



# A hierarchy of failures-based models: theory and application

Christie Bolton\*, Gavin Lowe

*Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK*

---

## Abstract

Consistency between a process and its specification expressed in CSP is typically presented as a refinement check. Within the traces model consistency is measured by examining only the traces of the systems, whilst in the finer stable failures model the possibility of subsequently refusing a combination of events is also taken into consideration.

In this paper, we begin by motivating the need for alternative measures of consistency. We then identify the *failures class*—a class of semantic models for describing concurrent systems in which each model is associated with a predicate that determines how much availability information is recorded. We show how refinement within members of this class corresponds to confirmation of non-standard measures of consistency, and identify application areas for these measures of consistency. We show how refinement in each model can be automatically tested.

We also carry out a theoretical examination of the failures class. We prove that the class forms a complete lattice, and investigate the positions of particular models within that lattice. We also identify the maximal subset of the language over which each model is compositional.

© 2004 Elsevier B.V. All rights reserved.

*Keywords:* Process algebra; Semantic models; CSP; Stable failures; Automatic verification

---

## 1. Introduction

Refinement is a standard technique that is used to ascertain whether or not a model of a system is in some way consistent with, or satisfies a property captured by, its specification.

---

\* Corresponding author.

*E-mail addresses:* [christie@comlab.ox.ac.uk](mailto:christie@comlab.ox.ac.uk), [christie@dcs.warwick.ac.uk](mailto:christie@dcs.warwick.ac.uk) (C. Bolton), [gavinl@comlab.ox.ac.uk](mailto:gavinl@comlab.ox.ac.uk) (G. Lowe).

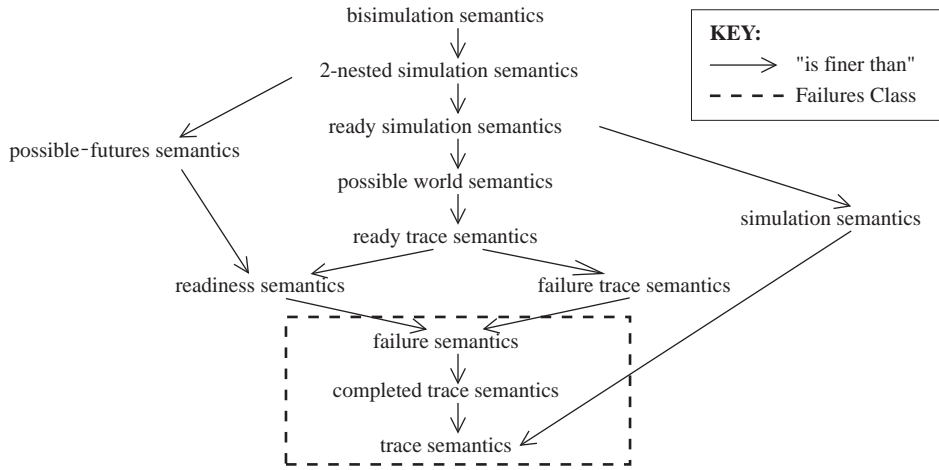


Fig. 1. Linear time-branching time spectrum of van Glabbeek [24], illustrating the positioning of the failures class of semantic models.

The measure of consistency depends on the language used and often on the choice of semantic model.

A variety of languages and semantic models have been proposed for describing and reasoning about the properties of concurrent systems, from process algebras such as CSP [11,20] and CCS [12,13] through modal logics [16,5,9] to Petri nets [15]. No one model is better than all the others: the choice and suitability of any given model depends on the requirements of the user.

We may impose an ordering on these models in terms of the number of identifications each makes: one model is *coarser* than another if whenever the second identifies two processes then so too does the first; in this case, we say that the second model is *finer* than the first. In [24], van Glabbeek considers various semantic equivalences for modelling finitely branching, concrete, sequential, non-deterministic processes. He presents them in a language-independent style, reducing them to eleven distinct models as illustrated in his linear time-branching time spectrum: see Fig. 1.

In the process algebra CSP a typical refinement check is of the form  $SPEC \sqsubseteq P$  for process  $P$  and specification  $SPEC$ . Non-trivial problems are generally verified using the model-checker FDR [19,10]. FDR checks that all behaviours of process  $P$  are possible behaviours of specification  $SPEC$ . The class of behaviours taken into consideration is determined by the choice of semantic model: the traces model [20] simply records all possible *traces*, or sequences of interaction, whereas the stable failures model [20] records also the potential to *refuse* sets of events after each such trace.

In Example 1 we motivate the need for alternative measures of consistency between a process and its specification, and hence the need for additional semantic models: we present a scenario in which the full strength of the stable failures model is too prescriptive but the traces model is not prescriptive enough.

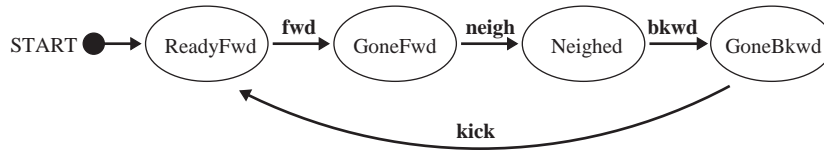


Fig. 2. A state diagram illustrating the implementation of a pantomime horse.

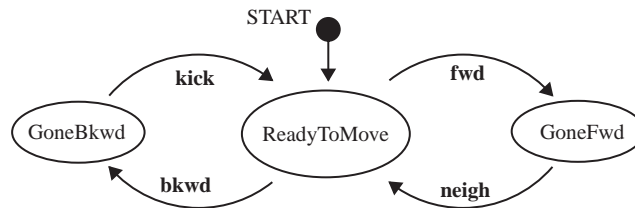


Fig. 3. A state diagram illustrating the specification of a pantomime horse.

### Example 1

The implementation of a pantomime horse<sup>1</sup> is modelled in the self-explanatory UML [23] state diagram in Fig. 2 below. The description says that the pantomime horse will repeatedly go forwards, neigh, go backwards and then kick. The original specification of the pantomime horse is modelled in Fig. 3. The specification can do any combination of going forwards then neighing and going backwards then kicking. Our question is “Is the implementation consistent with the specification?”

If we are working in the context of the traces model then the answer is “Yes”: every possible sequence of interaction of the implementation is a possible sequence of interaction of the specification. However if we consider full failures information then the answer is “No”: the implementation could initially refuse to go backwards whilst the specification could not.

But perhaps neither of these is the correct measure. It might be that in this instance our measure of consistency should be “Can the implementation refuse to move only when the specification can also refuse to move?” In other words, can the implementation refuse both *fwd* and *bkwd* only when the specification could refuse both these events?<sup>2</sup> If that is our measure then the answer is “Yes”: the implementation is consistent with the specification.

In this paper we put the bottom section of van Glabbeek’s spectrum under the microscope for further scrutiny. We identify the *failures class*, a particular class of semantic models of concurrent systems, lying between the traces and failures models. Each such model records possible sequences of interaction, along with particular *refusal sets*, sets which contain information about the subsequent availability of operations. They do not explicitly model *divergence*, the possible occurrence of an infinite sequence of hidden or

<sup>1</sup> The idea of modelling a pantomime horse was taken from Roscoe’s book [20].

<sup>2</sup> This is verifiable using the Actual Refusals model as introduced in Section 3.5.

internal actions. We use the process algebra Communicating Sequential Processes (CSP) [11,20] to present a generalised description of elements within this class.

Each semantic model from the failures class is associated with a predicate on refusal sets; this predicate determines how much availability information is recorded. We use our generalised model to:

- explore the relative strengths of models of this class;
- identify models—both established and new—within this class, and demonstrate application areas for their associated measures of consistency;
- identify the maximal subset of the language for which each model is compositional;
- consider for which such models is automatic refinement-checking possible.

The paper begins with a brief introduction to the syntax of CSP, introducing the standard operators along with processes that we will refer to throughout the rest of the paper; we follow this with a discussion of the stable failures model [20].

In Section 3 we formally identify the class of failures models central to the paper. Each such model is generated by a predicate that determines those refusal sets to be recorded; we identify the predicates associated with four established models—the stable failures model [20], the traces model [20], the singleton failures model [1], and the completed trace model [24]—and six non-established models. For each model under consideration we identify problem domains in which the associated measure of consistency would be useful and appropriate. Finally, we show that the failures class of semantic models forms a complete lattice, and in particular that the stable failures model is the top element and the traces model the bottom element.

In Section 4 we explore which models are compositional with respect to which subsets of the language, and apply these general results to the particular models already identified. In particular we prove that the traces model and the stable failures model are the only members of the class that are compositional over the entire language.

In Section 5 we consider the problem of automatic refinement checking, using existing tools. We demonstrate simple techniques that can be used within all the specific models considered in this paper, and also more complicated techniques that can be used for arbitrary models within the class.

We conclude in Section 6 with a discussion of related work. Throughout the paper we include in the main body of the text those proofs that shed light on the results they are establishing. The remaining proofs are left to the appendix.

## 2. Overview of CSP

In this section we give a brief overview of the CSP syntax that we will be using, and of the stable failures model for CSP [20].

### 2.1. Syntax

In CSP a *process* is a pattern of communication that describes the behaviour of a system. Examples of systems that might be modelled in this language are individual machines, networks and protocols. Moreover simple components may be combined to

create a composite process. Whatever the system, the behaviour is described in terms of *events* or synchronous atomic communications, marking points in the evolution of the system.

Channels carry sets of events; for example,  $c.5$  is an event of channel  $c$ . The notation  $\llbracket c \rrbracket$  represents the set of events corresponding to channel  $c$ .

The simplest process is *Stop*, the deadlocked process that will not perform any events and marks the end of a pattern of communication. The process *div* represents a divergent process, which performs unboundedly many internal events.

For any event  $a$  and process  $P$ , the process  $a \rightarrow P$  is willing to communicate event  $a$  and, if that event occurs, will subsequently behave as  $P$ . If  $A$  is a set of events, then  $?a : A \rightarrow P_a$  represents the process that is willing to communicate any of the events from  $A$ , and if event  $a$  is performed subsequently behaves as  $P_a$ .

CSP has two choice operators. The process  $P \square Q$  represents an *external* choice: the environment is given the choice between the initial events of  $P$  and  $Q$ . By contrast,  $P \sqcap Q$  represents an *internal* (or non-deterministic) choice between processes  $P$  and  $Q$ . The process  $\bigsqcap i : I | p(i) \bullet P_i$  represents an indexed internal choice between the processes  $P_i$  where  $i$  ranges over those members of  $I$  such that  $p(i)$  holds. The process  $P \triangle Q$  represents a process that initially acts like  $P$ , but at any point,  $P$  can be interrupted and control passed to  $Q$ .

Given processes  $P$  and  $Q$  and sets of events  $A$  and  $B$  (their respective interfaces), the process  $P_A \parallel B Q$  denotes the *parallel* combination of  $P$  and  $Q$ . In such a parallel combination, a process can perform only those events that are in its interface, and its cooperation is required if such an event is to occur; hence the processes synchronise on events in the intersection of their interfaces. By contrast,  $P ||| Q$  represents an interleaving of  $P$  and  $Q$ , i.e. a parallel composition with no synchronisation.

If  $P$  is a process and  $A$  a set of events, then the process  $P \setminus A$  behaves as  $P$  except that events from  $A$  are hidden (or made internal) so cannot be observed and do not require the cooperation of any other process.  $\mu X \bullet P$  represents a recursive process, where occurrences of  $X$  within  $P$  represent recursive calls; for example,  $\mu X \bullet a \rightarrow X$  represents a process that can perform an unbounded number of  $as$ . Such recursive processes are more normally defined equationally, for example  $P = a \rightarrow P$ . A collection of processes may also be defined by mutual recursion; see [20].

For later convenience we define also  $\text{Offer}(A) \triangleq ?a : A \rightarrow \text{div}$ , the process that offers the events from  $A$  and then diverges, and  $\rightsquigarrow$ , the non-deterministic interrupt operator defined as follows:  $P \rightsquigarrow Q \equiv P \sqcap (P \triangle Q)$ .

## 2.2. The stable failures model

The stable failures model represents each process  $P$  by a pair in which the first component is a set of *traces* and the second a set of *failures*. A trace is an element of the type  $\Sigma^*$ , where  $\Sigma$  is the set of all events, and corresponds to a possible sequence of interaction. A failure is an element of the type  $\Sigma^* \times \mathbb{P} \Sigma$ ; the first component is a trace and the second a *refusal set*, or set of events that might collectively be refused from a stable state (i.e. where no internal activity is possible) reached after the given trace.

### 2.3. Healthiness conditions

The semantics of a process  $P$  is given by the pair  $(traces(P), failures(P))$ . Furthermore, the functions  $traces$  and  $failures$  must satisfy the following healthiness conditions:

$$\langle \rangle \in traces(P), \quad (T1)$$

$$tr \hat{\ } tr' \in traces(P) \Rightarrow tr \in traces(P), \quad (T2)$$

$$(tr, X) \in failures(P) \Rightarrow tr \in traces(P), \quad (F1)$$

$$(tr, X \cup Y) \in failures(P) \Rightarrow (tr, X) \in failures(P), \quad (F2)$$

$$(tr, Y) \in failures(P) \wedge \forall x \in X \bullet tr \hat{\ } \langle x \rangle \notin traces(P) \Rightarrow (tr, X \cup Y) \in failures(P). \quad (F3)$$

The first condition (T1) states that the empty trace is a possible trace of every process and the second (T2) states that the set of traces of any process is prefix-closed. The third condition (F1) ensures consistency between failure and trace information. The fourth condition (F2) states that the set of refusal sets for every possible trace is subset closed. Finally, condition (F3) states that events that cannot be performed in a particular state may be added to a corresponding refusal set. Observe that the absence of the pair  $(tr, \emptyset)$  from the set  $failures(P)$  for some  $tr \in traces(P)$  indicates divergence.

### 2.4. Semantic equations

The functions  $traces$  and  $failures$  satisfy the following equations<sup>3</sup>:

$$traces(Stop) = \{\langle \rangle\},$$

$$failures(Stop) = \{(\langle \rangle, X) \mid X \in \mathbb{P} \Sigma\},$$

$$traces(div) = \{\langle \rangle\},$$

$$failures(div) = \{\},$$

$$traces(a \rightarrow P) = \{\langle \rangle\} \cup \{\langle a \rangle \hat{\ } tr \mid tr \in traces(P)\},$$

$$failures(a \rightarrow P) = \{(\langle \rangle, X) \mid X \in \mathbb{P} \Sigma \wedge a \notin X\} \cup \{(\langle a \rangle \hat{\ } tr, X) \mid (tr, X) \in failures(P)\},$$

$$traces(?a : A \rightarrow P_a) = \{\langle \rangle\} \cup \{\langle a \rangle \hat{\ } tr \mid a \in A \wedge tr \in traces(P_a)\},$$

$$failures(?a : A \rightarrow P_a) = \{(\langle \rangle, X) \mid X \in \mathbb{P} \Sigma \wedge A \cap X = \{\}\} \cup \{(\langle a \rangle \hat{\ } tr, X) \mid a \in A \wedge (tr, X) \in failures(P_a)\},$$

$$traces(P \square Q) = traces(P) \cup traces(Q),$$

$$failures(P \square Q) = \{(\langle \rangle, X) \in failures(P) \cap failures(Q)\} \cup \{(tr, X) \in failures(P) \cup failures(Q) \mid tr \neq \langle \rangle\},$$

<sup>3</sup> Note in particular that  $div$  has no stable failures, because it never reaches a stable state; and  $P \triangle Q$  has no failures corresponding to  $P$ , because in such states it can be interrupted at any point, so never stabilises.

$$\begin{aligned} \text{traces}(P \sqcap Q) &= \text{traces}(P) \cup \text{traces}(Q), \\ \text{failures}(P \sqcap Q) &= \text{failures}(P) \cup \text{failures}(Q), \end{aligned}$$

$$\begin{aligned} \text{traces}(\bigsqcap i : I \mid p(i) \bullet P_i) &= \bigcup \{ \text{traces}(P_i) \mid i \in I \wedge p(i) \}, \\ \text{failures}(\bigsqcap i : I \mid p(i) \bullet P_i) &= \bigcup \{ \text{failures}(P_i) \mid i \in I \wedge p(i) \}, \end{aligned}$$

$$\begin{aligned} \text{traces}(P \triangle Q) &= \{ tr \hat{\sim} tr' \mid tr \in \text{traces}(P) \wedge tr' \in \text{traces}(Q) \}, \\ \text{failures}(P \triangle Q) &= \{ (tr \hat{\sim} tr', X) \mid tr \in \text{traces}(P) \wedge \\ &\quad (tr', X) \in \text{failures}(Q) \}, \end{aligned}$$

$$\begin{aligned} \text{traces}(P \parallel_B Q) &= \{ tr \in (A \cup B)^* \mid tr \upharpoonright A \in \text{traces}(P) \wedge \\ &\quad tr \upharpoonright B \in \text{traces}(Q) \}, \\ \text{failures}(P \parallel_B Q) &= \{ (tr, X) \in (A \cup B)^* \times \mathbb{P} \Sigma \mid \\ &\quad \exists Y \in \mathbb{P} A; Z \in \mathbb{P} B; W \in \mathbb{P}(\Sigma - A - B) \bullet \\ &\quad (tr \upharpoonright A, Y) \in \text{failures}(P) \wedge \\ &\quad (tr \upharpoonright B, Z) \in \text{failures}(Q) \wedge \\ &\quad X = Y \cup Z \cup W \}, \end{aligned}$$

$$\begin{aligned} \text{traces}(P \parallel \parallel Q) &= \{ tr \mid \exists tr_P \in \text{traces}(P), tr_Q \in \text{traces}(Q) \bullet \\ &\quad tr \in tr_P \parallel \parallel tr_Q \}, \\ \text{failures}(P \parallel \parallel Q) &= \{ (tr, X) \mid \exists tr_P, tr_Q \bullet (tr_P, X) \in \text{failures}(P) \wedge \\ &\quad (tr_Q, X) \in \text{failures}(Q) \wedge \\ &\quad tr \in tr_P \parallel \parallel tr_Q \}, \end{aligned}$$

$$\begin{aligned} \text{traces}(P \setminus A) &= \{ tr \setminus A \mid tr \in \text{traces}(P) \}, \\ \text{failures}(P \setminus A) &= \{ (tr \setminus A, X) \mid (tr, A \cup X) \in \text{failures}(P) \}. \end{aligned}$$

In the equations for  $P \parallel \parallel Q$ , the notation  $tr_P \parallel \parallel tr_Q$  represents all ways of interleaving the traces  $tr_P$  and  $tr_Q$ ; see [20].

The semantics of recursive processes can be defined by

$$\begin{aligned} \text{traces}(\mu X \bullet F(X)) &= \bigcup \{ \text{traces}(F^n(\text{STOP})) \mid n \geq 0 \}, \\ \text{failures}(\mu X \bullet F(X)) &= \bigcup \{ \text{failures}(F^n(\text{STOP})) \mid n \geq 0 \}. \end{aligned}$$

These represent the least fixed points of the mapping on the semantic model corresponding to  $F$ . Semantics can be given to mutual recursions analogously [20].

For later convenience, we state also the equations for the process *Offer* and the non-deterministic interrupt operator  $\rightsquigarrow$ .

$$\begin{aligned} \text{traces}(\text{Offer}(A)) &= \{ \langle \rangle \} \cup \{ \langle a \rangle \mid a \in A \}, \\ \text{failures}(\text{Offer}(A)) &= \{ (\langle \rangle, X) \mid X \in \mathbb{P} \Sigma \wedge X \cap A = \emptyset \}, \\ \text{traces}(P \rightsquigarrow S) &= \text{traces}(P) \cup \\ &\quad \{ tr \hat{\sim} tr' \mid tr \in \text{traces}(P) \wedge tr' \in \text{traces}(S) \}, \end{aligned}$$

$$\begin{aligned} failures(P \rightsquigarrow S) = & failures(P) \cup \\ & \{(tr \frown tr', X) \mid \\ & tr \in traces(P) \wedge (tr', X) \in failures(S)\}. \end{aligned}$$

### 2.5. Refinement and equivalence

Equivalence and refinement in the stable failures model are defined as follows:

$$\begin{aligned} P \equiv_{\mathcal{F}} Q & \Leftrightarrow traces(P) = traces(Q) \wedge failures(P) = failures(Q), \\ P \sqsubseteq_{\mathcal{F}} Q & \Leftrightarrow traces(Q) \subseteq traces(P) \wedge failures(Q) \subseteq failures(P). \end{aligned}$$

The coarser traces model models a process only in terms of its traces. Equivalence and refinement in this model are defined by:

$$\begin{aligned} P \equiv_{\mathcal{T}} Q & \Leftrightarrow traces(P) = traces(Q), \\ P \sqsubseteq_{\mathcal{T}} Q & \Leftrightarrow traces(Q) \subseteq traces(P). \end{aligned}$$

## 3. The hierarchy of models

In this section we introduce the failures class of models; each model is associated with a predicate over failures sets that determines the amount of refusal information recorded. We then describe both established and non-established models within the failures class, identifying for each the associated predicate. For each model under consideration we indicate a potential application area; for a selected few we give also a more detailed example of a practical application. We present general results that determine the relative strengths of models in terms of their defining predicates, and conclude the section with a spectrum illustrating the relative strengths of the identified models.

All the models we consider in this paper represent processes by a pair comprising their traces and a *subset* of their failures. The subset of failures for any given model will be determined by a predicate  $p$  over refusal sets associated with that model; more precisely, a model associated with predicate  $p$  will include only those failures  $(tr, X)$  such that  $p(X)$  holds. We define:

$$failures_p(P) \hat{=} \{(tr, X) \in failures(P) \mid p(X)\},$$

to be those failures included in the model of predicate  $p$ . We then define the model  $\mathcal{M}_p$  to be the model that represents the process  $P$  by

$$\mathcal{M}_p[P] \hat{=} (traces(P), failures_p(P)).$$

We can define equivalence and refinement in the model  $\mathcal{M}_p$  by<sup>4</sup>:

$$\begin{aligned} P \equiv_p Q & \Leftrightarrow traces(P) = traces(Q) \wedge failures_p(P) = failures_p(Q), \\ P \sqsubseteq_p Q & \Leftrightarrow traces(Q) \subseteq traces(P) \wedge failures_p(Q) \subseteq failures_p(P). \end{aligned}$$

The following four established models are all instances of this class.

<sup>4</sup> Note that for succinctness and clarity, the equivalence and preorder operators will sometimes be indexed with the model and sometimes with the associated predicate.



### 3.1. Stable failures model

Roscoe's stable failures model ( $\mathcal{F}$ ) [20] records *full* trace and failure information. Two processes are equivalent within this model if they share the same traces and the same failures. Hence the predicate that generates the stable failures model is  $p(X) \hat{=} \text{true}$ : equivalently,  $\mathcal{F} = \mathcal{M}_{\lambda X} \bullet \text{true}$ .

The stable failures model may be used for reasoning about both safety and liveness properties for divergence-free processes. De Nicola [6] proves that for processes in which no internal events may occur (thereby precluding the use of the hiding operator) the stable failures semantic model is equivalent to his *testing equivalences* model [7].

### 3.2. Traces model

The traces model ( $\mathcal{T}$ ) [11,20] records *no* refusal information. Irrespective of their failures, two processes are equivalent within this model precisely when they share the same traces. Hence the predicate that generates the traces model is  $p(X) \hat{=} \text{false}$ : equivalently,  $\mathcal{T} = \mathcal{M}_{\lambda X} \bullet \text{false}$ .

The traces model may be used for reasoning about safety properties.

### 3.3. Singleton failures model

The singleton failures model ( $\mathcal{S}$ ) [1,2,24] records all trace information, and failures where the cardinality of the refusal set is at most one. Two processes are equivalent within this model if they share the same traces and if, after every such trace, they can refuse the same events *individually*. Hence the predicate that generates the singleton failures model is  $p(X) \hat{=} \#X \leq 1$ : equivalently,  $\mathcal{S} = \mathcal{M}_{\lambda X} \bullet \#X \leq 1$ .

This model was defined to coincide with the relational semantics of data types [8]: the refinement of data types is equivalent to the singleton failures refinement of their corresponding processes.

### 3.4. Completed trace model

As well as recording all trace information, the completed trace model ( $\mathcal{CT}$ ) [24] records all *completed traces*, that is traces after which no events can be performed. Two processes are equivalent within this model if firstly they share the same traces, and secondly if one can deadlock after a given trace then so can the other. Hence  $p(X) \hat{=} X = \Sigma$  is the predicate that generates the completed trace model: equivalently,  $\mathcal{CT} = \mathcal{M}_{\lambda X} \bullet X = \Sigma$ .

A means of automatically verifying whether one system can deadlock when another would not is necessary whenever we are designing a system that may reach a deadlocked state, but should do so only under certain conditions: for instance after reporting an error or successfully completing a task.

### Example

Consider an inference tool that can make a variety of deductive steps depending on its current knowledge. The specification might permit any possible deductive step from any

given state whilst, for efficiency reasons, the implementation might prioritise techniques: all possible deductive steps of one class must be completed before any deductive steps of another class may be performed. We require that every possible trace of the implementation must be a possible trace of the specification, but are not concerned that the implementation will have more refusals than the specification. Our concern is that whenever the implementation can make no more deductive steps and hence deadlocks, the specification might also deadlock. We need to verify that

$$\begin{aligned} & \{(tr, X) \in failures(InferenceToolImpl) \mid X = \Sigma\} \\ & \subseteq \{(tr, X) \in failures(InferenceToolSPEC) \mid X = \Sigma\}. \end{aligned}$$

This holds precisely when the following predicate is true.

$$InferenceToolSPEC \sqsubseteq_{CT} InferenceToolImpl.$$

We have identified four established semantic models that are members of the failures class. Obviously there are many more members of this class—as many as there are predicates on refusal sets, namely  $2^{2^{\#Z}}$ . Below we identify six predicates that yield potentially interesting or useful models.

### 3.5. Actual refusals model for $A$

The model generated by the predicate  $p(X) \hat{=} X = A$  records availability information only for a specified set  $A$ . Two particular instances of models within this subclass, models that merit their own names, are the completed trace model above (where  $A = \Sigma$ ) and the stable traces model below (where  $A = \{\}$ ). In the general case we will refer to models from this class as the *actual refusals* model for  $A$  ( $\mathcal{A}_A = \mathcal{M}_{\lambda X} \bullet_{X=A}$ ) for any given set  $A$ .

The actual refusals model will be useful whenever we have a system in which we are concerned about the availability of at least one of a given class of operations, although which operation is actually performed does not concern us.

#### Example

Suppose that a service provider has a collection of processors indexed by some set  $I$ . The acceptance and completion of a job by processor  $i$  are respectively modelled by events  $acceptJob.i$  and  $completeJob.i$ . An initial specification might be of the following form:

$$\begin{aligned} ServiceProviderSpec & \hat{=} |||_{i \in I} Processor(i), \\ Processor(i) & \hat{=} acceptJob.i \rightarrow completeJob.i \rightarrow Processor(i). \end{aligned}$$

To maximise efficiency, the service provider decides to incorporate a scheduler into the system when it comes to the actual implementation. The scheduler's role is to allocate jobs to processors. How it makes its decisions is not important for the purpose of this example. The final implementation is modelled as follows:

$$ServiceProviderImpl \hat{=} Scheduler \quad || \quad ServiceProviderSpec. \\ \{\{acceptJob\}\}$$

We are concerned not with which processor performs which job, but that the implementation cannot block the acceptance of a job where the specification would not. We need to verify that the following predicate holds:

$$\begin{aligned} & \{(tr, X) \in \text{ServiceProviderImpl} \mid X = \llbracket \text{acceptJob} \rrbracket\} \\ & \subseteq \{(tr, X) \in \text{ServiceProviderSpec} \mid X = \llbracket \text{acceptJob} \rrbracket\}. \end{aligned}$$

This is true precisely when

$$\text{ServiceProviderSpec} \sqsubseteq_{\mathcal{A}_{\llbracket \text{acceptJob} \rrbracket}} \text{ServiceProviderImpl}.$$

### 3.6. Stable traces model

The model generated by the predicate  $p(X) \hat{=} X = \{\}$ , a special case of the actual failures model which we will refer to as the *stable traces* model ( $\mathcal{ST} = \mathcal{M}_{\lambda X} \bullet_{X=\{\}}$ ), merits individual attention. The traces component records all possible traces whereas the failures component records only *stable* traces, traces after which the empty set can be refused. Hence this model, like Olderog and Hoare's divergence model [14] and Reed's untimed stability model [17], does not record the unavailability of events, but does distinguish between deadlock and divergence:

$$\mathcal{ST} \llbracket \text{Stop} \rrbracket = \{\{\langle \rangle\}, \{\langle \rangle, \{\}\}\}, \quad \mathcal{ST} \llbracket \text{div} \rrbracket = \{\{\langle \rangle\}, \{\}\}.$$

This model differs from the divergence and untimed stability models by the non-deterministic choice not being strict with respect to *div*.

### 3.7. Restricted refusals model for A

A model that is a little finer than the actual refusals model for A, is that generated by the predicate  $p(X) \hat{=} X \subseteq A$  for any A such that  $\{\} \subseteq A \subseteq \Sigma$ . We will refer to this as the *restricted refusals* model for A ( $\mathcal{R}_A = \mathcal{M}_{\lambda X} \bullet_{X \subseteq A}$ ). Such a model identifies two processes with the same traces if they agree upon refusals with events only from A. For  $A = \{\}$  and  $A = \Sigma$  this yields respectively the stable traces and the stable failures models. Such a model might be useful for situations in which we are concerned only about the availability of a particular subset of events.

### 3.8. Lower bounded refusals model for N

It is sometimes useful to impose a bound, either upper or lower, on the cardinality of refusal sets to be recorded. We introduce first the model that is generated by the predicate  $p(X) \hat{=} \#X \geq N$  for some integer N such that  $0 \leq N \leq \#\Sigma$ . We will refer to this as the *lower bounded refusals* model for N ( $\mathcal{L}_N = \mathcal{M}_{\lambda X} \bullet_{\#X \geq N}$ ).

Such a model identifies two processes with the same traces if they agree upon refusal sets of cardinality at least N. For  $N = 0$  and (assuming finiteness of  $\Sigma$ )  $N = \#\Sigma$ , this yields respectively the stable failures and the completed trace models.

Verifying that a set of at least  $N$  events can be refused by a process only when its specification could also refuse those events is of particular interest when we are considering support or safety systems, systems that under usual circumstances require a built-in margin for error.

### 3.9. Upper bounded refusals model for $N$

Another interesting model is that generated by the predicate  $p(X) \hat{=} \#X \leq N$  for some integer  $N$  such that  $0 \leq N \leq \#\Sigma$ . We will refer to this as the *upper bounded refusals* model for  $N$  ( $\mathcal{U}_N = \mathcal{M}_{\lambda X} \bullet \#X \leq N$ ). Such a model identifies two processes with the same traces if they agree upon refusal sets of cardinality at most  $N$ . For  $N = 0$ ,  $N = 1$  and (assuming finiteness of  $\Sigma$ )  $N = \#\Sigma$ , this yields respectively the stable traces, the singleton failures, and the stable failures models.

A potential application for the upper bounded refusals model is for analysing the behaviour of a distributed system of  $N$  processes provided no more than  $M < N$  processors had failed.

### 3.10. Bounded and restricted refusals model for $A$ and $N$

The last model we consider is essentially a hybrid of the lower bounded refusals and the restricted refusals models and is generated by the predicate  $p(X) \hat{=} X \subseteq A \wedge \#X \geq N$  for set of events  $A$  and integer  $N$ . We will refer to this as the *bounded and restricted refusals* model for  $A$  and  $N$  ( $\mathcal{BR}_{A,N} = \mathcal{M}_{\lambda X} \bullet X \subseteq A \wedge \#X \geq N$ ).

The bounded and restricted refusals model  $\mathcal{BR}_{A,N}$  identifies two processes with the same traces if they agree on all refusal sets that are both a subset of  $A$  and of cardinality greater than  $N$ .

A means of automatically verifying whether one system can refuse at least a certain number of events of a given class when another could not is of particular interest when we are modelling distributed systems involving the parallel combination or interleaving of an indexed collection of identical components.

#### Example

Suppose that a company is running the software for a large safety-critical distributed system comprising a dynamic network of  $R$  indexed nodes. The requirements imposed on the company state that (modulo an agreed latency period after any node crashes) the central monitor must always be able to communicate with at least  $k$  nodes. The ability of the central monitor to be able to communicate with node  $i \in R$  is modelled by the event *inContact.i*.

Suppose that the process *Network* below satisfies the necessary constraints:

$$\text{Network} \hat{=} \text{MonitorProcess} \quad || \quad \text{Nodes.} \\ \{\{inContact\}\}$$

The company now decides, for whatever reason, to update its nodes:

$$\text{NewNetwork} \hat{=} \text{MonitorProcess} \quad || \quad \text{UpdatedNodes,} \\ \{\{inContact\}\}$$

but then needs to verify that the new system can communicate with at least  $k$  of its processors at least as often as the old system could. They need to verify that whenever the new system can refuse  $N = R - K + 1$  or more *inContact* events the old system could also have refused these events. They need to verify that

$$\begin{aligned} \{(tr, X) \in \text{failures}(\text{NewNetwork}) \mid \#X \geq N \wedge X \subseteq \llbracket \text{contact} \rrbracket\} \subseteq \\ \{(tr, X) \in \text{failures}(\text{Network}) \mid \#X \geq N \wedge X \subseteq \llbracket \text{contact} \rrbracket\}. \end{aligned}$$

This could be expressed equivalently as follows:

$$\text{Network} \sqsubseteq_{\mathcal{BR}_{\llbracket \text{contact} \rrbracket, N}} \text{NewNetwork}.$$

### 3.11. Relative strengths

Recall that model  $\mathcal{M}$  is finer than  $\mathcal{M}'$  if  $\mathcal{M}$  distinguishes at least as many processes as  $\mathcal{M}'$ :

$$\forall P, Q \bullet P \equiv_{\mathcal{M}} Q \Rightarrow P \equiv_{\mathcal{M}'} Q.$$

This ordering on models corresponds to the ordering on predicates:

**Lemma 1.** *If predicate  $p$  is stronger than predicate  $q$  then semantic model  $\mathcal{M}_q$  is finer than semantic model  $\mathcal{M}_p$ .*

The following theorem follows immediately from this result.

**Theorem 2.** *The failures class of semantic models forms a complete lattice under the is-finer-than relation, which corresponds to implication of the corresponding predicates; the top element is the stable failures model ( $\mathcal{F} = \mathcal{M}_{\lambda X \bullet \text{true}}$ ), and the bottom element the traces model ( $\mathcal{T} = \mathcal{M}_{\lambda X \bullet \text{false}}$ ).*

**Theorem 3.** *If predicate  $q$  is not at least as strong as predicate  $p$ , i.e.  $\neg(\forall X \bullet q(X) \Rightarrow p(X))$ , then semantic model  $\mathcal{M}_q$  distinguishes processes identified by model  $\mathcal{M}_p$ .*

**Proof sketch.** Since  $\neg(\forall X \bullet q(X) \Rightarrow p(X))$  there must be some set of events  $X_{pq}$  such that  $q(X_{pq}) \wedge \neg p(X_{pq})$ . We identify processes  $P_{pq}$  and  $Q_{pq}$  such that  $\text{traces}(P_{pq}) = \text{traces}(Q_{pq})$  and  $\text{failures}(P_{pq})$  is the disjoint union of  $\text{failures}(Q_{pq})$  and  $\{(\langle \rangle, X_{pq})\}$ . Hence  $P_{pq} \equiv_p Q_{pq}$  but  $P_{pq} \not\equiv_q Q_{pq}$ . In the case where  $X_{pq}$  is non-empty this is true of

$$\begin{aligned} P_{pq} &\hat{=} \text{Offer}(\Sigma) \sqcap \text{Offer}(\Sigma - X_{pq}), \\ Q_{pq} &\hat{=} \text{Offer}(\Sigma) \sqcap (\bigsqcap Y : \mathbb{P} \Sigma \mid Y \subset X_{pq} \bullet \text{Offer}(\Sigma - Y)), \end{aligned}$$

where  $X_{pq} = \emptyset$ , it is true of  $P_{pq} \hat{=} \text{Offer}(\Sigma)$  and  $Q_{pq} \hat{=} \text{Offer}(\Sigma) \sqcap \text{div}$ .  $\square$

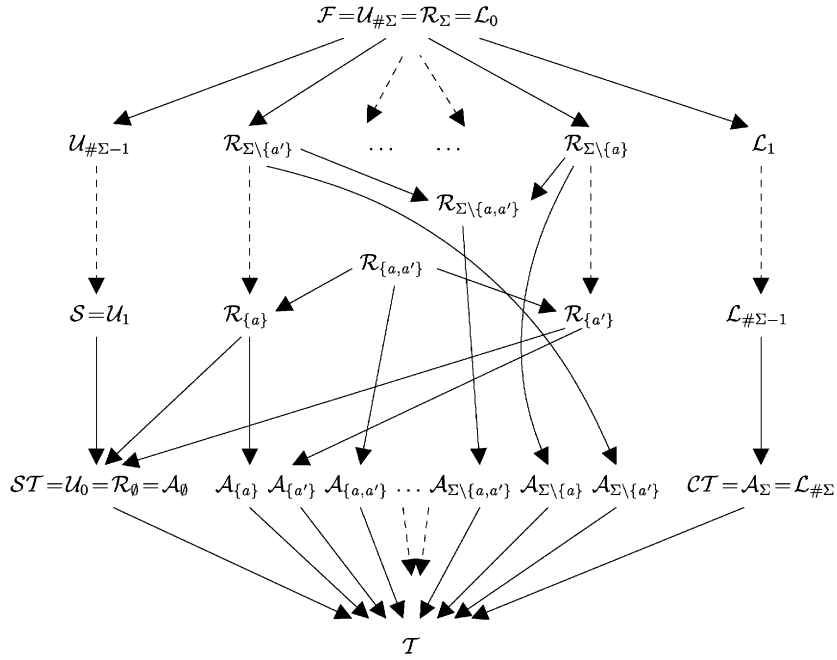


Fig. 4. Hierarchy of models (where  $a, a' \in \Sigma$  and  $a \neq a'$ ).

It follows directly from Theorems 2 and 3 that for processes with finite alphabet  $\Sigma$ , the relative strengths of the models identified in this section<sup>5</sup> are as illustrated in Fig. 4. For convenience, we summarise the models in Fig. 5.

Observe first that members of the actual refusals subclass of models,  $\{\mathcal{A}_A \mid A \in \mathbb{P} \Sigma\}$  including the stable failures model ( $\mathcal{ST} = \mathcal{A}_{\emptyset}$ ) and completed traces model ( $\mathcal{CT} = \mathcal{A}_{\Sigma}$ ), are independent of one another whereas members of the restricted refusals subclass of models,  $\{\mathcal{R}_A \mid A \in \mathbb{P} \Sigma\}$ , form a complete sub-lattice, with the stable failures model ( $\mathcal{F} = \mathcal{R}_{\Sigma}$ ) as top element, and the stable traces model ( $\mathcal{ST} = \mathcal{R}_{\emptyset}$ ) as bottom element. Observe also that for every set of events  $A$ , the restricted refusals model for  $A$  ( $\mathcal{R}_A$ ) is finer than the actual refusals model for  $A$  ( $\mathcal{A}_A$ ). The upper bounded refusals subclass of models form a chain with the stable failures model ( $\mathcal{F} = \mathcal{U}_{\# \Sigma}$ ) as the top element and the stable traces model ( $\mathcal{ST} = \mathcal{U}_0$ ), which is slightly coarser than the singleton failures model ( $\mathcal{S} = \mathcal{U}_1$ ), as the bottom element. Similarly, the lower bounded refusals subclass of models form a chain with the stable failures model ( $\mathcal{F} = \mathcal{L}_0$ ) as the top element and the completed trace model ( $\mathcal{CT} = \mathcal{L}_{\# \Sigma}$ ) as the bottom element.

<sup>5</sup> For clarity we have omitted the bounded and restricted refusals subclass of models from Fig. 4. However, for each set of events  $A$ , the subclass of models  $\{\mathcal{BR}_{A,N} \mid 0 \leq N \leq \#A\}$  form a chain with  $\mathcal{BR}_{A,0} = \mathcal{R}_A$  as the top element and  $\mathcal{BR}_{A,\#A} = \mathcal{A}_A$  as the bottom element; also, for  $N > \#A$ ,  $\mathcal{BR}_{A,N} = \mathcal{T}$ .

Model	Corresponding predicate	Description
$\mathcal{A}_A$	$\lambda X \bullet X = A$	actual refusals
$\mathcal{BR}_{A,N}$	$\lambda X \bullet X \subseteq A \wedge \#X \geq N$	bounded restricted refusals
$\mathcal{CT}$	$\lambda X \bullet X = \Sigma$	completed trace
$\mathcal{F}$	$\lambda X \bullet \text{true}$	stable failures
$\mathcal{L}_N$	$\lambda X \bullet \#X \geq N$	lower bounded refusals
$\mathcal{R}_A$	$\lambda X \bullet X \subseteq A$	restricted refusals
$\mathcal{S}$	$\lambda X \bullet \#X \leq 1$	singleton failures
$\mathcal{ST}$	$\lambda X \bullet X = \{\}$	stable traces
$\mathcal{T}$	$\lambda X \bullet \text{false}$	traces
$\mathcal{U}_N$	$\lambda X \bullet \#X \leq N$	upper bounded refusals

Fig. 5. Summary of models.

#### 4. Compositionality

A semantic model  $\mathcal{M}$  is *compositional* with respect to *unary* operator  $F$  if we may express  $\mathcal{M} \llbracket F(P) \rrbracket$  in terms of  $\mathcal{M} \llbracket P \rrbracket$ ; it is compositional with respect to *binary* operator  $\oplus$  if we may express  $\mathcal{M} \llbracket P \oplus Q \rrbracket$  in terms of  $\mathcal{M} \llbracket P \rrbracket$  and  $\mathcal{M} \llbracket Q \rrbracket$ . It is *compositional* (or is a *congruence*) if it is a compositional with respect to the entire language.

In this section we consider which models in our class are compositional with respect to which operators. Most established semantic models of CSP are compositional, and indeed the denotational semantics of a process is traditionally defined in a compositional manner. Several semantic equivalences for CCS [12,13] are not compositional; in such cases, it is common to consider the coarsest corresponding congruence. Compositionality is useful because it allows compositional reasoning. However, it is not essential, especially if the model has automatable verification techniques, as we will show, in Section 5, is the case with the failures class.

All the models associate the same traces with a given process, and the semantic equations for *traces* in Section 2.2 showed show that the traces of a composite process can always be expressed in terms of the traces of the components. Hence we need consider only failures below.

All the models within the failures class are compositional with respect to the operators  $\rightarrow$ ,  $\sqcap$ ,  $\sqcup$ ,  $\triangle$  and  $\parallel$ .

**Lemma 4.** *For every predicate  $p$ , the model  $\mathcal{M}_p$  is compositional on the subset of the language containing the operators  $\rightarrow$ ,  $\sqcap$ ,  $\sqcup$ ,  $\triangle$  and  $\parallel$ .*

**Proof.** For any predicate  $p$  the following all hold:

$$\begin{aligned}
 \text{failures}_p(a \rightarrow P) &= \{(\langle \rangle, X) \mid p(X) \wedge a \notin X\} \cup \\
 &\quad \{(\langle a \rangle \frown tr, X) \mid (tr, X) \in \text{failures}_p(P)\}, \\
 \text{failures}_p(?a : A \rightarrow P_a) &= \{(\langle \rangle, X) \mid p(X) \wedge A \cap X = \{\}\} \cup \\
 &\quad \{(\langle a \rangle \frown tr, X) \mid a \in A \wedge (tr, X) \in \text{failures}_p(P_a)\}, \\
 \text{failures}_p(P \sqcap Q) &= \text{failures}_p(P) \cup \text{failures}_p(Q),
 \end{aligned}$$

$$\begin{aligned}
failures_p(P \square Q) &= \{(\langle \rangle, X) \in failures_p(P) \cap failures_p(Q)\} \\
&\cup \\
&\quad \{(tr, X) \in failures_p(P) \cup failures_p(Q) \mid tr \neq \langle \rangle\}, \\
failures_p(P \triangle Q) &= \{(tr \hat{\ } tr', X) \mid tr \in traces(P) \wedge \\
&\quad (tr', X) \in failures_p(Q)\}, \\
failures_p(P \parallel Q) &= \{(tr, X) \mid \exists tr_P, tr_Q \bullet (tr_P, X) \in failures_p(P) \wedge \\
&\quad (tr_Q, X) \in failures_p(Q) \wedge \\
&\quad tr \in tr_P \parallel tr_Q\}.
\end{aligned}$$

Consider first the case of the external choice operator:

$$\begin{aligned}
&failures_p(P \square Q) \\
&= \quad \quad \quad [\text{def}^n \text{ of } failures_p \text{ and } failures] \\
&\quad \{(\langle \rangle, X) \in failures(P) \cap failures(Q) \mid p(X)\} \\
&\quad \cup \\
&\quad \{(tr, X) \in failures(P) \cup failures(Q) \mid tr \neq \langle \rangle \wedge p(X)\} \\
&= \quad \quad \quad [\text{set theory}] \\
&\quad \{(\langle \rangle, X) \in failures(P) \mid p(X)\} \cap \\
&\quad \{(\langle \rangle, X) \in failures(Q) \mid p(X)\} \\
&\quad \cup \\
&\quad \{(tr, X) \in failures(P) \mid tr \neq \langle \rangle \wedge p(X)\} \\
&\quad \cup \\
&\quad \{(tr, X) \in failures(Q) \mid tr \neq \langle \rangle \wedge p(X)\} \\
&= \quad \quad \quad [\text{def}^n \text{ of } failures_p] \\
&\quad \{(\langle \rangle, X) \in failures_p(P) \cap failures_p(Q)\} \\
&\quad \cup \\
&\quad \{(tr, X) \in failures_p(P) \cup failures_p(Q) \mid tr \neq \langle \rangle\}.
\end{aligned}$$

The proofs of the other results follow similar lines.  $\square$

However, as we will illustrate below, this result does not extend to subsets of the language containing either the parallel operator or the hiding operator.

#### 4.1. Hiding

In this section we show that the model  $\mathcal{M}_p$  is compositional with respect to hiding of set  $A$  if and only if  $p(X) \Rightarrow p(X \cup A)$  for all sets  $X$ . We begin by proving the “only if” result.

**Lemma 5.** *Semantic model  $\mathcal{M}_p$  is compositional with respect to hiding of set  $A$  only if  $p(X) \Rightarrow p(X \cup A)$  for all sets  $X$ .*

**Proof.** Suppose  $p(X)$  and  $\neg p(X \cup A)$ . We exhibit processes  $P$  and  $Q$  such that  $P \equiv_p Q$  but  $P \setminus A \not\equiv_p Q \setminus A$ . Let

$$P \hat{=} \bigcap Y : \mathbb{P} \Sigma \mid Y \subseteq X \cup A \bullet Offer(\Sigma - Y),$$



$$Q \triangleq \prod Y : \mathbb{P} \Sigma | Y \subseteq X \cup A \bullet \text{Offer}(\Sigma - Y).$$

Then

$$\begin{aligned} \text{traces}(P) &= \text{traces}(Q) = \{\langle \rangle\} \cup \{\langle a \rangle | a \in \Sigma\}, \\ \text{failures}(P) &= \{(\langle \rangle, Y) | Y \subseteq X \cup A\}, \\ \text{failures}(Q) &= \{(\langle \rangle, Y) | Y \subseteq X \cup A\}. \end{aligned}$$

Hence  $P \equiv_p Q$ : the only difference between  $P$  and  $Q$  is the failure  $(\langle \rangle, X \cup A)$  of  $P$ ; but  $X \cup A$  does not satisfy  $p$ , so this failure is not included in  $\text{failures}_p(P)$ .

However

$$(\langle \rangle, X) \in \text{failures}(P \setminus A) - \text{failures}(Q \setminus A),$$

because

$$(\langle \rangle, X \cup A) \in \text{failures}(P) - \text{failures}(Q),$$

so  $P \setminus A \not\equiv_p Q \setminus A$ , as required.  $\square$

We now prove the converse result.

**Lemma 6.** *If  $p(X) \Rightarrow p(X \cup A)$  for all sets  $X$ , then for all processes  $P$ , the set  $\text{failures}_p(P \setminus A)$  is expressible in terms of  $\text{failures}_p(P)$ .*

**Proof.** Suppose  $p(X) \Rightarrow p(X \cup A)$  for all sets  $X$ . We reason as follows:

$$\begin{aligned} &\text{failures}_p(P \setminus A) \\ &= \{(tr, X) \in \text{failures}(P \setminus A) \mid p(X)\} && [\text{def}^n \text{ of } \text{failures}_p] \\ &= && [\text{def}^n \text{ of } \text{failures}] \\ &\quad \{(tr \setminus A, X) \mid (tr, A \cup X) \in \text{failures}(P) \wedge p(X)\} \\ &= && [p(X) \Rightarrow p(A \cup X)] \\ &\quad \{(tr \setminus A, X) \mid (tr, A \cup X) \in \text{failures}(P) \wedge p(A \cup X) \wedge p(X)\} \\ &= && [\text{def}^n \text{ of } \text{failures}_p] \\ &\quad \{(tr \setminus A, X) \mid (tr, A \cup X) \in \text{failures}_p(P) \wedge p(X)\}. \quad \square \end{aligned}$$

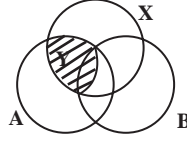
**Corollary 7.** *Model  $\mathcal{M}_p$  is compositional with respect to hiding of arbitrary sets precisely when predicate  $p$  is upwards closed.*

#### 4.2. Parallel composition

In this section we show that model  $\mathcal{M}_p$  is compositional with respect to the parallel composition  $-_A \parallel_B -$  if and only if

$$\forall X, Y : \mathbb{P} \Sigma | X \cap (A - B) \subseteq Y \subseteq X \cap A \bullet p(X) \Rightarrow p(Y), \quad (1)$$

$$\forall X, Z : \mathbb{P} \Sigma | X \cap (B - A) \subseteq Z \subseteq X \cap B \bullet p(X) \Rightarrow p(Z). \quad (2)$$

Fig. 6.  $X \cap (A - B) \subseteq Y \subseteq X \cap A$ .

The relationship  $X \cap (A - B) \subseteq Y \subseteq X \cap A$  is illustrated in Fig. 6. If  $X$  is a refusal of  $P_A \parallel_B Q$  then the corresponding refusals  $Y$  of  $P$  and  $Z$  of  $Q$  will satisfy  $X \cap (A - B) \subseteq Y \subseteq X \cap A$  and  $X \cap (B - A) \subseteq Z \subseteq X \cap B$ , respectively; conditions (1) and (2) say that if  $X$  satisfies  $p$  then so do  $Y$  and  $Z$ .

We begin by proving the “only if” result

**Lemma 8.** *Semantic model  $\mathcal{M}_p$  is compositional on a subset of the language containing the parallel operator  $-_A \parallel_B$  – only if conditions ‘(1) and (2) hold.*

**Proof.** By symmetry, it is enough to consider just the case where condition (1) does not hold. So suppose  $X \cap (A - B) \subseteq Y \subseteq X \cap A \wedge p(X) \wedge \neg p(Y)$ .

We construct processes  $P$ ,  $P'$  and  $Q$  such that  $P \equiv_p P'$  but  $P_A \parallel_B Q \not\equiv_p P'_A \parallel_B Q$  as follows:

$$\begin{aligned} P &\triangleq \square Z : \mathbb{P} \Sigma | Z \subseteq Y \bullet \text{Offer}(\Sigma - Z), \\ P' &\triangleq \square Z : \mathbb{P} \Sigma | Z \subset Y \bullet \text{Offer}(\Sigma - Z), \\ Q &\triangleq \text{Offer}(B \cap Y). \end{aligned}$$

Then

$$\begin{aligned} \text{traces}(P) &= \text{traces}(P') = \{ \langle \rangle \} \cup \{ \langle a \rangle | a \in \Sigma \}, \\ \text{failures}(P) &= \{ (\langle \rangle, Z) | Z \subseteq Y \}, \\ \text{failures}(P') &= \{ (\langle \rangle, Z) | Z \subset Y \}, \\ \text{failures}(Q) &= \{ (\langle \rangle, Z) | Z \cap (B \cap Y) = \{ \} \}. \end{aligned}$$

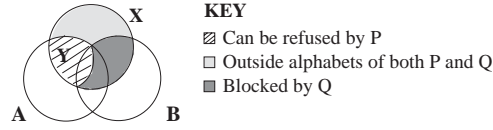
Hence  $P \equiv_p P'$ , because they differ only in the failure  $(\langle \rangle, Y)$  and  $Y$  does not satisfy  $p$ .

Observe that, as illustrated in Fig. 7, the process  $P_A \parallel_B Q$  can initially refuse the whole of set  $X$ : any element in  $X - (A \cup B)$  lies outside the alphabets of both processes; any element in  $X \cap (B - Y)$  will be blocked by  $Q$ ; and the remainder of  $X$ , i.e. the set  $Y$ , can be refused by  $P$ . However, since the process  $P'$  cannot refuse the whole of  $Y$ , we conclude that  $P'_A \parallel_B Q$  cannot initially refuse the whole of  $X$ . We see that

$$(\langle \rangle, X) \in \text{failures}(P_A \parallel_B Q) - \text{failures}(P'_A \parallel_B Q).$$

Hence, since  $p(X)$  is true, we conclude, as required, that  $P_A \parallel_B Q \not\equiv_p P'_A \parallel_B Q$ .  $\square$

We now prove the converse result.

Fig. 7. Process  $P_A \parallel_B Q$  can initially refuse the whole of  $X$ .

**Lemma 9.** *If conditions (1) and (2) hold, then for all processes  $P$  and  $Q$ , the set  $\text{failures}_p(P \parallel_B Q)$  is expressible in terms of the sets  $\text{failures}_p(P)$  and  $\text{failures}_p(Q)$ .*

**Proof.** We reason as follows:

$$\begin{aligned}
& \text{failures}_p(P \parallel_B Q) \\
&= \{(tr, X) \in \text{failures}(P \parallel_B Q) \mid p(X)\} && [\text{def}^n \text{ of } \text{failures}_p] \\
&= \{(tr, X) \in (A \cup B)^* \times \mathbb{P} \Sigma \mid p(X) \wedge \\
&\quad \exists Y \in \mathbb{P} A; Z \in \mathbb{P} B; W \in \mathbb{P}(\Sigma - A - B) \bullet \\
&\quad X = Y \cup Z \cup W \wedge \\
&\quad (tr \upharpoonright A, Y) \in \text{failures}(P) \wedge (tr \upharpoonright B, Z) \in \text{failures}(Q)\} \\
&= \{(tr, X) \in (A \cup B)^* \times \mathbb{P} \Sigma \mid p(X) \wedge \\
&\quad \exists Y \in \mathbb{P} A; Z \in \mathbb{P} B; W \in \mathbb{P}(\Sigma - A - B) \bullet \\
&\quad X = Y \cup Z \cup W \wedge \\
&\quad X \cap (A - B) \subseteq Y \subseteq X \cap A \wedge \\
&\quad X \cap (B - A) \subseteq Z \subseteq X \cap B \wedge \\
&\quad (tr \upharpoonright A, Y) \in \text{failures}(P) \wedge (tr \upharpoonright B, Z) \in \text{failures}(Q)\} && [\text{set theory}] \\
&= \{(tr, X) \in (A \cup B)^* \times \mathbb{P} \Sigma \mid p(X) \wedge \\
&\quad \exists Y \in \mathbb{P} A; Z \in \mathbb{P} B; W \in \mathbb{P}(\Sigma - A - B) \bullet \\
&\quad X = Y \cup Z \cup W \wedge p(Y) \wedge p(Z) \wedge \\
&\quad (tr \upharpoonright A, Y) \in \text{failures}(P) \wedge (tr \upharpoonright B, Z) \in \text{failures}(Q)\} && [\text{conditions (1) and (2)}] \\
&= \{(tr, X) \in (A \cup B)^* \times \mathbb{P} \Sigma \mid p(X) \wedge \\
&\quad \exists Y \in \mathbb{P} A; Z \in \mathbb{P} B; W \in \mathbb{P}(\Sigma - A - B) \bullet \\
&\quad X = Y \cup Z \cup W \wedge \\
&\quad (tr \upharpoonright A, Y) \in \text{failures}_p(P) \wedge (tr \upharpoonright B, Z) \in \text{failures}_p(Q)\}. \quad \square && [\text{def}^n \text{ of } \text{failures}_p]
\end{aligned}$$

**Corollary 10.** *Model  $\mathcal{M}_p$  is compositional with respect to parallel composition with arbitrary interface sets precisely when predicate  $p$  is downwards closed, i.e.:*

$$\forall X, Y : \mathbb{P} \Sigma \mid Y \subseteq X \bullet p(X) \Rightarrow p(Y). \quad (3)$$

**Proof.** We must show the above condition (3) is equivalent to the conjunctions of conditions (1) and (2) for all  $A$  and  $B$ .

Observe that if conditions (1) and (2) hold for all  $A$  and  $B$  then in particular condition (1) holds for the case when  $A = B = \Sigma$ . This gives condition (3). Conversely, assume condition (3) holds and suppose  $X \cap (A - B) \subseteq Y \subseteq X \cap A$ ; then  $Y \subseteq X$ , so  $p(X) \Rightarrow p(Y)$ . This yields conditions (1) and (2) for arbitrary  $A$  and  $B$ .  $\square$

#### 4.3. Summary

In Theorem 11 below we apply the results established in Lemma 4, and Corollaries 7 and 10 to identify for each subset of the language the constraints on predicate  $p$  that ensure model  $\mathcal{M}_p$  is compositional. Theorems 12 and 13, in which we consider specific models, follow directly from Theorem 11.

**Theorem 11.** *Model  $\mathcal{M}_p$  defined on a language with operators*

$$Ops \subseteq \{\rightarrow, \square, \sqcap, \triangle, \parallel, |||, \backslash\}$$

*is compositional precisely when the following two predicates hold:*

- $\parallel \in Ops \Rightarrow (\forall X, Y \in \mathbb{P} \Sigma \bullet p(X \cup Y) \Rightarrow p(X)),$
- $\backslash \in Ops \Rightarrow (\forall X, Y \in \mathbb{P} \Sigma \bullet p(X) \Rightarrow p(X \cup Y)).$

**Theorem 12.** *Both  $\mathcal{T}$  and  $\mathcal{F}$  are compositional with respect to the whole language. Moreover they are the only models within the class that satisfy this property.*

**Proof.** Theorem 11 tells us that model  $\mathcal{M}_p$  is a congruence upon the whole language precisely when the following predicate holds:

$$\forall X, Y \in \mathbb{P} \Sigma \bullet (p(X) \Rightarrow p(X \cup Y)) \wedge (p(X \cup Y) \Rightarrow p(X)) .$$

Equivalently, if  $p$  is ever true it must be true for the whole of  $\mathbb{P} \Sigma$ . This occurs precisely when  $p$  is identically true or identically false. Hence  $\mathcal{F}$  (or  $\mathcal{M}_{\lambda X \bullet \text{true}}$ ) and  $\mathcal{T}$  (or  $\mathcal{M}_{\lambda X \bullet \text{false}}$ ) are the only models that are congruences upon the whole language.  $\square$

**Theorem 13.** *Of the other specific models we have considered:*

- Models  $\mathcal{S}$ ,  $\mathcal{ST}$  and  $\mathcal{U}_N$  for  $0 \leq N < \# \Sigma$  (recall that  $\mathcal{U}_{\# \Sigma} = \mathcal{F}$ ) are compositional with respect to the sub-language that excludes the hiding operator.
- Models  $\mathcal{CT}$  and  $\mathcal{L}_N$  for  $0 < N \leq \# \Sigma$  (recall that  $\mathcal{L}_0 = \mathcal{F}$ ) are compositional with respect to the sub-language that excludes parallel composition.
- Model  $\mathcal{R}_A$  for  $A \subset \Sigma$  (recall that  $\mathcal{R}_{\Sigma} = \mathcal{F}$ ), is compositional with respect to the sub-language that excludes the hiding of set of events  $B \not\subseteq A$ .
- Models  $\mathcal{A}_A$  and  $\mathcal{BR}_{A,N}$  for every  $A$  and for  $0 < N \leq \# \Sigma$ , (recall that  $\mathcal{BR}_{A,0} = \mathcal{R}_A$ ) are compositional with respect to the sub-language that excludes parallel composition and the hiding of set of events  $B \not\subseteq A$ .

#### 5. Automatic analysis

FDR [19,10] is a powerful analysis tool for CSP, which can be used to automatically check refinement of finite-state CSP processes in the traces and stable *failures* models. In

this section we consider whether it can also be used to check for refinement in other models of our class, by encoding refinement in such models as *failures* and/or traces refinement checks.

We identify first such an encoding for the subclass of actual refusal models, models concerned only with one fixed refusal set. In Sections 5.2 and 5.3 we show that such an encoding is possible also for all models associated with predicates that are either downwards or upwards closed: this is the case with all the specific models of this paper other than the actual refusals models, considered in Section 5.1, and the bounded and restricted refusals models, for which we present an encoding in Section 5.4. In Section 5.5, we consider the general case, and discuss the practicability of our constructions.

### 5.1. Actual refusals model for $A$

We begin by considering refinement within the actual refusals model, a model concerned only with one fixed refusal set  $A$ :  $\mathcal{M}_{\lambda X \bullet X=A}$ . If  $A$  is the empty set then  $p$  is downwards closed and we may apply the results from Section 5.2. Hence we consider only the cases where  $A \neq \emptyset$ .

In order to express,  $P \sqsubseteq_{\lambda X \bullet X=A} Q$ , or equivalently

$$\{(tr, X) \in failures(Q) \mid X = A\} \subseteq \{(tr, X) \in failures(P) \mid X = A\}, \quad (4)$$

as a refinement check within the established models we must retain failures information when the refusal set is  $A$ , and mask all other failures information. Provided  $P$  is trace-refined by  $Q$  we can achieve this by:

- preserving the trace information of both  $Q$  and  $P$ ;
- removing all trace-refusal pairs  $(tr, X)$  where  $X \not\subseteq A$  from the failures sets of both  $Q$  and  $P$ ; and
- flooding the failures sets of both  $Q$  and  $P$  with all trace-refusal pairs  $(tr, X)$  where  $tr \in traces(Q)$  and  $X \subset A$ .

If the resulting processes are  $Q'$  and  $P'$ , then

$$\begin{aligned} traces(P') &= traces(P), \\ traces(Q') &= traces(Q), \\ failures(P') &= \{(tr, X) \in failures(P) \mid X = A\} \cup \\ &\quad \{(tr, X) \mid tr \in traces(P) \wedge X \subset A\}, \\ failures(Q') &= \{(tr, X) \in failures(Q) \mid X = A\} \cup \\ &\quad \{(tr, X) \mid tr \in traces(Q) \wedge X \subset A\}. \end{aligned}$$

Eq. (4) will then hold precisely when  $P \sqsubseteq_{\mathcal{T}} Q \wedge P' \sqsubseteq_{\mathcal{F}} Q'$ .

Removing failures can be achieved by interleaving with a suitable process—in this case a process in which every event outside  $A$  is initially available. Flooding with additional failures at each trace can be achieved by non-deterministically allowing interruption by a suitable process—in this case a process which can refuse every *strict* subset of  $A$ .

In Theorem 14 below we prove our desired result for an arbitrary non-empty set  $A$ .

**Theorem 14.** For any processes  $P$  and  $Q$  defined over finite set of events  $\Sigma$ , and for any given set of events  $A \neq \emptyset$ ,

$$P \sqsubseteq_{\lambda X} \bullet_{X=A} Q \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge (P ||| R_A) \rightsquigarrow S_A \sqsubseteq_{\mathcal{F}} (Q ||| R_A) \rightsquigarrow S_A,$$

where  $R_A = \text{Offer}(\Sigma - A)$  and  $S_A = \bigcap Y : \mathbb{P} \Sigma \mid Y \subset A \bullet \text{Offer}(\Sigma - Y)$ .

**Proof.** Observe that since  $A \neq \emptyset$ ,

$$\text{failures}(S_A) = \{(\langle \rangle, X) \mid X \subset A\},$$

$$\text{failures}(R_A) = \{(\langle \rangle, X) \mid X \subseteq A\},$$

Hence

$$\begin{aligned} \text{failures}(P ||| R_A) &= \{(tr, X) \mid (tr, X) \in \text{failures}(P) \wedge \\ &\quad (\langle \rangle, X) \in \text{failures}(R_A)\} \\ &= \{(tr, X) \in \text{failures}(P) \mid X \subseteq A\}. \end{aligned} \tag{5}$$

We can calculate as follows.

$$\begin{aligned} &\text{failures}((P ||| R_A) \rightsquigarrow S_A) \\ &= \text{[def}^n \text{ of } \rightsquigarrow] \\ &\quad \text{failures}(P ||| R_A) \cup \\ &\quad \{(tr \hat{\ } tr', X) \mid tr \in \text{traces}(P ||| R_A) \wedge (tr', X) \in \text{failures}(S_A)\} \\ &= \text{[set theory; def}^n \text{ of } S_A] \\ &\quad \{(tr, X) \in \text{failures}(P ||| R_A) \mid X \not\subseteq A\} \cup \\ &\quad \{(tr, X) \in \text{failures}(P ||| R_A) \mid X \subset A\} \cup \\ &\quad \{(tr, X) \mid tr \in \text{traces}(P ||| R_A) \wedge X \subset A\} \\ &= \text{[equation (5); condition F1]} \\ &\quad \{(tr, X) \in \text{failures}(P) \mid X \subseteq A \wedge X \not\subseteq A\} \cup \\ &\quad \{(tr, X) \mid tr \in \text{traces}(P ||| R_A) \wedge X \subset A\} \\ &= \text{[def}^n \text{ of } R_A] \\ &\quad \{(tr, X) \mid (tr, X) \in \text{failures}(P) \wedge X = A\} \cup \\ &\quad \{(tr, X) \mid tr \in \text{traces}(P ||| R_A) \wedge X \subset A\}. \end{aligned}$$

And similarly for  $\text{failures}((Q ||| R_A) \rightsquigarrow S_A)$ .

In the reasoning below, we make use of the well-known result that all the CSP operators are monotonic with respect to inclusion of traces [20]. Given processes  $P$  and  $Q$  and set of

events  $A \neq \emptyset$ , we reason as follows:

$$\begin{aligned}
& P \sqsubseteq_{\mathcal{T}} Q \wedge (P ||| R_A) \rightsquigarrow S_A \sqsubseteq_{\mathcal{F}} (Q ||| R_A) \rightsquigarrow S_A \\
& \Leftrightarrow \quad \text{[def}^n \text{s of } \sqsubseteq_{\mathcal{T}} \text{ and } \sqsubseteq_{\mathcal{F}} \text{]} \\
& \quad \text{traces}(Q) \subseteq \text{traces}(P) \wedge \\
& \quad \text{traces}((Q ||| R_A) \rightsquigarrow S_A) \subseteq \text{traces}((P ||| R_A) \rightsquigarrow S_A) \wedge \\
& \quad \text{failures}((Q ||| R_A) \rightsquigarrow S_A) \subseteq \text{failures}((P ||| R_A) \rightsquigarrow S_A) \\
& \Leftrightarrow \quad \text{[monotonicity; above calculation]} \\
& \quad \text{traces}(Q) \subseteq \text{traces}(P) \wedge \\
& \quad \{(tr, X) | (tr, X) \in \text{failures}(Q) \wedge X = A\} \cup \\
& \quad \{(tr, X) | tr \in \text{traces}(Q ||| R_A) \wedge X \subset A\} \\
& \quad \subseteq \\
& \quad \{(tr, X) | (tr, X) \in \text{failures}(P) \wedge X = A\} \cup \\
& \quad \{(tr, X) | tr \in \text{traces}(P ||| R_A) \wedge X \subset A\} \\
& \Leftrightarrow \quad \text{[set theory]} \\
& \quad \text{traces}(Q) \subseteq \text{traces}(P) \wedge \\
& \quad \{(tr, X) | (tr, X) \in \text{failures}(Q) \wedge X = A\} \subseteq \\
& \quad \{(tr, X) | (tr, X) \in \text{failures}(P) \wedge X = A\} \wedge \\
& \quad \{(tr, X) | tr \in \text{traces}(Q ||| R_A) \wedge X \subset A\} \subseteq \\
& \quad \{(tr, X) | tr \in \text{traces}(P ||| R_A) \wedge X \subset A\} \\
& \Leftrightarrow \quad \text{[monotonicity; condition (F1)]} \\
& \quad \text{traces}(Q) \subseteq \text{traces}(P) \wedge \\
& \quad \{(tr, X) | (tr, X) \in \text{failures}(Q) \wedge X = A\} \subseteq \\
& \quad \{(tr, X) | (tr, X) \in \text{failures}(P) \wedge X = A\} \\
& \Leftrightarrow P \sqsubseteq_{\lambda X \bullet X=A} Q. \quad \text{[def}^n \text{ of } \sqsubseteq_p \text{]}
\end{aligned}$$

□

## 5.2. Downwards closed predicates

We now present an encoding of refinement within  $\mathcal{M}_p$  for any predicate  $p$  that is *downwards* closed and not identically false (if  $p$  is identically false, then  $\mathcal{M}_p = \mathcal{T}$ ), in terms of refinement within  $\mathcal{T}$  and  $\mathcal{F}$ . This may be used to verify refinement within  $\mathcal{S}$ ,  $\mathcal{ST}$ ,  $\mathcal{R}_A$  for any set of events  $A$ , and  $\mathcal{U}_N$  for  $0 \leq N \leq \#\Sigma$ .

We use techniques similar to those in Section 5.1: we interleave with the process  $T_p$  that can initially refuse any set  $Y$  such that  $p(Y)$  is *true*, and that diverges after any event is performed:

$$T_p \triangleq \prod Y : \mathbb{P} \Sigma | p(Y) \bullet \text{Offer}(\Sigma - Y).$$

We observe that

$$\begin{aligned}
 \text{traces}(T_p) &= \{\langle \rangle\} \cup \{\langle a \rangle \mid \exists Y \bullet p(Y) \wedge a \in \Sigma - Y\} && [\text{def}^n \text{ of } T_p] \\
 &= [\text{p is downwards closed and not identically false so } P(\{\})] \\
 &\quad \{\langle \rangle\} \cup \{\langle a \rangle \mid a \in \Sigma\} \\
 \text{failures}(T_p) &= \{(\langle \rangle, X) \mid \exists Y \bullet p(Y) \wedge X \subseteq Y\} && [\text{def}^n \text{ of } T_p] \\
 &= \{(\langle \rangle, X) \mid p(X)\} . && [\text{p is downwards closed}]
 \end{aligned}$$

The interleaving of process  $T_p$  with any process  $P$  then yields a process whose stable failures are equal to  $\text{failures}_p(P)$ :

$$\begin{aligned}
 \text{failures}(P \parallel T_p) &= \{(tr, X) \mid (tr, X) \in \text{failures}(P) \wedge (\langle \rangle, X) \in \text{failures}(T_p)\} \\
 &= \{(tr, X) \mid (tr, X) \in \text{failures}(P) \wedge p(X)\} \\
 &= \text{failures}_p(P),
 \end{aligned}$$

and similarly for  $Q \parallel T_p$ .

**Theorem 15.** Suppose predicate  $p$  is downwards closed, and  $p \neq \lambda X \bullet \text{false}$ . Then

$$P \sqsubseteq_p Q \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge P \parallel T_p \sqsubseteq_{\mathcal{F}} Q \parallel T_p,$$

where  $T_p$  is as defined above.

**Proof.**

$$\begin{aligned}
 &P \sqsubseteq_{\mathcal{T}} Q \wedge P \parallel T_p \sqsubseteq_{\mathcal{F}} Q \parallel T_p \\
 \Leftrightarrow &\text{traces}(Q) \subseteq \text{traces}(P) \wedge && [\text{def}^n \text{ of } \sqsubseteq_{\mathcal{T}} \text{ and } \sqsubseteq_{\mathcal{F}}] \\
 &\quad \text{traces}(Q \parallel T_p) \subseteq \text{traces}(P \parallel T_p) \wedge \\
 &\quad \text{failures}(Q \parallel T_p) \subseteq \text{failures}(P \parallel T_p) \\
 \Leftrightarrow & && [\text{monotonicity; above result}] \\
 &\text{traces}(Q) \subseteq \text{traces}(P) \wedge \text{failures}_p(Q) \subseteq \text{failures}_p(P) \\
 \Leftrightarrow &P \sqsubseteq_p Q. && [\text{def}^n \text{ of } \sqsubseteq_p]
 \end{aligned}$$

□

**Corollary 16.** Refinement within  $\mathcal{S}$ ,  $\mathcal{ST}$ ,  $\mathcal{R}_A$  for any set of events  $A$ , and  $\mathcal{U}_N$  for  $0 \leq N \leq \# \Sigma$  and can be verified using the following checks:

$$\begin{aligned}
 P \sqsubseteq_{\mathcal{S}} Q &\Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge P \parallel T_{\mathcal{S}} \sqsubseteq_{\mathcal{F}} Q \parallel T_{\mathcal{S}}, \\
 P \sqsubseteq_{\mathcal{ST}} Q &\Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge P \parallel T_{\mathcal{ST}} \sqsubseteq_{\mathcal{F}} Q \parallel T_{\mathcal{ST}}, \\
 P \sqsubseteq_{\mathcal{R}_A} Q &\Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge P \parallel T_{\mathcal{R}_A} \sqsubseteq_{\mathcal{F}} Q \parallel T_{\mathcal{R}_A}, \\
 P \sqsubseteq_{\mathcal{U}_N} Q &\Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge P \parallel T_{\mathcal{U}_N} \sqsubseteq_{\mathcal{F}} Q \parallel T_{\mathcal{U}_N},
 \end{aligned}$$



where

$$\begin{aligned} T_S &\triangleq \sqcap Y : \mathbb{P} \Sigma | \#Y \leq 1 \bullet \text{Offer}(\Sigma - Y), \\ T_{ST} &\triangleq \text{Offer}(\Sigma), \\ T_{\mathcal{R}_A} &\triangleq \sqcap Y : \mathbb{P} \Sigma | Y \subseteq A \bullet \text{Offer}(\Sigma - Y), \\ T_{\mathcal{U}_N} &\triangleq \sqcap Y : \mathbb{P} \Sigma | \#Y \leq N \bullet \text{Offer}(\Sigma - Y). \end{aligned}$$

### 5.3. Upwards closed predicates

We now present an encoding of refinement within  $\mathcal{M}_p$  for any predicate  $p$  that is *upwards* closed and not identically true (if  $p$  is identically true, then  $\mathcal{M}_p = \mathcal{F}$ ), in terms of refinement within  $\mathcal{T}$  and  $\mathcal{F}$ . This may be used to verify refinement within  $\mathcal{CT}$  and  $\mathcal{L}_N$  for  $0 < N \leq \# \Sigma$ .

We again use techniques similar to those in Section 5.1. We introduce the process  $U_p$  that can initially refuse any set  $Y$  such that  $p(Y)$  is *false*, and that diverges after any event is performed:

$$U_p \triangleq \sqcap Y : \mathbb{P} \Sigma | \neg p(Y) \bullet \text{Offer}(\Sigma - Y).$$

We observe that

$$\begin{aligned} \text{traces}(U_p) &= \{\langle \rangle\} \cup \{\langle a \rangle | \exists Y \bullet \neg p(Y) \wedge a \in \Sigma - Y\} && [\text{def}^n \text{ of } U_p] \\ &= [\text{p is upwards closed and not identically true so } \neg p(\{\})] \\ &\quad \{\langle \rangle\} \cup \{\langle a \rangle | a \in \Sigma\} \\ \text{failures}(U_p) &= \{\langle \langle \rangle, X \rangle | \exists Y \bullet \neg p(Y) \wedge X \subseteq Y\} && [\text{def}^n \text{ of } U_p] \\ &= \{\langle \langle \rangle, X \rangle | \neg p(X)\} && [\text{p is upwards closed}] \end{aligned}$$

The effect of non-deterministically interrupting process  $P$  with the process  $U_p$  is to augment the failures set with trace refusal pairs  $(tr, X)$  such that  $tr$  is a trace of  $P$  and  $p(X)$  is false:

$$\begin{aligned} \text{failures}(P \rightsquigarrow U_p) &= \text{failures}(P) \cup \\ &\quad \{(tr \hat{\ } tr', X) | tr \in \text{traces}(P) \wedge (tr', X) \in \text{failures}(U_p)\} \\ &= \text{failures}(P) \cup \{(tr, X) | tr \in \text{traces}(P) \wedge \neg p(X)\}, \end{aligned}$$

and similarly for  $Q \rightsquigarrow U_p$ .

**Theorem 17.** Suppose predicate  $p$  is upwards closed, and  $p \neq \lambda X \bullet \text{true}$ . Then

$$P \sqsubseteq_p Q \Leftrightarrow P \rightsquigarrow U_p \sqsubseteq_{\mathcal{F}} Q \rightsquigarrow U_p$$

where  $U_p$  is as defined above.

**Proof.**

$$\begin{aligned}
& P \rightsquigarrow U_p \sqsubseteq_{\mathcal{F}} Q \rightsquigarrow U_p \\
& \Leftrightarrow \text{traces}(Q \rightsquigarrow U_p) \subseteq \text{traces}(P \rightsquigarrow U_p) \wedge \text{failures}(Q \rightsquigarrow U_p) \subseteq \text{failures}(P \rightsquigarrow U_p) & [\text{def}^n \text{ of } \sqsubseteq_{\mathcal{F}}] \\
& \Leftrightarrow \text{traces}(Q \rightsquigarrow U_p) \subseteq \text{traces}(P \rightsquigarrow U_p) \wedge \text{failures}(Q) \cup \{(tr, X) \mid tr \in \text{traces}(Q) \wedge \neg p(X)\} \subseteq \text{failures}(P) \cup \{(tr, X) \mid tr \in \text{traces}(P) \wedge \neg p(X)\} & [\text{above result}] \\
& \Leftrightarrow & [\text{for all } X, p(X) \text{ or } \neg p(X)] \\
& \quad \text{traces}(Q \rightsquigarrow U_p) \subseteq \text{traces}(P \rightsquigarrow U_p) \wedge \{(tr, X) \in \text{failures}(Q) \mid p(X)\} \cup \{(tr, X) \in \text{failures}(Q) \mid \neg p(X)\} \cup \{(tr, X) \mid tr \in \text{traces}(Q) \wedge \neg p(X)\} \subseteq \{(tr, X) \in \text{failures}(P) \mid p(X)\} \cup \{(tr, X) \in \text{failures}(P) \mid \neg p(X)\} \cup \{(tr, X) \mid tr \in \text{traces}(P) \wedge \neg p(X)\} \\
& \Leftrightarrow & [\text{def}^n \text{ of } \text{failures}_p; \text{condition (F1)}] \\
& \quad \text{traces}(Q \rightsquigarrow U_p) \subseteq \text{traces}(P \rightsquigarrow U_p) \wedge \text{failures}_p(Q) \subseteq \text{failures}_p(P) \wedge \{(tr, X) \mid tr \in \text{traces}(Q) \wedge \neg p(X)\} \subseteq \{(tr, X) \mid tr \in \text{traces}(P) \wedge \neg p(X)\} \\
& \Leftrightarrow \text{traces}(Q \rightsquigarrow U_p) \subseteq \text{traces}(P \rightsquigarrow U_p) \wedge \text{failures}_p(Q) \subseteq \text{failures}_p(P) \wedge \text{traces}(Q) \subseteq \text{traces}(P) & [p \text{ is not identically true}] \\
& \Leftrightarrow \text{failures}_p(Q) \subseteq \text{failures}_p(P) \wedge \text{traces}(Q) \subseteq \text{traces}(P) & [\text{monotonicity}] \\
& \Leftrightarrow P \sqsubseteq_p Q. & [\text{def}^n \text{ of } \sqsubseteq_p]
\end{aligned}$$

□

**Corollary 18.** *Refinement within  $\mathcal{CT}$  and  $\mathcal{L}_N$  for  $0 < N \leq \# \Sigma$  can be verified using the following checks:*

$$\begin{aligned}
P \sqsubseteq_{\mathcal{CT}} Q & \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge P \rightsquigarrow U_{\mathcal{CT}} \sqsubseteq_{\mathcal{F}} Q \rightsquigarrow U_{\mathcal{CT}}, \\
P \sqsubseteq_{\mathcal{L}_N} Q & \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge P \rightsquigarrow U_{\mathcal{L}_N} \sqsubseteq_{\mathcal{F}} Q \rightsquigarrow U_{\mathcal{L}_N},
\end{aligned}$$

where

$$\begin{aligned}
U_{\mathcal{CT}} & \triangleq \prod Y : \mathbb{P} \Sigma \mid Y \neq \Sigma \bullet \text{Offer}(\Sigma - Y), \\
U_{\mathcal{L}_N} & \triangleq \prod Y : \mathbb{P} \Sigma \mid \#Y < N \bullet \text{Offer}(\Sigma - Y).
\end{aligned}$$

#### 5.4. Bounded and restricted refusals model for $A$ and $N$

For  $\{\} \subset A \subset \Sigma$  and  $0 < N < \#\Sigma$ , the predicate associated with the bounded and restricted refusals model  $\mathcal{BR}_{A,N}$ ,  $\lambda X \bullet X \subseteq A \wedge \#X \geq N$ , is neither upwards nor downwards closed. Refinement within this model may however be captured easily in terms of refinement within  $\mathcal{T}$  and  $\mathcal{F}$ , as shown by the following theorem.

**Theorem 19.** *For any processes  $P$  and  $Q$  defined over a finite set of events  $\Sigma$ , and for  $\{\} \subset A \subset \Sigma$  and  $0 < N < \#\Sigma$ ,*

$$P \sqsubseteq_{\lambda X \bullet X \subseteq A \wedge \#X \geq N} Q \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge ((P ||| R_A) \rightsquigarrow V_{A,N}) \sqsubseteq_{\mathcal{F}} ((Q ||| R_A) \rightsquigarrow V_{A,N}),$$

where

$$R_A \hat{=} \text{Offer}(\Sigma - A), \\ V_{A,N} \hat{=} \bigcap Y : \mathbb{P} \Sigma \mid Y \subseteq A \wedge \#Y < N \bullet \text{Offer}(\Sigma - Y).$$

The proof is in Appendix A.

#### 5.5. General predicates

In this section we identify rules for verifying refinement within models for which the associated predicate is neither upwards nor downwards closed. We prove first that checkability is closed under disjunctions over corresponding predicates: if  $\sqsubseteq_p$  and  $\sqsubseteq_q$  are checkable then so is  $\sqsubseteq_{p \vee q}$ .

**Lemma 20.** *Given any processes  $P$  and  $Q$  and predicates  $p$  and  $q$ ,*

$$P \sqsubseteq_{p \vee q} Q \Leftrightarrow P \sqsubseteq_p Q \wedge P \sqsubseteq_q Q.$$

The proof is in Appendix A.

It follows immediately from the above result, the associativity of  $\wedge$  and  $\vee$ , and Theorem 14 in which we presented techniques for verifying refinement within  $\mathcal{A}_A$ , that  $\sqsubseteq_p$  is checkable for any predicate  $p$ .

**Theorem 21.** *For any predicate  $p$  over a finite alphabet  $\Sigma$  we may express refinement within model  $\mathcal{M}_p$  in terms of refinement within  $\mathcal{T}$  and  $\mathcal{F}$ . In particular,*

$$P \sqsubseteq_p Q \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge \bigwedge_{X_i \in \mathbb{P} \Sigma \mid p(X_i)} (P ||| R_{X_i}) \rightsquigarrow S_{X_i} \sqsubseteq_{\mathcal{F}} (Q ||| R_{X_i}) \rightsquigarrow S_{X_i}.$$

where  $R_A = \text{Offer}(\Sigma - A)$  and  $S_A = \bigcap Y : \mathbb{P} \Sigma \mid Y \subset A \bullet \text{Offer}(\Sigma - Y)$ .

Note that the check done for each  $X_i$  corresponds to the check for refinement in  $\mathcal{A}_{X_i}$ , the actual refusals model for  $X_i$ , from Section 5.1.

The proof is in Appendix A.

In Theorem 21 above we have shown that, in the worst case, verifying refinement within model  $\mathcal{M}_p$  will require  $\#\{X \in \mathbb{P} \Sigma \mid p(X)\} + 1$  refinement checks. However, it should be noted that this is the *worst-case* scenario. Indeed, in Sections 5.1, 5.2, 5.3, and 5.4 we showed that verification of refinement within all models under consideration in this paper, as well as any other models in which the associated predicate is either upwards or downwards closed, is relatively straightforward requiring only one refinement check in each of  $\mathcal{T}$  and  $\mathcal{F}$ .

## 6. Discussion

In this paper we have motivated the need for means of verifying non-standard measures of consistency and hence the need for alternative semantic models. We identified the *failures class*, a particular family of semantic models for describing concurrent systems, each model recording all trace information and possibly some information about subsequent availability of events. The amount of availability—or rather possibility of refusal—information recorded by each model is determined by the predicate on sets of events with which it is associated.

To put this work in a wider context, we have put under the microscope a small section of van Glabbeek’s linear time-branching time spectrum [24], presenting an entire sub-lattice. The top and bottom elements of our sub-lattice are Roscoe’s stable failures and traces models—identified respectively as “failures semantics” and “trace semantics” within van Glabbeek’s spectrum—the other model they share being the completed trace model.

We discussed in detail four established models that are members of this class: Roscoe’s traces and stable failures model [20]; Bolton’s singleton failures model [1,2]; and van Glabbeek’s completed trace model [24]. We examined also several families of non-established models.

We proved that the failures class forms a complete lattice, and identified the position within the lattice of each of these models, thereby exposing their relative strengths. We identified the maximal sub-language over which each model is compositional, verifying that only the traces and the stable failures semantics are fully compositional.

For each model under consideration we identified various problem domains in which the associated measure of consistency would be useful and appropriate. For a selected few of the models we explored in greater detail a particular example within one of these domains. We expressed the predicate defining consistency for each measure in terms of refinement checks within the associated model.

Finally, we presented techniques for using the model-checker FDR [19,10] to automatically verify such measures of consistency. We presented a general technique for expressing refinement in arbitrary models of the failures class as refinements within the traces and stable failures models, and presented particular, efficient, techniques that could be applied in each of the specific models we have considered.

It is interesting to observe that the techniques for verifying consistency presented in this paper are not unique: other operators and combinations could be used to mask the unwanted failures.

In [22], Roscoe considers which predicates  $R$  over the semantics of processes can be captured as refinement checks. He shows that all predicates that are closed under the metric

topology can be captured as refinement checks. He gives simplified refinement checks in the cases that  $R$  is refinement-closed (i.e. such that if  $P$  satisfies  $R$  and  $P \sqsubseteq Q$  then  $Q$  satisfies  $R$ ) and/or distributive (i.e. such that if each process in the set  $S$  satisfies  $R$ , then so does  $\prod S$ ). However, his refinement checks are not, in general, finite-state, and so cannot be automated using a tool such as FDR.

More concretely, Roscoe has come up with an alternative method for determining whether a process can refuse a set of events  $A$  only when its specification could refuse the same set of events [21]. This work arose from a problem posed by Reed and Sinclair, following on from [18], in which their predicate “is as live as” could be verified using techniques similar to those presented in Section 3.5. Roscoe’s technique is as follows. Let  $e \notin \Sigma$  be a fresh event, and let  $R_A$  be the renaming relation that maps each element  $a$  of  $A$  to both itself and  $e$ , and is otherwise the identity function. Let

$$\begin{aligned} \text{CheckAll}_A(Q) &\triangleq Q \parallel_{\Sigma \cup \{e\}} \text{CantBlock}_e, \\ \text{CantBlock}_e &\triangleq (?x : \Sigma \rightarrow \text{CantBlock}_e \sqcap \text{STOP}) \square e \rightarrow \text{STOP}. \end{aligned}$$

$\text{CheckAll}_A(Q)$  can perform any trace of  $Q$ , but can also perform an  $e$  when  $Q$  can do an event from  $A$ , after which it must stop. After any trace it can refuse anything except for  $e$ ; it can refuse  $e$  only when  $Q$  can refuse the whole of  $A$ . Hence the requirement that  $P$  refuses  $A$  only when  $\text{SPEC}$  refuses  $A$  can be captured by the following refinement check:

$$\text{CheckAll}_A(\text{SPEC}) \sqsubseteq_{\mathcal{F}} \text{CheckAll}_A(P).$$

This check and the one introduced in Section 3.5 take comparable times.

## Acknowledgements

We would like to thank Bill Roscoe, Michael Goldsmith and Irfan Zakiuddin for interesting discussions on this work.

This work is partly funded by the Ministry of Defence through QinetiQ. Earlier versions were presented at the IWFm’03 and EXPRESS’03 workshops [4,3]; we would like to thank the referees and participants of those workshops as well as the referees of this journal for useful comments.

## Appendix A. Proofs

**Proof of Theorem 19.** First observe that, since  $N > 0$ ,

$$\text{failures}(V_{A,N}) = \{(\langle \rangle, X) \mid \#X < N \wedge X \subseteq A\}.$$

Hence, by definition of  $\triangle$ , we deduce that for any  $Q$

$$\begin{aligned} \text{failures}(Q \triangle V_{A,N}) &= \{(tr \hat{\ } tr', X) \mid tr \in \text{traces}(Q) \wedge (tr', X) \in \text{failures}(V_{A,N})\} \\ &= \{(tr, X) \mid tr \in \text{traces}(Q) \wedge \#X < N \wedge X \subseteq A\}. \end{aligned} \tag{A.1}$$

We now reason as follows:

$$\begin{aligned}
& failures((P|||R_A) \rightsquigarrow V_{A,N}) \\
&= failures(P|||R_A) \cup failures((P|||R_A) \triangle V_{A,N}) \quad [\text{def}^n \text{ of } \rightsquigarrow, \sqcap] \\
&= \{(tr, X) \in failures(P) \mid X \subseteq A\} \cup \{(tr, X) \mid tr \in traces(P|||R_A) \wedge \#X < N \wedge X \subseteq A\} \quad [\text{equations (5) and (A.1)}] \\
&= \{(tr, X) \in failures(P) \mid \#X \geq N \wedge X \subseteq A\} \cup \{(tr, X) \in failures(P) \mid \#X < N \wedge X \subseteq A\} \cup \{(tr, X) \mid tr \in traces(P|||R_A) \wedge \#X < N \wedge X \subseteq A\} \quad [\text{set theory}] \\
&= \{(tr, X) \in failures(P) \mid \#X \geq N \wedge X \subseteq A\} \cup \{(tr, X) \mid tr \in traces(P|||R_A) \wedge \#X < N \wedge X \subseteq A\} \quad [\text{def}^n \text{ of } |||; \text{Condition F1}]
\end{aligned}$$

and similarly for  $Q$ .

Suppose, then  $P \sqsubseteq_{\mathcal{T}} Q$ . We calculate as follows:

$$\begin{aligned}
& (P|||R_A) \rightsquigarrow V_{A,N} \sqsubseteq_{\mathcal{F}} (Q|||R_A) \rightsquigarrow V_{A,N} \\
& \Leftrightarrow \begin{aligned} & traces((Q|||R_A) \rightsquigarrow V_{A,N}) \subseteq traces((P|||R_A) \rightsquigarrow V_{A,N}) \\ & \wedge \\ & failures((Q|||R_A) \rightsquigarrow V_{A,N}) \subseteq failures((P|||R_A) \rightsquigarrow V_{A,N}) \end{aligned} \quad [\text{def}^n \text{ of } \sqsubseteq_{\mathcal{F}}] \\
& \Leftrightarrow \begin{aligned} & [P \sqsubseteq_{\mathcal{T}} Q, \text{monotonicity, above calculation}] \\ & \{(tr, X) \in failures(Q) \mid \#X \geq N \wedge X \subseteq A\} \cup \\ & \{(tr, X) \mid tr \in traces(Q|||R_A) \wedge \#X < N \wedge X \subseteq A\} \\ & \subseteq \\ & \{(tr, X) \in failures(P) \mid \#X \geq N \wedge X \subseteq A\} \cup \\ & \{(tr, X) \mid tr \in traces(P|||R_A) \wedge \#X < N \wedge X \subseteq A\} \end{aligned} \\
& \Leftrightarrow \begin{aligned} & \{(tr, X) \in failures(Q) \mid \#X \geq N \wedge X \subseteq A\} \subseteq \\ & \{(tr, X) \in failures(P) \mid \#X \geq N \wedge X \subseteq A\} \\ & \wedge \\ & \{(tr, X) \mid tr \in traces(Q|||R_A) \wedge \#X < N \wedge X \subseteq A\} \subseteq \\ & \{(tr, X) \mid tr \in traces(P|||R_A) \wedge \#X < N \wedge X \subseteq A\} \end{aligned} \quad [\text{set theory}] \\
& \Leftrightarrow \begin{aligned} & [P \sqsubseteq_{\mathcal{T}} Q, \text{monotonicity, set theory}] \\ & \{(tr, X) \in failures(Q) \mid \#X \geq N \wedge X \subseteq A\} \subseteq \\ & \{(tr, X) \in failures(P) \mid \#X \geq N \wedge X \subseteq A\}. \end{aligned} \\
& \Leftrightarrow P \sqsubseteq_{\lambda X} \bullet_{X \subseteq A \wedge \#X \geq N} Q \quad \square \quad [SPEC \sqsubseteq_{\mathcal{T}} P \text{ and def}^n \text{ of } \sqsubseteq_p]
\end{aligned}$$

**Proof of Lemma 20.**

$$\begin{aligned}
& P \sqsubseteq_p Q \wedge P \sqsubseteq_q Q \\
& \Leftrightarrow \text{traces}(P) \supseteq \text{traces}(Q) \wedge \quad [\text{def}^n \text{ of refinement}] \\
& \quad \{(tr, X) \in \text{failures}(P) | p(X)\} \supseteq \\
& \quad \{(tr, X) \in \text{failures}(Q) | p(X)\} \wedge \\
& \quad \{(tr, X) \in \text{failures}(P) | q(X)\} \supseteq \\
& \quad \{(tr, X) \in \text{failures}(Q) | q(X)\} \\
& \Leftrightarrow \text{traces}(P) \supseteq \text{traces}(Q) \wedge \quad [\text{set theory}] \\
& \quad \{(tr, X) \in \text{failures}(P) | p(X) \vee q(X)\} \supseteq \\
& \quad \{(tr, X) \in \text{failures}(Q) | p(X) \vee q(X)\} \\
& \Leftrightarrow P \sqsubseteq_{p \vee q} Q. \quad [\text{def}^n \text{ of refinement}]
\end{aligned}$$

□

**Proof of Theorem 21.**

Given processes  $P$  and  $Q$  and predicate  $p$  we reason as follows:

$$\begin{aligned}
& P \sqsubseteq_p Q \\
& \Leftrightarrow \quad [\text{def}^n \text{ of } \sqsubseteq_p] \\
& \quad \text{traces}(P) \supseteq \text{traces}(Q) \wedge \\
& \quad \{(tr, X) \in \text{failures}(P) | p(X)\} \supseteq \{(tr, X) \in \text{failures}(Q) | p(X)\} \\
& \Leftrightarrow \bigwedge_{X_i \in \mathbb{P} \Sigma | p(X_i)} \quad [\text{set theory, distributivity}] \\
& \quad \text{traces}(P) \supseteq \text{traces}(Q) \wedge \\
& \quad \{(tr, X) \in \text{failures}(P) | X = X_i\} \supseteq \\
& \quad \{(tr, X) \in \text{failures}(Q) | X = X_i\} \\
& \Leftrightarrow \bigwedge_{X_i \in \mathbb{P} \Sigma | p(X_i)} P \sqsubseteq_{\lambda X \bullet X = X_i} Q \quad [\text{Lemma 20; def}^n \text{ of refinement}] \\
& \Leftrightarrow \bigwedge_{X_i \in \mathbb{P} \Sigma | p(X_i)} \quad [\text{Theorem 14}] \\
& \quad P \sqsubseteq_{\mathcal{T}} Q \wedge (P ||| R_{X_i}) \rightsquigarrow S_{X_i} \sqsubseteq_{\mathcal{F}} (Q ||| R_{X_i}) \rightsquigarrow S_{X_i} \\
& \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge \quad [\text{distributivity}] \\
& \quad \bigwedge_{X_i \in \mathbb{P} \Sigma | p(X_i)} (P ||| R_{X_i}) \rightsquigarrow S_{X_i} \sqsubseteq_{\mathcal{F}} (Q ||| R_{X_i}) \rightsquigarrow S_{X_i}. \quad \square
\end{aligned}$$

**References**

- [1] C. Bolton, On the refinement of state-based and event-based models, D. Phil., University of Oxford, 2002.
- [2] C. Bolton, J. Davies, A singleton failures semantics for communicating sequential processes, 2001. Formal Aspects of Computing, to appear.
- [3] C. Bolton, G. Lowe, A hierarchy of failures-based models, in: Proc. 10th International Workshop on Expressiveness in Concurrency: EXPRESS'03, 2003.

- [4] C. Bolton, G. Lowe, On the automatic verification of non-standard measures of consistency, in: Proc. International Workshop on Formal Methods: IWFM'03, 2003.
- [5] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronisation skeletons using branching-time temporal logic, in: Proc. International Workshop on Logic of Programs, Vol. 131, Lecture Notes in Computer Science, 1981, pp. 52–71.
- [6] R. de Nicola, Extensional equivalences for transition systems, *Acta Inform.* 24 (1987).
- [7] R. de Nicola, M. Hennessy, Testing equivalences for processes, *Theoret. Comput. Sci.* 34 (1984).
- [8] W.-P. de Roever, K. Engelhardt, Data refinement: model-oriented proof methods and their comparison, Cambridge Tracts in Theoretical Computer Science, 1998.
- [9] E.A. Emerson, J.Y. Halpern, Sometimes and not never revisited: on branching versus linear time, *J. ACM* 33 (1) (1986) 151–178.
- [10] Formal Systems (Europe) Ltd. Failures-Divergence Refinement—FDR 2 User Manual, 1999. Available via URL [http://www.fsel.com/fdr\\_2.manual.html](http://www.fsel.com/fdr_2.manual.html).
- [11] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice Hall, Englewood Cliffs, NJ, 1985.
- [12] R. Milner, *A Calculus of Communicating Systems*, volume 92 of Lecture Notes in Computer Science, Springer-Verlag, 1980.
- [13] R. Milner, *Communications and Concurrency*, Prentice Hall, Englewood Cliffs, NJ, 1989.
- [14] E.-R. Olderog, C.A.R. Hoare, Specification-oriented semantics for communicating processes, *Acta Inform.* 23 (1986).
- [15] J.L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall International, 1981.
- [16] A. Pnueli, The temporal logic of programs, in: Proc. 18th International Symposium on Foundations of Computer Science, 1977, pp. 46–57.
- [17] G.M. Reed, A uniform mathematical theory for real-time distributed computing, Ph.D. Thesis, University of Oxford, 1988.
- [18] J.N. Reed, J.E. Sinclair, Combining independent specifications, in: Proc. Fundamental Approaches to Software Engineering, Vol. 2029, Lecture Notes in Computer Science, 2001.
- [19] A.W. Roscoe, Model-checking CSP, in: *A Classical Mind, Essays in Honour of C.A.R. Hoare*, Prentice-Hall, Englewood Cliffs, NJ, 1994.
- [20] A.W. Roscoe, *The Theory and Practice of Concurrency*, Prentice Hall, Englewood Cliffs, NJ, 1997.
- [21] A.W. Roscoe, Personal Communication, 2003.
- [22] A.W. Roscoe, On the expressiveness of CSP refinement checking, in: Proc. Third Workshop on Automated Verification of Critical Systems (AVoCS 2003), 2003.
- [23] J. Rumbaugh, I. Jacobson, G. Booch, *The Unified Modeling Language reference manual*, Addison-Wesley, 1997.
- [24] R.J. 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*, Elsevier, Reading, MA, 2001.