corresponding to all paths in $T$ from the root $q^I(\mathcal{L})$ to the states in $\mathcal{Q}(\mathcal{L})$. Since a collection of (nontrivial) paths in a tree is prefix-closed, then so is $V$, and furthermore, since the path lengths in $T$ obviously do not exceed $n$, the same holds for traces in $V$. $\square$

**Example 4.1.1.** Consider the $\mathcal{OTS}$ induced by a language $\mathcal{L}$ from Fig. 3. The set $V = \{\epsilon, \{b,c\}.a, \{b,c\}.a.\bullet.a, \{b,c\}.a.\bullet.a.\{c\}.a\}$ is a normal state cover for $\mathcal{L}$.

### 4.4. State identification

One can use a state cover to reach the states of the specification and so devise tests that execute transitions of the SUT that correspond to those of the specification. However, such tests need not find faults that entail a transition of the SUT going to the wrong state (state transfer faults). In order to detect state transfer faults we use refusal traces that separate two language states [15].

**Definition 4.3.** Refusal trace $\pi$ is said to *separate* language states $q_1$ and $q_2$ of $\mathcal{L}$ if either $\pi \in q_1 \setminus q_2$ or $\pi \in q_2 \setminus q_1$.

A characterising set is a set of traces that separates all pairs of states.

**Definition 4.4.** A set $W$ of refusal traces is a *characterising set* for language $\mathcal{L}$ if for every pair $q_1, q_2$ of distinct language states in $\mathcal{Q}(\mathcal{L})$, there is some $\pi \in W$ that separates $q_1$ and $q_2$.

One can place an upper bound on the size of the smallest characterisation set.

**Proposition 4.2.** *If $\mathcal{Q}(\mathcal{L})$ is finite, with $n = |\mathcal{Q}(\mathcal{L})|$, then $\mathcal{L}$ has a characterising set $W$ such that:*

1. *$W$ contains at most $n-1$ refusal traces; and*
2. *Every trace in $W$ has length at most $n-1$.*

**Proof.** By definition, two different language states $q_1$ and $q_2$ define different sets of refusal traces and so are separable. As a result, since $\mathcal{Q}(\mathcal{L})$ is finite, $\mathcal{L}$ has a finite characterising set.

We now prove that given two language states $q_1$ and $q_2$, there is a refusal trace $w$ that separates $q_1$ and $q_2$ such that $|w| \leq n-1$.

Let us fix a context of some language $\mathcal{L}$ and its corresponding OTS $\mathcal{OTS}(\mathcal{L})$. For pairs of states in $\mathcal{Q}(\mathcal{L})$, we define $minsep(q,q')$ as the length of a minimal trace separating $q$ and $q'$. We shall prove the following statement:

(*) $minsep(q,q') > 1 \implies \exists s, s' \in \mathcal{Q}(\mathcal{L}) : minsep(s,s') = minsep(q,q') - 1$

Take any $q_1, q_2 \in \mathcal{Q}(\mathcal{L})$ such that $minsep(q_1,q_2) > 1$ and let $\pi$ be their separating trace of minimal length. Assume w.l.o.g. that $\pi \in \mathbf{OT}_{\mathcal{L}}(q_1) \setminus \mathbf{OT}_{\mathcal{L}}(q_2)$.

Let $\pi = X.a.\pi'$ (note that $|\pi'| \geq 1$ hence $\pi' \neq \epsilon$). Let $q_1' = q_1 \mid X.a$.

Observe that $X.a$ is a refusal trace of $q_2$ (otherwise $\pi$ would not be a minimal trace separating $q_1$ and $q_2$). Let $q_2' = q_2 \mid X.a$. Observe that the trace $\pi'$ separates $q_1'$ and $q_2'$ and there is no shorter trace separating $q_1'$ and $q_2'$. Indeed, if there was such $\lambda$, then $X.a.\lambda$ would separate $q_1$ and $q_2$, contradicting the minimality of $\pi$. Since $|\pi'| = |X.a.\pi'| - 1 = |\pi| - 1$, we obtain $minsep(q_1', q_2') = minsep(q_1, q_2) - 1$. We have thus shown that (*) holds.

Let us define a family of equivalence relations $\sim^i$ on language states by: $q \sim^i q'$ if and only if $q$ and $q'$ are not separated by any refusal trace of length at most $i$. Observe that for any $q, q'$ such that $k = minsep(q, q') > 1$ we have $q \nsim^k q'$ and $q \sim^i q'$ for all $i \in \{1, \dots, k-1\}$.

Let $q$ and $q'$ be a pair of states for which the value $minsep(q, q')$ is maximal. Let $k = minsep(q, q')$. Suppose that $k > 1$ (the other case is trivial). From $(*)$ we know that there is a sequence of pairs of states $\{(s_i, s_i') \mid i \in \{1, \dots, k\}\}$ such that $minsep(s_i, s_i') = i$. This means that for all $i \in \{2, \dots, k\}$, $\sim^i$ has more equivalence classes than $\sim^{i-1}$. Moreover, since $s_1 \nsim^1 s_1'$, we know that $\sim^1$ has at least two equivalence classes. Since the number of equivalence classes is bounded by the number of states $n$, we thus obtain that $k \leq n-1$.

What remains to be shown is that there is a characterising set $W$ that contains at most $n-1$ refusal traces and where every refusal trace in $W$ contains at most $n-1$ refusals. Let us suppose that $W = \{w_1, \dots, w_k\}$ is a minimal (smallest) characterising set whose refusal traces all contain at most $n-1$ actions. For all $1 \leq i \leq k$. let $W_i = \{w_1, \dots, w_i\}$. Further, we define the equivalence relation $\sim_i$ by: $q_1 \sim_i q_2$ if $q_1$ and $q_2$ are not separated by any refusal trace in $W_i$. Since $W$ is minimal, the equivalence relations $\sim_1, \dots, \sim_k$ are distinct and have different (increasing) numbers of equivalence classes. Now observe that $\sim_1$ has at least two equivalence classes and so $\sim_k$ has at least $k+1$ equivalence classes. It is now sufficient to note that the number of equivalence classes is bounded above by $n$ and so $k+1 \leq n$ as required. □

**Example 4.2.1.** Consider the $\mathcal{OTS}$ induced by a language $\mathcal{L}$ from Fig. 3. The set $W = \{\{b, c\}, \bullet.b, \bullet.c\}$ is a characterising set for $\mathcal{L}$. Indeed, observe that:

- $\{b, c\}$ separates $q_1$ from every other state in $\mathcal{Q}(\mathcal{L}) \setminus \{q_1\}$
- $\bullet.b$ separates $q_0$ from every other state in $\mathcal{Q}(\mathcal{L}) \setminus \{q_0\}$
- $\bullet.c$ separates states in $\{q_1, q_3\}$ from states in $\{q_0, q_2\}$.

## 5. Test generation

In this section we introduce a complete test generation technique for testing from a (finite) OTS induced by a specification language $\mathcal{L}_S$. The aim is to test whether the implementation defines the same language as the specification; we are testing for equivalence.

We now show how one can test to check properties of the state cover $V$ and characterising set $W$. We will use the notion of two languages $\mathcal{L}_1$ and $\mathcal{L}_2$, normally the specification and SUT languages, agreeing on a set $A$ of refusal traces.

**Definition 5.1.** Given set $A$ of refusal traces, languages $\mathcal{L}_1$ and $\mathcal{L}_2$ agree on $A$, written $\mathcal{L}_1 \equiv_A \mathcal{L}_2$, if $\mathcal{L}_1 \cap A = \mathcal{L}_2 \cap A$.

**Proposition 5.1.** *Let us suppose that the specification language $\mathcal{L}_S$ induces finite state OTS $\mathcal{OTS}(\mathcal{L}_S)$, with state set $\mathcal{Q}(\mathcal{L}_S)$ of cardinality $n = |\mathcal{Q}(\mathcal{L}_S)|$, and the implementation language $\mathcal{L}_I$ induces OTS $\mathcal{OTS}(\mathcal{L}_I)$, with state set $\mathcal{Q}(\mathcal{L}_I)$. Further, let us suppose that $V$ is a state cover for $\mathcal{OTS}(\mathcal{L}_S)$ and $W$ is a characterising set for $\mathcal{OTS}(\mathcal{L}_S)$. If $\mathcal{L}_S \equiv_{V \cup V.W} \mathcal{L}_I$ then the following hold.*

1. *$V \subseteq \mathcal{L}_I$ and $V$ reaches $n$ distinct language states of $\mathcal{OTS}(\mathcal{L}_I)$.*
2. *$W$ separates the language states of $\mathcal{OTS}(\mathcal{L}_I)$ reached by $V$.*

**Proof.** That $V \subseteq \mathcal{L}_I$ follows immediately from $\mathcal{L}_S \equiv_{V \cup V.W} \mathcal{L}_I$.

By definition, each refusal trace in $V$ must reach a unique specification state in $\mathcal{Q}(\mathcal{L}_S)$. Now consider two language states $s$ and $s'$ of $\mathcal{L}_S$ reached by distinct refusal traces $v, v' \in V$. Let us suppose that $q$ and $q'$ are the language states of $\mathcal{OTS}(\mathcal{L}_I)$ reached by $v$ and $v'$ respectively.

Since $W$ is a characterising set for $\mathcal{OTS}(\mathcal{L}_S)$, there must exist $\sigma \in W$ that separates $s$ and $s'$. Without loss of generality, we assume that $\sigma \in s \setminus s'$. Then $v.\sigma \in \mathcal{L}_S$ and $v'.\sigma \notin \mathcal{L}_S$. Since $\mathcal{L}_S \equiv_{V.W} \mathcal{L}_I$, we therefore have that $v.\sigma \in \mathcal{L}_I$ and $v'.\sigma \notin \mathcal{L}_I$, hence $\sigma \in q \setminus q'$, and so $\sigma$ separates $q$ and $q'$.

We now know that the language states of $\mathcal{OTS}(\mathcal{L}_I)$ reached by $V$ are separated by $W$. This also implies that these language states must be distinct as required. □

### 5.1. A simplified scenario

We now adapt the W-method to our scenario. We start by first considering a simplified setting where the number of language states of the implementation does not exceed that of the specification.

Using the notions of a state cover and a characterising set, our test suite needs to ensure that the transitions are properly executed. As a result of non-determinism, when testing from the OTS induced by $\mathcal{L}_S$, there may be more than one transition that can be followed if an event $a$ is received in state $s$. We therefore define a set of refusal traces that extends $V$ to execute the transitions of $\mathcal{OTS}(\mathcal{L}_S)$.

**Definition 5.2.** Given specification $\mathcal{L}_S$ and normal state cover $V$ for $\mathcal{L}_S$, we let $ReqTr(V, \mathcal{L}_S)$ and $ReqRef(V, \mathcal{L}_S)$ be defined as follows.

$$\begin{aligned} ReqTr(V, \mathcal{L}_S) &= \{v.X.a \in \mathbf{FRT}(\mathcal{L}_S) \mid v \in V \wedge a \in \Sigma \wedge X \in \mathcal{P}(\Sigma) \cup \{\bullet\}\} \\ ReqRef(V, \mathcal{L}_S) &= \{v.X \in \mathbf{FRT}(\mathcal{L}_S) \mid v \in V \wedge X \in \mathcal{P}(\Sigma) \cup \{\bullet\}\} \end{aligned}$$

Note that all refusal traces in the above sets are required to be present in the implementation (corresponding to, respectively, required transitions and refusals). We use these sets, as well as $V$ and $W$, to construct the following test suite that contains both *positive tests* (refusal traces of the specification language) and *negative tests* (refusal traces that are not refusal traces of the specification language).

**Definition 5.3.** Given specification $\mathcal{L}_S$, state cover $V$ for $\mathcal{L}_S$, and characterising set $W$ for $\mathcal{L}_S$:

$$T1(\mathcal{L}_S, V, W) = V \cup V.W \cup ReqTr(V, \mathcal{L}_S) \cup ReqTr(V, \mathcal{L}_S).W \cup ReqRef(V, \mathcal{L}_S)$$

The above tests will be used to check for the presence of required transitions and refusals, as well as the absence of transitions to incorrect states, but it is also necessary to check that the SUT does not have any additional, forbidden behaviours: refusal traces of the form $v.X$ or $v.X.a$ that are not refusal traces of the specification. We therefore define the following set of additional negative tests: refusal traces that the SUT should *not* have.

**Definition 5.4.** Given specification $\mathcal{L}_S$ and state cover $V$ for $\mathcal{L}_S$:

$$\begin{array}{lcl} ForbRef(\mathcal{L}_S, V) & = & \{v.X \notin \mathcal{L}_S \mid v \in V \wedge X \in \mathcal{P}(\Sigma)\} \\ ForbTr(\mathcal{L}_S, V) & = & \{v.X.a \notin \mathcal{L}_S \mid v \in V \wedge X \in \mathcal{P}(\Sigma) \cup \{\bullet\} \wedge v.X \in \mathcal{L}_S\} \\ T2(\mathcal{L}_S, V) & = & ForbRef(\mathcal{L}_S, V) \cup ForbTr(\mathcal{L}_S, V) \\ \mathcal{T}_{simple}(\mathcal{L}_S, V, W) & = & T1(\mathcal{L}_S, V, W) \cup T2(\mathcal{L}_S, V) \end{array}$$

**Theorem 5.2.** *Let us suppose that $\mathcal{L}_S$ and $\mathcal{L}_I$ are languages, $\mathcal{Q}(\mathcal{L}_S)$ is finite, and $\mathcal{OTS}(\mathcal{L}_I)$ has no more states than $\mathcal{OTS}(\mathcal{L}_S)$. Further, let us suppose that $V$ is a state cover and $W$ is a characterising set for $\mathcal{OTS}(\mathcal{L}_S)$. If $\mathcal{L}_S \equiv_{\mathcal{T}_{simple}(\mathcal{L}_S, V, W)} \mathcal{L}_I$ then $\mathcal{OTS}(\mathcal{L}_S)$ and $\mathcal{OTS}(\mathcal{L}_I)$ are isomorphic.*

**Proof.** From Proposition 5.1 we know that $V \subseteq \mathcal{L}_I$, the refusal traces in $V$ reach $|\mathcal{Q}(\mathcal{L}_S)|$ distinct states of $\mathcal{OTS}(\mathcal{L}_I)$, and that these states are separated by $W$. This in particular means that $|\mathcal{Q}(\mathcal{L}_S)| = |\mathcal{Q}(\mathcal{L}_I)|$, and hence $V$ reaches all states in $\mathcal{Q}(\mathcal{L}_I)$. Since also $\epsilon \in V$, $V$ is a state cover for $\mathcal{Q}(\mathcal{L}_I)$.

$V$ induces a relation $\sim_V \subseteq \mathcal{Q}(\mathcal{L}_S) \times \mathcal{Q}(\mathcal{L}_I)$ defined by: $s \sim_V q$ if and only if there exists $v \in V$ such that $v$ reaches $s$ in $\mathcal{OTS}(\mathcal{L}_S)$ and $v$ reaches $q$ in $\mathcal{OTS}(\mathcal{L}_I)$. Since $V$ is a state cover of both $\mathcal{OTS}(\mathcal{L}_S)$ and $\mathcal{OTS}(\mathcal{L}_I)$, one can easily observe that $\sim_V$ is in fact a bijection.

By Proposition 5.1 and $|\mathcal{Q}(\mathcal{L}_S)| = |\mathcal{Q}(\mathcal{L}_I)|$, we have that $W$ separates the states of $\mathcal{OTS}(\mathcal{L}_I)$. As a result, $W$ induces the relation $\simeq_W \subseteq \mathcal{Q}(\mathcal{L}_S) \times \mathcal{Q}(\mathcal{L}_I)$ defined by $s \simeq_W q$ if and only if $s \equiv_W q$. Again, since $W$ is a characterising set of both $\mathcal{L}_S$ and $\mathcal{L}_I$, one can easily show that $\simeq_W$ is a bijection.

From $\mathcal{L}_S \equiv_{V.W} \mathcal{L}_I$ we have $\sim_V \subseteq \simeq_W$, and since $\sim_V$ and $\simeq_W$ are both bijections between $\mathcal{Q}(\mathcal{L}_S)$ and $\mathcal{Q}(\mathcal{L}_I)$, we have in fact $\sim_V = \simeq_W$. We now need to prove that the bijection is an isomorphism, that is, preserves initial states, and transitions.

Due to $\epsilon \in V$, $\sim_V = \simeq_W$ obviously preserves the initial states; we proceed to show that transitions are preserved as well.

Take any $v \in V$ and let $s$ and $q$ be the states of $\mathcal{Q}(\mathcal{L}_S)$ and $\mathcal{Q}(\mathcal{L}_I)$ respectively that are reached by $v$.

The first important observation is that $\mathbf{FR}(s) = \mathbf{FR}(q)$. Indeed, we have:

$X \in \mathbf{FR}(s) \iff$ [def. of $\mathbf{FR}$]
$X \in s \wedge \forall a \in \Sigma \setminus X : X.a \in s \iff [s = \mathcal{L}_S \mid v]$
$v.X \in \mathcal{L}_S \wedge \forall a \in \Sigma \setminus X : v.X.a \in \mathcal{L}_S \iff [\mathcal{L}_S \equiv_{ReqRef(V,\mathcal{L}_S) \cup ForbRef(V,\mathcal{L}_S)} \mathcal{L}_I]$
$v.X \in \mathcal{L}_I \wedge \forall a \in \Sigma \setminus X : v.X.a \in \mathcal{L}_S \iff [\mathcal{L}_S \equiv_{ReqTr(V,\mathcal{L}_S) \cup ForbTr(V,\mathcal{L}_S)} \mathcal{L}_I]$
$v.X \in \mathcal{L}_I \wedge \forall a \in \Sigma \setminus X : v.X.a \in \mathcal{L}_I \iff [q = \mathcal{L}_I \mid v]$
$X \in q \wedge \forall a \in \Sigma \setminus X : X.a \in q \iff$ [def. of $\mathbf{FR}$]
$X \in \mathbf{FR}(q)$

We now proceed to show that all pairs of successor states of $s$ and $q$ in their respective OTSs are related by $\simeq_W$.

Consider an arbitrary $X.a \in s \cup q$. We will first show that $X.a \in s \cap q$. There are two cases:

- if $X.a \in s$, then also $\mathsf{top}_s(X) \in s$, hence $v.\mathsf{top}_s(X).a \in \mathcal{L}_S$. Since $\mathcal{L}_S \equiv_{ReqTr(V,\mathcal{L}_S)} \mathcal{L}_I$, we have $v.\mathsf{top}_s(X).a \in \mathcal{L}_I$, hence $\mathsf{top}_s(X).a \in q$, and we obtain $X.a \in q$ from RT1.
- if $X.a \in q$, then $v.X.a \in \mathcal{L}_I$. Were it the case that $v.X.a \notin \mathcal{L}_S$, we would have either $v.X.a \in ForbTr(V, \mathcal{L}_S)$ or $v.X \in ForbRef(V, \mathcal{L}_S)$, but since $\mathcal{L}_S \equiv_{ForbTr(V,\mathcal{L}_S)} \mathcal{L}_I$ and $\mathcal{L}_S \equiv_{ForbRef(V,\mathcal{L}_S)} \mathcal{L}_I$, this would entail either $v.X.a \notin \mathcal{L}_I$ or $v.X \notin \mathcal{L}_I$, a contradiction. Hence $v.X.a \in \mathcal{L}_S$, from which we obtain $X.a \in s$.

Hence we have shown in particular that transition labelled with $X.a$ is defined for both $s$ and $q$.

Since $\mathbf{FR}(s) = \mathbf{FR}(q)$, $s \mid X.a = s \mid \mathsf{top}_s(X).a$, and $q \mid X.a = q \mid \mathsf{top}_q(X).a$, we can further assume that $X \in \mathbf{FR}(s) = \mathbf{FR}(q)$. This in particular means that $X.a \in ReqTr(V, \mathcal{L}_S)$.

Let $s' = s \mid X.a = \mathcal{L}_S \mid v.X.a$ and $q' = q \mid X.a = \mathcal{L}_I \mid v.X.a$. Take an arbitrary $w \in W$.

We have:

$w \in s' \iff$
$v.X.a.w \in \mathcal{L}_S \iff [\mathcal{L}_S \equiv_{ReqTr(V,\mathcal{L}_S).W} \mathcal{L}_I]$
$v.X.a.w \in \mathcal{L}_I \iff$
$w \in q'$

We therefore have that $s' \equiv_W q'$ and so $s' \simeq_W q'$. Therefore $\simeq_W$ (and so also $\sim_V$) is preserved under transitions and hence is an isomorphism between $\mathcal{OTS}(\mathcal{L}_S)$ and $\mathcal{OTS}(\mathcal{L}_I)$ as required. □

**Example 5.2.1.** We shall now generate the test suite for our running example, i.e. language $\mathcal{L}$ of LTS from Fig. 2, whose $\mathcal{OTS}(\mathcal{L})$ is shown in Fig. 3.

Recall from Examples 4.1.1 and 4.2.1 that we have the following state cover and characterising set for $\mathcal{L}$:

$$\begin{aligned} V &= \{\epsilon,\ \{b,c\}.a,\ \{b,c\}.a.\bullet.a,\ \{b,c\}.a.\bullet.a.\{c\}.a\} \\ W &= \{\{b,c\},\ \bullet.b,\ \bullet.c\} \end{aligned}$$

We therefore obtain:

$$\begin{aligned} V.W \;=\; & \{\{b,c\},\ \bullet.b,\ \bullet.c, \\ & \{b,c\}.a.\{b,c\},\ \{b,c\}.a.\bullet.b,\ \{b,c\}.a.\bullet.c, \\ & \{b,c\}.a.\bullet.a.\{b,c\},\ \{b,c\}.a.\bullet.a.\bullet.b,\ \{b,c\}.a.\bullet.a.\bullet.c, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\{b,c\},\ \{b,c\}.a.\bullet.a.\{c\}.a.\bullet.b,\ \{b,c\}.a.\bullet.a.\{c\}.a.\bullet.c\} \end{aligned}$$

For convenience in further construction, the following table lists, in the respective columns: all traces $v$ in $V$, their corresponding states in $\mathcal{OTS}(\mathcal{L})$, and for each $v$, all fundamental refusals in $\mathcal{L} \mid v$.

| $v$ | state in $\mathcal{OTS}(\mathcal{L})$ | $\mathbf{FR}(\mathcal{L} \mid v)$ |
|---|---|---|
| $\epsilon$ | $q_0$ | $\{b,c\}$ |
| $\{b,c\}.a$ | $q_1$ | $\emptyset, \{b\}, \{c\}$ |
| $\{b,c\}.a.\bullet.a$ | $q_2$ | $\{c\}, \{b,c\}$ |
| $\{b,c\}.a.\bullet.a.\{c\}.a$ | $q_3$ | $\emptyset, \{b\}, \{c\}, \{b,c\}$ |

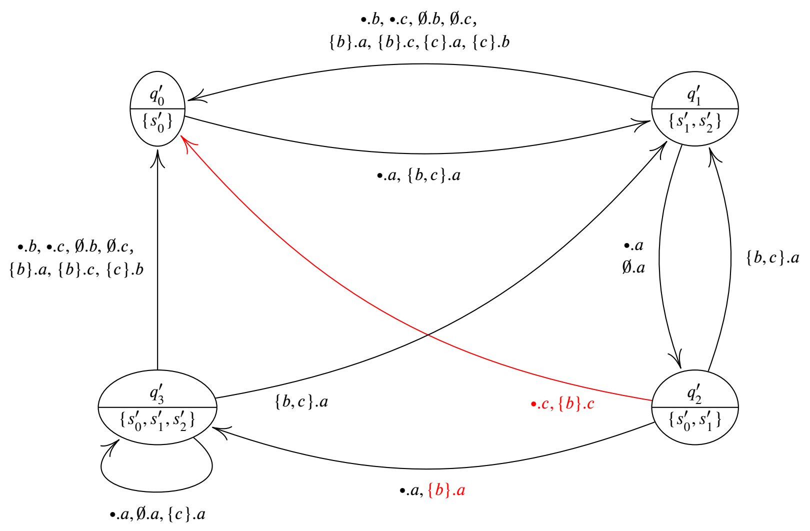
The set $ReqRef(V,\mathcal{L}_S)$ can be constructed by taking, for each trace $v \in V$ (first column above), all its extensions $v.X$ where $X \in \mathbf{FR}(\mathcal{L} \mid v)$ (last column), plus the trace $v.\bullet$:

$$\begin{aligned} ReqRef(V,\mathcal{L}_S) \;=\; & \{\bullet,\ \{b,c\}, \\ & \{b,c\}.a.\bullet,\ \{b,c\}.a.\emptyset,\ \{b,c\}.a.\{b\},\ \{b,c\}.a.\{c\}, \\ & \{b,c\}.a.\bullet.a.\bullet,\ \{b,c\}.a.\bullet.a.\{c\},\ \{b,c\}.a.\bullet.a.\{b,c\}, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\bullet,\ \{b,c\}.a.\bullet.a.\{c\}.a.\emptyset, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\{b\},\ \{b,c\}.a.\bullet.a.\{c\}.a.\{c\}, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\{b,c\}\} \end{aligned}$$

The set $ReqTr(V,\mathcal{L}_S)$ can be derived in a simple manner by taking each $v.X$ in the (already generated) $ReqRef(V,\mathcal{L}_S)$ and including, for each action $a$, the trace $v.X.a$ whenever $v.X.a \in \mathcal{L}$ (this in turn can be checked by e.g. inspecting the transitions in Fig. 3):

$$\begin{aligned} ReqTr(V,\mathcal{L}_S) \;=\; & \{\bullet.a,\ \{b,c\}.a, \\ & \{b,c\}.a.\bullet.a,\ \{b,c\}.a.\bullet.b,\ \{b,c\}.a.\bullet.c, \\ & \{b,c\}.a.\emptyset.a,\ \{b,c\}.a.\emptyset.b,\ \{b,c\}.a.\emptyset.c, \\ & \{b,c\}.a.\{b\}.a,\ \{b,c\}.a.\{b\}.c, \\ & \{b,c\}.a.\{c\}.a\ \{b,c\}.a.\{c\}.b, \\ & \{b,c\}.a.\bullet.a.\bullet.a,\ \{b,c\}.a.\bullet.a.\bullet.b, \\ & \{b,c\}.a.\bullet.a.\{c\}.a,\ \{b,c\}.a.\bullet.a.\{c\}.b, \\ & \{b,c\}.a.\bullet.a.\{b,c\}.a, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\bullet.a,\ \{b,c\}.a.\bullet.a.\{c\}.a.\bullet.b,\ \{b,c\}.a.\bullet.a.\{c\}.a.\bullet.c, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\emptyset.a,\ \{b,c\}.a.\bullet.a.\{c\}.a.\emptyset.b,\ \{b,c\}.a.\bullet.a.\{c\}.a.\emptyset.c, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\{b\}.b,\ \{b,c\}.a.\bullet.a.\{c\}.a.\{b\}.c, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\{c\}.b,\ \{b,c\}.a.\bullet.a.\{c\}.a.\{c\}.a, \\ & \{b,c\}.a.\bullet.a.\{c\}.a.\{b,c\}.a\} \end{aligned}$$

The remaining component of the first part of our test suite, i.e. $T1(\mathcal{L}_S,V,W)$ is the concatenation $ReqTr(V,\mathcal{L}_S).W$ For brevity, we only show its fragment:



$\bullet.b$, $\bullet.c$, $\emptyset.b$, $\emptyset.c$,
$\{b\}.a$, $\{b\}.c$, $\{c\}.a$, $\{c\}.b$
$q'_0$
$\{s'_0\}$
$q'_1$
$\{s'_1, s'_2\}$
$\bullet.a$, $\{b,c\}.a$
$\bullet.b$, $\bullet.c$, $\emptyset.b$, $\emptyset.c$,
$\{b\}.a$, $\{b\}.c$, $\{c\}.b$
$\bullet.a$
$\emptyset.a$
$\{b,c\}.a$
$q'_3$
$\{s'_0, s'_1, s'_2\}$
$\{b,c\}.a$
$\bullet.c$, $\{b\}.c$
$q'_2$
$\{s'_0, s'_1\}$
$\bullet.a$, $\{b\}.a$
$\bullet.a, \emptyset.a, \{c\}.a$

**Fig. 4.** OTS induced by a language consisting of a faulty candidate implementation. Transitions differing from those of the specification OTS in Fig. 3 have been highlighted.

$$
\begin{aligned}
ReqTr(V,\mathcal{L}_S).W \quad = \quad & \{\bullet.a.\{b,c\},\ \{b,c\}.a.\{b,c\},\\
& \{b,c\}.a.\bullet.a.\{b,c\},\ \{b,c\}.a.\bullet.b.\{b,c\},\ \{b,c\}.a.\bullet.c.\{b,c\},\\
& \dots\\
& \bullet.a.\bullet.b,\ \{b,c\}.a.\bullet.b,\\
& \{b,c\}.a.\bullet.a.\bullet.b,\ \{b,c\}.a.\bullet.b.\bullet.b,\ \{b,c\}.a.\bullet.c.\bullet.b,\\
& \dots\\
& \bullet.a.\bullet.c,\ \{b,c\}.a.\bullet.c,\\
& \{b,c\}.a.\bullet.a.\bullet.c,\ \{b,c\}.a.\bullet.b.\bullet.c,\ \{b,c\}.a.\bullet.c.\bullet.c,\\
& \dots\\
& \}
\end{aligned}
$$

We now proceed to generate the second part of the test suite $T2(\mathcal{L}_S, V)$, which consists of two sets of forbidden behaviours. $ForbRef(\mathcal{L}_S, V)$ contains, for each $v \in V$, all forbidden refusal extensions $v.X$. Note that we can optimise this set by including only minimal refusals $X$ forbidden after $v$. The second set, $ForbTr(\mathcal{L}_S, V)$, consists of forbidden continuations of $v$ of the form $X.a$.

| $v$ | minimal forbidden refusals in $\mathcal{L} \mid v$ | forbidden transitions after $v$ |
|---|---|---|
| $\epsilon$ | $\{a\}$ | $\{b\}.c, \{c\}.b$ |
| $\{b,c\}.a$ | $\{a\}, \{b,c\}$ | |
| $\{b,c\}.a.\bullet.a$ | $\{a\}, \{b\}$ | $\bullet.c, \{b\}.c$ |
| $\{b,c\}.a.\bullet.a.\{c\}.a$ | $\{a\}$ | |

We therefore obtain:

$$
\begin{aligned}
ForbRef(\mathcal{L}_S, V) \quad = \quad & \{\{a\},\\
& \{b,c\}.a.\{a\},\ \{b,c\}.a.\{b,c\},\\
& \{b,c\}.a.\bullet.a.\{a\},\ \{b,c\}.a.\bullet.a.\{b\},\\
& \{b,c\}.a.\bullet.a.\{c\}.a.\{a\}\}
\end{aligned}
$$

$$
\begin{aligned}
ForbTr(\mathcal{L}_S, V) \quad = \quad & \{\{b\}.c,\ \{c\}.b,\\
& \{b,c\}.a.\bullet.a.\bullet.c\}
\end{aligned}
$$

**Example 5.2.2.** Suppose an incorrect candidate implementation $\mathcal{L}_I$ results from a modified LTS from Fig. 2, where the direction of the $a$-transition from $s_1$ to $s_2$ has been switched. The corresponding $\mathcal{OTS}$ induced by the language $\mathcal{L}_I$ is given in Fig. 4. Observe that in particular, for $v = \{b,c\}.a.\bullet.a$ and $w = \bullet.c$ we have a negative test $\{b,c\}.a.\bullet.a.\bullet.c \in V.W \cap \mathcal{L}_I$, proving $\mathcal{L}_I \neq \mathcal{L}$.

### 5.2. The general case

We now extend our approach to the general case where there is an (arbitrary but fixed in advance) upper bound $m \geq n$ on the number of states of $\mathcal{OTS}(\mathcal{L}_I)$. To handle the possible larger number of states in $\mathcal{OTS}(\mathcal{L}_I)$, an exhaustive exploration with arbitrary (sub)traces outside $V$ and $W$ is required.

We start by defining some sets of traces, parameterised with a number $k$, which corresponds to the difference between the aforementioned bounds on the number of OTS states i.e. $m - n$, as well as the length of the additional arbitrary subtraces that need to be added to the tests.

**Definition 5.5.** Given specification $\mathcal{L}_S$ and state cover $V$ for $\mathcal{L}_S$, we let $ReqTr_k(V, \mathcal{L}_S)$ and $ReqRef_k(V, \mathcal{L}_S)$ be defined as follows.

$$\begin{aligned} ReqTr_k(V, \mathcal{L}_S) &= \{ v.\sigma.X.a \in \mathcal{L}_S \cap \mathbf{RT}_\mathsf{A} \mid v \in V \wedge |\sigma| \leq k \} \\ ReqRef_k(V, \mathcal{L}_S) &= \{ v.\sigma.X \in \mathcal{L}_S \cap \mathbf{RT}_\mathsf{R} \mid v \in V \wedge |\sigma| \leq k \} \end{aligned}$$

We then obtain the following (partial) test suite that checks that the required transitions have been implemented.

**Definition 5.6.** Given specification $\mathcal{L}_S$, state cover $V$ for $\mathcal{L}_S$, and characterising set $W$ for $\mathcal{L}_S$, we define:

$$\begin{aligned} T1_k(\mathcal{L}_S, V, W) = \; & V \cup V.W \cup ReqTr_k(V, \mathcal{L}_S) \cup ReqRef_k(V, \mathcal{L}_S) \\ & \cup ReqTr_k(V, \mathcal{L}_S).W \end{aligned}$$

Naturally, we also need to extend the set of negative tests defined earlier, so that it includes longer refusal traces that follow $V$.

**Definition 5.7.** Given specification $\mathcal{L}_S$ and state cover $V$ for $\mathcal{L}_S$, we define:

$$\begin{aligned} ForbTr_k(\mathcal{L}_S, V) =& \{ v.\sigma.X.a \in \mathbf{RT}_\mathsf{A} \mid \sigma \in \mathbf{RT}^k(\mathcal{L}_S \,|\, v) \wedge v.\sigma.X \in \mathcal{L}_S \\ & \wedge v.\sigma.X.a \notin \mathbf{RT}(\mathcal{L}_S) \} \\ ForbRef_k(\mathcal{L}_S, V) =& \cup \{ v.\sigma.X \in \mathbf{RT}_\mathsf{R} \mid \sigma \in \mathbf{RT}^k(\mathcal{L}_S \,|\, v) \\ & \wedge v.\sigma.X \notin \mathbf{RT}(\mathcal{L}_S) \} \\ T2_k(\mathcal{L}_S, V) =& ForbTr_k(\mathcal{L}_S, V) \cup ForbRef_k(\mathcal{L}_S, V) \end{aligned}$$

$$\mathcal{T}_k(\mathcal{L}_S, V, W) = T1_k(\mathcal{L}_S, V, W) \cup T2_k(\mathcal{L}_S, V)$$

We can now prove that the generalised test suite is complete for $k = m - n$.

**Theorem 5.3.** *Let us suppose that $\mathcal{L}_S$ and $\mathcal{L}_I$ are languages, $Q(\mathcal{L}_S)$ has $n$ states, and $\mathcal{OTS}(\mathcal{L}_I)$ has no more than $m$ states for some $m \geq n$. Further, let us suppose that $V$ is a normal state cover and $W$ is a characterising set for $\mathcal{OTS}(\mathcal{L}_S)$. Let $k = m - n$. If $\mathcal{L}_S \equiv_{\mathcal{T}_k(\mathcal{L}_S, V, W)} \mathcal{L}_I$, then $\mathcal{L}_S = \mathcal{L}_I$.*

**Proof.** Observe first that we only need to prove the statement for $m$ strictly greater than $n$, since if $m = n$, our test suite is contained in the one defined in Section 5.1, and hence the result follows from Theorem 5.2. We thus assume throughout the proof that $m > n$, and hence $k > 2$.

Suppose that $\mathcal{L}_S \equiv_{\mathcal{T}_k(\mathcal{L}_S, V, W)} \mathcal{L}_I$. Recall relations $\simeq_W$ and $\sim_V$ from the proof of Theorem 5.2; using a similar reasoning, we can show that $\simeq_W$ and $\sim_V$ coincide – with the proviso that codomains are restricted to the image of $Q(\mathcal{L}_1)$ under $\sim_V$; in other words, $\simeq_W$ and $\sim_V$ are bijections and coincide when restricted to $Q(\mathcal{L}_S) \times \{ \mathcal{L}_I \,|\, v \mid v \in V \}$.

We now proceed to prove the main statement. We will use proof by contradiction, assuming that $\mathcal{L}_S \neq \mathcal{L}_I$. Since $\epsilon \in V$, we know that every $\sigma \in \mathbf{RT}(\mathcal{L}_S) \cup \mathbf{RT}(\mathcal{L}_I)$ is of the form $v.\sigma'$ for some $v \in V$.

Since $\mathcal{L}_S \neq \mathcal{L}_I$, we can choose some *shortest* refusal trace $\sigma$ such that *either* of the following holds:

1. there is some $v_\sigma \in V$ where $v_\sigma.\sigma$ separates $\mathcal{L}_S$ and $\mathcal{L}_I$, or
2. there is some $v_\sigma \in V$ and $w \in W$ such that $v_\sigma.\sigma.w$ separates $\mathcal{L}_S$ and $\mathcal{L}_I$.

Since $\mathcal{L}_S \equiv_{\mathcal{T}_k(\mathcal{L}_S, V, W)} \mathcal{L}_I$ and $k \geq 1$, we have that $|\sigma| \geq 1$. Let $\sigma = \sigma_1.\chi$ where $\chi$ is of the form $\chi = X.a$ or $\chi = X$. In the remainder of the proof, we will reason about two subsets of $Q(\mathcal{L}_I)$: language states reached by $V$, and language states reached by extensions of $v_\sigma$ with prefixes of $\sigma_1$. Formally, we define:

$$\begin{aligned} Q_V &\triangleq \{ \mathcal{L}_I \,|\, v \mid v \in V \} \\ Q_V^+ &\triangleq \{ \mathcal{L}_I \,|\, v_\sigma.\sigma_1' \mid \sigma_1' \text{ is a nonempty prefix of } \sigma_1 \} \end{aligned}$$

Observe that by Proposition 5.1, $Q_V$ contains $n$ states and these are all in different equivalence classes of $\sim_W$.

Consider some non-empty $\sigma_2$ with $\sigma_1 = \sigma_2.\sigma_3$ and the state $s'$ of the specification reached by the trace $v_\sigma.\sigma_2$. Let $q'$ denote the state of $\mathcal{OTS}(\mathcal{L}_I)$ reached by $v_\sigma.\sigma_2$. By the minimality of $\sigma$, we have that $s' \simeq_W q'$. By the definition of $\sigma$, we have that either $\sigma_3.X.a$ separates $s'$ and $q'$ or there is some $w \in W$ such that $\sigma_3.X.a.w$ separates $s'$ and $q'$. Let $v' \in V$ be such that $s' = \mathcal{L}_S \mid v'$.

If $q'$ is a state in $Q_V$, then $q'$ must be the state in $Q_V$ that is reached by $v'$; otherwise we would have $s' \nsim_V q'$, contradicting $s' \simeq_W q'$. But then either $v'.\sigma_3.X.a$ separates $\mathcal{L}_I$ and $\mathcal{L}_S$ or there is some $w \in W$ such that $v'.\sigma_3.X.a.w$ separates $\mathcal{L}_I$ and $\mathcal{L}_S$ and this contradictions the minimality of $\sigma$. We therefore have that $Q_V \cap Q_V^+ = \emptyset$.

Now let us suppose that $\sigma_2$ and $\sigma_3$ are distinct non-empty prefixes of $\sigma_1$, $|\sigma_2| < |\sigma_3|$, $v_\sigma.\sigma_2$ reaches state $q_2$ of $\mathcal{OTS}(\mathcal{L}_I)$ and $v_\sigma.\sigma_3$ reaches state $q_3$ of $\mathcal{OTS}(\mathcal{L}_I)$. We will consider two cases, in each case proving that $q_2 \neq q_3$. First, if $v_\sigma.\sigma_2$ and $v_\sigma.\sigma_3$ reach different states of the specification then $q_2$ and $q_3$ must be separated by $W$ and so $q_2 \neq q_3$ is immediate. In the second case, $v_\sigma.\sigma_2$ and $v_\sigma.\sigma_3$ reach the same state $s$ of the specification. Let us suppose that $q_2 = q_3$ and define $\sigma_4$ as the (possibly empty) trace such that $\sigma_1 = \sigma_3.\sigma_4$. By the definition of $\sigma$, either $v_\sigma.\sigma_3.\sigma_4.X.a$ separates the specification and SUT or there is some $w \in W$ such that $v_\sigma.\sigma_3.\sigma_4.X.a.w$ separates the specification and SUT. But then, since $q_2 = q_3$, we must have that either $v_\sigma.\sigma_2.\sigma_4.X.a$ separates the specification and SUT or there is some $w \in W$ such that $v_\sigma.\sigma_2.\sigma_4.X.a.w$ separates the specification and SUT. This contradicts the minimality of $\sigma$ and so we must have that $q_2 \neq q_3$.

As a result of the above, we know that for any two distinct non-empty prefixes $\sigma_2$ and $\sigma_3$ of $\sigma_1$, we have that $v_\sigma.\sigma_2$ and $v_\sigma.\sigma_3$ reach distinct states of $\mathcal{OTS}(\mathcal{L}_I)$. We therefore have that $|Q_V^+| = |\sigma_1|$.

We now have that $|Q_V| = n$, $|Q_V^+| = |\sigma_1|$, and $Q_V \cap Q_V^+ = \emptyset$. Thus, $|Q_V^+ \cup Q_V| = n + |\sigma_1|$. Since $\mathcal{OTS}(\mathcal{L}_I)$ has at most $m$ states, we therefore have that $n + |\sigma_1| \leq m$, and so $|\sigma_1| \leq m - n$. But this means that $v.\sigma$ is of the form $v.\sigma_1.\chi$ where $|\sigma_1| \leq k$ and $\chi$ is of the form $\chi = X.a$ or $\chi = X$. Hence $v.\sigma \in \mathcal{T}_k(\mathcal{L}_S, V, W)$, providing a contradiction as required. □

## 5.3. An optimised test suite

In this section, we present an optimisation to our test suite for the general case. One component that can particularly contribute to its size is the enumeration of arbitrary traces of length $k$. We will show that, apart from possibly final refusals, we can restrict ourselves to traces containing fundamental refusals (of the specification) only.

We start by defining the optimised test suite.

**Definition 5.8.** Given specification $\mathcal{L}_S$ and normal state cover $V$ for $\mathcal{L}_S$, we let $ReqTr_k^{\mathsf{O}}(V, \mathcal{L}_S)$ and $ReqRef_k^{\mathsf{O}}(V, \mathcal{L}_S)$ be defined as follows.

$$\begin{aligned} ReqTr_k^{\mathsf{O}}(V, \mathcal{L}_S) &= \{v.\sigma.X.a \in \mathbf{FRT}(\mathcal{L}_S) \cap \mathbf{RT}_{\mathsf{A}} \mid v \in V \wedge |\sigma| \leq k\} \\ ReqRef_k^{\mathsf{O}}(V, \mathcal{L}_S) &= \{v.\sigma.X \in \mathbf{FRT}(\mathcal{L}_S) \cap \mathbf{RT}_{\mathsf{R}} \mid v \in V \wedge |\sigma| \leq k\} \end{aligned}$$

Furthermore, we define:

$$T1_k^{\mathsf{O}}(\mathcal{L}_S, V, W) = V \cup V.W \cup ReqTr_k^{\mathsf{O}}(V, \mathcal{L}_S) \cup ReqRef_k^{\mathsf{O}}(V, \mathcal{L}_S) \cup ReqTr_k^{\mathsf{O}}(V, \mathcal{L}_S).W$$

**Definition 5.9.** Given specification $\mathcal{L}_S$ and normal state cover $V$ for $\mathcal{L}_S$, we define:

$$\begin{aligned} ForbTr_k^{\mathsf{O}}(\mathcal{L}_S, V) =& \{v.\sigma.X.a \in \mathbf{RT}_{\mathsf{A}} \mid \sigma \in \mathbf{FRT}^k(\mathcal{L}_S \mid v) \wedge v.\sigma.X \in \mathcal{L}_S \\ & \wedge v.\sigma.X.a \notin \mathbf{RT}(\mathcal{L}_S)\} \\ ForbRef_k^{\mathsf{O}}(\mathcal{L}_S, V) =& \cup \{v.\sigma.X \in \mathbf{RT}_{\mathsf{R}} \mid \sigma \in \mathbf{FRT}^k(\mathcal{L}_S \mid v) \\ & \wedge v.\sigma.X \notin \mathbf{RT}(\mathcal{L}_S)\} \\ T2_k^{\mathsf{O}}(\mathcal{L}_S, V) =& ForbTr_k^{\mathsf{O}}(\mathcal{L}_S, V) \cup ForbRef_k^{\mathsf{O}}(\mathcal{L}_S, V) \end{aligned}$$

$$\mathcal{T}_k^{\mathsf{O}}(\mathcal{L}_S, V, W) = T1_k^{\mathsf{O}}(\mathcal{L}_S, V) \cup T2_k^{\mathsf{O}}(\mathcal{L}_S, V)$$

We shall need the following technical lemmas.

**Lemma 5.4.** *For any two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ we have:*

$$\mathbf{FR}(\mathcal{L}_1) = \mathbf{FR}(\mathcal{L}_2) \iff \mathbf{RT}^1(\mathcal{L}_1) = \mathbf{RT}^1(\mathcal{L}_2)$$

**Proof.** We shall use the following simple observation

$$(*) \quad X \in \mathcal{L} \iff \exists \widehat{X} \in \mathbf{FR}(\mathcal{L}) : X \leq_{\mathbf{RT}} \widehat{X}$$

The above can be shown immediately by considering $\widehat{X} = \mathsf{top}_{\mathcal{L}}(X)$.
"$\Longrightarrow$": Suppose $\mathbf{FR}(\mathcal{L}_1) = \mathbf{FR}(\mathcal{L}_2)$. Take any $\sigma \in \mathbf{RT}^1$. There are two possible cases:

- $\sigma = X$: We have:
  $X \in \mathbf{RT}^1(\mathcal{L}_1) \iff [(*)]$
  $\exists \hat{X} \in \mathbf{FR}(\mathcal{L}_1) : X \leq_{\mathbf{RT}} \hat{X} \iff [\mathbf{FR}(\mathcal{L}_1) = \mathbf{FR}(\mathcal{L}_2)]$
  $\exists \hat{X} \in \mathbf{FR}(\mathcal{L}_2) : X \leq_{\mathbf{RT}} \hat{X} \iff [(*)]$
  $X \in \mathbf{RT}^1(\mathcal{L}_2)$
  - $\sigma = X.a$: First, observe that:

    $$(\dagger)\ \ \hat{X} \in \mathbf{FR}(\mathcal{L}) \wedge \hat{X}.a \in \mathcal{L} \iff \hat{X} \in \mathbf{FR}(\mathcal{L}) \wedge a \notin \hat{X}$$

    We have:
    $X.a \in \mathbf{RT}^1(\mathcal{L}_1) \iff [(*)]$
    $\exists \hat{X} \in \mathbf{FR}(\mathcal{L}_1) : X \leq_{\mathbf{RT}} \hat{X} \wedge \hat{X}.a \in \mathcal{L}_1 \iff [(\dagger)]$
    $\exists \hat{X} \in \mathbf{FR}(\mathcal{L}_1) : X \leq_{\mathbf{RT}} \hat{X} \wedge a \notin \hat{X} \iff [\mathbf{FR}(\mathcal{L}_1) = \mathbf{FR}(\mathcal{L}_2)]$
    $\exists \hat{X} \in \mathbf{FR}(\mathcal{L}_2) : X \leq_{\mathbf{RT}} \hat{X} \wedge a \notin \hat{X} \iff [(\dagger)]$
    $\exists \hat{X} \in \mathbf{FR}(\mathcal{L}_2) : X \leq_{\mathbf{RT}} \hat{X} \wedge \hat{X}.a \in \mathcal{L}_2 \iff [(*)]$
    $X.a \in \mathbf{RT}^1(\mathcal{L}_2)$
- "$\Longleftarrow$": Immediate, as the predicate defining "$X \in \mathbf{FR}(\mathcal{L})$" contains only refusal traces from $\mathbf{RT}^1(\mathcal{L})$. □

**Lemma 5.5.** *For any two languages $\mathcal{L}_1$ and $\mathcal{L}_2$, and any $\ell \geq 1$ we have:*

$$\mathbf{RT}^{\ell}(\mathcal{L}_1) = \mathbf{RT}^{\ell}(\mathcal{L}_2) \iff \mathbf{RT}^{\ell-1}(\mathcal{L}_1) = \mathbf{RT}^{\ell-1}(\mathcal{L}_2)$$
$$\wedge\ \forall \sigma \in \mathbf{FRT}^{\ell-1}(\mathcal{L}_1) : \mathbf{FR}(\mathcal{L}_1 \,|\, \sigma) = \mathbf{FR}(\mathcal{L}_2 \,|\, \sigma)$$

**Proof.** We only need to show the direction from right to left, as the other one is very straightforward.

Suppose that $\mathbf{RT}^{\ell-1}(\mathcal{L}_1) = \mathbf{RT}^{\ell-1}(\mathcal{L}_2)$ and for all $\sigma \in \mathbf{FRT}^{\ell-1}(\mathcal{L}_1)$, we have $\mathbf{FR}(\mathcal{L}_1 \,|\, \sigma) = \mathbf{FR}(\mathcal{L}_2 \,|\, \sigma)$. We shall prove that $\mathbf{RT}^{\ell}(\mathcal{L}_1) \subseteq \mathbf{RT}^{\ell}(\mathcal{L}_2)$ (the proof of other inclusion is symmetric).

Observe first that we in fact have $\mathbf{FR}(\mathcal{L}_1 \,|\, \sigma) = \mathbf{FR}(\mathcal{L}_2 \,|\, \sigma)$ for an arbitrary $\sigma \in \mathbf{RT}^{\ell-1}(\mathcal{L}_1)$. Indeed, from Corollary 3.6.1 it follows that $\mathbf{FR}(\mathcal{L}_1 \,|\, \sigma) = \mathbf{FR}(\mathcal{L}_1 \,|\, \mathsf{top}_{\mathcal{L}}(\sigma)) = \mathbf{FR}(\mathcal{L}_2 \,|\, \mathsf{top}_{\mathcal{L}}(\sigma)) = \mathbf{FR}(\mathcal{L}_2 \,|\, \sigma)$.

Let $\sigma \in \mathbf{RT}^{\ell}(\mathcal{L}_1)$. If $\sigma \in \mathbf{RT}^{\ell-1}(\mathcal{L}_1)$, then immediately from the initial assumption we have $\sigma \in \mathbf{RT}^{\ell-1}(\mathcal{L}_2) \subseteq \mathbf{RT}^{\ell}(\mathcal{L}_2)$. Hence we assume that $|\sigma| = \ell$ and $\sigma = \sigma_0.\chi$ where $|\sigma_0| = \ell - 1$.

From the initial assumption we know that $\mathbf{FR}(\mathcal{L}_1 \,|\, \sigma_0) = \mathbf{FR}(\mathcal{L}_2 \,|\, \sigma_0)$; furthermore, from Lemma 5.4, we obtain $\mathbf{RT}^1(\mathcal{L}_1 \,|\, \sigma_0) = \mathbf{RT}^1(\mathcal{L}_2 \,|\, \sigma_0)$. Since $\chi \in \mathbf{RT}^1(\mathcal{L}_1 \,|\, \sigma_0)$, we therefore have $\chi \in \mathbf{RT}^1(\mathcal{L}_2 \,|\, \sigma_0)$, from which $\sigma = \sigma_0.\chi \in \mathbf{RT}^{\ell}(\mathcal{L}_2 \,|\, \sigma_0)$ follows. □

**Theorem 5.6.** *Let us suppose that $\mathcal{L}_S$ and $\mathcal{L}_I$ are languages, $Q(\mathcal{L}_S)$ has $n$ states, and $\mathcal{OTS}(\mathcal{L}_I)$ has no more than $m$ states for some $m \geq n$. Further, let us suppose that $V$ is a normal state cover and $W$ is a characterising set for $\mathcal{OTS}(\mathcal{L}_S)$. Let $k = m - n$. If $\mathcal{L}_S \equiv_{\mathcal{T}_k^o(\mathcal{L}_S,V,W)} \mathcal{L}_I$, then $\mathcal{L}_S = \mathcal{L}_I$.*

**Proof.** The proof boils down to showing that $\mathcal{L}_S \equiv_{\mathcal{T}_k(\mathcal{L}_S,V,W)} \mathcal{L}_I$; the statement will then follow immediately from Theorem 5.3.

We shall first prove the following statement:

(*) For any $s \in Q(\mathcal{L}_S)$ and $q \in Q(\mathcal{L}_I)$ such that $s \sim_V q$, we have $\mathbf{RT}^{k+1}(s) = \mathbf{RT}^{k+1}(q)$.

We prove (*) by showing that $\mathbf{RT}^{\ell}(s) = \mathbf{RT}^{\ell}(q)$ for all $0 \leq \ell \leq k+1$ by induction on $\ell$. For $\ell = 0$ the statement holds trivially. For $\ell = 1$, we have already shown in the proof of Theorem 5.2 that $\mathbf{FR}(s) = \mathbf{FR}(q)$, and from Lemma 5.4 we obtain $\mathbf{RT}^1(s) = \mathbf{RT}^1(q)$. Hence the base case holds.

For the inductive step, we assume that $\mathbf{RT}^{\ell}(s) = \mathbf{RT}^{\ell}(q)$ for some $1 \leq \ell \leq k$; we need to show that $\mathbf{RT}^{\ell+1}(s) = \mathbf{RT}^{\ell+1}(q)$. Due to Lemma 5.5, it suffices to show that for all $\sigma \in \mathbf{FRT}^{\ell}(s)$, we have $\mathbf{FR}(s \,|\, \sigma) = \mathbf{FR}(q \,|\, \sigma)$.

Let $v \in V$ be such that $s = \mathcal{L}_S \,|\, v$ and $q = \mathcal{L}_I \,|\, v$. We start by unfolding the two statements in the bi-implication that we need to show, i.e. $X \in \mathbf{FR}(s \,|\, \sigma) \iff X \in \mathbf{FR}(q \,|\, \sigma)$.

We have:
$X \in \mathbf{FR}(s \,|\, \sigma) \iff$ [def. of $\mathbf{FR}$]
$X \in s \,|\, \sigma \wedge \forall a \in \Sigma \setminus X : X.a \in s \,|\, \sigma \iff$ [def. of $s \,|\, \sigma$]
$\sigma.X \in s \wedge \forall a \in \Sigma \setminus X : \sigma.X.a \in s \iff [s = \mathcal{L}_S \,|\, v]$
$v.\sigma.X \in \mathcal{L}_S \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_S$
On the other hand:
$X \in \mathbf{FR}(q \,|\, \sigma) \iff$ [def. of $\mathbf{FR}$]
$X \in q \,|\, \sigma \wedge \forall a \in \Sigma \setminus X : X.a \in q \,|\, \sigma \iff$ [def. of $q \,|\, \sigma$]
$\sigma.X \in q \wedge \forall a \in \Sigma \setminus X : \sigma.X.a \in q \iff [q = \mathcal{L}_I \,|\, v]$
$v.\sigma.X \in \mathcal{L}_I \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_I$
Hence it suffices to show that:

$$(\dagger)\ \ v.\sigma.X \in \mathcal{L}_S \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_S \iff$$
$$v.\sigma.X \in \mathcal{L}_I \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_I$$

- "$\Longrightarrow$": Suppose that

$$v.\sigma.X \in \mathcal{L}_S \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_S$$

Observe that, since $v.\sigma.X \in \mathbf{FRT}(\mathcal{L}_S)$, and $|\sigma| \leq \ell \leq k$, we have $v.\sigma.X \in ReqRef_k^{\mathrm{O}}(V, \mathcal{L}_S)$, hence from $\mathcal{L}_S \equiv_{ReqRef_k^{\mathrm{O}}(V,\mathcal{L}_S)} \mathcal{L}_I$, we obtain

$$v.\sigma.X \in \mathcal{L}_I \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_S$$

Moreover, for any $a \in \Sigma \setminus X$, from $v.\sigma.X.a \in \mathbf{FRT}(\mathcal{L}_S)$ and $|\sigma| \leq \ell \leq k$, we have $v.\sigma.X.a \in ReqTr_k^{\mathrm{O}}(V, \mathcal{L}_S)$. Hence from $\mathcal{L}_S \equiv_{ReqTr_k^{\mathrm{O}}(V,\mathcal{L}_S)} \mathcal{L}_I$ and $\ell + 1 \leq k$, we obtain

$$v.\sigma.X \in \mathcal{L}_I \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_I$$

- "$\Longleftarrow$": Suppose that

$$v.\sigma.X \in \mathcal{L}_I \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_I$$

Observe first that it must be the case that $v.\sigma.X \in \mathcal{L}_S$; otherwise, as $v.\sigma \in \mathbf{FRT}(\mathcal{L}_S)$, we would have $v.\sigma.X \in ForbRef_k^{\mathrm{O}}(V, \mathcal{L}_S)$, which, combined with $v.\sigma.X \in \mathcal{L}_I$, would contradict $\mathcal{L}_S \equiv_{ForbRef_k^{\mathrm{O}}(V,\mathcal{L}_S)} \mathcal{L}_I$.

Let $\widehat{X} = \mathsf{top}_{\mathcal{L}_S, v.\sigma}(X)$. We will show that $\widehat{X} = X$. Suppose, towards contradiction, that $\widehat{X} \setminus X \neq \emptyset$, and let $a \in \widehat{X} \setminus X$. From the definition of $\mathsf{top}_{\mathcal{L}_S, v.\sigma}$, we have $v.\sigma.X.a \notin \mathcal{L}_S$. Observe that $v.\sigma.X.a$ also meets other conditions from the definition of $ForbTr_k^{\mathrm{O}}(V, \mathcal{L}_S)$, hence $v.\sigma.X.a \in ForbTr_k^{\mathrm{O}}(V, \mathcal{L}_S)$. However, $v.\sigma.X.a \notin \mathcal{L}_S$ and $v.\sigma.X.a \in \mathcal{L}_I$ contradicts $\mathcal{L}_S \equiv_{ForbTr_k^{\mathrm{O}}(V,\mathcal{L}_S)} \mathcal{L}_I$.

We have thus shown that $X = \mathsf{top}_{\mathcal{L}_S, v.\sigma}(X)$, hence $X \in \mathbf{FR}(\mathcal{L}_S \,|\, v.\sigma)$, and from the definition of $\mathbf{FR}$ we have $\forall a \in \Sigma \setminus X$ : $v.\sigma.X.a \in \mathcal{L}_S$. We have thus finally established that

$$v.\sigma.X \in \mathcal{L}_S \wedge \forall a \in \Sigma \setminus X : v.\sigma.X.a \in \mathcal{L}_S$$

concluding the proof of (†) and (∗).

We now proceed to prove that $\mathcal{L}_S \equiv_{\mathcal{T}_k(\mathcal{L}_S, V, W)} \mathcal{L}_I$.

Take any $\pi \in (\mathcal{T}_k(\mathcal{L}_S, V, W)) \setminus (\mathcal{T}_k^{\mathrm{O}}(\mathcal{L}_S, V, W))$. Observe that $\pi$ must be of the form $\pi = v.\sigma.\chi$ where $v \in V$ and $|\sigma| \leq k$. Let $s = \mathcal{L}_S \,|\, v$ and $q = \mathcal{L}_I \,|\, v$.

We have:

$v.\sigma.\chi \in \mathcal{L}_S \iff$

$\sigma.\chi \in s \iff$ [Corollary 3.6.1]

$\mathsf{top}_s(\sigma).\chi \in s \iff$

$v.\mathsf{top}_s(\sigma).\chi \in \mathcal{L}_S \iff [\mathcal{L}_S \equiv_{\mathcal{T}_k^{\mathrm{O}}(\mathcal{L}_S,V,W)} \mathcal{L}_I]$

$v.\mathsf{top}_s(\sigma).\chi \in \mathcal{L}_I \iff$ [(*) implies $\mathbf{FRT}^{k+1}(s) = \mathbf{FRT}^{k+1}(q)$]

$v.\mathsf{top}_q(\sigma).\chi \in \mathcal{L}_I \iff$

$\mathsf{top}_q(\sigma).\chi \in q \iff$ [Corollary 3.6.1]

$\sigma.\chi \in q \iff$

$v.\sigma.\chi \in \mathcal{L}_I$ □

### 5.4. Coffee machine example

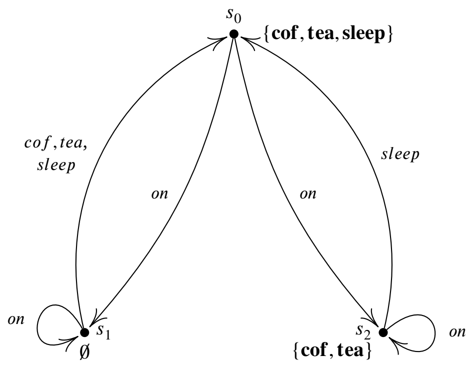
In this section, we consider an example of a simple coffee machine whose idea is based on the system from Fig. 1, i.e. a machine that offers both tea and coffee, or no beverage at all. We shall start by defining a somewhat improved specification; in particular, we wish to work with a system that operates perpetually rather than terminating after one beverage is produced. To this end, we consider the system given in Fig. 5. Its initial state ($s_0$) can be seen as a "sleeping" mode, from which it can be activated by pressing the *on* button. After that the system proceeds to one of two active states: $s_1$, in which the user is offered to choose between tea and coffee, or $s_2$ that models an active state without authorisation to use the machine. After a beverage is provided, the system returns to the original sleeping state, which may also happen after a period of inactivity ($sleep$ action). Note that the *on* button can be pressed in active states ($s_1$ and $s_2$ as well); in such case, the system remains in its current state.

We can provide the following state cover $V$ and characterising set $W$:

$$\begin{aligned} V &= \{\epsilon,\ \{c,t,s\}.on,\ \{c,t,s\}.on.\{c,t\}.on\} \\ W &= \{\{s\},\ \{c,t\}\} \end{aligned}$$

$W$ is a characterising set since:

- $\{s\}$ separates $q_0$ from $q_1$ and $q_2$
- $\{c,t\}$ separates $q_2$ from $q_1$



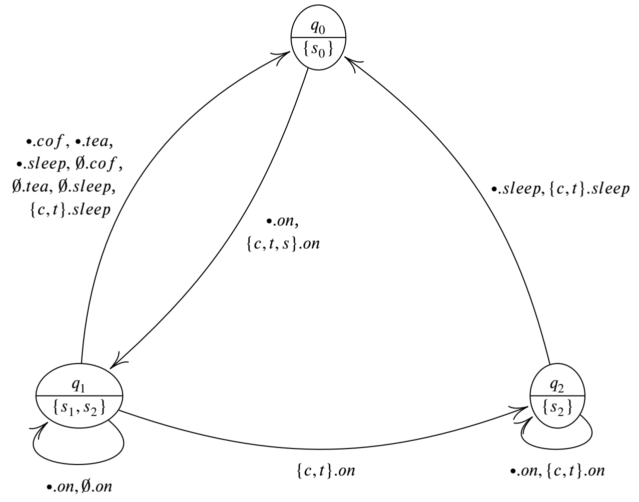


**Fig. 5.** Specification of a simple coffee machine (above) and its corresponding OTS (below).

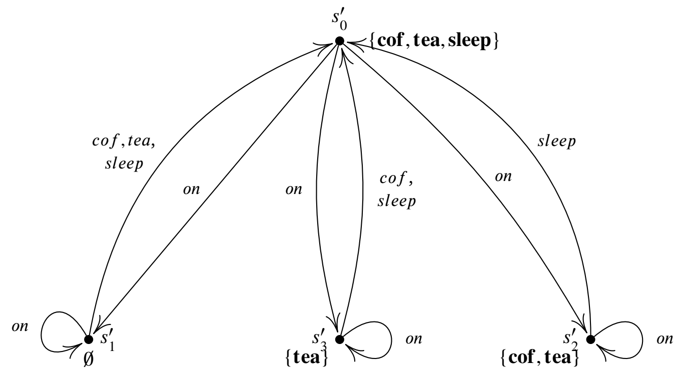
We further obtain:

$$\begin{aligned} V.W \;=\; & \{\{s\},\ \{c,t\},\\ & \{c,t,s\}.on.\{s\},\ \{c,t,s\}.on.\{c,t\},\\ & \{c,t,s\}.on.\{c,t\}.on.\{s\},\ \{c,t,s\}.on.\{c,t\}.on.\{c,t\}\} \end{aligned}$$

We shall assume that an implementation OTS may have at most two more states than the specification OTS ($n = 3$), hence $m = 5$, and $k = 5 - 3 = 2$. This means that our test suite needs to contain additional trace components resulting from exhaustive exploration of paths up to length 2. This, combined with the presence of multiple labels for paths between the same states, makes it undesirable to provide tests with such a level of detail as in Example 5.2.1. We shall therefore present only the general idea and some examples of tests, and subsequently demonstrate the way the faults are uncovered on two instances of erroneous implementations.

Compared with the simple test suite from Example 5.2.1, the additional feature in our test suite will be subtraces of length 2 (denoted in the test suite Definitions 5.8 and 5.9 by $\sigma$) that can be added in between $v$ and the remainder of the trace. To give some idea of how this affects the size of the test suite (even on such a small example and value of $k$) while not enumerating all the possible $\sigma$-subtraces, we instead list all the possible sequences/paths of OTS states that can be traversed with OTS transitions:

| $v$ | state in $\mathcal{OTS}(\mathcal{L})$ | State paths contributing additional subtraces $\sigma$ |
|---|---|---|
| $\epsilon$ | $q_0$ | $q_0{\cdot}q_1$, $q_0{\cdot}q_1{\cdot}q_0$, $q_0{\cdot}q_1{\cdot}q_1$, $q_0{\cdot}q_1{\cdot}q_2$ |
| $\{c,t,s\}.on$ | $q_1$ | $q_1{\cdot}q_0$, $q_1{\cdot}q_1$, $q_1{\cdot}q_2$, $q_1{\cdot}q_0{\cdot}q_1$, $q_1{\cdot}q_1{\cdot}q_0$, $q_1{\cdot}q_1{\cdot}q_1$, $q_1{\cdot}q_1{\cdot}q_2$, $q_1{\cdot}q_2{\cdot}q_0$, $q_1{\cdot}q_2{\cdot}q_2$ |
| $\{c,t,s\}.on.\{c,t\}.on$ | $q_2$ | $q_2{\cdot}q_0$, $q_2{\cdot}q_2$, $q_2{\cdot}q_0{\cdot}q_1$, $q_2{\cdot}q_2{\cdot}q_0$, $q_2{\cdot}q_2{\cdot}q_2$ |



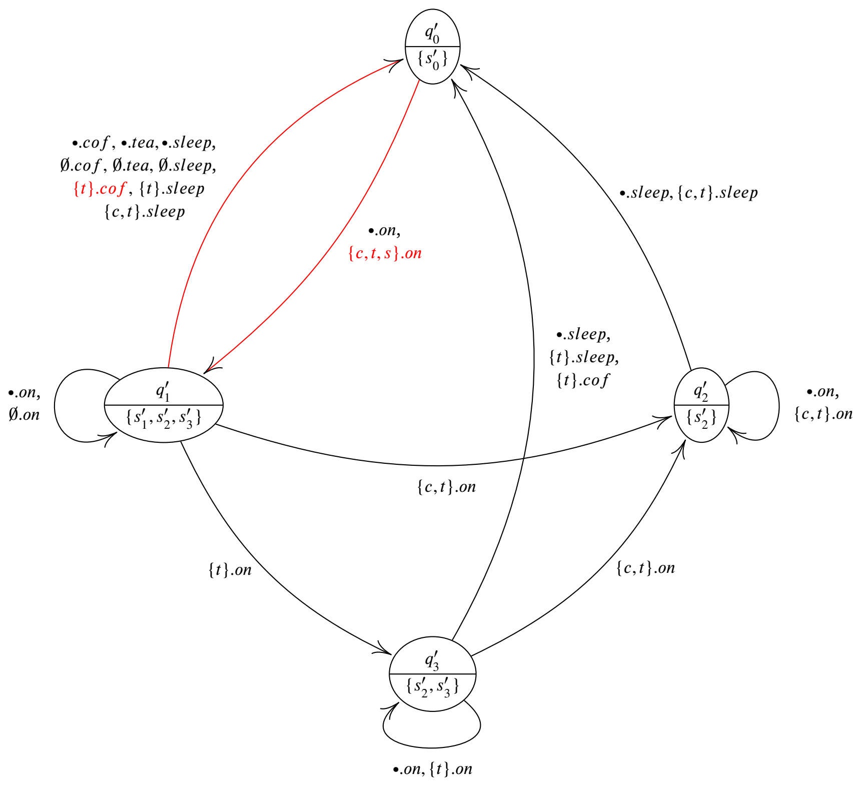


**Fig. 6.** LTS illustrating an erroneous implementation of the coffee machine from Fig. 5 (above) and its corresponding OTS (below).

Each state path in the last column contributes potentially multiple subtraces $\sigma$: for instance, for the path $q_1.q_0.q_1$, we have 14 corresponding $\sigma$ traces, since there are 7 distinct transitions from $q_1$ to $q_0$ and 2 transitions in the opposite direction. Below, we provide examples of tests generated for $v = \{c,t,s\}.on$ involving some of the aforementioned subtraces $\sigma$ contributed by the path $q_1.q_0.q_1$:

- $ReqTr_2^{\circ}(V, \mathcal{L}_S)$: $\{c,t,s\}.on.\bullet.cof.\bullet.on.\{c,t\}.on$, $\{c,t,s\}.on.\bullet.tea.\{c,t,s\}.on.\emptyset.on$, $\{c,t,s\}.on.\bullet.sleep.\bullet.on.\bullet.on$
- $ReqRef_2^{\circ}(V, \mathcal{L}_S)$: $\{c,t,s\}.on.\emptyset.cof.\{c,t,s\}.on.\{c,t\}$
- $ForbTr_2^{\circ}(\mathcal{L}_S, V)$: $\{c,t,s\}.on.\emptyset.sleep.\{c,t,s\}.on.\{t\}.cof$, $\{c,t,s\}.on.\{c,t\}.sleep.\bullet.on.\{c\}.tea$

**Example 5.6.1.** Fig. 6 depicts an incorrect candidate implementation $\mathcal{L}_I$ of the specification $\mathcal{L}$ given in Fig. 5. Observe that in particular, for $v = \{cof, tea, sleep\}.on$, $\sigma = \epsilon$, we have

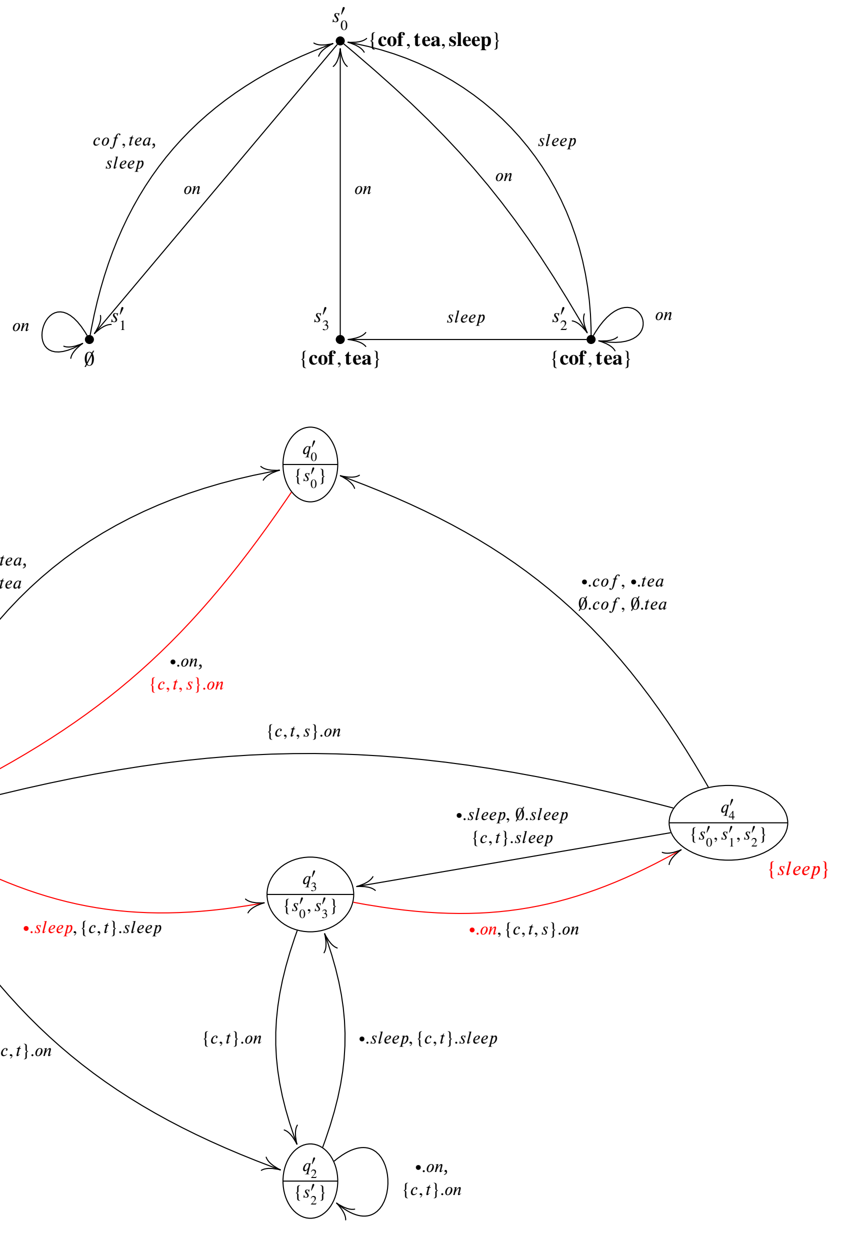

$s'_0$
$\{\mathbf{cof}, \mathbf{tea}, \mathbf{sleep}\}$
$cof, tea, sleep$
$on$
$on$
$sleep$
$on$
$on$
$s'_1$
$\emptyset$
$s'_3$
$\{\mathbf{cof}, \mathbf{tea}\}$
$sleep$
$s'_2$
$\{\mathbf{cof}, \mathbf{tea}\}$
$on$
$q'_0$ $\{s'_0\}$
$\bullet.cof, \bullet.tea, \emptyset.cof, \emptyset.tea$
$\bullet.cof, \bullet.tea$ $\emptyset.cof, \emptyset.tea$
$\bullet.on, \{c,t,s\}.on$
$\{c,t,s\}.on$
$q'_1$ $\{s'_1, s'_2\}$
$\bullet.sleep, \emptyset.sleep$ $\{c,t\}.sleep$
$q'_4$ $\{s'_0, s'_1, s'_2\}$
$\{sleep\}$
$q'_3$ $\{s'_0, s'_3\}$
$\bullet.sleep, \{c,t\}.sleep$
$\bullet.on, \{c,t,s\}.on$
$\{c,t\}.on$
$\{c,t\}.on$
$\bullet.sleep, \{c,t\}.sleep$
$q'_2$ $\{s'_2\}$
$\bullet.on, \{c,t\}.on$

**Fig. 7.** LTS illustrating an erroneous implementation of the coffee machine from Fig. 5 (above) and its corresponding OTS (below).

$$v.\sigma.\{tea\}.cof = \{cof, tea, sleep\}.on.\{tea\}.cof \in ForbTr_2^{\circ}(\mathcal{L}_S, V) \cap \mathcal{L}_I$$

proving $\mathcal{L}_I \neq \mathcal{L}$. We note that in fact we have

$$\{cof, tea, sleep\}.on.\{tea\}.cof \in ForbTr_0^{\circ}(\mathcal{L}_S, V) \cap \mathcal{L}_I$$

hence the fault is revealed in this instance by the simplest test suite for $\mathcal{L}$, i.e. $\mathcal{T}_{simple}(\mathcal{L}, V, W) = \mathcal{T}_0(\mathcal{L}, V, W)$.

**Example 5.6.2.** Fig. 7 depicts an incorrect candidate implementation $\mathcal{L}'_I$ of the specification $\mathcal{L}$ given in Fig. 5. Observe that in particular, for $v = \{cof, tea, sleep\}.on$, $\sigma = \bullet.sleep.\bullet.on$ and refusal $\{sleep\}$, forbidden after $v.\sigma$, we have

$$v.\sigma.\{sleep\} = \{cof, tea, sleep\}.on.\bullet.sleep.\bullet.on.\{sleep\} \in ForbRef_2^{\circ}(\mathcal{L}_S, V) \cap \mathcal{L}_I$$

proving $\mathcal{L}'_I \neq \mathcal{L}$.

We note that, in contrast to the previous example, the simplest test suite $\mathcal{T}_{simple}(\mathcal{L}, V, W)$ does not uncover the fault.

## 6. Conclusions

In this work, we have developed a model-independent method of generating finite test suites for refusal trace semantics, based on languages of refusal traces alone. We set off by showing how to produce an observation transition system from a given language of refusal traces. Subsequently, we have provided a method of generating a finite complete test suite from a language LTS, provided that the LTS is finite. The method is inspired by the W method from the realm of finite state machines.

There are a number of potential lines for future research. Firstly, we can explore the possibility to further reduce the size of test suites through equivalence-based testing [14] that utilises alphabet equivalence. Another type of optimisation could be accomplished by employing methods from [6,7] to remove redundant traces. We also wish to extend the scope of our work so that it can accommodate inputs and outputs. Another interesting extension involves discrete time, where a special action tock is used [2].

## CRediT authorship contribution statement

**Maciej Gazda:** Writing – review & editing, Writing – original draft. **Robert M. Hierons:** Writing – review & editing, Writing – original draft, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work was supported by EPSRC grant EP/R025134/2, RoboTest: Systematic Model-Based Testing and Simulation of Mobile Autonomous Robots.

## References

[1] Philip J. Armstrong, Gavin Lowe, Joël Ouaknine, Bill Roscoe, Model checking timed CSP, in: Andrei Voronkov, Margarita V. Korovina (Eds.), HOWARD-60: A Festschrift on the Occasion of Howard Barringer's 60th Birthday, in: EPiC Series in Computing, vol. 42, EasyChair, 2014, pp. 13–33.
[2] James Baxter, Ana Cavalcanti, Maciej Gazda, Robert M. Hierons, Testing using CSP models: time, inputs, and outputs, ACM Trans. Comput. Log. 24 (2) (Jan 2023), https://doi.org/10.1145/3572837.
[3] Ana Cavalcanti, Robert M. Hierons, Sidney C. Nogueira, Inputs and outputs in CSP: a model and a testing theory, ACM Trans. Comput. Log. 21 (3) (2020) 24:1–24:53, https://doi.org/10.1145/3379508.
[4] Tsun S. Chow, Testing software design modelled by finite state machines, IEEE Trans. Softw. Eng. 4 (1978) 178–187.
[5] Marie-Claude Gaudel, Testing can be formal too, in: 6th International Joint Conference CAAP/FASE Theory and Practice of Software Development (TAPSOFT'95), in: Lecture Notes in Computer Science, vol. 915, Springer, 1995, pp. 82–96.
[6] Maciej Gazda, Robert M. Hierons, Removing redundant refusals: minimal complete test suites for failure trace semantics, in: 2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), 2021, pp. 1–13.
[7] Maciej Gazda, Robert M. Hierons, Removing redundant refusals: minimal complete test suites for failure trace semantics, Inf. Comput. 291 (2023) 105009, https://doi.org/10.1016/j.ic.2023.105009.
[8] R. van Glabbeek, The linear time-branching time spectrum I. The semantics of concrete, sequential processes, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, North Holland, 2001, Chapter 1.
[9] Wolfgang Grieskamp, Nicolas Kicillof, Keith Stobie, Victor Braberman, Model-based quality assurance of protocol documentation: tools and methodology, J. Softw. Testing Verif. Reliability 21 (1) (2011) 55–71.
[10] Lex Heerink, Jan Tretmans, Refusal testing for classes of transition systems with inputs and outputs, in: Formal Description Techniques and Protocol Specification, Testing and Verification (FORTE X/PSTV XVII), in: IFIP Conference Proceedings, vol. 107, Chapman & Hall, 1997, pp. 23–38.
[11] F.C. Hennie, Fault-detecting experiments for sequential circuits, in: Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design, Princeton, New Jersey, November 1964, pp. 95–110.
[12] Robert M. Hierons, Kirill Bogdanov, Jonathan P. Bowen, Rance Cleaveland, John Derrick, Jeremy Dick, Marian Gheorghe, Mark Harman, Kalpesh Kapoor, Paul Krause, Gerald Lüttgen, Anthony J.H. Simons, Sergiy A. Vilkomir, Martin R. Woodward, Hussein Zedan, Using formal specifications to support testing, ACM Comput. Surv. 41 (2) (2009) 9:1–9:76.
[13] Wen-ling Huang, Jan Peleska, Complete model-based equivalence class testing for nondeterministic systems, Form. Asp. Comput. 29 (2) (2017) 335–364, https://doi.org/10.1007/s00165-016-0402-2.
[14] Wen-ling Huang, Jan Peleska, Model-based testing strategies and their (in)dependence on syntactic model representations, Int. J. Softw. Tools Technol. Transf. 20 (4) (2018) 441–465, https://doi.org/10.1007/s10009-017-0479-9.
[15] David Lee, Mihalis Yannakakis, Testing finite-state machines: state identification and verification, IEEE Trans. Comput. 43 (3) (1994) 306–320.
[16] G.L. Luo, G.v. Bochmann, A. Petrenko, Test selection based on communicating nondeterministic finite-state machines using a generalized wp-method, IEEE Trans. Softw. Eng. 20 (2) (1994) 149–161.
[17] E.F. Moore, Gedanken-experiments, in: C. Shannon, J. McCarthy (Eds.), Automata Studies, Princeton University Press, 1956.
[18] Joël Ouaknine, Discrete analysis of continuous behaviour in real-time concurrent systems, PhD thesis, University of Oxford, UK, 2000, https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.365293.
[19] Jan Peleska, Wen-ling Huang, Ana Cavalcanti, Finite complete suites for CSP refinement testing, Sci. Comput. Program. 179 (2019) 1–23.
[20] A. Petrenko, N. Yevtushenko, A. Lebedev, A. Das, Nondeterministic state machines in protocol conformance testing, in: Proceedings of Protocol Test Systems, VI (C-19), 28–30 September, Elsevier Science (North-Holland), Pau, France, 1994, pp. 363–378.
[21] Alexandre Petrenko, Nina Yevtushenko, Adaptive testing of nondeterministic systems with FSM, in: 15th International IEEE Symposium on High-Assurance Systems Engineering, HASE 2014, Miami Beach, FL, USA, January 9-11, 2014, IEEE Computer Society, 2014, pp. 224–228.
[22] I. Phillips, Refusal testing, Theor. Comput. Sci. 50 (3) (1987) 241–284.

[23] Jan Tretmans, Model based testing with labelled transition systems, in: Formal Methods and Testing, in: Lecture Notes in Computer Science, vol. 4949, Springer, 2008, pp. 1–38.
[24] H. Ural, X. Wu, F. Zhang, On minimizing the lengths of checking sequences, IEEE Trans. Comput. 46 (1) (1997) 93–99.
[25] Rob van Glabbeek, Reactive bisimulation semantics for a process algebra with time-outs, in: 31st International Conference on Concurrency Theory (CONCUR 2020), in: LIPIcs, vol. 171, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 6:1–6:23.
[26] M.P. Vasilevskii, Failure diagnosis of automata, Cybernetics 4 (1973) 653–665.