A Suspension-trace Semantics for CSP

Ana Cavalcanti*, Robert M. Hierons†, Sidney Nogueira‡, and Augusto Sampaio§
*University of York, UK
Email: Ana.Cavalcanti@york.ac.uk
†Brunel University London, UK
‡Universidade Federal Rural de Pernambuco, Brazil
§Universidade Federal de Pernambuco, Brazil

***Abstract*—CSP is well established as a process algebra for refinement. Most refinement relations for CSP do not differentiate between inputs and outputs, and so are unsuitable for testing. This paper provides CSP with a denotational semantics based on suspension traces; it gives the traditional CSP operators a novel view, catering for the differences between inputs and outputs. We identify healthiness conditions for the suspension-traces model and include a treatment of termination not contemplated in the context of input-output labelled transition systems. Using our suspension-traces semantics, we provide for CSP a characterisation of the conformance relation ioco, which is widely used in testing. Finally, we propose a strategy to mechanise the verification of conformance according to ioco and suspension-trace refinement using CSP tools. This work provides the basis for a theory of testing for CSP with inputs and outputs, and opens up the possibility of studying algebraic laws and compositional reasoning techniques based on ioco. Ultimately, it contributes to making CSP models useful for both design and testing of systems.**

I. Introduction

As a notation for refinement in the context of reactive programming, CSP is very successful. It has popular model checkers [1], [2] that have encouraged both academic and industrial take up. A variety of semantic models and refinement relations based on traces, refusals and divergences have been studied and related to ensure consistency [3].

The use of CSP in model-based testing, however, has been limited. It does have a theory of testing [4], but, crucially, lacks a notion of inputs and outputs, which is essential in testing.

In CSP, inputs and outputs are both mere synchronisations: there is no differentiated controllability by either the process or the environment. An adaptation of the standard CSP stable-failures model to handle inputs and outputs has been presented in [5]. It, however, entails severe restrictions on parallelism and renaming; it forbids, for example, sychronisation on inputs and renaming inputs to outputs or vice-versa. In addition, its input-output failures refinement relation is incomparable to ioco [6], a conformance relation that is widely used by the testing community.

Testing with inputs and outputs has been extensively studied using input-output labelled transition systems (IOLTS), with ioco as a conformance relation [6]. Recently, CSP has been used to encode the testing theory of IOLTS with ioco, with the CSP refinement model checker FDR used as a technology to generate tests [7]. In that work, a notion of IO process is defined, which partitions the alphabet of a process into input and output events. CSP operators are redefined for IO processes and a conformance relation, cspio, is introduced. It is shown that, provided quiescence is annotated as a special output event, cspio conformance verification (automated as traces-refinement assertions using FDR) corresponds precisely to ioco. This work, however, does not present a formal semantic model to formalise IO processes.

A semantic treatment of ioco in the context of CSP is what we address here. Our work opens to the testing community the possibility of using process algebraic models. In this way, the same models used to reason about specifications and designs can also be used in testing. CSP models can be automatically generated, for example, from diagrammatic domain-specific models for (control) systems, from Java programs [8]–[10], and even from natural language requirements [11].

We give CSP a novel denotational semantics based on suspension traces [6]. This is the traditional model used for characterisation of ioco (in the context of input-output labelled transition systems). With this new model for CSP, we can reason about arbitrary processes using ioco rather than the restricted class of IO processes from [7]. In addition, our semantics gives the traditional CSP operators a different denotational view that caters for inputs and outputs.

Our work provides a basis for a theory of testing for CSP with inputs and outputs, with a conformance relation that corresponds to ioco, and with a justification based on a denotational model. Moreover, it opens the possibility of studying algebraic laws and compositional reasoning for ioco.

Specifically, we make three main contributions here.

1) Firstly, we give a characterisation of a semantic model of suspension traces that identifies its healthiness conditions and includes a treatment of termination not contemplated in the context of IOLTSs.
2) Secondly, we give the definition of some CSP operators in this new model.
3) Finally, we provide a general characterisation of ioco for CSP, and a strategy to use FDR's recent facilities [1] for reasoning about priority to mechanise conformance verification according to ioco and suspension-trace refinement.

In the next section, we briefly present CSP and the standard definition of suspension traces. The healthiness conditions of the suspension traces model are identified in Section III. The

CSP operators are defined in Section IV. The conformance relation and a strategy for mechanised conformance verification are addressed in Section V. Finally, we conclude considering also related and future work in Section VI.

II. Preliminaries

In this section, we describe the background material of our work. Namely, we introduce CSP (Section II-A), its input-output refusal traces model (Section II-B), and define suspension traces (Section II-C).

A. CSP

Processes are the main feature of CSP, used to model systems and components. Processes engage in atomic and instantaneous events that represent visible actions. Events take place synchronously as an agreement between processes and their environment. The alphabet $\Sigma^{\checkmark} = \Sigma \cup \{\checkmark\}$ of a process includes the set Σ of events that can be explicitly used in process definitions, and the special event $\checkmark$ to represent successful termination (as opposed to a deadlock or livelock).

Two basic CSP processes are *STOP* and *SKIP*. The process *STOP* represents the canonical deadlock: a process that never communicates and does not perform internal actions. The process *SKIP* communicates $\checkmark$ and terminates successfully.

For $a \in \Sigma$, the process $a \rightarrow Q$, read a prefix Q, offers the event a and waits for a synchronisation. Upon the occurrence of a, it behaves as the process Q.

An external choice $a \rightarrow P \,\Box\, b \rightarrow Q$ offers a and b to the environment. If it synchronises on a, then the choice resolves to P; if the environment synchronises on b, it is resolved to Q. An internal choice $a \rightarrow P \sqcap b \rightarrow Q$ behaves either as $a \rightarrow P$ or $b \rightarrow Q$. The initial state is not stable due to the existence of an internal event, corresponding to the choice of process. When interacting with $a \rightarrow P \sqcap b \rightarrow Q$, the environment can fail to synchronise if it insists on a or b, but not if it offers both, since exactly one is nondeterministically offered.

The process $P \,[\![X]\!]\, Q$ is the generalised parallel composition of the processes P and Q with synchronisation set $X \subseteq \Sigma$. In this parallelism, P and Q must agree on events that belong to X, but can proceed independently for events not in X. Such a composition terminates successfully if, and only if, P and Q terminate (that is, we have distributed termination). The process $P \,|||\, Q$ is the interleaving between P and Q. In such a composition P and Q communicate any event freely (without synchronisation); $P \,|||\, Q$ is equivalent to $P \,[\![\, \{\} \,]\!]\, Q$.

The CSP process $P \setminus X$, read P hide X, has the same behaviour as P, except that the events that belong to the set X are internalized, and so are not available for synchronisation. The notation $\setminus$ stands for the hiding operator.

An interrupt operator allows us to define a process $P \,\triangle\, Q$, which behaves like P, but continuously offers Q in choice. If, at any point an event offered by Q is chosen, P is interrupted, and the process carries on with the execution of Q. We can think of $P \,\triangle\, Q$ as a distributed external choice of Q over P.

CSP has several semantic models [3]. In the next section, we give an overview of the input-output refusal traces model [12], which can be used to establish the connection between our suspension-traces model and the CSP standard models.

B. Input-Output refusal traces

The input-output refusal traces model of CSP ($\mathcal{RT}^{\mathcal{O}}$) [12] is based on the stable refusal-testing model $\mathcal{RT}$ [3], [13]–[15], adapting the notion of a refusal trace in a manner that distinguishes inputs and outputs.

Classically, a refusal trace includes events a_i and refusals X_i; it has the form $\langle X_0, a_1, X_1, \ldots, a_n, X_n \rangle$, where for $1 \leq i \leq n$ we have that either $X_i = \bullet$ or $X_i \in \mathbb{P}(\Sigma^{\checkmark})$ is a refusal set [3]. The recording of $\bullet$ indicates that no actual refusals have been observed, possibly, due to instability. If $X_i \in \mathbb{P}(\Sigma^{\checkmark})$ and $i < n$, then a_{i+1} is an event in $\Sigma^{\checkmark}$ such that $a_{i+1} \notin X_i$, since the events in X_i are refused immediately before the observation of a_{i+1}. The essential idea is that refusals are observed between events; in contrast, in failures we only observe a refusal set at the end of a trace. Similar to failures, a refusal set can only be observed in a stable state. If a state is unstable then the corresponding X_i must take the value $\bullet$.

The input-output refusal traces model follows the same principle adopted in input-output stable failures in [5]. States that communicate outputs are not stable, so no refusals are observed. In general terms, if we let $rtraces[\![P]\!]$ denote the set of refusal traces of a process P, then the set $IOrtraces^{\mathcal{O}}[\![P]\!]$ of input-output refusal traces of P can be described as follows.

$$
\begin{array}{l}
IOrtraces^{\mathcal{O}}[\![P]\!] = \\
\quad \{ \langle X_0, a_1, X_1, \ldots, a_n, X_n \rangle \mid \exists\, X'_0, \ldots, X'_n \bullet \\
\qquad \langle X'_0, a_1, X'_1, \ldots, a_n, X'_n \rangle \in rtraces[\![P]\!] \wedge \\
\qquad \forall\, 1 \leq i \leq n \bullet (X_i = \bullet \vee X'_i = X_i \cup \mathcal{O}) \\
\quad \}
\end{array}
$$

In words, for a refusal trace $\langle X_0, a_1, X_1, \ldots, a_n, X_n \rangle$ to be an input-output refusal trace we require that its refusals X_i are either $\bullet$ or have been observed in a stable state. Stability now requires that no outputs are available, so in a corresponding refusal trace $\langle X'_0, a_1, X'_1, \ldots, a_n, X'_n \rangle$ of $rtraces[\![P]\!]$ no outputs are enabled so that $X_i \cup \mathcal{O}$ is a refusal that can be observed.

The $\mathcal{RT}$ model is more discriminating than the standard CSP stable-failures model, and similarly $\mathcal{RT}^{\mathcal{O}}$ is more discriminating than the input-output stable failures model. In addition, it is more discriminating than ioco and suspension traces, since it allows the refusal of inputs to be observed. As a result, it is possible to obtain suspension traces from the $\mathcal{RT}^{\mathcal{O}}$ model. This is detailed in the next section.

C. Suspension traces

Suspension traces were originally described [6] in the context of testing from an IOLTS. Suspension traces are similar to traces except that they can also include the observation of quiescence. A system is quiescent if it cannot produce an output or change state without first receiving an input.

The observation of quiescence is normally denoted δ. This is a special type of refusal: one in which the system can refuse the whole set of outputs. For instance, the trace $\langle \delta, in, \delta \rangle$

records that the process is initially stable and does not communicate outputs. Subsequently, the input *in* is communicated and quiescence is again observed.

The interest of the testing community in suspension traces has come from the feasibility to observe them in testing. Typically, we observe quiescence through an implementation-dependent timeout. Additionally, suspension traces are more discriminating than traces. As an example, we consider $in \rightarrow STOP \sqcap out \rightarrow STOP$ and $in \rightarrow STOP \,\Box\, out \rightarrow STOP$, where *in* is an input and *out* is an output. These processes have the same sets of traces, but only $in \rightarrow STOP \sqcap out \rightarrow STOP$ can be quiescent before the event *in*.

Quiescence is the only type of refusal recorded in suspension traces. A suspension trace can be seen as an input-output refusal trace in which refusals are reported only when all outputs are refused, and in this case we simply report that outputs are refused (that is, we do not report refused inputs).

Input-output failures and suspension traces semantics are incomparable. An input-output failure $(s, X \cup O)$ only records quiescence at the end of a trace s, but suspension traces can observe quiescence at any point in the trace. On the other hand, suspension traces only record refusal of the set O (quiescence), while input-output failures record any refusal.

A function, which we call *st*, maps a refusal trace, or equally an input-output refusal trace, to its corresponding suspension trace. If *st* is applied to $\langle X_0, a_1, X_1, \ldots, a_n, X_n \rangle$, the resulting suspension trace retains the events $a_1, \ldots, a_n$, and a refusal set X_i is either removed, if $\mathcal{O} \not\subseteq X_i$, or replaced by δ, if $\mathcal{O} \subseteq X_i$. More precisely, we have the following definition.

$$st(\langle X_0, a_1, X_1, \ldots, a_n, X_n \rangle) = \\ f_\delta(X_0) \frown \langle a_1 \rangle \frown f_\delta(X_1) \frown \ldots \frown \langle a_n \rangle \frown f_\delta(X_n)$$

For a set X of events, $f_\delta(X) = \langle \delta \rangle$ if $\mathcal{O} \subseteq X$ and otherwise $f_\delta(X) = \langle \rangle$. With this, we can define the set $straces[\![P]\!]$ of suspension traces of a process P in terms of the set $IOrtraces^{\mathcal{O}}[\![P]\!]$ of input-output refusal traces of P.

Definition 1:

$$straces[\![P]\!] = st(\!| \; IOrtraces^{\mathcal{O}}[\![P]\!] \; |\!)$$

$IOrtraces^{\mathcal{O}}[\![P]\!]$ is defined in terms of the set $rtraces[\![P]\!]$. A formal account of $rtraces[\![P]\!]$, $IOrtraces^{\mathcal{O}}[\![P]\!]$, and *st* is in [12]. The operator $\_(\!|\ \_\ |\!)$ is relational image.

Example 1: The set of suspension traces for the process $P = out \rightarrow STOP$ is $straces[\![P]\!] = \{\langle\rangle, \langle out \rangle, \langle out, \delta \rangle\}$, for $out \in \mathcal{O}$. □

The ioco conformance relation is traditionally defined in terms of suspension traces. Given a suspension trace σ of a process P, the standard definition of ioco refers to the possible states of P after σ, and the outputs that are possible in these states. We next explore the suspension traces model, and its relation to other CSP models via Definition 1. Suspension traces are used to define ioco in Section V.

TABLE I
SUSPENSION-TRACES MODEL: HEALTHINESS CONDITIONS

ST0	$\sigma_1 \frown \langle \delta, \delta \rangle \frown \sigma_2 \notin ST$
ST1	$\langle\rangle \in ST$
ST2	$\sigma_2 \in ST \wedge \sigma_1 \leq \sigma_2 \Rightarrow \sigma_1 \in ST$
ST3	$\sigma_1 \frown \langle \delta \rangle \frown \sigma_2 \Rightarrow \sigma_1 \frown \sigma_2 \in ST$
ST4	$\sigma \frown \langle e \rangle \in ST \wedge e \in \Sigma^{\checkmark} \Rightarrow \checkmark \notin \operatorname{ran} \sigma$

III. HEALTHINESS CONDITIONS

This section introduces a characterisation of the model of suspension traces, namely, by defining its healthiness conditions. As explained above, the approach we adopt defines suspension traces in terms of the set of input-output refusal traces of a process. Thus, our model can express successful termination behaviour, which is particular to the theory of CSP and not defined for suspension traces of IOLTSs.

We call *STrace* the set of all suspension traces. Using the variable *ST* to stand for an arbitrary set of suspension traces, we define in Table I the conditions that characterise the healthy sets that represent a CSP process. We use variables σ, σ_1, σ_2, and so on, to stand for suspension traces in *STrace*; the notation δ^n to represent a sequence of δ events of size n, for $n \geq 0$; and, $\mathcal{O}_\delta$ to represent $\mathcal{O} \cup \{\delta\}$.

The condition ST0 indicates a difference between the suspension-traces model for CSP and that for IOLTSs: quiescence is not recorded repeatedly. In the model for IOLTSs, if we observe a suspension trace $\sigma_1 \frown \langle \delta \rangle \frown \sigma_2$, then we can also observe all suspension traces of the form $\sigma_1 \frown \delta^n \frown \sigma_2$. This is due to self-loop δ transitions in the stable states of IOLTS. As stated in [16], however, the target state after a δ transition is always the same as the start state. Hence, we can replace any non-empty subsequence δ^n in a trace $\sigma \in ST$ by a single δ without loss of expressiveness in the context of reasoning using ioco or suspension-traces inclusion.

The next conditions are counterparts for healthiness conditions of sets of (input-output) refusal traces [12]. For example, the healthiness conditions RT1 of the refusal traces model requires every healthy set of refusal traces to include $\langle \bullet \rangle$, and RT2 requires that they are prefixed closed. Correspondingly, ST1 and ST2 require the empty trace to be in *ST* and prefix closure. ST3 requires that *ST* is closed under the removal of δ. Finally, the healthiness conditions ST4 addresses termination. If it is recorded, using $\checkmark$, this is the last event in $\Sigma^{\checkmark}$ to appear in the suspension trace; after a $\checkmark$, only δ can appear.

Example 2: The set containing the traces $\langle\rangle$, $\langle \delta \rangle$, $\langle in \rangle$, $\langle \delta, in \rangle$, $\langle in, \checkmark \rangle$, $\langle \delta, in, \checkmark \rangle$, $\langle in, \checkmark, \delta \rangle$, $\langle \delta, in, \checkmark, \delta \rangle$ is an example of a healthy set of suspension traces. This set represents the suspension traces of the process $in \rightarrow SKIP$, for $\mathcal{O} = \emptyset$. □

We prove below that the sets of suspension traces characterised by $straces[\![P]\!]$ are healthy.

ST0*:* Proof by contradiction.

$\sigma_1 \frown \langle \delta, \delta \rangle \frown \sigma_2 \in straces[\![P]\!]$

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet st(\rho) = \sigma_1 \frown \langle \delta, \delta \rangle \frown \sigma_2$
[definition of $straces[\![P]\!]$]

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ i : 1..\#\rho \bullet$
$st(\rho)(i) = \delta \wedge st(\rho)(i+1) = \delta$
[properties of sequences]

This contradicts the definition of *RTrace*, the type of ρ.

ST1:

$\langle \bullet \rangle \in rtraces[\![P]\!]$ [healthiness condition MRT0 for $\mathcal{RT}$]
$\Rightarrow \langle \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]$ [definition of $IOrtraces^{\mathcal{O}}[\![P]\!]$]
$\Rightarrow st(\langle \bullet \rangle) \in straces[\![P]\!]$ [definition of $straces[\![P]\!]$]
$\Rightarrow \langle\rangle \in straces[\![P]\!]$ [definition of st]

ST2:

$\sigma_2 \in straces[\![P]\!] \wedge \sigma_1 \leq \sigma_2$

$\Rightarrow \exists\, \rho_2 : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet \sigma_2 = st(\rho_2) \wedge \sigma_1 \leq \sigma_2$
[definition of $straces[\![P]\!]$]

$\Rightarrow \exists\, \rho_1, \rho_2 : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet$ [st is surjective]
$\sigma_2 = st(\rho_2) \wedge \sigma_1 = st(\rho_1) \wedge \sigma_1 \leq \sigma_2$

$\Rightarrow \exists\, \rho_1, \rho_2 : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet$
$\sigma_2 = st(\rho_2) \wedge \sigma_1 = st(\rho_1) \wedge \rho_1 \leq_{RT} \rho_2$
[property of st: distribution over $\frown$,]
[and prefixing $\leq_{RT}$ for refusal traces]

$\Rightarrow \exists\, \rho_1 : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet \sigma_1 = st(\rho_1)$ [simplification]

$\Rightarrow \sigma_1 \in straces[\![P]\!]$ [definition of $straces[\![P]\!]$]

ST3:

$\sigma_1 \frown \langle \delta \rangle \frown \sigma_2 \in straces[\![P]\!]$

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet st(\rho) = \sigma_1 \frown \langle \delta \rangle \frown \sigma_2$
[definition of $straces[\![P]\!]$]

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ \rho_1, \rho_2 : RTrace;\ X : Refusal \bullet$
$\rho = \rho_1 \frown \langle X \rangle \frown \rho_2 \wedge \sigma_1 = st(\rho_1) \wedge$
$\sigma_2 = st(\rho_2) \wedge refusal\, \mathcal{O} \subseteq_{RT} X$
[property of st]

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ \rho_1, \rho_2 : RTrace \bullet$ [RT1]
$\rho = \rho_1 \frown \langle \bullet \rangle \frown \rho_2 \wedge \sigma_1 = st(\rho_1) \wedge \sigma_2 = st(\rho_2)$

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ \rho_1, \rho_2 : RTrace \bullet$ [property of st]
$\rho = \rho_1 \frown \langle \bullet \rangle \frown \rho_2 \wedge \sigma_1 \frown \sigma_2 = st(\rho)$

$\Rightarrow \sigma_1 \frown \sigma_2 \in straces[\![P]\!]$ [definition of $straces[\![P]\!]$]

ST4: In the following, $e \in \Sigma^{\checkmark}$.

$\sigma \frown \langle e \rangle \in straces[\![P]\!]$

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet \sigma \frown \langle e \rangle = st(\rho)$
[definition of $straces[\![P]\!]$]

$\Rightarrow \exists\, \rho, \rho_1 : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet$
$\rho = \rho_1 \frown \langle e, X \rangle \wedge st(\rho) = \sigma \frown \langle e \rangle$
[definition of st and RT1]

$\Rightarrow \exists\, \rho, \rho_1 : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet$
$\rho = \rho_1 \frown \langle e, X \rangle \wedge st(\rho_1 \frown \langle e, X \rangle) = \sigma \frown \langle e \rangle$
[predicate calculus]

$\Rightarrow \exists\, \rho, \rho_1 : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet$ [RT3]
$\rho = \rho_1 \frown \langle e, X \rangle \wedge st(\rho_1 \frown \langle e, X \rangle) = \sigma \frown \langle e \rangle \wedge$
$\checkmark \notin \operatorname{ran} \rho_1$

$\Rightarrow \exists\, \rho, \rho_1 : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet$ [definition of st]
$\rho = \rho_1 \frown \langle e, X \rangle \wedge st(\rho_1 \frown \langle e, X \rangle) = \sigma \frown \langle e \rangle \wedge$
$\checkmark \notin \operatorname{ran} \sigma$

$\Rightarrow \checkmark \notin \operatorname{ran} \sigma$ [predicate calculus]

An extra condition ST5 defined below holds for the divergence-free processes, since they eventually reach a stable state. Divergent behaviour cannot be observed (or distinguished from a deadlock) by testing, hence a model for divergence-free processes is sufficient here.

ST5 $\quad \sigma \in ST \wedge (\sigma = \langle\rangle \vee \sigma \neq \langle\rangle \wedge \operatorname{last} \sigma \neq \delta) \Rightarrow$
$\exists\, o : \mathcal{O}_\delta \bullet \sigma \frown \langle o \rangle \in ST$

ST5 asserts that it is always possible to observe output or quiescence (δ). A condition similar to ST5 is found in [6, Proposition 4.17, item 4] for IOLTS.

We show below that $straces[\![P]\!]$ is ST5-healthy, provided P is divergence free. In the refusal-traces model, we can characterise the sets of traces RT that represent divergence-free processes using the healthiness condition below [14].

MRT2 $\quad \sigma \frown \langle \bullet \rangle \in RT \Rightarrow \sigma \frown \langle refusal\, \emptyset \rangle \in RT$

MRT2 establishes that a proper refusal, as opposed to $\bullet$, can always be observed, and so the process is always stable.

$\sigma \in straces[\![P]\!] \wedge (\sigma = \langle\rangle \vee \sigma \neq \langle\rangle \wedge \operatorname{last} \sigma \neq \delta)$

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet$ [definition of $straces[\![P]\!]$]
$\sigma = st(\rho) \wedge (\sigma = \langle\rangle \vee \sigma \neq \langle\rangle \wedge \operatorname{last} \sigma \neq \delta)$

$\Rightarrow \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi;\ X : Refusal \bullet$
$\rho = \phi \frown \langle X \rangle \wedge \sigma = st(\rho) \wedge refusal\, \mathcal{O} \not\subseteq_{RT} X$
[definition of st and $(\sigma = \langle\rangle \vee \sigma \neq \langle\rangle \wedge \operatorname{last} \sigma \neq \delta)$]

$\Rightarrow \exists\, \rho, \rho_2 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi;\ X : Refusal \bullet$ [RT1]
$\rho = \phi \frown \langle X \rangle \wedge \rho_2 = \phi \frown \langle \bullet \rangle \wedge$
$\sigma = st(\rho) \wedge refusal\, \mathcal{O} \not\subseteq_{RT} X$

$\Rightarrow \exists\, \rho_2 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$ [definition of st]
$\rho_2 = \phi \frown \langle \bullet \rangle \wedge \sigma = st(\rho_2)$

$\Rightarrow \exists\, \rho_2, \rho_3 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$ [MRT2]
$\rho_2 = \phi \frown \langle \bullet \rangle \wedge \rho_3 = \phi \frown \langle refusal\, \emptyset \rangle \wedge \sigma = st(\rho_2)$

$\Rightarrow \exists\, \rho_3 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$ [definition of st]
$\quad \rho_3 = \phi \frown \langle refusal\, \emptyset \rangle \wedge \sigma = st(\rho_3)$

$\Rightarrow \exists\, \rho_3 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$
$\quad \rho_3 = \phi \frown \langle refusal\, \emptyset \rangle \wedge \sigma = st(\rho_3) \wedge$
$\quad (\ (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \vee$
$\quad\ \ \neg\, (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \ )$
[predicate calculus]

$\Rightarrow \exists\, \rho_3 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$ [RT2]
$\quad \rho_3 = \phi \frown \langle refusal\, \emptyset \rangle \wedge \sigma = st(\rho_3) \wedge$
$\quad (\ (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \vee$
$\quad\ \ \neg\, (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \wedge$
$\quad\ \ \phi \frown \langle refusal\, \emptyset \cup_{RT} refusal\, \mathcal{O} \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!] \ )$

$\Rightarrow \exists\, \rho_3 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$ [property of $\cup_{RT}$]
$\quad \rho_3 = \phi \frown \langle refusal\, \emptyset \rangle \wedge \sigma = st(\rho_3) \wedge$
$\quad (\ (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \vee$
$\quad\ \ \neg\, (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \wedge$
$\quad\ \ \phi \frown \langle refusal\, \mathcal{O} \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!] \ )$

$\Rightarrow \exists\, \rho_3 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$
$\quad \rho_3 = \phi \frown \langle refusal\, \emptyset \rangle \wedge \sigma = st(\rho_3) \wedge$
$\quad (\ (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \vee$
$\quad\ \ \neg\, (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \wedge$
$\quad\ \ st(\phi \frown \langle refusal\, \mathcal{O} \rangle) \in straces[\![P]\!] \ )$
[definition of $straces[\![P]\!]$]

$\Rightarrow \exists\, \rho_3 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$ [definition of st]
$\quad \rho_3 = \phi \frown \langle refusal\, \emptyset \rangle \wedge \sigma = st(\rho_3) \wedge$
$\quad (\ (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \vee$
$\quad\ \ \neg\, (\exists\, o : \mathcal{O} \bullet \phi \frown \langle refusal\, \emptyset, o, \bullet \rangle \in IOrtraces^{\mathcal{O}}[\![P]\!]) \wedge$
$\quad\ \ \sigma \frown \langle \delta \rangle \in straces[\![P]\!] \ )$

$\Rightarrow \exists\, \rho_3 : IOrtraces^{\mathcal{O}}[\![P]\!];\ \phi \bullet$
$\quad \rho_3 = \phi \frown \langle refusal\, \emptyset \rangle \wedge \sigma = st(\rho_3) \wedge$
$\quad (\ (\exists\, o : \mathcal{O} \bullet \sigma \frown \langle o \rangle \in straces[\![P]\!]) \vee$
$\quad\ \ (\sigma \frown \langle \delta \rangle \in straces[\![P]\!]) \ )$
[definition of $straces[\![P]\!]$ and st]

$\Rightarrow \exists\, o : \mathcal{O}_\delta \bullet \sigma \frown \langle o \rangle \in straces[\![P]\!]$ [definition of $\mathcal{O}_\delta$]

Next, we show how to calculate suspension traces for processes defined using the CSP operators, including *SKIP* and $in \rightarrow SKIP$, mentioned above.

IV. Definition of CSP operators

Table II presents the definition of a suspension-traces semantics for the main CSP operators. Using Definition 1, and a definition of input-output refusal traces [12], it is possible to calculate these definitions. For illustration, we present below the calculation for *STOP*, *SKIP*, prefixing and internal choice.

STOP: We recall from Section II-A that *STOP* refuses all events and does not progress. Accordingly, the suspension traces for *STOP* are the empty trace $\langle\rangle$ and the singleton sequence recording just quiescence ($\langle\delta\rangle$). This reflects the fact that *STOP* is always quiescent (although we do not repeat quiescence due to healthiness condition ST0). The calculation of $straces[\![STOP]\!]$ is as follows.

$straces[\![STOP]\!]$

$= st(\!|\ IOrtraces^{\mathcal{O}}[\![STOP]\!]\ |\!)$
[definition of $straces[\![STOP]\!]$]

$= st(\!|\ \{X : Refusal \bullet \langle X \rangle\}\ |\!)$
[input-output refusal traces of *STOP* [12]]

$= \{\sigma : STrace \mid \exists\, X : Refusal \bullet \sigma = st\, \langle X \rangle\}$
[property of relational image]

$= \{\sigma : STrace \mid \exists\, X : Refusal \bullet$ [definition of st]
$\quad \neg\, (refusal\, \mathcal{O} \subseteq_{RT} X) \wedge \sigma = \langle\,\rangle \vee$
$\quad refusal\, \mathcal{O} \subseteq_{RT} X \wedge \sigma = \langle \delta \rangle$
$\quad \}$

$= \{\sigma : STrace \mid \sigma = \langle\,\rangle \vee \sigma = \langle \delta \rangle\ \}$ [predicate calculus]

$= \{\langle\,\rangle, \langle \delta \rangle\}$ [property of sets]

SKIP: The suspension traces for *SKIP* are $\langle \checkmark, \delta \rangle$ and its prefixes. We cannot observe quiescence before the $\checkmark$ event because the process is not stable until it terminates. Upon termination, that is, after recording $\checkmark$, *SKIP* behaves as a deadlock. This semantics can be calculated as follows.

$straces[\![SKIP]\!]$

$= st(\!|\ IOrtraces^{\mathcal{O}}[\![SKIP]\!]\ |\!)$ [definition of $straces[\![SKIP]\!]$]

$= st(\!|\ \{\langle \bullet \rangle\} \cup \{X : Refusal \bullet \langle \bullet, event(\checkmark), X \rangle\}\ |\!)$
[input-output refusal traces of *SKIP* [12]]

$= \{\sigma : STrace \mid \sigma = st(\langle \bullet \rangle)\} \cup$ [relational image]
$\quad \{\sigma : STrace \mid \exists\, X : Refusal \bullet \sigma = st(\langle \bullet, event(\checkmark), X \rangle)\}$

$= \{\langle\,\rangle\} \cup$ [definition of st]
$\quad \{\sigma : STrace \mid \exists\, X : Refusal \bullet$
$\quad\ \ \sigma = st(\langle \bullet \rangle) \frown \langle \checkmark \rangle \frown st(\langle X \rangle)$
$\quad \}$

$= \{\langle\,\rangle\} \cup$ [definition of st]
$\quad \{\sigma : STrace \mid \exists\, X : Refusal \bullet$
$\quad\ \ \sigma = \langle \checkmark \rangle \wedge \neg (refusal\, O \subseteq X) \vee$
$\quad\ \ \sigma = \langle \checkmark, \delta \rangle \wedge refusal\, O \subseteq X$
$\quad \}$

$= \{\langle\,\rangle\} \cup \{\sigma : STrace \mid \sigma = \langle \checkmark \rangle \vee \sigma = \langle \checkmark, \delta \rangle\}$
[predicate calculus]

$= \{\langle\,\rangle, \langle \checkmark \rangle, \langle \checkmark, \delta \rangle\}$ [predicate calculus]

TABLE II
SUSPENSION TRACES MODEL

Process P	$straces[\![P]\!]$
$STOP$	$\{\langle\rangle, \langle\delta\rangle\}$
$SKIP$	$\{\langle\rangle, \langle\checkmark\rangle, \langle\checkmark, \delta\rangle\}$
$a \rightarrow P$	$\{\sigma : STrace \mid \sigma = \langle\rangle \lor (a \notin \mathcal{O} \land \sigma = \langle\delta\rangle)\} \cup$ $\{\sigma_1, \sigma_2 : STrace \mid (\sigma_1 = \langle\rangle \lor (a \notin \mathcal{O} \land \sigma_1 = \langle\delta\rangle)) \land \sigma_2 \in straces[\![P]\!] \bullet \sigma_1 \frown \langle a\rangle \frown \sigma_2\}$
$P \sqcap Q$	$straces[\![P]\!] \cup straces[\![Q]\!]$
$P \,\square\, Q$	$\begin{cases} straces[\![P]\!] \cup straces[\![Q]\!] & \text{if } \langle\delta\rangle \in straces[\![P]\!] \cap straces[\![Q]\!] \\ strip(straces[\![P]\!] \cup straces[\![Q]\!]) & \text{otherwise} \end{cases}$
$P \,[\![X]\!]\, Q$	$\bigcup\{\sigma_1 : straces[\![P]\!];\ \sigma_2 : straces[\![Q]\!] \bullet (\sigma_1 \,[\![X]\!]_{ST}\, \sigma_2)\}$

Prefixing: Given a process P, the suspension traces of the prefixing $a \rightarrow P$ include those of the form $\langle a\rangle \frown \sigma$, where σ is a suspension trace of P, and the prefixes of $\langle a\rangle$. In addition, if a is not an output, the initial state of the prefixing is stable and quiescence is recorded. In this case, the trace $\langle\delta, a\rangle \frown \sigma$ and the prefixes of $\langle\delta, a\rangle$ are also suspension traces of $a \rightarrow P$.

Example 3: For input event *in* and output event *out*, we consider the processes $P1 = in \rightarrow STOP$ and $P2 = out \rightarrow STOP$. The set $straces[\![P1]\!] = \{\langle\rangle, \langle\delta\rangle, \langle in\rangle, \langle\delta, in\rangle, \langle in, \delta\rangle, \langle\delta, in, \delta\rangle\}$ defines that $P1$ is initially stable and waits for the environment to synchronise on *in*. After that, $P1$ deadlocks. Similarly, $straces[\![P2]\!] \mathrel{\widehat{=}} \{\langle\rangle, \langle out\rangle, \langle out, \delta\rangle\}$. This records that $P2$ is not stable initially because *out* is offered. $P2$ becomes stable afterwards. □

In calculating the model of prefixing, we use two lemmas. The first considers the traces before the event a takes place.

Lemma 1:

$$\left(\begin{array}{l} \exists X : Refusal \bullet \\ \quad (X = \bullet \lor event\, a \notin_{RT} X \land a \notin \mathcal{O}) \land \sigma = st(\langle X\rangle) \end{array}\right) = \sigma = \langle\rangle \lor (a \notin \mathcal{O} \land \sigma = \langle\delta\rangle)$$

Proof:

$$\begin{array}{ll} & \exists X : Refusal \bullet \\ & \quad (X = \bullet \lor event\, a \notin_{RT} X \land a \notin \mathcal{O}) \land \sigma = st(\langle X\rangle) \\ = & \quad (\exists X : Refusal \bullet X = \bullet \land \sigma = st(\langle X\rangle)) \lor \\ & \quad a \notin \mathcal{O} \land (\exists X : Refusal \bullet event\, a \notin_{RT} X \land \sigma = st(\langle X\rangle)) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[predicate calculus]} \\ = & \quad \sigma = \langle\rangle \lor \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[definition of } st\text{]} \\ & \quad (\ a \notin \mathcal{O} \land \\ & \qquad (\ (\exists X : Refusal \bullet \\ & \qquad\quad event\, a \notin_{RT} X \land \neg\, (refusal\, \mathcal{O} \subseteq_{RT} X) \land \\ & \qquad\quad \sigma = \langle\rangle) \lor \\ & \qquad (\exists X : Refusal \bullet \\ & \qquad\quad event\, a \notin_{RT} X \land refusal\, \mathcal{O} \subseteq_{RT} X \land \sigma = \langle\delta\rangle)\ )\ ) \\ = & \quad \sigma = \langle\rangle \lor \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[definition of } st\text{]} \\ & \quad (\ a \notin \mathcal{O} \land \\ & \qquad (\ (\exists X : Refusal \bullet \\ & \qquad\quad event\, a \notin_{RT} X \land \neg\, (refusal\, \mathcal{O} \subseteq_{RT} X)) \land \\ & \qquad\quad \sigma = \langle\rangle \lor \\ & \qquad (\exists X : Refusal \bullet \\ & \qquad\quad event\, a \notin_{RT} X \land refusal\, \mathcal{O} \subseteq_{RT} X) \land \sigma = \langle\delta\rangle\ )\ ) \\ = & \quad \sigma = \langle\rangle \lor (a \notin \mathcal{O} \land (\sigma = \langle\rangle \lor \sigma = \langle\delta\rangle)) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[property of sets and predicate calculus]} \\ = & \quad \sigma = \langle\rangle \lor (a \notin \mathcal{O} \land \sigma = \langle\delta\rangle) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[predicate calculus]} \end{array}$$

□

For the traces after a occurs, we have the following result.

Lemma 2:

$$\begin{array}{l} \{\sigma : STrace \mid \exists \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet \\ \quad (X = \bullet \lor event\, a \notin_{RT} X \land a \notin \mathcal{O}) \land \\ \quad \sigma = st(\langle X, event\, a\rangle \frown \rho) \\ \} \\ = \\ \{\sigma_X, \sigma_\rho : STrace \mid \\ \qquad (\sigma_X = \langle\rangle \lor (a \notin \mathcal{O} \land \sigma_X = \langle\delta\rangle)) \land \sigma_\rho \in straces[\![P]\!] \\ \quad \bullet \sigma_X \frown \langle a\rangle \frown \sigma_\rho \\ \} \end{array}$$

Proof:

$$\begin{array}{ll} & \{\sigma : STrace \mid \exists \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet \\ & \quad (X = \bullet \lor event\, a \notin_{RT} X \land a \notin \mathcal{O}) \land \\ & \quad \sigma = st(\langle X, event\, a\rangle \frown \rho) \\ & \} \\ = & \{\sigma : STrace \mid \exists \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet \\ & \quad (X = \bullet \lor event\, a \notin_{RT} X \land a \notin \mathcal{O}) \land \\ & \quad \sigma = st(\langle X\rangle) \frown \langle a\rangle \frown st(\rho) \\ & \} \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[definition of } st\text{]} \end{array}$$

$$
\begin{array}{ll}
= & \{\sigma_X, \sigma_a, \sigma_\rho : STrace \mid \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet \\
& \quad (X = \bullet \vee event\, a \notin_{RT} X \wedge a \notin \mathcal{O}) \wedge \\
& \quad \sigma_X = st(\langle X \rangle) \wedge \sigma_a = \langle a \rangle \wedge \sigma_\rho = st(\rho) \\
& \quad \bullet\ \sigma_X \frown \sigma_a \frown \sigma_\rho \\
& \}
\end{array}
$$

[property of sets]

$$
\begin{array}{ll}
= & \{\sigma_X, \sigma_a, \sigma_\rho : STrace \mid \\
& \quad (\exists X : Refusal \bullet \\
& \quad\quad (X = \bullet \vee event\, a \notin_{RT} X \wedge a \notin \mathcal{O}) \wedge \\
& \quad\quad \sigma_X = st(\langle X \rangle)) \wedge \\
& \quad \sigma_a = \langle a \rangle \wedge \\
& \quad (\exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet \sigma_\rho = st(\rho)) \\
& \quad \bullet\ \sigma_X \frown \sigma_a \frown \sigma_\rho \\
& \}
\end{array}
$$

[predicate calculus]

$$
\begin{array}{ll}
= & \{\sigma_X, \sigma_a, \sigma_\rho : STrace \mid \\
& \quad (\sigma_X = \langle\rangle \vee (a \notin \mathcal{O} \wedge \sigma_X = \langle\delta\rangle)) \wedge \\
& \quad \sigma_a = \langle a \rangle \wedge \\
& \quad (\exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet \sigma_\rho = st(\rho)) \\
& \quad \bullet\ \sigma_X \frown \sigma_a \frown \sigma_\rho \\
& \}
\end{array}
$$

[Lemma 1]

$$
\begin{array}{ll}
= & \{\sigma_X, \sigma_\rho : STrace \mid \\
& \quad (\sigma_X = \langle\rangle \vee (a \notin \mathcal{O} \wedge \sigma_X = \langle\delta\rangle)) \wedge \\
& \quad \sigma_\rho \in \{\rho : IOrtraces^{\mathcal{O}}[\![P]\!] \bullet st(\rho)\} \\
& \quad \bullet\ \sigma_X \frown \langle a \rangle \frown \sigma_\rho \\
& \}
\end{array}
$$

[property of sets]

$$
\begin{array}{ll}
= & \{\sigma_X, \sigma_\rho : STrace \mid \\
& \quad (\sigma_X = \langle\rangle \vee (a \notin \mathcal{O} \wedge \sigma_X = \langle\delta\rangle)) \wedge \\
& \quad \sigma_\rho \in st(\!|\ IOrtraces^{\mathcal{O}}[\![P]\!]\ |\!) \\
& \quad \bullet\ \sigma_X \frown \langle a \rangle \frown \sigma_\rho \\
& \}
\end{array}
$$

[property of relational image]

$$
\begin{array}{ll}
= & \{\sigma_X, \sigma_\rho : STrace \mid \\
& \quad (\sigma_X = \langle\rangle \vee (a \notin \mathcal{O} \wedge \sigma_X = \langle\delta\rangle)) \wedge \\
& \quad \sigma_\rho \in straces[\![P]\!] \\
& \quad \bullet\ \sigma_X \frown \langle a \rangle \frown \sigma_\rho \\
& \}
\end{array}
$$

[definition of $straces[\![P]\!]$]

□

Finally, we proceed with the calculation of the suspension traces of a prefixing $a \to P$ as follows.

$straces[\![a \to P]\!]$

$= st(\!|\ IOrtraces^{\mathcal{O}}[\![a \to P]\!]\ |\!)$ [definition of $straces[\![a \to P]\!]$]

$$
\begin{array}{ll}
= st(\!| & \{X : Refusal \mid \\
& \quad X = \bullet \vee event\, a \notin_{RT} X \wedge a \notin \mathcal{O} \bullet \langle X \rangle \\
& \} \cup \\
& \{\rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \mid \\
& \quad X = \bullet \vee event\, a \notin_{RT} X \wedge a \notin \mathcal{O} \\
& \quad \bullet\ \langle X, event\, a \rangle \frown \rho \\
& \}\ |\!)
\end{array}
$$

[definition of $IOrtraces^{\mathcal{O}}[\![a \to P]\!]$]

$$
\begin{array}{ll}
= & \{\sigma : STrace \mid \exists X : Refusal \bullet \\
& \quad (X = \bullet \vee event\, a \notin_{RT} X \wedge a \notin \mathcal{O}) \wedge \sigma = st(\langle X \rangle)\} \cup \\
& \{\sigma : STrace \mid \exists\, \rho : IOrtraces^{\mathcal{O}}[\![P]\!];\ X : Refusal \bullet \\
& \quad (X = \bullet \vee event\, a \notin_{RT} X \wedge a \notin \mathcal{O}) \wedge \\
& \quad \sigma = st(\langle X, event\, a \rangle \frown \rho)\}
\end{array}
$$

[property of relational image]

$$
\begin{array}{ll}
= & \{\sigma : STrace \mid \sigma = \langle\rangle \vee (a \notin \mathcal{O} \wedge \sigma = \langle\delta\rangle)\} \cup \\
& \{\sigma_X, \sigma_\rho : STrace \mid \\
& \quad (\sigma_X = \langle\rangle \vee (a \notin \mathcal{O} \wedge \sigma_X = \langle\delta\rangle)) \wedge \\
& \quad \sigma_\rho \in straces[\![P]\!] \\
& \quad \bullet\ \sigma_X \frown \langle a \rangle \frown \sigma_\rho \\
& \}
\end{array}
$$

[Lemmas 1 and 2]

$P \sqcap Q$: The suspension traces of the internal choice of the processes P and Q is the union of the suspension traces of P and the suspension traces of Q.

$straces[\![P \sqcap Q]\!]$

$= st(\!|\ IOrtraces^{\mathcal{O}}[\![P \sqcap Q]\!]\ |\!)$ [definition of $straces[\![P \sqcap Q]\!]$]

$= st(\!|\ IOrtraces^{\mathcal{O}}[\![P]\!] \cup IOrtraces^{\mathcal{O}}[\![Q]\!]\ |\!)$

[input-output refusal traces of $P \sqcap Q$ [12]]

$= st(\!|\ IOrtraces^{\mathcal{O}}[\![P]\!]\ |\!) \cup st(\!|\ IOrtraces^{\mathcal{O}}[\![Q]\!]\ |\!)$

[property of relational image]

$= straces[\![P]\!] \cup straces[\![Q]\!]$ [definition of $straces$]

Example 4: We show below the suspension traces of the internal choice $P1 \sqcap P2$.

$$
\begin{array}{l}
straces[\![P1 \sqcap P2]\!] = \\
\quad \{\langle\rangle, \langle\delta\rangle, \langle in\rangle, \langle out\rangle, \langle\delta, in\rangle, \langle in, \delta\rangle, \langle out, \delta\rangle, \langle\delta, in, \delta\rangle\}
\end{array}
$$

The process $P1 \sqcap P2$ is not stable in its initial state, since it can change through an internal event to a state where it behaves as $P1$, or to a state where it behaves as $P2$. Once an internal choice has been made the process behaves like either $P1$ or $P2$ and so can then be stable (despite no observations having been made). This justifies a trace like $\langle\delta\rangle$, which starts with δ even though the initial state of $P1 \sqcap P2$ is unstable. This is a trace of $P1$ as shown in Example 3. □

External choice: The suspension traces of an external choice $P \,\Box\, Q$ reflect the fact that its initial state is quiescent only if P and Q are both initially quiescent themselves. Otherwise, the choice can be made without the agreement of the environment and we cannot observe quiescence.

Accordingly, as usual in CSP models, in defining external choice, we have to consider separately the behaviours before and after the choice is made. The definition of $straces[\![P \,\Box\, Q]\!]$ is given by the union of the suspension traces of P and the suspension traces of Q, if P and Q record quiescence in the first state. Otherwise, the suspension traces are defined by $strip(\!|\ straces[\![P]\!] \cup straces[\![Q]\!]\ |\!)$, that is, the set obtained by applying the function $strip$ to all elements of

$straces[\![P]\!] \cup straces[\![Q]\!]$. Basically, this is again the union of the sets of traces for P and Q, but any leading δ events in these traces are removed. As explained above, in this case, quiescence is not really observed in the choice, even if it may be individually in P or Q. Stripping of the leading δ events is achieved by applying the function *strip*, defined below.

$$strip(\langle\rangle) = \langle\rangle$$
$$strip(\langle\delta\rangle \frown \sigma) = \sigma$$
$$strip(\langle e\rangle \frown \sigma) = \langle e\rangle \frown \sigma \quad \text{if } e \neq \delta$$

As a consequence of the above definitions, internal and external choices can be distinguished in the suspension traces model if one process in the choice is stable and the other is not.

Example 5: The model for $P1 \,\Box\, P2$, where $P1$ and $P2$ are as defined in Example 3, can be calculated as follows.

$$straces[\![P1 \,\Box\, P2]\!]$$
$$= strip(\!| \; straces[\![P1]\!] \cup straces[\![P2]\!] \; |\!) \quad [\langle\delta\rangle \in straces[\![P1]\!] \text{ and } \langle\delta\rangle \notin straces[\![P2]\!]]$$
$$= strip \; (\!| \; \{ \langle\rangle, \langle\delta\rangle, \langle in\rangle, \langle out\rangle, \langle\delta, in\rangle, \langle in, \delta\rangle, \langle out, \delta\rangle, \langle\delta, in, \delta\rangle \} \; |\!) \quad [straces[\![P1]\!] \cup straces[\![P2]\!] \text{ (see Example 4)}]$$
$$= \{\langle\rangle, \langle in\rangle, \langle out\rangle, \langle in, \delta\rangle, \langle out, \delta\rangle, \langle in, \delta\rangle\} \quad [\text{definition of } strip]$$

In $P1 \,\Box\, P2$, the input *in* and the output *out* are offered in choice, so in $straces[\![P1 \,\Box\, P2]\!]$ stability cannot be observed before *in*. Comparing the sets $straces[\![P1 \sqcap P2]\!]$ (see Example 4)) and $straces[\![P1 \,\Box\, P2]\!]$, we can observe that the trace $\langle\delta, in\rangle$ belongs to the former but not to the latter. □

We now give an example where internal and external choice are indistinguishable via suspension traces.

Example 6: We consider $P3 = out2 \rightarrow STOP$, whose model is $straces[\![P3]\!] = \{\langle\rangle, \langle out2\rangle, \langle out2, \delta\rangle\}$. The internal and external choices between $P2$ and $P3$ are indistinguishable.

$$straces[\![P2 \sqcap P3]\!] = straces[\![P2 \,\Box\, P3]\!] = \{\langle\rangle, \langle out\rangle, \langle out2\rangle, \langle out, \delta\rangle, \langle out2, \delta\rangle\}$$

The processes $P2$ and $P3$ communicate outputs in the first state; none of them is stable at that stage. □

Parallelism: The set of suspension traces for the generalised parallel composition $P \,[\![\, X \,]\!]\, Q$ is defined in the usual way, as the combination of the traces of P and Q according to an operator $|[X]|_{ST}$ for traces. Like in all CSP models, the parallel composition $\sigma_a \,|[X]|\, \sigma_b$ of suspension traces σ_a and σ_b defines the set of suspension traces representing the interleaved execution of σ_a and σ_b synchronising on X. The traces are obtained by merging σ_a and σ_b, to represent their interleaved execution. On events in the set X, however, they need to agree. If they do not, the parallel execution does not lead to any trace. If they do, just one occurrence of the corresponding event is included in the merged traces. To explain the issues particular to composition of suspension traces, we use a few examples.

Example 7: The traces of $P4 = out1 \rightarrow in \rightarrow STOP$ and $P5 = in \rightarrow out2 \rightarrow STOP$ are enumerated below.

$$straces[\![P4]\!] = \{ \langle\rangle, \langle out1\rangle, \langle out1, \delta\rangle, \langle out1, in\rangle, \langle out1, \delta, in\rangle, \langle out1, in, \delta\rangle, \langle out1, \delta, in, \delta\rangle \}$$
$$straces[\![P5]\!] = \{ \langle\rangle, \langle\delta\rangle, \langle in\rangle, \langle\delta, in\rangle, \langle in, out2\rangle, \langle\delta, in, out2\rangle, \langle in, out2, \delta\rangle, \langle\delta, in, out2, \delta\rangle \}$$

For $P4 \,[\![\, \{in\} \,]\!]\, P5$, the suspension traces are defined below.

$$straces[\![P4 \,[\![\, \{in\} \,]\!]\, P5]\!] = \{ \langle\rangle, \langle out1\rangle, \langle out1, \delta\rangle, \langle out1, in\rangle, \langle out1, \delta, in\rangle, \langle out1, in, out2\rangle, \langle out1, \delta, in, out2\rangle, \langle out1, in, out2, \delta\rangle, \langle out1, \delta, in, out2, \delta\rangle \}$$

The first state of the parallel composition is not stable since $P4$ can communicate the output $out1$, and $P5$ cannot communicate *in* as it belongs to the synchronisation set. After $out1$, the composition behaves as the process $in \rightarrow STOP \,[\![\, \{in\} \,]\!]\, in \rightarrow out2 \rightarrow STOP$, which is stable and can synchronise on the input *in*. After that, it behaves as $STOP \,[\![\, \{in\} \,]\!]\, out2 \rightarrow STOP$, which is again not stable due to the offer of the output $out2$. Finally, after communicating $out2$, the process is quiescent and behaves as *STOP*. □

So, the composition of $\langle\rangle$ with a suspension trace $\langle\delta\rangle \frown \sigma$ that records quiescence as the first event does not record stability, since stability occurs only if both sides are stable.

Example 8: The composition of $\langle\rangle$ and $\langle\delta, in\rangle$, from the the processes $P4$ and $P5$ in Example 7, gives the traces in $\langle\rangle \,[\![\, \{in\} \,]\!]_{ST}\, \langle in\rangle$, that is, no traces at all. □

The composition of $\langle\delta\rangle$ with itself yields $\langle\delta\rangle$: when both processes are stable, the composition is as well. The composition of $\langle\delta\rangle$ with a trace $\langle y\rangle \frown \sigma$, starting with an event y not in X, yields traces that start with y and continue with traces in $\langle\delta\rangle \,[\![X]\!]_{ST}\, \sigma$. The occurrence of y does not affect the quiescence recorded in $\langle\delta\rangle$, and so it is still considered in the composition.

Example 9: The composition of $\langle\delta\rangle$ and $\langle out1, \delta\rangle$, which belong to the processes $P4$ and $P5$ in Example 7 is $\langle out1, \delta\rangle$. □

If the trace composed with $\langle\delta\rangle$ records quiescence before y, the resulting traces start with δ because the composed traces record stability. The traces after δ are those in $\langle\delta\rangle \,[\![X]\!]_{ST}\, \langle y\rangle \frown \sigma$.

The composition of a trace $\langle\delta, x\rangle \frown \sigma_1$ that offers initially an event in X in a stable state with another trace $\langle y\rangle \frown \sigma_2$ starting with an event not in X yields traces that start with $\langle y\rangle$ and proceed with those in $\langle\delta, x\rangle \frown \sigma_1 \,[\![\, X \,]\!]_{ST}\, \sigma_2$. Since x has not occurred, the quiescence remains. The composition of $\langle\delta, y\rangle \frown \sigma_1$, which offers y in a stable state, with $\langle x\rangle \frown \sigma_2$ gives traces that do not record quiescence, since δ is just in one of the composed traces and is discarded; the composition is just $\langle y\rangle \frown \sigma_1 \,[\![X]\!]_{ST}\, \langle x\rangle \frown \sigma_2$. The composition of traces $\langle\delta, x\rangle \frown \sigma_1$

and $\langle \delta, y \rangle \frown \sigma_2$ whose first events are offered in stable states, but one is in X and the other is not contains the traces that start with $\langle \delta, y \rangle$: a δ is recorded because both traces record stability, and y follows because it is not in X. The traces following y are those in $\langle \delta, x \rangle \frown \sigma_1 \, [\![X]\!]_{ST} \, \sigma_2$. The quiescence before x is maintained because x is offered in the same (quiescent) state, although, after y we can have another (non stable) state.

Example 10: The composition of $\langle out1, \delta \rangle$ and $\langle \delta, in \rangle$ from $P4$ and $P5$ is empty, because we get traces starting with $out1$ followed by those in $\langle \delta, in \rangle$ and $\langle \delta \rangle$, but since in requires synchronisation, we have a deadlock: no traces. □

The composition of traces $\langle \delta, x_1 \rangle \frown \sigma_1$ and $\langle \delta, x_2 \rangle \frown \sigma_2$ that initially offer events in X in a stable state gives traces that record stability (δ) in addition to synchronisation, if possible. If we have different events in X, no traces are yielded.

Example 11:

$$\begin{aligned}
&\langle out1, \delta, in \rangle \, [\![\{in\}]\!]_{ST} \, \langle \delta, in \rangle \\
&= \{\sigma : \langle \delta, in \rangle \, [\![\{in\}]\!]_{ST} \, \langle \delta, in \rangle \bullet \langle out1 \rangle \frown \sigma\} \\
&= \{\sigma : \langle \delta, in \rangle \bullet \langle out1 \rangle \frown \sigma\} \\
&= \{\langle out1, \delta, in \rangle\}
\end{aligned}$$

□

A trace that records termination is of the form $\sigma_1 \frown \langle \checkmark \rangle \frown \sigma_2$, where σ_2 is either $\langle\rangle$ or $\langle \delta \rangle$. A parallel composition terminates only if both parallel processes terminate, so we can think of $\checkmark$ as an implicit event of every synchronisation set. Accordingly, the composition of a trace that records termination ($\sigma_1 \frown \langle \checkmark \rangle \frown \sigma_2$) with another that does not (σ_3), yields no traces. On the other hand, if both traces to be composed record termination, $\sigma_1 \frown \langle \checkmark \rangle \frown \sigma_2 \, [\![X]\!]_{ST} \, \sigma_3 \frown \langle \checkmark \rangle \frown \sigma_4$, the composition can record termination. Such a composition yields the traces in the form $\sigma_5 \frown \langle \checkmark \rangle \frown \sigma_6$, where σ_5 is a trace yielded by the composition of the traces before $\checkmark$, and σ_6 is a trace yielded by the composition of the traces after the $\checkmark$.

Example 12: The first state of $SKIP \, [\![X]\!] \, out \rightarrow SKIP$ is not stable, since this process offers the output out. After communicating out, it behaves as $SKIP \, [\![X]\!] \, SKIP$, which is equivalent to $SKIP$. Accordingly, we have

$$\begin{aligned}
&traces[\![SKIP \, [\![X]\!] \, out \rightarrow SKIP]\!] = \\
&\quad \{\langle\rangle, \langle out \rangle, \langle out, \checkmark \rangle, \langle out, \checkmark, \delta \rangle\}
\end{aligned}$$

This set is obtained by combining the traces of $SKIP$ with those of $out \rightarrow SKIP$. For instance, $\langle \checkmark \rangle \, [\![X]\!]_{ST} \, \langle out, \checkmark \rangle$ is $\{\langle out, \checkmark \rangle\}$. □

Calculation of the definitions of external choice and parallelism, and of other CSP operators, is work in progress. In the next section, we discuss the possibility of mechanising verification based on suspension traces.

V. Conformance based on suspension traces

For input-enabled specifications, ioco and suspension-traces inclusion are equivalent. In general, however, the latter is strictly stronger because it restricts the inputs that can be accepted by the implementation based on the specification. Moreover, ioco is defined only for input-enabled implementations. We define below cspioco, our account of ioco in the setting of CSP with a suspension-traces semantics. Our definition is very similar to the classical definition of ioco. Here, however, we can give a precise definition of the output events of a process after a trace.

Definition 2: For processes P and Q, with P input enabled.

$$P \, \text{cspioco} \, Q \mathrel{\widehat{=}} \forall \sigma : traces[\![Q]\!] \bullet out(P, \sigma) \subseteq out(Q, \sigma)$$

where

$$out(R, \sigma) = \{a : \mathcal{O}_\delta \mid \sigma \frown \langle a \rangle \in traces[\![R]\!] \}$$

For a process R and a suspension trace σ, the set $out(R, \sigma)$ contains the output events (including δ) of R, after the trace σ. As with ioco, the relation cspioco establishes that any output event observed in an implementation P, after any suspension trace σ of Q is also observed in the specification Q.

Example 13: As an example, we consider the processes S, I_1 and I_2 below, with input alphabet $\{coin\}$ and output alphabet $\{coffee, choc\}$, where S is a sample specification, and I_1 and I_2 are (input enabled) candidate implementations.

$$\begin{aligned}
S &= coin \rightarrow coffee \rightarrow S \\
I_1 &= coin \rightarrow \quad (coffee \rightarrow \quad I_1 \\
&\qquad\qquad \Box \; coin \rightarrow \quad (choc \rightarrow I_1 \; \Box \; I_1)) \\
I_2 &= coin \rightarrow \quad (coffee \rightarrow I_2 \\
&\qquad\qquad \Box \; choc \rightarrow I_2 \\
&\qquad\qquad \Box \; I_2)
\end{aligned}$$

We note that $I_1 \, \text{cspioco} \, S$ holds, but $I_2 \, \text{cspioco} \, S$ does not. Despite the fact that I_1 performs an output ($choc$) that is not specified in S, this output happens after traces that are not in $traces[\![S]\!]$, such as, $\langle coin, coin \rangle$. However, because $out(I_2, \langle coin \rangle) = \{coffee, choc\}$ is not a subset of $out(S, \langle coin \rangle) = \{coffee\}$, the trace $\langle coin \rangle$ is a counter-example for $I_2 \, \text{cspioco} \, S$. □

Although the suspension traces model is not implemented in any refinement checker for CSP, we can automate verification according to Definition 2 using a traces-refinement checker. The following theorem captures our proposed strategy.

Theorem 1 (Verification of cspioco*):*

$$P \, \text{cspioco} \, Q \Leftrightarrow Q_\delta \sqsubseteq_T (Q_\delta \; \triangle \; ANY(\mathcal{O}_\delta, STOP)) \, [\![\Sigma_\delta]\!] \, P_\delta$$

For any process P, we define a corresponding process P_δ that outputs δ in all quiescent states of P.

Definition 3: $P_\delta \mathrel{\widehat{=}} \text{prioritise}(P \, ||| \, RUN(\{\delta\}), \langle \mathcal{O}, \{\delta\} \rangle)$

The general form of the prioritise operator is $\text{prioritise}(P, \langle X_1, \ldots, X_n \rangle)$. Its behaviour is similar to that of P, but it prevents any event in X_i (for $i > 1$) from taking place when τ (that is, an internal event), $\checkmark$ (that is, termination) or an event in some X_j, with $j < i$, is possible.

We provide now a sketch for the proof of the above theorem. As shown in [7], if all quiescences are identified in the traces as the special output δ, then cspioco can be verified as a traces-

refinement assertion. P cspioco Q holds if, and only if,

$$Q \sqsubseteq_T (Q \triangle ANY(\mathcal{O}_\delta, STOP)) \llbracket \Sigma_\delta \rrbracket P \quad (1)$$

where $ANY(X, R) = \Box\ a : X \bullet a \rightarrow R$, and $\Sigma_\delta = \Sigma \cup \{\delta\}$.

We cannot compare P to Q directly since, as shown in Example 13, P can have traces that are not traces of Q: it can accept extra inputs and offer any output after a trace not in Q. We, therefore, consider whether Q is traces refined by the parallelism $(Q \triangle ANY(\mathcal{O}_\delta, STOP)) \llbracket \Sigma_\delta \rrbracket P$, which masks the traces of P that are not relevant for the verification of cspioco.

The parallelism blocks inputs of Q that are not accepted by P. These are allowed by cspioco and do not need to be checked. The interruption with the process $ANY(\mathcal{O}_\delta, STOP)$, on the other hand, prevents the parallelism from refusing outputs that P can perform but Q cannot; $ANY(\mathcal{O}_\delta, STOP)$ allows these outputs to be communicated to P. If P can perform such outputs, then they appear in the traces of the parallelism, which falsifies the refinement, as required.

Example 14: For illustration, we consider the verification of I_1 cspioco S using the above assertion. The processes I_1 and S are those in Example 13. I_1 can engage in two *coin* events in a row, which is not possible for S. The parallel composition $(S \triangle ANY(\mathcal{O}_\delta, STOP)) \llbracket \Sigma_\delta \rrbracket I_1$ cannot do the same, as only I_1 is ready to accept the second *coin* event. Because refused events are not recorded in the traces model, they are not in the traces of the parallelism. New inputs of the implementation are, therefore, allowed by the above refinement.

In the case of I_2 cspioco S, we have a forbidden output. When I_2 engages in the output *choc*, this is recorded in the trace of the parallelism because, although S does not offer this event, $ANY(\mathcal{O}_\delta, STOP)$ takes over and engages in it. This event, however, is not part of any trace of S, as it is never offered by S. Therefore, the refinement does not hold. □

Although the refinement assertion (1) captures cspioco conformance, it does not show how quiescent states of P and Q are effectively signalled using the event δ. To take advantage of (1) in a mechanisation of cspioco conformance verification, we use a notion of priority for CSP processes in [3]. P_δ defined above behaves like P whenever P can engage in an event in $\mathcal{O}$ (that is, an output event). When no outputs are available, P_δ behaves like $RUN(\{\delta\})$, which models a δ-loop. This additional behaviour happens only in the absence of output because of the order of the sets $\langle \mathcal{O}, \{\delta\} \rangle$ in the prioritise operator, which gives priority to output events over δ. Input events are incomparable to $\mathcal{O}$ and to δ.

The prioritise operator is implemented in FDR3, which can then be used to check for mechanical verification of cspioco and suspension-trace inclusion.

VI. Conclusion

We have defined a suspension traces model for CSP, including its healthiness conditions and denotations for some CSP operators. Our approach is to calculate the suspension traces for each operator from an input-output refusal traces model [12]. Our goal is to introduce the foundations for reasoning about suspension traces in a process algebra. It is suitable not only to explore approaches to test-case generation, but also compositional conformance verification, due to the rich repertoire of operators available.

We have also introduced the cspioco relation between CSP processes, defined in terms of suspension traces. This relation is weaker than suspension-traces inclusion, but the two relations coincide if specifications are input enabled. Additionally, we have shown that cspioco verification can be mechanised as a refinement assertion in the standard traces model, provided a notion of priority is used to identify quiescence in the specification and implementation models.

Despite the extensive adoption of suspension traces in testing theories based on ioco [6], [16], [18], as far as we are aware, our denotational definition is a novel contribution. Suspension traces have been characterised previously only in the context of operational models, particularly IOLTS.

Previous work [7], [19] explored test generation and compositional verification in the context of CSP, distinguishing input and output events. In those works, a relation cspio was defined and used for test-case generation via counterexamples of refinement in the traces model. However, no formal account of suspension traces was provided and, as a consequence, no proper treatment of quiescence was given. The work here is a formal basis to justify the syntactic encodings in [7], [19].

Perhaps the main benefit of a process algebraic characterisation of suspension traces is as a suitable framework to study compositionality. We plan to explore compositional conformance verification; particularly, for an implementation model I and partial specifications $S_1, \ldots, S_n$, we intend to uncover the conditions to ensure that I cspioco $S_1, \ldots, I$ cspioco $S_n \Rightarrow I$ cspioco $S_1 \parallel \ldots \parallel S_n$. We aim to do the same for all other CSP operators, and relax restrictions on input enabledness of the specification, present in current results on compositionality [19], [20].

Acknowledgment

We thank Bill Roscoe for useful discussions. The work was carried out with the support of EPSRC hiJaC project, the CNPq (Brazil), INES, and the grants FACEPE 573964/2008-4, APQ-1037-1.03/08, CNPq 573964/2008-4, 476821/2011-8 and 249710/2013-7.

References

[1] T. Gibson-Robinson, P. Armstrong, A. Boulgakov, and A. W. Roscoe, "FDR3 - A Modern Refinement Checker for CSP," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2014, pp. 187–201.

[2] J. Sun, Y. Liu, and J. S. Dong, "Model Checking CSP Revisited: Introducing a Process Analysis Toolkit," in *Proceedings of 3rd ISoLA*, ser. CCIS, vol. 17. Springer, 2008, pp. 307–322.

[3] A. W. Roscoe, *Understanding Concurrent Systems*, ser. Texts in Computer Science. Springer, 2011.

[4] A. L. C. Cavalcanti and M.-C. Gaudel, "Testing for Refinement in CSP," in *9th International Conference on Formal Engineering Methods*, ser. Lecture Notes in Computer Science, vol. 4789. Springer-Verlag, 2007, pp. 151–170.

[5] A. L. C. Cavalcanti and R. M. Hierons, "Testing with Inputs and Outputs in CSP," in *Fundamental Approaches to Software Engineering*, ser. Lecture Notes in Computer Science, vol. 7793, 2013, pp. 359–374.

[6] J. Tretmans, "Test Generation with Inputs, Outputs and Repetitive Quiescence," *Software—Concepts and Tools*, vol. 17, no. 3, pp. 103–120, 1996.

[7] S. Nogueira, A. C. A. Sampaio, and A. C. Mota, "Test generation from state based use case models," *Formal Aspects of Computing*, vol. 26, no. 3, pp. 441–490, 2014.

[8] F. Zeyda and A. L. C. Cavalcanti, "Mechanised Translation of Control Law Diagrams into *Circus*," in *Integrated Formal Methods*, ser. Lecture Notes in Computer Science, M. Leuschel and H. Wehrheim, Eds., vol. 5423. Springer-Verlag, 2009, pp. 151–166.

[9] A. L. C. Cavalcanti, F. Zeyda, A. Wellings, J. C. P. Woodcock, and K. Wei, "Safety-critical Java programs from *Circus* models," *Real-Time Systems*, vol. 49, no. 5, pp. 614–667, 2013.

[10] L. Lima, A. Miyazawa, A. L. C. Cavalcanti, M. Cornélio, J. Iyoda, A. C. A. Sampaio, R. Hains, A. Larkham, and V. Lewis, "An integrated semantics for reasoning about SysML design models using refinement," *Software & Systems Modeling*, pp. 1 – 28, 2015.

[11] G. Carvalho, F. A. Barros, A. Carvalho, A. L. C. Cavalcanti, A. Mota, and A. C. A. Sampaio, "NAT2TEST tool: From natural language requirements to test cases based on CSP," in *13th International Conference Software Engineering and Formal Methods*, ser. Lecture Notes in Computer Science, R. Calinescu and B. Rumpe, Eds., vol. 9276. Springer, 2015, pp. 283 – 290.

[12] A. L. C. Cavalcanti, R. M. Hierons, and S. Nogueira, "Input and outputs in CSP: a model and a testing theory," University of York, https://www.cs.york.ac.uk/, Tech. Rep., 2015.

[13] R. J. van Glabbeek, "The Linear Time - Branching Time Spectrum II," in *4th International Conference on Concurrency Theory*, ser. LNCS, E. Best, Ed., vol. 715. Springer, 1993, pp. 66–81.

[14] A. Mukarram, "A Refusal Testing Model for CSP," Ph.D. dissertation, Department of Computer Science, University of Oxford, 1993.

[15] I. Phillips, "Refusal testing," *Theoretical Computer Science*, vol. 50, no. 3, pp. 241–284, 1987.

[16] J. Tretmans, in *Formal Methods and Testing*, R. M. Hierons, J. P. Bowen, and M. Harman, Eds. Springer-Verlag, 2008, ch. Model Based Testing with Labelled Transition Systems, pp. 1–38.

[17] ——, "Test Generation with Inputs, Outputs, and Quiescence," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, vol. 1055. Springer-Verlag, 1996, pp. 127–146.

[18] C. Jard and T. Jéron, "TGV: theory, principles and algorithms, a tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems," *Software Tools for Technology Transfer*, vol. 6, 2004.

[19] A. C. A. Sampaio, S. Nogueira, A. Mota, and Y. Isobe, "Sound and mechanised compositional verification of input-output conformance," *STVR*, vol. 24, no. 4, pp. 289–319, 2014.

[20] M. van der Bijl, A. Rensink, and J. Tretmans, "Compositional Testing with ioco," in *3rd International Workshop on Formal Approaches to Testing of Software*, 2003, pp. 86–100.