A Hierarchy of Failures-Based Models

Christie Bolton, Gavin Lowe [1,2]

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

Abstract

In this paper we identify the *failures class*, a class of semantic models for describing concurrent systems. Each such model records all possible sequences of interaction, and gives some information about subsequent availability. Each model is associated with a predicate that determines how much availability information is recorded.

The general contribution of the paper is three-fold: we identify the relative strengths of models in terms of their defining predicates; we identify the maximal subset of the language over which each model induces a congruence; and we show how refinement in each model can be automatically tested.

More concretely, we apply these general results to specific instances of the class. In particular we construct a spectrum showing the relative strengths of four established models and three interesting new models, and we prove that only Roscoe's stable failures and traces models define congruences over the *whole* language.

1 Introduction

A variety of languages and semantic models, from process algebras such as CSP [10] and CCS [11,12] through modal logics [15,4,8] to Petri nets [14], have been proposed for describing and reasoning about the properties of concurrent systems. 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 [19], van Glabbeek considers various semantic equivalences for modelling finitely branching, concrete, sequential, non-deterministic processes. He presents them in a language-

[1] Email: christie@comlab.ox.ac.uk

[2] Email: gavinl@comlab.ox.ac.uk

This is a preliminary version. The final version will be published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

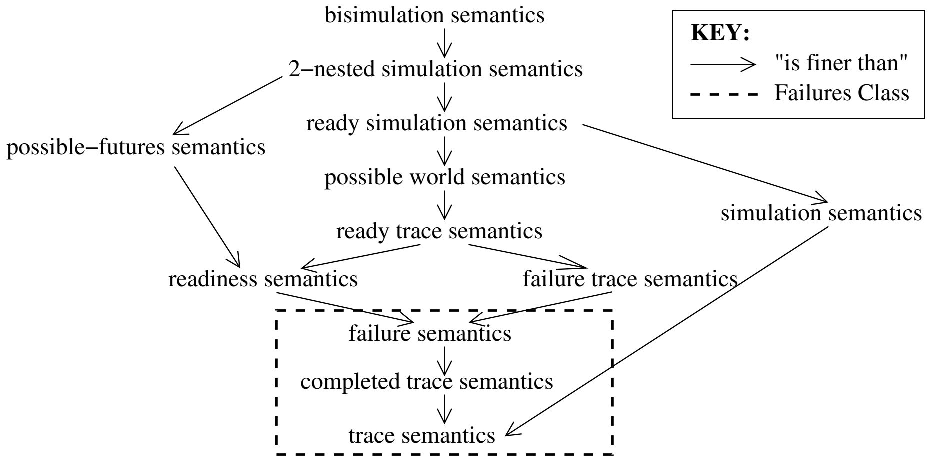

Fig. 1. Linear time – branching time spectrum of van Glabbeek [19], illustrating the positioning of the failures class of semantic models.

independent style, reducing them to eleven distinct models as illustrated in his linear time – branching time sprectrum: see Figure 1.

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. Each such model records possible sequences of interaction, along with *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 internal actions. We use the process algebra Communicating Sequential Processes (CSP) [10,18] 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; to identify the maximal subset of the language for which each model defines a congruence; to consider for which such models is automatic refinement-checking possible; and to identify and consider the properties of models—both established and new—within this class.

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 [18].

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 [18]; the traces model [18]; the singleton failures model [1]; and the completed trace model [19]. Furthermore we identify non-standard models within the class and discuss their potential applications. 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 congruences with respect to which operators, 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 congruences 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 [18].

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.

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. For later convenience we define $\mathit{Offer}(A) \mathrel{\widehat{=}} ?a : A \rightarrow \mathsf{div}$, the process that offers the events from A, and then diverges.

CSP has two choice operators: $P \mathrel{\Box} Q$ represents the *external* choice and $P \sqcap Q$ the *internal* (or non-deterministic) choice between processes P and Q; the process $\bigsqcap i : I \mid p(i) \bullet P_i$ 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 $P \mathrel{\triangle} Q$ 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 $P \mathbin{{}_A\|_B} Q$ 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, $P \mathbin{|||} Q$ 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.

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 Σ^*, where Σ 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.

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

$$\langle\rangle \in \mathit{traces}(P), \tag{T1}$$

$$\mathit{tr} \frown \mathit{tr}' \in \mathit{traces}(P) \;\Rightarrow\; \mathit{tr} \in \mathit{traces}(P), \tag{T2}$$

$$(\mathit{tr}, X) \in \mathit{failures}(P) \;\Rightarrow\; \mathit{tr} \in \mathit{traces}(P), \tag{F1}$$

$$(\mathit{tr}, X \cup Y) \in \mathit{failures}(P) \;\Rightarrow\; (\mathit{tr}, X) \in \mathit{failures}(P), \tag{F2}$$

$$\begin{aligned}&(\mathit{tr}, Y) \in \mathit{failures}(P) \wedge \forall\, x \in X \bullet \mathit{tr} \frown \langle x\rangle \notin \mathit{traces}(P) \;\Rightarrow\\ &\qquad (\mathit{tr}, X \cup Y) \in \mathit{failures}(P).\end{aligned} \tag{F3}$$

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 $(\mathit{tr}, \emptyset)$ from the set $\mathit{failures}(P)$ for some $\mathit{tr} \in \mathit{traces}(P)$ indicates divergence. Semantic equations for the functions *traces* and *failures* can be found in Appendix A.

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

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

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

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

3 The hierarchy of 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:

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

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

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

We can define equivalence and refinement in the model $\mathcal{M}_p$ by:

$$P \equiv_p Q \;\Leftrightarrow\; traces(P) = traces(Q) \wedge failures_p(P) = failures_p(Q),$$
$$P \sqsubseteq_p Q \;\Leftrightarrow\; traces(Q) \subseteq traces(P) \wedge failures_p(Q) \subseteq failures_p(P).$$

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

Stable failures model

Roscoe's stable failures model ($\mathcal{F}$) [18] 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 $p(X) \mathrel{\widehat{=}} \text{true}$: 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 [5] 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 [6].

Traces model

The traces model ($\mathcal{T}$) [10,18] 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 $p(X) \mathrel{\widehat{=}} \text{false}$: equivalently, $\mathcal{T} = \mathcal{M}_{\lambda X \bullet \text{false}}$. The traces model may be used for reasoning about safety properties.

Singleton failures model
The singleton failures model ($\mathcal{S}$) [1,2,19] 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 $p(X) \mathrel{\widehat{=}} \#X \leq 1$: equivalently, $\mathcal{S} = \mathcal{M}_{\lambda X \bullet \#X \leq 1}$. This model was defined to coincide with the relational semantics of data types [7]: the refinement of data types is equivalent to the singleton failures refinement of their corresponding processes.

Completed trace model
As well as recording all trace information, the completed trace model ($\mathcal{CT}$) [19] 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 $p(X) \mathrel{\widehat{=}} X = \Sigma$ is the predicate that generates the completed trace model: equivalently, $\mathcal{CT} = \mathcal{M}_{\lambda X \bullet X = \Sigma}$. The completed trace model may be used for reasoning about safety and deadlocking properties.

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—and below we identify three predicates that yield potentially interesting or useful models.

Stable traces model
The model generated by the predicate $p(X) \mathrel{\widehat{=}} X = \{\}$, which we will refer to as the *stable traces* model ($\mathcal{ST} = \mathcal{M}_{\lambda X \bullet X = \{\}}$), merits 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 [13] and Reed's untimed stability model [16], does not record the unavailability of events, but does distinguish between deadlock and divergence:

$$\mathcal{ST}\,[\![\mathit{Stop}]\!] = \{\{\langle\rangle\}, \{(\langle\rangle, \{\})\}\}\ , \qquad \mathcal{ST}\,[\![\,\mathsf{div}\,]\!] = \{\{\langle\rangle\}, \{\}\}\ .$$

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

Bounded refusals model for N
Another interesting model is that generated by the predicate $p(X) \mathrel{\widehat{=}} \#X \leq N$ for any integer N such that $0 \leqslant N \leqslant \#\Sigma$. We will refer to this as the *bounded refusals* model for N ($\mathcal{BR}_N = \mathcal{M}_{\lambda X \bullet \#X \leq N}$). 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 Σ) $N = \#\Sigma$, this yields respectively the stable traces, the singleton failures, and the stable

failures models. A potential application for the bounded refusals model is for determining whether N processors within a distributed system of $M > N$ processors could fail.

Restricted refusals model for A
A final potentially useful model within the failures class is that generated by the predicate $p(X) \mathrel{\widehat{=}} X \subseteq A$ for any A such that $\{\} \subseteq A \subseteq \Sigma$. We will refer to this as the *restricted refusals* model for A ($\mathcal{RR}_A = \mathcal{M}_{\lambda X \bullet X \subseteq A}$). 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.

The following two results consider the relative expressiveness of the models in our class.

Theorem 3.1 *The failures class of semantic models forms a complete lattice, ordered according to implication of the corresponding predicates, with top element the stable failures model ($\mathcal{F} = \mathcal{M}_{\lambda X \bullet \mathrm{true}}$), and bottom element the traces model ($\mathcal{T} = \mathcal{M}_{\lambda X \bullet \mathrm{false}}$).*

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

Theorem 3.2 *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 $traces(P_{pq}) = traces(Q_{pq})$ and $failures(P_{pq}) = failures(Q_{pq}) \cup \{(\langle\rangle, X_{pq})\}$ so that $P_{pq} \equiv_p Q_{pq}$ but $P_{pq} \not\equiv_q Q_{pq}$. In the case where X_{pq} is non-empty this is true of*

$$P_{pq} \mathrel{\widehat{=}} \mathit{Offer}(\Sigma) \sqcap \mathit{Offer}(\Sigma - X_{pq}),$$
$$Q_{pq} \mathrel{\widehat{=}} \mathit{Offer}(\Sigma) \sqcap (\textstyle\bigsqcap Y : \mathbb{P}\,\Sigma \mid Y \subset X_{pq} \bullet \mathit{Offer}(\Sigma - Y)).$$

Where $X_{pq} = \emptyset$, it is true of $P_{pq} \mathrel{\widehat{=}} \mathit{Offer}(\Sigma)$ and $Q_{pq} \mathrel{\widehat{=}} \mathit{Offer}(\Sigma) \,\square\, \mathsf{div}$.

It follows directly from Theorems 3.1 and 3.2 that for processes with finite alphabet Σ, the relative strengths of the models identified in this section are as illustrated in Figure 2. Observe that the restricted refusals models, $\{\mathcal{RR}_A \mid A \in \mathbb{P}\,\Sigma\}$, form a complete sub-lattice, with the stable failures model ($\mathcal{F} = \mathcal{RR}_\Sigma$) as top element, and the stable traces model ($\mathcal{ST} = \mathcal{RR}_{\{\}}$) as bottom element.

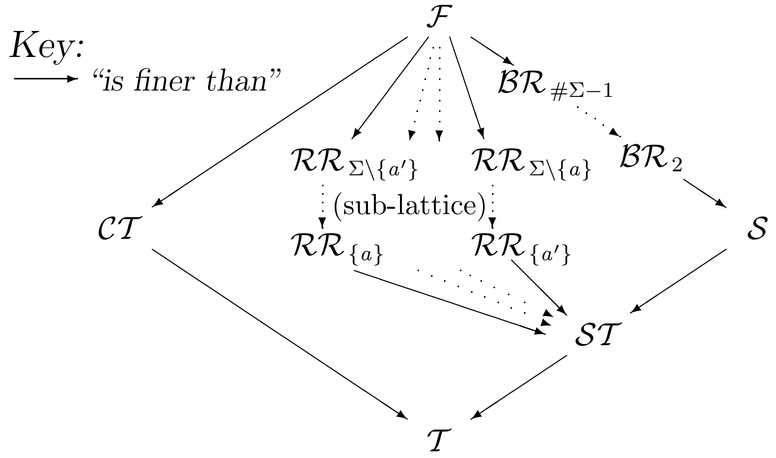

Fig. 2. Hierarchy of models (where $a, a' \in \Sigma$ and $a \neq a'$).

4 For which operators is $\mathcal{M}_p$ a congruence?

In this section we consider which models in our class are congruences with respect to which operators, identifing the maximal subset of the language for which each model induces a conguence and hence for which its semantics is compositional. A semantic model $\mathcal{M}$ is a congruence with respect to unary operator F if we may express $\mathcal{M}\,[\![F(P)]\!]$ in terms of $\mathcal{M}\,[\![P]\!]$; and it is a congruence with respect to binary operator $\oplus$ if we may express $\mathcal{M}\,[\![P \oplus Q]\!]$ in terms of $\mathcal{M}\,[\![P]\!]$ and $\mathcal{M}\,[\![Q]\!]$.

All the models associate the same traces with a given process, and the semantic equations for *traces* in Appendix A 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 congruences with respect to the operators $\rightarrow$, $\sqcap$, $\Box$, $\triangle$ and $|||$.

Lemma 4.1 *For every predicate p, the model $\mathcal{M}_p$ defines a congruence on the subset of the language containing the operators $\rightarrow$, $\sqcap$, $\Box$, $\triangle$ and $|||$.*

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 a congruence 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 in Lemma 4.2 the "only if" result for which we include the proof. The simpler proof for Lemma 4.3 is included in Appendix B.

Lemma 4.2 *Semantic model $\mathcal{M}_p$ defines a congruence 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*

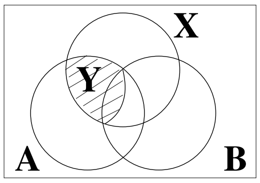

Fig. 3. $X \cap (A - B) \subseteq Y \subseteq X \cap A$

that $P \equiv_p Q$ *but* $P \setminus A \not\equiv_p Q \setminus A$. *Let*

$$P \mathrel{\widehat{=}} \textstyle\bigsqcap Y : \mathbb{P}\Sigma \mid Y \subseteq X \cup A \bullet \mathit{Offer}(\Sigma - Y),$$

$$Q \mathrel{\widehat{=}} \textstyle\bigsqcap Y : \mathbb{P}\Sigma \mid Y \subset X \cup A \bullet \mathit{Offer}(\Sigma - Y).$$

Then

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

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*. However* $(\langle\rangle, X) \in \mathit{failures}(P \setminus A) - \mathit{failures}(Q \setminus A)$*, so* $P \setminus A \not\equiv_p Q \setminus A$*, as required.*

We now prove the converse result.

Lemma 4.3 *If* $p(X) \Rightarrow p(X \cup A)$ *for all sets* X*, then for all processes* P*, the set* $\mathit{failures}_p(P \setminus A)$ *is expressible in terms of* $\mathit{failures}_p(P)$*.*

Corollary 4.4 *Model* $\mathcal{M}_p$ *defines a congruence 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 a congruence with respect to the parallel composition $\_\,{}_A\|_B\,\_$ if and only if

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

The relationship $X \cap (A - B) \subseteq Y \subseteq X \cap A$ is illustrated in Figure 3. We begin by proving in Lemma 4.5 the "only if" result for which we include the proof. The simpler proof for Lemma 4.6 is included in the appendix.

If X is a refusal of $P\,{}_A\|_B\,Q$ then the corresponding refusals Y of P and Z of Q, as well as satisfying the predicate $Y \cup Z = X \cap (A \cup B)$, 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.

Lemma 4.5 *Semantic model* $\mathcal{M}_p$ *defines a congruence on a subset of the language containing the parallel operator* $\_\,{}_A\|_B\,\_$ *only if conditions (1) and (2) hold.*

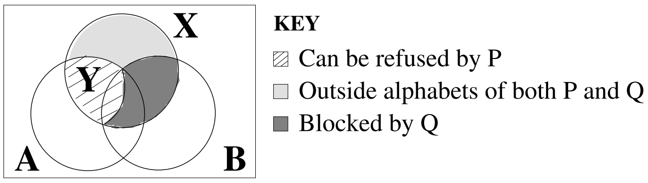

Fig. 4. Process $P\ {}_A\|_B\ Q$ can initially refuse the whole of X.

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* $P\ {}_A\|_B\ Q \not\equiv_p P'\ {}_A\|_B\ Q$ *as follows:*

$$P \mathrel{\widehat{=}} \textstyle\bigsqcap Z : \mathbb{P}\,\Sigma \mid Z \subseteq Y \bullet \mathit{Offer}(\Sigma - Z),$$
$$P' \mathrel{\widehat{=}} \textstyle\bigsqcap Z : \mathbb{P}\,\Sigma \mid Z \subset Y \bullet \mathit{Offer}(\Sigma - Z),$$
$$Q \mathrel{\widehat{=}} \mathit{Offer}(B \cap Y).$$

Then

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

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 Figure 4, the process $P\ {}_A\|_B\ Q$ *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* $P'\ {}_A\|_B\ Q$ *cannot initially refuse the whole of* X*. We see that*

$$(\langle\rangle, X) \in \mathit{failures}(P\ {}_A\|_B\ Q) - \mathit{failures}(P'\ {}_A\|_B\ Q).$$

Hence, since $p(X)$ *is true, we conclude, as required, that* $P\,{}_A\|_B\,Q \not\equiv_p P'\,{}_A\|_B\,Q$.

We now prove the converse result.

Lemma 4.6 *If conditions (1) and (2) hold, then for all processes* P *and* Q*, the set* $\mathit{failures}_p(P\ {}_A\|_B\ Q)$ *is expressible in terms of the sets* $\mathit{failures}_p(P)$ *and* $\mathit{failures}_p(Q)$.

Corollary 4.7 *Model* $\mathcal{M}_p$ *defines a congruence with respect to parallel composition with arbitrary interface sets precisely when predicate* p *is downwards closed, i.e.:*

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

4.3 Summary

In Theorem 4.8 below we apply the results establised in Lemma 4.1, and Corollaries 4.4 and 4.7 to identify for each subset of the language the constraints on predicate p that ensure model $\mathcal{M}_p$ is a congruence. Theorems 4.9 and 4.10, in which we consider specific models, follow directly from Theorem 4.8.

Theorem 4.8 *Model $\mathcal{M}_p$ defined on a language with operators*

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

is a congruence precisely when the following two predicates hold:

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

Theorem 4.9 *Both $\mathcal{T}$ and $\mathcal{F}$ define a congruence upon the whole language introduced in this paper. Moreover they are the* only *models within the class that satisfy this property.*

Theorem 4.10 *Of the other specific models we have considered:*

- *The Singleton Failures Model, the Stable Traces Model, and the Bounded Refusals Models are congruences with respect to all the operators except for hiding.*
- *The Restricted Refusals Model for A ($\mathcal{RR}_A$) is a congruence with respect to all the operators except for hiding of B for $B \not\subseteq A$.*
- *The Completed Traces Model is a congruence with respect to all the operators except for parallel composition.*

5 Automatic analysis

FDR [17,9] 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 show first that such an encoding is possible for all models associated with predicates that are either upwards or downwards closed, as is the case with all the specific models we have considered in this paper. Then, in Section 5.3 we prove a corresponding result for the general case and discuss the practicalities of our rule. The proofs are included in the appendix.

5.1 Downwards closed predicates

To prove, for any predicate p that is downwards closed, that refinement within model $\mathcal{M}_p$ may be expressed in terms of refinement within $\mathcal{T}$ and $\mathcal{F}$, we introduce the process R_p that can initially refuse any set Y such that $p(Y)$ is *true*, and that diverges after any event is performed:

$$R_p \mathrel{\widehat=} \bigsqcap Y : \mathbb{P}\,\Sigma \mid p(Y) \bullet \mathit{Offer}(\Sigma - Y).$$

We observe that

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

The interleaving of process R_p with any process P then yields a process whose stable failures are equal to $\mathit{failures}_p(P)$.

Theorem 5.1 *Suppose predicate p is downwards closed, and $p \neq \lambda\, X \bullet \mathit{false}$. Then*

$$P \sqsubseteq_p Q \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge P \mathbin{|||} R_p \sqsubseteq_{\mathcal{F}} Q \mathbin{|||} R_p.$$

where R_p is as defined above.

5.2 Upwards closed predicates

To prove, for any predicate p that is upwards closed, that refinement within model $\mathcal{M}_p$ may be expressed in terms of refinement within $\mathcal{T}$ and $\mathcal{F}$, we introduce the process S_p that can initially refuse any set Y such that $p(Y)$ is *false*, and that diverges after any event is performed:

$$S_p \mathrel{\widehat=} \bigsqcap Y : \mathbb{P}\,\Sigma \mid \neg\, p(Y) \bullet \mathit{Offer}(\Sigma - Y).$$

We observe that

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

For convenience, we define a new syntactic operator, the non-deterministic interrupt operator: $P \rightsquigarrow S \mathrel{\widehat=} P \sqcap (P \bigtriangleup S)$. Process S non-deterministically may or may not be able to interrupt process P. Note that

$$traces(P \rightsquigarrow S) = traces(P) \cup \{tr \frown tr' \mid tr \in traces(P) \wedge tr' \in traces(S)\},$$

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

The effect of non-deterministically interrupting any process P with the process S_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.

Theorem 5.2 *Suppose predicate p is upwards closed, and $p \neq \lambda X \bullet true$. Then*

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

where S_p is as defined above.

5.3 General predicates

In this section we identify, for arbitrary predicate p, a rule for expressing refinement within model $\mathcal{M}_p$ in terms of refinement within $\mathcal{T}$ and $\mathcal{F}$. We begin by considering refinement within 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 5.1. Hence we consider only the cases where $A \neq \emptyset$.

We use techniques similar to those applied in Sections 5.1 and 5.2: we interleave with the process $Offer(\Sigma - A)$ to remove all failures in which the refusal set is not a subset of A, and then non-deterministically interrupt with the process $\sqcap Y \mid Y \subset A \bullet Offer(\Sigma - Y)$ thereby adding failures in which the refusal set is a strict subset of A.

Lemma 5.3 *Given any processes P and Q and any set of events $A \neq \emptyset$,*

$$P \sqsubseteq_{\lambda X \bullet X = A} Q \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \wedge ((P \mid\mid\mid T_A) \rightsquigarrow U_A) \sqsubseteq_{\mathcal{F}} ((Q \mid\mid\mid T_A) \rightsquigarrow U_A)$$

where $T_A = Offer(\Sigma - A)$ and $U_A = \sqcap Y : \mathbb{P}\Sigma \mid Y \subset A \bullet Offer(\Sigma - Y)$.

We now prove 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 5.4 *Given any processes P and Q and predicates p and q,*

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

It follows immediately from the previous two results, and the associativity of $\wedge$ and $\vee$ that $\sqsubseteq_p$ is checkable for any predicate p.

Theorem 5.5 *For any predicate p over a finite alphabet Σ we may express refinement within model $\mathcal{M}_p$ in terms of refinement within $\mathcal{T}$ and $\mathcal{F}$. In* *particular,*

$$
\begin{aligned}
&P \sqsubseteq_p Q \\
&\qquad \Leftrightarrow P \sqsubseteq_{\mathcal{T}} Q \;\wedge \\
&\qquad\quad \textstyle\bigwedge_{X_i \in \mathbb{P}\,\Sigma \mid p(X_i)} \\
&\qquad\qquad (P \mid\mid\mid \mathit{Offer}(\Sigma - X_i)) \rightsquigarrow (\textstyle\bigsqcap Y \mid Y \subset X_i \bullet \mathit{Offer}(\Sigma - Y)) \\
&\qquad\qquad\quad \sqsubseteq_{\mathcal{F}} \\
&\qquad\qquad\quad (Q \mid\mid\mid \mathit{Offer}(\Sigma - X_i)) \rightsquigarrow (\textstyle\bigsqcap Y \mid Y \subset X_i \bullet \mathit{Offer}(\Sigma - Y)).
\end{aligned}
$$

In Sections 5.1 and 5.2 we showed that verification of refinement in model $\mathcal{M}_p$ for upwards or downwards closed predicate p is relatively straightforward requiring only one refinement check in each of $\mathcal{T}$ and $\mathcal{F}$. However, we observe from Theorem 5.5 above, that such a refinement is not so straightforward in general. Indeed, in the worst case—when there are no subsets of $\{X \in \mathbb{P}\,\Sigma \mid p(X)\}$ that are either upwards or downwards closed over Σ—verifying refinement in model $\mathcal{M}_p$ will require $O(\#\{X \in \mathbb{P}\,\Sigma \mid p(X)\})$ refinement checks.

6 Discussion

In this paper we have 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.

We discussed in detail four established models that are members of this class: Roscoe's traces and stable failures model [18]; Bolton's singleton failures model [1,2]; and van Glabbeek's completed trace model [19]. We examined also three non-established models within this family. Having proved that the failures class forms a complete lattice we identified the position within the lattice of each of these models, thereby exposing their relative strengths.

For each model we identified the maximal sublanguage over which the model induces a congruence, verifying that only the traces and the stable failures semantics are fully compositional. Finally, by expressing such a refinement in terms of refinement within the traces and stable failures models, we presented techniques for using the model-checker FDR [17,9] to verify refinement within any model in this class.

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 [19], 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.

A related paper [3] extends and explores practical applications of the work described here. Non-standard measures of consistency are identified and motivated. Verification of consistency within each such measure can be seen as a refinement in one of the models of the current paper and hence can be performed automatically using existing tools.

References

[1] C. Bolton. *On the Refinement of State-Based and Event-Based Models*. D.Phil., University of Oxford, 2002.

[2] C. Bolton and J. Davies. A singleton failures semantics for communicating sequential processes, 2001. Submitted to Formal Aspects of Computing.

[3] C. Bolton and G. Lowe. On the automatic verification of non-standard measures of consistency. In J.M. Morris, editor, *Proceedings of the International Workshop on Formal Methods*, Electronic Workshops in Computing (eWiC), 2003.

[4] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronisation skeletons using branching-time temporal logic. In *Proceedings of the International Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.

[5] R. de Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24, 1987.

[6] R. de Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34, 1984.

[7] W.-P. de Roever and K. Engelhardt. *Data refinment: model-oriented proof methods and their comparison*. Cambridge Tracts in Theoretical Computer Science, 1998.

[8] E. A. Emerson and J. Y. Halpern. Sometimes and not never revisited: On branching versus linear time. *Journal of the ACM*, 33(1):151–178, 1986.

[9] Formal Systems (Europe) Ltd. *Failures-Divergence Refinement—FDR 2 User Manual*, 1999. Available via URL `http://www.fsel.com/fdr2_manual.html`.

[10] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[11] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[12] R. Milner. *Communications and concurrency*. Prentice Hall, 1989.

[13] E.-R. Olderog and C. A. R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23, 1986.

[14] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall International, 1981.

[15] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th International Smposium on Foundations of Computer Science*, pages 46–57, 1977.

[16] G. M. Reed. *A uniform mathematical theory for real-time distributed computing.* PhD thesis, University of Oxford, 1988.

[17] A. W. Roscoe. Model-checking CSP. In *A Classical Mind, Essays in Honour of C. A. R. Hoare.* Prentice-Hall, 1994.

[18] A. W. Roscoe. *The Theory and Practice of Concurrency.* Prentice Hall, 1997.

[19] R. J. van Glabbeek. The linear time – branching time spectrum I: the semantics of concrete, sequential processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra.* Elsevier, 2001.

A Semantic equations

The functions *traces* and *failures* satisfy the following equations[3]:

$$\begin{aligned}
traces(Stop) &= \{\langle\rangle\},\\
failures(Stop) &= \{(\langle\rangle, X) \mid X \in \mathbb{P}\,\Sigma\},\\[4pt]
traces(\mathsf{div}) &= \{\langle\rangle\},\\
failures(\mathsf{div}) &= \{\},\\[4pt]
traces(a \to P) &= \{\langle\rangle\} \cup \{\langle a\rangle \frown tr \mid tr \in traces(P)\},\\
failures(a \to P) &= \{(\langle\rangle, X) \mid X \in \mathbb{P}\,\Sigma \wedge a \notin X\} \;\cup\\
&\qquad \{(\langle a\rangle \frown tr, X) \mid (tr, X) \in failures(P)\},\\[4pt]
traces(?a : A \to P_a) &= \{\langle\rangle\} \cup \{\langle a\rangle \frown tr \mid a \in A \wedge tr \in traces(P_a)\},\\
failures(?a : A \to P_a) &= \{(\langle\rangle, X) \mid X \in \mathbb{P}\,\Sigma \wedge A \cap X = \{\}\} \cup\\
&\qquad \{(\langle a\rangle \frown tr, X) \mid a \in A \wedge (tr, X) \in failures(P_a)\},\\[4pt]
traces(Offer(A)) &= \{\langle\rangle\} \cup \{\langle a\rangle \mid a \in A\},\\
failures(Offer(A)) &= \{(\langle\rangle, X) \mid X \in \mathbb{P}\,\Sigma \wedge A \cap X = \{\}\},\\[4pt]
traces(P \;\Box\; Q) &= traces(P) \cup traces(Q),\\
failures(P \;\Box\; Q) &= \{(\langle\rangle, X) \in failures(P) \cap failures(Q)\} \cup\\
&\qquad \{(tr, X) \in failures(P) \cup failures(Q) \mid tr \neq \langle\rangle\},
\end{aligned}$$

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

$$traces(P \sqcap Q) = traces(P) \cup traces(Q),$$
$$failures(P \sqcap Q) = failures(P) \cup failures(Q),$$

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

$$traces(P \bigtriangleup Q) = \{tr \frown tr' \mid tr \in traces(P) \wedge tr' \in traces(Q)\},$$
$$failures(P \bigtriangleup Q) = \{(tr \frown tr', X) \mid tr \in traces(P) \wedge (tr', X) \in failures(Q)\},$$

$$traces(P \mathbin{{}_A\|_B} Q) = \{tr \in (A \cup B)^* \mid tr \restriction A \in traces(P) \wedge tr \restriction B \in traces(Q)\},$$

$$\begin{aligned} failures(P \mathbin{{}_A\|_B} Q) = \{(tr, X) \in (A \cup B)^* \times \mathbb{P}\,\Sigma \mid \\ \exists\, Y \in \mathbb{P}\,A;\ Z \in \mathbb{P}\,B;\ W \in \mathbb{P}(\Sigma - A - B) \bullet \\ (tr \restriction A, Y) \in failures(P) \wedge \\ (tr \restriction B, Z) \in failures(Q) \wedge \\ X = Y \cup Z \cup W\}, \end{aligned}$$

$$traces(P \mathbin{|||} Q) = \{tr \mid \exists\, tr_P \in traces(P), tr_Q \in traces(Q) \bullet tr \in tr_P \mathbin{|||} tr_Q\},$$

$$\begin{aligned} failures(P \mathbin{|||} Q) = \{(tr, X) \mid \exists\, tr_P, tr_Q \bullet (tr_P, X) \in failures(P) \wedge \\ (tr_Q, X) \in failures(Q) \wedge \\ tr \in tr_P \mathbin{|||} tr_Q\}, \end{aligned}$$

$$traces(P \setminus A) = \{\, tr \setminus A \mid tr \in traces(P)\,\},$$
$$failures(P \setminus A) = \{(tr \setminus A, X) \mid (tr, A \cup X) \in failures(P)\}.$$

In the equations for $P \mathbin{|||} Q$, the notation $tr_P \mathbin{|||} tr_Q$ represents all ways of interleaving the traces tr_P and tr_Q; see [18].

B Proofs

Proof of Lemma 4.1

For any predicate p the following all hold:

$$\begin{aligned} failures_p(a \to P) = \{\, (\langle\rangle, X) \mid p(X) \wedge a \notin X \,\} \cup \\ \{\, (\langle a \rangle \frown tr, X) \mid (tr, X) \in failures_p(P) \,\}, \end{aligned}$$

$$\begin{aligned}
\mathit{failures}_p(?a : A \to P_a) &= \{\, (\langle\rangle, X) \mid p(X) \wedge A \cap X = \{\} \,\} \cup \\
&\quad \{\, (\langle a \rangle \frown tr, X) \mid a \in A \wedge (tr, X) \in \mathit{failures}_p(P_a) \,\},
\end{aligned}$$

$$\mathit{failures}_p(P \sqcap Q) = \mathit{failures}_p(P) \cup \mathit{failures}_p(Q),$$

$$\begin{aligned}
\mathit{failures}_p(P \mathbin{\Box} Q) &= \{\, (\langle\rangle, X) \in \mathit{failures}_p(P) \cap \mathit{failures}_p(Q) \,\} \\
&\quad \cup \\
&\quad \{\, (tr, X) \in \mathit{failures}_p(P) \cup \mathit{failures}_p(Q) \mid tr \neq \langle\rangle \,\},
\end{aligned}$$

$$\begin{aligned}
\mathit{failures}_p(P \bigtriangleup Q) &= \{(tr \frown tr', X) \mid tr \in \mathit{traces}(P) \wedge \\
&\qquad\qquad (tr', X) \in \mathit{failures}_p(Q)\},
\end{aligned}$$

$$\begin{aligned}
\mathit{failures}_p(P \mathbin{|||} Q) &= \{(tr, X) \mid \exists\, tr_P, tr_Q \bullet (tr_P, X) \in \mathit{failures}_p(P) \wedge \\
&\qquad\qquad (tr_Q, X) \in \mathit{failures}_p(Q) \wedge \\
&\qquad\qquad tr \in tr_P \mathbin{|||} tr_Q\}.
\end{aligned}$$

Consider first the case of the external choice operator:

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

The proofs of the other results follow similar lines.

Proof of Lemma 4.3

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

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

Proof of Lemma 4.6

We reason as follows:

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

$$
\begin{aligned}
= \{ (tr, X) \in (A \cup B)^* \times \mathbb{P}\,\Sigma \mid p(X) \ \wedge & \qquad\qquad [\text{def}^n \text{ of } \mathit{failures}_p] \\
\exists\, Y \in \mathbb{P}\,A;\ Z \in \mathbb{P}\,B;\ W \in \mathbb{P}(\Sigma - A - B) \bullet & \\
X = Y \cup Z \cup W \ \wedge & \\
(tr \upharpoonright A, Y) \in \mathit{failures}_p(P) \ \wedge & \\
(tr \upharpoonright B, Z) \in \mathit{failures}_p(Q) \}. &
\end{aligned}
$$

Proof of Corollary 4.7

We must show the above condition (3) is equivalent to the conjunctions of conditions (1) and (2) for all A and B. Firstly, taking $A = B = \Sigma$ in condition (1) gives condition (3). Conversely, assume condition (3) and suppose $X \cap (A - B) \subseteq Y \subseteq X \cap A$; then $Y \subseteq X$, so $p(X) \Rightarrow p(Y)$.

Proof of Theorem 4.9

Theorem 4.8 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.

Proof of Theorem 5.1

Note that for process R_p with traces and failures as identified in Section 5.1:

$$\mathit{traces}(P \mathbin{|||} R_p) = \{tr \mid \exists\, tr' \in \mathit{traces}(P), a \in \Sigma \bullet tr \in tr' \mathbin{|||} \langle a \rangle\},$$

and similarly for $Q \mathbin{|||} R_p$; in particular

$$\mathit{traces}(Q) \subseteq \mathit{traces}(P) \Rightarrow \mathit{traces}(Q \mathbin{|||} R_p) \subseteq \mathit{traces}(P \mathbin{|||} R_p).$$

Further

$$
\begin{aligned}
&\mathit{failures}(P \mathbin{|||} R_p) \\
&\quad = \{(tr, X) \mid (tr, X) \in \mathit{failures}(P) \wedge (\langle\rangle, X) \in \mathit{failures}(R_p)\} \\
&\quad = \{(tr, X) \mid (tr, X) \in \mathit{failures}(P) \wedge p(X)\} \\
&\quad = \mathit{failures}_p(P),
\end{aligned}
$$

and similarly for $Q \mathbin{|||} R_p$. Hence

$$
\begin{aligned}
&P \sqsubseteq_{\mathcal{T}} Q \wedge P \mathbin{|||} R_p \sqsubseteq_{\mathcal{F}} Q \mathbin{|||} R_p \\
&\quad \Leftrightarrow\ \mathit{traces}(Q) \subseteq \mathit{traces}(P) \ \wedge \qquad\qquad [\text{def}^n \text{ of } \sqsubseteq_{\mathcal{T}} \text{ and } \sqsubseteq_{\mathcal{F}}] \\
&\qquad\quad \mathit{traces}(Q \mathbin{|||} R_p) \subseteq \mathit{traces}(P \mathbin{|||} R_p) \ \wedge \\
&\qquad\quad \mathit{failures}(Q \mathbin{|||} R_p) \subseteq \mathit{failures}(P \mathbin{|||} R_p)
\end{aligned}
$$

$$\Leftrightarrow\ traces(Q) \subseteq traces(P) \wedge failures_p(Q) \subseteq failures_p(P) \qquad \text{[above results]}$$

$$\Leftrightarrow\ P \sqsubseteq_p Q. \qquad [\text{def}^n \text{ of } \sqsubseteq_p]$$

Proof of Theorem 5.2

Note that for process S_p with traces and failures as identified in Section 5.2:
(B.1)$traces(Q) \subseteq traces(P) \Rightarrow traces(Q \rightsquigarrow S_p) \subseteq traces(P \rightsquigarrow S_p)$.
Further:

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

and similarly for $Q \rightsquigarrow S_p$. Hence

$$P \rightsquigarrow S_p \sqsubseteq_{\mathcal{F}} Q \rightsquigarrow S_p$$

$$\begin{aligned} \Leftrightarrow\ & traces(Q \rightsquigarrow S_p) \subseteq traces(P \rightsquigarrow S_p) \wedge && [\text{def}^n \text{ of } \sqsubseteq_{\mathcal{F}}] \\ & failures(Q \rightsquigarrow S_p) \subseteq failures(P \rightsquigarrow S_p) \end{aligned}$$

$$\begin{aligned} \Leftrightarrow\ & traces(Q \rightsquigarrow S_p) \subseteq traces(P \rightsquigarrow S_p) \wedge && [\text{above result}] \\ & failures(Q) \cup \{(tr, X) \mid tr \in traces(Q) \wedge \neg\, p(X)\} \subseteq \\ & \qquad failures(P) \cup \{(tr, X) \mid tr \in traces(P) \wedge \neg\, p(X)\} \end{aligned}$$

$$\begin{aligned} \Leftrightarrow\ & traces(Q \rightsquigarrow S_p) \subseteq traces(P \rightsquigarrow S_p) \wedge && [\text{for all } X,\ p(X) \text{ or } \neg\, p(X)] \\ & \{(tr, X) \in failures(Q) \mid p(X)\} \cup \\ & \qquad \{(tr, X) \in failures(Q) \mid \neg\, p(X)\} \cup \\ & \qquad \{(tr, X) \mid tr \in traces(Q) \wedge \neg\, p(X)\} \\ & \subseteq \\ & \{(tr, X) \in failures(P) \mid p(X)\} \cup \\ & \qquad \{(tr, X) \in failures(P) \mid \neg\, p(X)\} \cup \\ & \qquad \{(tr, X) \mid tr \in traces(P) \wedge \neg\, p(X)\} \end{aligned}$$

$$\begin{aligned} \Leftrightarrow\ & && [\text{def}^n \text{ of } failures_p;\ \text{condition (F1)}] \\ & traces(Q \rightsquigarrow S_p) \subseteq traces(P \rightsquigarrow S_p) \wedge \\ & failures_p(Q) \subseteq failures_p(P) \wedge \\ & \{(tr, X) \mid tr \in traces(Q) \wedge \neg\, p(X)\} \subseteq \\ & \qquad \{(tr, X) \mid tr \in traces(P) \wedge \neg\, p(X)\} \end{aligned}$$

$$\Leftrightarrow \ \mathit{traces}(Q \rightsquigarrow S_p) \subseteq \mathit{traces}(P \rightsquigarrow S_p) \wedge \qquad [p \text{ is not identically } \mathit{true}]$$
$$\mathit{failures}_p(Q) \subseteq \mathit{failures}_p(P) \wedge$$
$$\mathit{traces}(Q) \subseteq \mathit{traces}(P) \wedge$$

$$\Leftrightarrow \ \mathit{failures}_p(Q) \subseteq \mathit{failures}_p(P) \wedge \mathit{traces}(Q) \subseteq \mathit{traces}(P) \qquad [\text{equation (B.1)}]$$

$$\Leftrightarrow \ P \sqsubseteq_p Q. \qquad [\text{def}^n \text{ of } \sqsubseteq_p]$$

Proof of Lemma 5.3

Observe that since $A \neq \emptyset$,

$$\mathit{failures}(U_A) = \{(\langle\rangle, X) \mid X \subset A\},$$
$$\mathit{failures}(T_A) = \{(\langle\rangle, X) \mid X \subseteq A\},$$
$$\begin{aligned}\mathit{failures}(P \mid\mid\mid T_A) &= \{(\mathit{tr}, X) \mid (\mathit{tr}, X) \in \mathit{failures}(P) \wedge (\langle\rangle, X \in \mathit{failures}(T_A)\} \\ &= \{(\mathit{tr}, X) \in \mathit{failures}(P) \mid X \subseteq A\}, \end{aligned} \tag{B.2}$$

and similarly for $\mathit{failures}(Q \mid\mid\mid T_A)$.

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 [18]. Given processes P and Q and set of events $A \neq \emptyset$, we reason as follows:

$$P \sqsubseteq_{\mathcal{T}} Q \wedge (P \mid\mid\mid T_A) \rightsquigarrow U_A \sqsubseteq_{\mathcal{F}} (Q \mid\mid\mid T_A) \rightsquigarrow U_A$$

$$\Leftrightarrow \qquad [\text{def}^n\text{s of } \sqsubseteq_{\mathcal{T}} \text{ and } \sqsubseteq_{\mathcal{F}}]$$
$$\mathit{traces}(Q) \subseteq \mathit{traces}(P) \ \wedge$$
$$\mathit{traces}((Q \mid\mid\mid T_A) \rightsquigarrow U_A) \subseteq \mathit{traces}((P \mid\mid\mid T_A) \rightsquigarrow U_A) \ \wedge$$
$$\mathit{failures}((Q \mid\mid\mid T_A) \rightsquigarrow U_A) \subseteq \mathit{failures}((P \mid\mid\mid T_A) \rightsquigarrow U_A)$$

$$\Leftrightarrow \qquad [\text{monotonicity}]$$
$$\mathit{traces}(Q) \subseteq \mathit{traces}(P) \ \wedge$$
$$\mathit{failures}((Q \mid\mid\mid T_A) \rightsquigarrow U_A) \subseteq \mathit{failures}((P \mid\mid\mid T_A) \rightsquigarrow U_A)$$

$$\Leftrightarrow \qquad [\text{def}^n \text{ of } \rightsquigarrow]$$
$$\mathit{traces}(Q) \subseteq \mathit{traces}(P) \ \wedge$$
$$\mathit{failures}(Q \mid\mid\mid T_A) \cup$$
$$\{(\mathit{tr} \frown \mathit{tr}', X) \mid \mathit{tr} \in \mathit{traces}(Q \mid\mid\mid T_A) \wedge (\mathit{tr}', X) \in \mathit{failures}(U_A)\}$$
$$\subseteq$$
$$\mathit{failures}(P \mid\mid\mid T_A) \cup$$
$$\{(\mathit{tr} \frown \mathit{tr}', X) \mid \mathit{tr} \in \mathit{traces}(P \mid\mid\mid T_A) \wedge (\mathit{tr}', X) \in \mathit{failures}(U_A)\}$$

$\Leftrightarrow$ [one point rule and def^n of U_A]

$$traces(Q) \subseteq traces(P) \;\wedge$$
$$failures(Q \mathbin{|||} T_A) \cup \{(tr, X) \mid tr \in traces(Q \mathbin{|||} T_A) \wedge X \subset A\}$$
$$\subseteq failures(P \mathbin{|||} T_A) \cup \{(tr, X) \mid tr \in traces(P \mathbin{|||} T_A) \wedge X \subset A\}$$

$\Leftrightarrow$ [set theory and condition F1]

$$traces(Q) \subseteq traces(P) \;\wedge$$
$$\{(tr, X) \in failures(Q \mathbin{|||} T_A) \mid X \not\subset A\}$$
$$\subseteq \{(tr, X) \in failures(P \mathbin{|||} T_A) \mid X \not\subset A\} \;\wedge$$
$$\{(tr, X) \mid tr \in traces(Q \mathbin{|||} T_A) \wedge X \subset A\}$$
$$\subseteq \{(tr, X) \mid tr \in traces(P \mathbin{|||} T_A) \wedge X \subset A\}$$

$\Leftrightarrow$ [set theory and monotonicity]

$$traces(Q) \subseteq traces(P) \;\wedge$$
$$\{(tr, X) \in failures(Q \mathbin{|||} T_A) \mid X \not\subset A\}$$
$$\subseteq \{(tr, X) \in failures(P \mathbin{|||} T_A) \mid X \not\subset A\}$$

$\Leftrightarrow$ [equation (B.2)]

$$traces(Q) \subseteq traces(P) \;\wedge$$
$$\{(tr, X) \in failures(Q) \mid X \subseteq A \wedge X \not\subset A\}$$
$$\subseteq \{(tr, X) \in failures(P) \mid X \subseteq A \wedge X \not\subset A\}$$

$\Leftrightarrow$ [predicate calculus and set theory]

$$traces(Q) \subseteq traces(P) \;\wedge$$
$$\{(tr, X) \in failures(Q) \mid X = A\} \subseteq \{(tr, X) \in failures(P) \mid X = A\}$$

$\Leftrightarrow$ [def^n of $\sqsubseteq_p$]

$$P \sqsubseteq_{\lambda X \bullet X = A} Q.$$

Proof of Lemma 5.4

$P \sqsubseteq_p Q \wedge P \sqsubseteq_q Q$

$\Leftrightarrow$ $traces(P) \supseteq traces(Q) \wedge$ [def^n of refinement]

$$\{(tr, X) \in failures(P) \mid p(X)\} \supseteq \{(tr, X) \in failures(Q) \mid p(X)\} \wedge$$
$$\{(tr, X) \in failures(P) \mid q(X)\} \supseteq \{(tr, X) \in failures(Q) \mid q(X)\}$$

$$\Leftrightarrow\ traces(P) \supseteq traces(Q) \wedge \qquad \text{[set theory]}$$

$$\{(tr, X) \in failures(P) \mid p(X) \vee q(X)\} \supseteq$$

$$\{(tr, X) \in failures(Q) \mid p(X) \vee q(X)\}$$

$$\Leftrightarrow\ P \sqsubseteq_{p \vee q} Q. \qquad \text{[def}^n\text{ of refinement]}$$

Proof of Theorem 5.5

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

$$P \sqsubseteq_p Q$$

$$\Leftrightarrow\ \bigwedge_{X_i \in \mathbb{P}\,\Sigma \mid p(X_i)} P \sqsubseteq_{\lambda\, X \bullet X = X_i} Q \qquad \text{[Lemma 5.4 and associativity]}$$

$$\Leftrightarrow\ \bigwedge_{X_i \in \mathbb{P}\,\Sigma \mid p(X_i)} \qquad \text{[Lemma 5.3]}$$

$$P \sqsubseteq_{\mathcal{T}} Q\ \wedge$$

$$(P \mid\mid\mid \mathit{Offer}(\Sigma - X_i)) \leadsto (\textstyle\bigsqcap Y \mid Y \subset X_i \bullet \mathit{Offer}(\Sigma - Y))$$

$$\sqsubseteq_{\mathcal{F}}$$

$$(Q \mid\mid\mid \mathit{Offer}(\Sigma - X_i)) \leadsto (\textstyle\bigsqcap Y \mid Y \subset X_i \bullet \mathit{Offer}(\Sigma - Y))$$

$$\Leftrightarrow\ P \sqsubseteq_{\mathcal{T}} Q\ \wedge \qquad \text{[distributivity]}$$

$$\bigwedge_{X_i \in \mathbb{P}\,\Sigma \mid p(X_i)}$$

$$(P \mid\mid\mid \mathit{Offer}(\Sigma - X_i)) \leadsto (\textstyle\bigsqcap Y \mid Y \subset X_i \bullet \mathit{Offer}(\Sigma - Y))$$

$$\sqsubseteq_{\mathcal{F}}$$

$$(Q \mid\mid\mid \mathit{Offer}(\Sigma - X_i)) \leadsto (\textstyle\bigsqcap Y \mid Y \subset X_i \bullet \mathit{Offer}(\Sigma - Y)).$$