Revivals, stuckness and the hierarchy of CSP models

A.W. Roscoe*

December 9, 2007

Abstract

We give details of a new model for CSP introduced in response to work by Fournet *et al* [8]. This is the *stable revivals* model $\mathcal{R}$ alluded to in [22]. We provide the full semantics for CSP in this model, indicate why this is operationally congruent, and provide proofs of the full abstraction properties asserted in [22]. We study the place of $\mathcal{R}$ in the hierarchy of CSP models, and show how this generates several extensions of $\mathcal{R}$ handling infinite behaviours. In doing this we discover more about the hierarchy and several known models within it. This includes results that show that the traces model, failures model and are new one are somehow "essential" or "Platonic". We present the normal form of processes in the new model (an essential precursor to any automation of refinement checking) and discuss the surprisingly large implication this has for the algebraic semantics of CSP. We set out a number of conjectures and challenges for future workers in this area.

1 Introduction

The author has long worked on mathematical models for concurrent systems, in particular Hoare's CSP [13]. It therefore came as something as a surprise, when studying the work of Fournet, Hoare, Rehof and Rajamani in [8], for him to realise that there was a new congruence sitting squarely in the middle of the known ones. This is based on the idea that we might extend the familiar concept of a failure – a pair (s, X) where s is a trace and X is a set of events that might be refused indefinitely after s – by adding an event that the process might accept after this refusal. [8] introduced this via an equivalence on CCS processes called *conformance*. In [22] we showed, in a section of comparisons with [8], that this idea could be turned into a model for CSP. There we stated the healthiness conditions for the new model, re-christened with the more descriptive name *stable revivals*, but did not have the space to give a full semantics or to justify the full abstraction claims that were made.

The purpose of this paper is to make up for these omissions, to study further details of this model, and to re-examine the hierarchy of CSP models in the light of this new one. By "CSP" here, we mean the untimed process algebra described in [13] and [23], using the second of these as our primary reference. So we are studying models for that language, not the ones for its continuously or discretely timed variants.

The paper is organised as follows: in the first section we summarise the language of CSP and its the existing hierarchy of models. In the next we introduce the new model and give the semantics for CSP over it. We then establish its congruence with the standard

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

operational semantics of CSP and use that to state formally the full abstraction properties that this model has with respect to issues discussed in [21, 22, 8], and prove them. In essence it is fully abstract with respect to detecting when some system of processes can fail to make progress despite one or more of them having unfinished business with other(s), or revealing when a process can offer some event from a stable state. There is then a section on revivals models that include representations of divergence and perhaps infinite traces – behaviours that take an infinite time to observe – together with appropriate full abstraction properties. Turning to algebraic semantics, we then discuss normal forms for revivals models and show how it implies that the basic structure of the standard set of CSP laws needs to be generalised, as well as setting out a programme to develop an algebraic framework that should address all the model-driven equivalences. In Section 3 we are able to use the results and methods developed in this paper to prove that our new congruence has an important place in the CSP hierarchy. To be precise we show that all finite-observation congruences that are not finite traces, stable failures and the trivial congruence are refinements of the stable revivals model $\mathcal{R}$.

In the conclusions we discuss the role of additional equivalences like revivals, discussing their potential impact, both theoretical and practical.

In this paper we rely on the methods and notation of the author's book [23], where in Chapters 7–11 the reader can find many of the same calculations being done for some of the models of CSP known at the time it was written. We provide an extended appendix of notation and brief details of the theoretical ideas used from [23], as well as detailed references to the Internet version of that book. We will assume that the alphabet Σ of all communication events is finite, since this is necessary to make some of our arguments work.

As part of this paper's contribution to the development of the hierarchy of CSP models, at various points in this paper we set out conjectures, open questions and pieces of work still to be done. These are highlighted by the symbol ¶ in the margin.

2 The CSP language

When we discuss congruences, denotational models and full abstraction we need to establish what language we are using, since the expressive power of that language has enormous effects on the results we are able to prove. In this paper we adopt the core CSP language described in [23], with the addition of two less central operators from that book ($\rhd$ and $\triangle$) for reasons that will become apparent later.

In the following, Σ is a nonempty set of communications that are visible and can only happen when the observing environment permits via handshaken communication. The actions of every process are taken from $\Sigma \cup \{\checkmark, \tau\}$ where τ is the invisible internal action and $\checkmark$ is a signal processes communicate when they have terminated successfully.

The constant processes of CSP are

- *STOP* which does nothing – a representation of deadlock.
- **div** which performs (only) an infinite sequence of internal τ actions – a representation of divergence or livelock.
- *CHAOS* which can do anything except diverge.
- *SKIP* which simply terminates successfully by communicating the signal $\checkmark$.

The following operators introduce communication:

- $a \rightarrow P$ communicates the event $a \in \Sigma$ before behaving like P. This is *prefixing*.
- $?x : A \rightarrow P(x)$ communicates any event from $A \subseteq \Sigma$ and then behaves like the appropriate $P(x)$. This is *prefix choice*.

There are three forms of binary choice between a pair of processes:

- $P \sqcap Q$ lets the process decide to behave like P or like Q: this is *nondeterministic* or *internal* choice.
- $P \Box Q$ offers the environment the choice between the initial Σ-events of P and Q. If the one selected is unambiguous then it continues to behave like the one chosen; if it is an initial event of both then the subsequent behaviour is nondeterministic. The occurrence of τ in one of P and Q does *not* resolve the choice (unlike CCS $+$), and if one of P and Q can terminate then so can $P \Box Q$. This is *external* choice.
- $P \rhd Q$ may choose to offer the visible actions of P but, unless one of these is followed, *must* offer the initial choices of Q. This is *asymmetric* or *sliding* choice and can be said to give an abstract (and untimed) representation of P timing out, if none of its initial actions are accepted, and becoming Q.

  We choose to regard the asymmetric choice operator $\rhd$ as primitive rather than deriving it from other operators as has usually been done. It is equivalent to $(P \Box a \rightarrow Q) \setminus \{a\}$ for any event a that does not appear in P or Q. In this paper we will always give $\rhd$ the following operational semantics taken from page 169 of [23], closely analogous to this example: the first rule says that P can perform internal actions without resolving the choice

$$\frac{P \xrightarrow{\tau} P'}{P \rhd Q \xrightarrow{\tau} P' \rhd Q}$$

  Any visible action from P decides the choice in its favour

$$\frac{P \xrightarrow{a} P'}{P \rhd Q \xrightarrow{a} P'} \quad (a \neq \tau)$$

  while at any moment (as we have no way of modelling time directly in this semantics) the combination can time out and become Q.

$$\frac{}{P \rhd Q \xrightarrow{\tau} Q}$$

The first two of these choice operators are commonly applied to sets of processes since they are idempotent[1] and associative. $\Box S$ can be applied to any finite set, and $\sqcap S$ can be applied to any nonempty set. If it is applied to an infinite set then it introduces *unbounded nondeterminism*, a topic we will be discussing in Section 3.2.

Various conditional choice constructs are also used within CSP, but these are resolved by non-process identifiers and are not CSP operators in the same sense as the above.

We only have a single parallel operator in our core language since all the usual ones of CSP can be defined in terms of it as discussed in Chapter 2 etc of [23].

[1] However we will see in Section 7 that there are (infrequently used) models where $\Box$ is not idempotent. Evidently anyone using these should not use $\Box$ over sets, rather over lists or multi-sets.

- $P \underset{X}{\parallel} Q$ runs P and Q in parallel, allowing each of them to perform any action in $\Sigma - X$ independently, whereas actions in X must be synchronised between the two. It terminates when both P and Q have, a rule which is equivalent to stating that $\checkmark$ is synchronised like members of X.

There are two operators that change the nature of a process's communications.

- $P \setminus X$, for $X \subseteq \Sigma$, *hides* X by turning all P's X-actions into τs.
- $P[\![R]\!]$ applies the *renaming* relation $R \subseteq \Sigma \times \Sigma$ to P: if $(a, b) \in R$ and P can perform a, then $P[\![R]\!]$ can perform b.

We will see that both of these forms are vital to full abstraction and related arguments.

We introduce a new notation for a particular type of renaming $P[\![a \mapsto A]\!]$ will mean that whenever P can perform a, the renamed process can perform any member of the set $A \subseteq \Sigma$. Similarly $P[\![A \mapsto a]\!]$ will be the process that performs a when P performs a member of A.

There are two operators that allow one process to follow another:

- $P;\ Q$ runs P until it terminates ($\checkmark$) and then runs Q. The $\checkmark$ of P becomes a τ. This is *sequential composition*.
- $P \triangle Q$ runs like P but if at any time the environment communicates an initial visible action of Q, then (nondeterministically if that event is also currently offered by P) P shuts down and the process continues like Q. This is the *interrupt* operator.

The final CSP construct is recursion: this can be single or mutual (including mutual recursions over infinite parameter spaces), can be defined by systems of equations or (in the case of single recursion) in line via the notation $\mu\, p.P$, for a term P that may include the free process identifier p.

3 The hierarchy of CSP models

Before we describe our new model it is helpful to understand the hierarchy within which it sits, particularly since that hierarchy demonstrates the existence of some sibling models that sit alongside our new one.

CSP models traditionally represent processes by sets of observations which can be made of a process. This is essentially the same idea as *testing equivalences* [6].[2] By varying the type(s) of behaviour observed we get different models.

3.1 Models based on finite observations

Each model consists of one or more sets of observations, with these being restricted by a number of healthiness conditions that ensure that each point in the model is "realistic". For example in the traces model each set of traces must be nonempty and prefix-closed.

[2]The significant difference is one of *intention*: the testing equivalences are defined over processes whose fundamental definition is operational. It is not essential that they are a congruence, and in particular there is no need of a fixed point theory for recursions. On the other hand, CSP models are expected to be potentially stand-alone, and to be capable of supporting a denotational semantics.

The tradition in CSP is to judge abstract models by means of observations of an operational semantics and by characterisation in terms of algebraic laws. Firstly, the set of processes captured by the healthiness conditions should be equal to, or at least have as a dense subset, the natural images of labelled transition systems under the abstraction map that observes a node's evolving behaviour. Secondly, the value predicted of a process P in a model by the denotational semantics should equal the same abstraction of the operational semantic value of P. Thirdly, there should be a set of (hopefully natural) algebraic laws such as $P \sqcap P = P$ which, together with a rule to handle infinitary processes, completely characterises the equivalence. Chapter 7 of [23] introduces the operational semantics of CSP, Chapter 8 introduces the traces ($\mathcal{T}$), stable failures ($\mathcal{F}$), and failures-divergences ($\mathcal{N}$) models of CSP and the denotational semantics if the language in each. Chapter 9 analyses these semantics, and in particular Sections 9.3 and 9.4 show how to prove the full abstraction results and congruence results that relate operational and denotational semantics. Chapter 10 extends the ideas of Chapters 8 and 9 to equivalences with infinite traces, introducing models $\mathcal{I}$ and $\mathcal{U}$ that handle divergences and infinite traces (respectively with finite traces, and failures), and Chapter 11 shows how to develop an algebraic semantics based on the systematic transformation of any finitary program to a normal form.

The original and simplest model consists of finite traces $\mathcal{T}$ [12], in which a process is represented by the set of finite sequences of visible events it can perform. Because this model only considers finite traces, all of the observations it makes of processes can be completed in a finite time. The basic model we will be studying also falls into this category, alongside a number of others[3]:

- The *stable failures model* $\mathcal{F}$ [23], in which processes are represented by their finite traces and stable failures (pairs (s, X) where s is a finite trace and X is a set of events some implementation state reachable after s can refuse.) In our type of LTS with the signal action $\checkmark$, a state can refuse a set of events if it is *either* stable (has no τ or signal event) *or* can perform a signal, since the right way to model signals in failures models is to define that any process that can perform a signal can opt to do this independently and can therefore refuse all other events. We will call a state $\checkmark$-stable if it satisfies either of these requirements.

- The *stable ready sets model* $\mathcal{A}$, adapted from [16], which is the same except that failures are replaced by pairs (s, A) in which s is a finite trace and A is the precise set of events offered by some stable state reachable after s. (This is sometimes called the (stable) *acceptance sets* model.) The difference between a ready set and the complement of a refusal set is that the latter is closed under superset, but the former is not.

- The *refusal testing model* $\mathcal{RT}$ ([15], based on [18]), in which a process is represented by a finite alternating sequence of the form

  $$\langle X_0, a_0, X_1, a_1, \ldots, a_n, X_{n+1} \rangle$$

  in which each a_i is a visible event and each X_i is either a $\checkmark$-stable refusal observable at the appropriate time or a marker $\bullet$ to say that no refusal has been observed. Note that in this notation we could re-cast $\mathcal{F}$ with processes having only one set of behaviours, failures with $\bullet$ being allowed as a "refusal" so that $(s, \bullet)$ would represent the simple trace s.

[3]The forms quoted here are in some cases not precisely those in which they were originally presented. What we do here is strip them down to finite observations only.

These are certainly not the only finite observation models that exist: after all the purpose of this paper is to introduce another one. It seems reasonable to define a *finite observation* model to be any that is defined in terms of behaviours that can be observed of processes

(i) that take a finite amount of time to observe,

(ii) that only record things that can be seen on a single interaction with the process – they are *linear*,

(iii) are restricted to what can reasonably be observed of a standard labelled transition system in which, from one state, one cannot "see ahead" to the range of behaviours that can follow its initial actions. This would be contrary to the spirit of "linearity".

The only things that we can observe are thus the sequence of visible actions that occur, the stability (and conceivably the instability) of the states from which they occur and the final state reached, and, in the case of stability, the set of visible actions offered and consequences of this such as refusals.

Certainly all the models listed above, and any conceivable model of this form, must give an equivalence that is coarser than observing all sequences of the forms

$$(A_0, a_1, A_2, \ldots, A_{n-1}, a_n, A_n) \qquad \text{and} \qquad (A_0, a_1, A_2, \ldots, A_{n-1}, a_n, \bullet, \checkmark)$$

possible for a process, where each $a_i \in A_{i-1}$, and $A_i \subseteq \Sigma$ or $A_i = \bullet$. Here, $\bullet$ is recorded just when the state prior to a visible event is unstable and otherwise A_i is the set of events offered from the stable state from which a_{i+1} occurred.[4] It seems highly undesirable to the author to be able to make the particular distinction implied by observing positively the instability of the state from which an event occurs. One argument for this is that it would mean

$$a \rightarrow STOP \not\equiv (a \rightarrow STOP) \rhd (a \rightarrow STOP)$$

Another is that *observing* instability of a state from which some a occurs implies we can see that a state has τ actions without following one.

For this reason the author proposes that there is no *positive* observation of instability, so that $\bullet$ simply means that our process has not been observed to be stable. Thus, when any observation of our process of one of the above forms is possible, so is the same one with any selection of the A_is replaced by $\bullet$. The observations possible of each of the processes displayed above will then be the same: sequences (X) and (X, a, Y) where X is either $\bullet$ or $\{a\}$ and Y is either $\bullet$ or $\emptyset$.

Clearly the set of $\mathcal{FL}$-observations of a process (i.e. those of the two forms above that can be made of its operational semantics) are also closed under prefix (initial subsequence), and whenever $(A_0, a_1, A_1, \ldots, a_r, A_r, a_{r+1}, A_{r+1})$ belongs to a process and $A_r \neq \bullet$ then $(A_0, a_1, A_1, \ldots, a_r, A_r, b, \bullet)$ also belongs for all elements b of A_r. Here, $\mathcal{FL}$ stands for "finite linear".

If β is such a behaviour, β' is either a proper prefix of β or is a prefix of β with at least one proper acceptance replaced by $\bullet$ then we will write $\beta' < \beta$.

We postulate that every reasonable finite observation model will induce a coarser equivalence than this equivalence. It seems certain to the author that it induces a CSP model

[4] We use round brackets $(\cdot)$ rather then the usual sequence ones $\langle\cdot\rangle$ here to distinguish this class of behaviour from $\mathcal{RT}$ ones visually.

itself – the finest possible in this class – which, like the behaviours used to construct it, we will term $\mathcal{FL}$. The details of its healthiness conditions, operator definitions etc, remain to be set out and checked. For the purposes of this paper we will assume that every finite observation model $\mathcal{M}$ represents a process as a finite tuple $(M_1, \ldots, M_r)$ of sets of observed behaviours, where each M_i is the image of the process's representation in $\mathcal{FL}$ under some relation. ¶

It is interesting to compare the above set of models with the equivalences described in Von Glabbeek's papers [9, 10] where he describes a hierarchy which extends through ones based on the sorts of behaviour we have looked at all the way to bisimulation. In the first of these papers there are analogues there for all the equivalences we have discussed above, including "ready traces" which corresponds to $\mathcal{FL}$. There is an essential difference, however, namely that the equivalences of [9] are described over process trees *without* τ *actions*. It follows that the issues of actions occuring without stability having been observed are irrelevant, and the subtleties (such as $\bullet$) associated with that phenomenon in our treatment of all the above models other than $\mathcal{T}$ are not present. In [10], where τ actions are considered, this issue still does not seem to be addressed. Without proper modelling of this phenomenon, there is no prospect of any model as fine as $\mathcal{RT}$ being a congruence for a CSP-like language involving hiding, and if the language contains $\triangle$ no model finer than $\mathcal{T}$ is possible without it.

There is no analogue in [9, 10] of the revivals models that are the main topic of this paper.

3.2 Models involving divergences and other infinite behaviours

The CSP model that is perhaps the most familiar, failures-divergences ($\mathcal{N}$) [4], does not fall into this finite-observation category because, as well as failures, it also records a process's *divergences*: finite traces after which the process can execute an infinite unbroken sequence of τ actions.

This makes a great difference in calculating the semantics of a process for two distinct reasons; both result in the modelling capacity of $\mathcal{N}$ being less than some might like.

- The first problem comes in calculating the divergences of the hiding operator $P \setminus X$. Since $\mathcal{N}$ does not represent infinite traces directly, we have to infer infinite sequences of X-actions that will map to divergence from the set of finite traces. This is only accurate if P has no infinite branching on any X-action or τ – we can then apply König's Lemma giving an infinite path through the that parts of P's execution tree whose trace is a prefix of a chosen infinite trace.

  It follows that $\mathcal{N}$ (unlike the models recording finite behaviour only) is only valid for finitely nondeterministic CSP – the language with no infinite, or *unbounded* nondeterminism $\sqcap \mathcal{S}$, and no infinite-to-one renaming or $P \setminus X$ with X infinite in the case where the overall alphabet is infinite.

- The second problem is that for two separate reasons – avoiding unbounded nondeterminism being created by finitely nondeterministic operators[5], and calculating the

[5]See [26] for an example of how this arises. In effect it is because the infinite sequence of states a process passes through during a divergence can each have a branch labelled with the same action x but leading to different results. This is sufficiently close to having all these actions leading from the same state to cause the same problems as unbounded nondeterminism. The infinite path through the tree created by König's

correct fixed points for recursions – $\mathcal{N}$ has to adopt the principle of *divergence strictness*: once a process can diverge on trace s, we have no interest in its other behaviour on s or its extensions. This is manifest by the healthiness conditions that say that if s is a divergence then so is $s^\frown t$, and any $(s^\frown t, X)$ is a failure. This means that $\mathcal{N}$ does not in fact record strictly more information than $\mathcal{F}$, as one would think at first: over $\mathcal{N}$ all processes that can diverge immediately are identified with each other, and this is certainly not the case over $\mathcal{F}$.

A noteworthy feature of $\mathcal{N}$ is that, for each operator, the calculation of the divergences of a process are completely independent of the failures *per se* of its arguments, only their traces and divergences. This is something we can expect, since no refusal information is recorded in a divergence, and the refusal at the end of a failure is the last thing that is observed. Equally, it is clear that if we have two different models that both contain records of all strict-divergent traces (or indeed any other sort of observation) then they must agree on them if they are accurate.

We will see in this section that there is, in effect, a two-dimensional structure of CSP models. On the one hand we can classify a model by the level of detail it records about a process's finitely observable behaviour, and on the the other we can do so based on what is recorded about its infinite behaviour.

From this perspective we can regard $\mathcal{N}$ as the divergence-strict extension of $\mathcal{F}$ to include divergences, where we can be confident that exactly the same traces and divergences will be calculated as in the corresponding model where we only record finite traces and divergences. We will use the notation $\mathcal{M}^{\Downarrow}$ for this type of extension, so $\mathcal{N} = \mathcal{F}^{\Downarrow}$.

We can similarly extend $\mathcal{A}$ to $\mathcal{A}^{\Downarrow}$. The extension of $\mathcal{RT}$ is more complex since here we should be interested in what is refused leading up to a divergence: refusal is no longer final. Here the natural form of a divergence takes the form $\langle X_0, a_0, X_1, a_1, \ldots, X_n, a_n\rangle$ and there are healthiness conditions that if this is a divergence then so are all others whose presence can be deduced from this one, for example by pointwise subset on the refusal arguments. This is in fact the model of [15].

In any case where $\mathcal{M}$ observes nothing other than a trace prior to a divergence, $\mathcal{M}^{\Downarrow}$ is simply formed by adding the set of finite divergence traces and forcing divergence strictness. In cases like $\mathcal{RT}$ where observations can carry on long enough to observe divergence after some refusal, acceptance or similar is recorded, it is necessary to used more complex "divergences" than just traces.

The problems highlighted above restricting $\mathcal{N}$ to finite nondeterminism and divergence strictness have been solved in the years since $\mathcal{N}$ was developed, in two steps. Both solutions complicate the original model.

The solution to the first problem is to add infinite traces to $\mathcal{N}$, creating a model ($\mathcal{U}$) where processes' representations have three components: failures, divergences, and infinite traces. This remains divergence strict. There is no real conceptual difficulty here, since in many ways it is more natural to represent infinite traces directly rather than to try to deduce them from the finite ones. We can now distinguish between the process that nondeterministically chooses to perform any finite number of as, and the one that has these choices that can also choose to perform an infinite number. Note that the first of these, on hiding a, should become $STOP$ since it cannot diverge, while the second one becomes

---

lemma may just be the divergence, and fail to have any x action on it. This does not matter in a divergence strict model, because divergence on a prefix of $(s \setminus A)^\frown\langle x\rangle$ implies it on this trace itself. But in a model where we have to know if $P \setminus A$ can diverge after precisely $(s^\frown\langle x\rangle) \setminus A$, it is serious.

divergent.

The complications here arise from the structure of the resulting model, which is no longer a complete partial order, and establishing the operational congruence result, which is significantly harder. Alternative methods were developed in [1, 25] for proving the existence of least fixed points despite incompleteness for any CSP-definable recursion, and also in [25] for proving operational congruence and hence full abstraction.

These methods work for all the models in which any refusal information is only at the end of a trace, meaning that for each of them there is a strict divergence and infinite trace extension $\mathcal{M}^{\Downarrow,\omega}$, so that $\mathcal{U} = \mathcal{F}^{\Downarrow,\omega}$ and $\mathcal{I} = \mathcal{T}^{\Downarrow,\omega}$.

For $\mathcal{RT}$, as with divergences, the situation will be more complex since it is natural to want to know infinite refusal testing information of the form:

$$\langle X_0, a_0, X_1, a_1, \ldots, a_n, X_{n+1}, \ldots \rangle$$

The details of this model, and the necessary proofs, have not been worked out at the time of writing as far as the author is aware. The work in this paper, particularly in Section 6.1, offers some guidance on the structure of that model, and we will briefly discuss it again there. ¶

The solution to the second problem (the need for divergence strictness) was published relatively recently [26]. As stated above, the need for divergence strictness comes in $\mathcal{N}$ from problems with unbounded nondeterminism and recursion. The issue with unbounded nondeterminism is solved by infinite traces, and cannot be solved without them [26]. The issue with fixed points comes from reconciling the following pair of facts:

(A) With finitary models like $\mathcal{T}$ and $\mathcal{N}$, the correct denotation of any recursion is always the least fixed point with respect to the component-wise subset order. This is easy to see: any finitely observable behaviour will appear after some finite number of unwindings of the recursion in the operational semantics. This is closely related to the fact that all CSP operators are continuous with respect to this order in all known models, guaranteeing that this fixed point is reached in ω unwindings: $\bigcup\{F^n(\bot) \mid n \in \mathbb{N}\}$.

(B) We cannot expect infinite behaviours to appear in a finite number of unwindings, and the above fixed point does not work for models with infinitary components. The least fixed point with respect to the superset, or refinement order does work, but only if we impose divergence strictness. Some operators are not continuous in this direction.

They *need* reconciling because (A) and (B) together appear to tell us that the finitely observable behaviour recorded in infinite observation models can be correctly calculated by either a least or a greatest fixed point! One of the reasons why divergence strictness helps in (B) is that in all known models one can show that all finite behaviour none of whose prefixes is divergent is *uniquely* determined by a recursion, neatly resolving the paradox of how we can use both least and greatest fixed points for finite behaviour. Since there is only one solution for the finite behaviour left visible by divergence strictness, the greatest and least fixed points give the same answer.

If we take away divergence strictness, it is fairly easy to see by example that neither the greatest nor the least fixed point is correct. The term $\mu\, p.p$ has every member of a model as a fixed point, but its correct denotation (the process that just divergences on the empty trace) is not refinement minimal, but does have a behaviour (its divergence) that not all fixed points have.

As shown in [26], it is possible to calculate the correct value by a non-standard fixed point process in a model with nearly all divergence strictness removed. Let $\mathcal{M}^{\#}$ be the equivalence determined by the finite behaviours of $\mathcal{M}$, with divergence and infinite trace information as in $\mathcal{M}^{\Downarrow,\omega}$

- without the divergence strictness assumptions, but
- with the assumption that if the infinite non-divergent behaviour u (usually an infinite trace) has an infinite number of divergent prefixes, then u is present in the representation whether it is really possible or not. We term this *weak divergence strictness*.

The last assumption is necessary: it is shown in [26] (and independently in [14]) that it is impossible to give a denotational fixed point theory over the model with this assumption removed. These models are very closely related to congruences of Puhakka and Valmari [19, 20], discovered there as the weakest ones that predict all divergence traces (by themselves, or with deadlock).

The operationally correct fixed point theory involves first calculating the refinement-least fixed point Λ of a recursion $p = F(p)$, then stripping away all post-divergence behaviour to get a value $\hat{\Lambda}$. The interval between Λ and $\hat{\Lambda}$ is mapped into itself by F, and the second stage of the fixed point process involves calculating the subset-least fixed point in this interval. This construction – called the *reflected* fixed point – is similar to one used in related circumstances by Broy in [5].

So for each of the models $\mathcal{T}$, $\mathcal{F}$, $\mathcal{A}$ and conjecturally for $\mathcal{RT}$ with richer infinite behaviours, we have a family of four: ¶

- The original model $\mathcal{M}$, valid for full CSP but not recording any infinite behaviours.
- The extension $\mathcal{M}^{\Downarrow}$ which adds strict divergences but can only handle finitely nondeterministic processes.
- The extension $\mathcal{M}^{\Downarrow,\omega}$ which adds infinite traces so it can handle the full language, but is still divergence strict.
- The model $\mathcal{M}^{\#}$ that removes almost all of the divergence strictness assumption. The main model discussed in [26], there termed $\mathcal{SBD}$, is $\mathcal{T}^{\#}$.

Considered as abstractions of an arbitrary LTS, $\mathcal{M}^{\#}$ is the finest equivalence, $\mathcal{M}^{\Downarrow,\omega}$ is finer than $\mathcal{M}^{\Downarrow}$, but both of these last two are incomparable with $\mathcal{M}$.

The choice of which CSP model to use very much depends on two things: whether unbounded nondeterminism is possible in the processes under consideration, and what level of detail is required. The majority of practical processes are both divergence free and finitely nondeterministic, so we might well want to use $\mathcal{T}^{\Downarrow}$ to establish divergence freedom, if required, and then use whichever finite observation model is required. We will return to this question in Section 6.4.

Our new model turns out to be one for which the various extensions only involve infinite *traces* and divergence *traces*, so we can, if we we wish, concentrate on the finite observations, confident that its three siblings are obtained by adding the same sets of divergences and infinite traces as with $\mathcal{T}$ and $\mathcal{F}$.

4 The stable revivals model

Over an alphabet $\Sigma^{\checkmark} = \Sigma \cup \{\checkmark\}$ of visible events ($\checkmark$ being CSP's special termination signal[6]), the stable revivals model identifies each process P with a a triple $(Tr, Dead, Rev)$, where

- $Tr \subseteq \Sigma^{*\checkmark}$ consists of all P's finite traces (perhaps ending in $\checkmark$).
- $Dead \subseteq \Sigma^*$ consists of all traces (other than terminated ones) after which P can deadlock (reach a state from which no further action is possible).
- $Rev \subseteq \Sigma^* \times \mathcal{P}(\Sigma) \times \Sigma$ consists of all P's revivals. (s, X, a) means that P can perform s, stably refuse X, and then perform a. It is important to note that we have neither allowed $\checkmark$ to be the "reviving event" a nor to appear in the refusal set X. We could have chosen to do either or both of these things, but with proper healthiness conditions they would not have changed the expressiveness of the model. Because of our interpretation of $\checkmark$, we know that when P has the trace $s\hat{}\langle\checkmark\rangle$ it can definitely choose to offer $\checkmark$ and only $\checkmark$ after s. Furthermore we know that whenever a process is stably offering a non-$\checkmark$ event a in the revival (s, X, a), the corresponding state is stable and unable to perform $\checkmark$. We do not have to have observations recorded to tell us these things.[7] Thus revivals are generated purely by stable states as opposed to $\checkmark$-stable states.

  The same structure would work if there were more than one signal. The reason why we are able to take this decision in this model and not in the stable failures model is because the mew model has the separate representation of deadlock. Without this, the only way of distinguishing between $SKIP$ and $SKIP \sqcap STOP$ is to observe the refusal of sets including $\checkmark$ in the latter.

Not all such triples $(Tr, Dead, Rev)$ represent a possible real process, so as with other CSP models we adopt a set of healthiness conditions. These are:

Tr1 Tr is nonempty and prefix-closed: if $s\hat{}t \in Tr$, then $s \in Tr$.

Dead1 $Dead \subseteq Tr$. Every deadlock trace is a trace.

Rev1 $(s, X, a) \in Rev \Rightarrow s\hat{}\langle a\rangle \in Tr$ This simply says that every trace implied by a revival is recorded in Tr.

Rev2 $(s, X, a) \in Rev \wedge Y \subseteq X \Rightarrow (s, Y, a) \in Rev$ This says that the state which refuses X and accepts a also refuses any subset of X.

Rev3 $(s, X, a) \in Rev \wedge b \in \Sigma \Rightarrow ((s, X, b) \in Rev \vee (s, X \cup \{b\}, a) \in Rev)$ In other words, whatever state refuses X and accepts a, either accepts or refuses b.

[6] $\checkmark$ is treated differently to other events because (i) it is always final in a trace and (ii) there is no need for other processes to agree to it – they just observe it. We include it, and the related CSP operations $SKIP$ and $P;\ Q$, in this paper primarily because one of the main motivations is developing the ideas of [8] on models of network termination. Its relationship with refusal sets also provides an excellent prototype for including more general signal events in the model, should this be desired. The role of $\checkmark$ as a signal event is discussed in detail in [23], particularly Chapter 6.

[7] Note that the presentation here is different – and hopefully cleaner – than the one in [22], since the latter allowed revivals of the form $(s, X, \checkmark)$ though the healthiness conditions prevented them from adding extra distinctions between processes.

The reader might want to compare these to the corresponding healthiness conditions for $\mathcal{F}$ discussed on page 212 of [23] (which uses some conditions defined on page 196).

The stable revivals model $\mathcal{R}$ is defined to be the set of all triples satisfying the above conditions. Following tradition, we define $P = (Tr_P, Dead_P, Rev_P) \sqsubseteq_R Q = (Tr_Q, Dead_Q, Rev_Q)$ (Q refines P) if and only if

$$Tr_Q \subseteq Tr_P \quad \text{and} \quad Dead_Q \subseteq Dead_P \quad \text{and} \quad Rev_Q \subseteq Rev_P$$

The refinement-minimum element of $\mathcal{R}$ is $CHAOS$, which contains all possible behaviours in each of its three components. One CSP representation of $CHAOS$ is

$$CHAOS = STOP \sqcap SKIP \sqcap \bigsqcap\{a \rightarrow CHAOS \mid a \in \Sigma\}$$

Replacing the last clause with $?x : \Sigma \rightarrow CHAOS$ (which works for $\mathcal{F}$) does not work in this model, because it would not have any revival of the form (s, X, a) $(a \neq \checkmark)$ when X is non-empty. The definition displayed above does work in $\mathcal{RT}$ but not in $\mathcal{A}$, where the nondeterministic choice would need to be extended to cover all initial acceptance sets.

The maximum element is the process $(\{\langle\rangle\}, \emptyset, \emptyset\})$ which corresponds to the process **div**, which simply diverges. This strange-seeming phenomenon occurs simply because we are choosing not to record divergence in our model: all the finite-behaviour models described in Section 3 map **div** to their greatest elements. Adding a representation of divergence, as we will do in the next section, (whether divergence-strictly or not) will mean that there is no top element: in $\mathcal{F}$, **div** is more refined than either $STOP$ or $a \rightarrow STOP$, but this is not true in models with representations of divergence, since each of the three processes has a recorded behaviour that neither of the other two does (respectively immediate divergence, immediate deadlock, and the trace $\langle a \rangle$).

Given that in order to be a complete lattice it is sufficient that a partial order has greatest lower bound for any subset, and all of the healthiness conditions above are easily seen to be closed under nondeterministic choice of nonempty sets, the statement of the following proposition is its own proof.

PROPOSITION 4.1 *$\mathcal{R}$ is a complete lattice under refinement, with greatest lower bound given by nondeterministic choice (equivalent to component-wise union), for nonempty sets, and* **div** *is the greatest lower bound of the empty set.*

It should be noted that the inclusion of the traces component Tr in $\mathcal{R}$ is not always essential to get a congruence which includes revivals, as can be discerned from the semantics below. To be precise, as over $\mathcal{F}$ (as alluded to in [23], page 239), the traces component of the model is only forced by the presence of revivals when one has an operator which has the capability of "switching off" one of its arguments when the latter might be diverging. The only such operator in CSP is the interrupt operator $P \triangle Q$, which turns off P as soon as Q performs a visible event. Thus if $Q = b \rightarrow STOP$ and $P = \mathbf{div} \,\Box\, a \rightarrow P$, we see that P has arbitrarily long traces, but no deadlocks or revivals; $P \triangle Q$ has deadlocks $\langle a, a, \ldots, a, b \rangle$ for any number of a's. We would not be able to discern this unless we recorded P's traces.

Without this operator we would get a congruence without the trace component, and $\mathcal{R}$ as we have defined it would not be fully abstract with respect to stuckness and *RespondsTo* in the sense discussed later. The author believes, however, that except for very specialised purposes (e.g. getting full abstraction theorems!) models recording finite behaviour should include traces, as these are the most basic tool in safety specification. It was to avoid this tension that we have adopted the interrupt operator as part of the language for this paper.

The semantics of CSP

As with all the usual CSP models, it is possible to calculate a process's semantics in $\mathcal{R}$ in different ways. One is to take the operational semantics as an LTS and perform "observations" on its value there. Doing this for traces and deadlocks is completely standard: we formally observe a process P in terms of the sequences of actions and states (*trajectories*)

$$P = P_0 \xrightarrow{a_1} P_1 \ldots \xrightarrow{a_n} P_n$$

it can perform, where $s = \langle a_i \mid i \in \langle 1 \ldots n \rangle \wedge a_i \neq \tau \rangle$. For traces, we record the trace which is the sequence

$$\langle a_i \mid 1 \leq i \leq n \wedge a_i \neq \tau \rangle$$

This trace is observed as a deadlock if P_n has no (outgoing) actions at all, and $a_n \neq \checkmark$. Similarly, we can observe the revival (s, X, a) if P_n is stable (no τ or $\checkmark$ actions) and our trajectory can be extended to

$$P = P_0 \xrightarrow{a_1} P_1 \ldots \xrightarrow{a_n} P_n \xrightarrow{a} Q$$

for some Q.

For any LTS node N, we can form a natural abstraction $\Phi(N) = (Tr_N, Dead_N, Rev_N)$ where the three components are the sets of traces, deadlocks and revivals that can be observed as indicated here of any trajectory of N. It is easy to show that $\Phi(N) \in \mathcal{R}$.

We could take P's value in a less abstract model such as $\mathcal{RT}$ or $\mathcal{A}$ and extract the $\mathcal{R}$ value from that. Over $\mathcal{R}$ a process has the revival $(\langle a_1, \ldots, a_n \rangle, X, a_{n+1})$ if and only if it has the refusal trace $\langle \bullet, a_1, \bullet, \ldots, a_n, X, a_{n+1}, \bullet \rangle$. Over $\mathcal{A}$, it has (s, X, a) if and only if it has a trace/ready set combination (s, Y) where $a \in Y$ and $Y \cap X = \emptyset$.

The way that really characterises $\mathcal{R}$ as a CSP model is to calculate the value directly by means of a denotational semantics: clauses which show how to derive the value of any CSP operator, or recursion, applied to simpler term(s).

The semantic clauses for *traces* are, of course, identical to those for the traces model (and for $\mathcal{F}$).

$$\begin{aligned}
traces(STOP) &= \{\langle\rangle\} \\
traces(SKIP) &= \{\langle\rangle, \langle\checkmark\rangle\} \\
traces(a \rightarrow P) &= \{\langle\rangle\} \cup \{\langle a\rangle \hat{} s \mid s \in traces(P)\} \\
traces(?x : A \rightarrow P) &= \{\langle\rangle\} \cup \{\langle a\rangle \hat{} s \mid a \in A \wedge s \in traces(P[a/x])\} \\
traces(P \sqcap Q) &= traces(P) \cup traces(Q) \\
traces(P \,\Box\, Q) &= traces(P) \cup traces(Q) \\
traces(P \rhd Q) &= traces(P) \cup traces(Q) \\
traces(P \underset{X}{\parallel} Q) &= \bigcup\{s \underset{X}{\parallel} t \mid s \in traces(P) \wedge t \in traces(Q)\} \\
traces(P; Q) &= traces(P) \cap \Sigma^* \cup \\
&\quad \{s \hat{} t \mid s \hat{} \langle\checkmark\rangle \in traces(P) \wedge t \in traces(Q)\} \\
traces(P[\![R]\!]) &= \{s' \mid \exists\, s \in traces(P) \mid s \,R\, s'\} \\
traces(P \setminus X) &= \{s \setminus X \mid s \in traces(P)\} \\
traces(P \triangle Q) &= traces(P) \cup \\
&\quad \{s \hat{} t \mid s \in traces(P) \cap \Sigma^* \wedge t \in traces(Q)\}
\end{aligned}$$

The calculation of most cases of $deadlocks(P)$ can be presented as typical denotational semantic clauses:

$$\begin{aligned}
deadlocks(STOP) &= \{\langle\rangle\} \\
deadlocks(SKIP) &= \emptyset \\
deadlocks(a \rightarrow P) &= \{\langle a\rangle\hat{}s \mid s \in deadlocks(P)\} \\
deadlocks(?x : A \rightarrow P) &= \{\langle a\rangle\hat{}s \mid a \in A \wedge s \in deadlocks(P[a/x])\} \\
deadlocks(P \,\Box\, Q) &= ((deadlocks(P) \cup deadlocks(Q)) \cap \{s \mid s \neq \langle\rangle\}) \\
&\quad \cup (deadlocks(P) \cap deadlocks(Q)) \\
deadlocks(P \sqcap Q) &= deadlocks(P) \cup deadlocks(Q) \\
deadlocks(P \rhd Q) &= deadlocks(Q) \cup \{s \in deadlocks(P) \mid s \neq \langle\rangle\} \\
deadlocks(P;\ Q) &= deadlocks(P) \\
&\quad \cup \{s\hat{}t \mid s\hat{}\langle\checkmark\rangle \in traces(P) \wedge t \in deadlocks(Q)\} \\
deadlocks(P \,\triangle\, Q) &= \{s\hat{}t \mid s \in traces(P) \cap \Sigma^* \wedge t \in deadlocks(Q)\} \\
deadlocks(P[\![R]\!]) &= \{t \mid \exists\, s \in deadlocks(P).s\ R\ t\} \\
deadlocks(P \setminus X) &= \{s \setminus X \mid s \in deadlocks(P)\}
\end{aligned}$$

But this breaks down for parallel operators involving synchronisation, and specifically $\underset{X}{\|}$. If, for example $R = (a \rightarrow R) \sqcap (b \rightarrow R)$, then $deadlocks(R \underset{\{a,b\}}{\|} R) = \{a, b\}^*$ even though $deadlocks(R) = \{\}$. In other words, a deadlock can occur in a parallel network when none of the components is deadlocked. This non-compositionality of deadlock traces under parallel is intimately related to the full abstraction of the stable failures model of CSP with respect to deadlock. In the case of $\mathcal{R}$, however, we will know $failures(P)$ and $failures(Q)$ for any pair of processes we combine in parallel, thanks to the following calculation.

If $P = (Tr, Dead, Rev)$ is a process represented in $\mathcal{R}$ we can easily calculate:

$$\begin{aligned}
failures(P) &= \{(s, X) \mid X \subseteq \Sigma^{\checkmark} \wedge s \in Dead\} \\
&\quad \cup \{(s, X), (s, X \cup \{\checkmark\}) \mid (s, X, a) \in Rev\} \\
&\quad \cup \{(s, X) \mid s\hat{}\langle\checkmark\rangle \in Tr \wedge X \subseteq \Sigma\} \\
&\quad \cup \{(s\hat{}\langle\checkmark\rangle, X) \mid s\hat{}\langle\checkmark\rangle \in Tr \wedge X \subseteq \Sigma^{\checkmark}\}
\end{aligned}$$

Note that this definition introduces failures consistent with the way in which $\checkmark$ is handled in $\mathcal{F}$.

We can therefore extract the final clause of the $deadlocks(P)$ semantics as follows.

$$\begin{aligned}
deadlocks(P \underset{X}{\|} Q) &= \{u \mid \exists (s, Y) \in failures(P), (t, Z) \in failures(Q). \\
&\quad Y - (X \cup \{\checkmark\}) = Z - (X \cup \{\checkmark\}) \\
&\quad \wedge\, u \in s \underset{X}{\|} t \cap \Sigma^* \\
&\quad \wedge\, \Sigma^{\checkmark} = Y \cup Z\}
\end{aligned}$$

Here, $s \underset{X}{\|} t$ is the set of traces that can result from s and t running and synchronising on X. See [23] (pages 69/70 and 148) for details.

It is also useful to define the set of failures recording only refusals from stable (as opposed to ✓-stable) states

$$\begin{array}{lcl} \mathit{failures}^\flat(P) & = & \{(s,X) \mid X \subseteq \Sigma \wedge s \in \mathit{Dead}\} \\ & & \cup \{(s,X) \mid (s,X,a) \in \mathit{Rev}\} \end{array}$$

The only things which remain to be constructed for our semantics are the clauses for $\mathit{revivals}(P \oplus Q)$, quoted below for each operator, and the method by which the value of a recursive term is computed. The following satisfies the first of these obligations. Once again we make use of the fact that $\mathit{failures}(P)$ can be derived from the value in $\mathcal{R}$ of P.

$$\mathit{revivals}(\mathit{STOP}) = \emptyset$$

$$\mathit{revivals}(\mathit{SKIP}) = \emptyset$$

$$\begin{array}{lcl} \mathit{revivals}(a \rightarrow P) & = & \{(\langle\rangle, X, a) \mid a \notin X\} \\ & & \cup \{(\langle a\rangle\hat{}s, X, b) \mid (s, X, b) \in \mathit{revivals}(P)\} \end{array}$$

$$\begin{array}{lcl} \mathit{revivals}(?x : A \rightarrow P) & = & \{(\langle\rangle, X, a) \mid X \cap A = \emptyset \wedge a \in A\} \\ & & \cup \{(\langle a\rangle\hat{}s, X, b) \mid a \in A \\ & & \quad \wedge\ (s, X, b) \in \mathit{revivals}(P[a/x])\} \end{array}$$

$$\mathit{revivals}(P \sqcap Q) = \mathit{revivals}(P) \cup \mathit{revivals}(Q)$$

$$\mathit{revivals}(\bigsqcap S) = \bigcup\{\mathit{revivals}(P) \mid P \in S\} \text{ for } S \text{ a non-empty set of processes}$$

$$\begin{array}{lcl} \mathit{revivals}(P \,\Box\, Q) & = & \{(\langle\rangle, X, a) \mid (\langle\rangle, X) \in \mathit{failures}^\flat(P) \cap \mathit{failures}^\flat(Q) \\ & & \qquad \wedge\ (\langle\rangle, X, a) \in \mathit{revivals}(P) \cup \mathit{revivals}(Q)\} \\ & & \cup \{(s, X, a) \mid (s, X, a) \in \mathit{revivals}(P) \cup \mathit{revivals}(Q) \wedge s \neq \langle\rangle\} \end{array}$$

$$\mathit{revivals}(P \rhd Q) = \{(s, X, a) \in \mathit{revivals}(P) \mid s \neq \langle\rangle\} \cup \mathit{revivals}(Q)$$

$$\begin{array}{lcl} \mathit{revivals}(P \underset{X}{\parallel} Q) & = & \{(u, Y \cup Z, a) \mid \\ & & \exists\, s, t.(s, Y) \in \mathit{failures}(P) \wedge (t, Z) \in \mathit{failures}(Q) \\ & & \wedge\ u \in s \underset{X}{\parallel} t \cap \Sigma^* \\ & & \wedge\ Y - (X \cup \{\checkmark\}) = Z - (X \cup \{\checkmark\}) \\ & & \wedge\ ((a \in X \wedge (s, Y, a) \in \mathit{revivals}(P) \\ & & \qquad \wedge\ (t, Z, a) \in \mathit{revivals}(Q)) \\ & & \quad \vee\ a \notin X \wedge ((s, Y, a) \in \mathit{revivals}(P) \\ & & \quad \vee\ a \notin X \wedge (t, Z, a) \in \mathit{revivals}(Q))))\} \end{array}$$

$$\mathit{revivals}(P \setminus X) = \{(s \setminus X, Y, a) \mid (s, Y \cup X, a) \in \mathit{revivals}(P)\}$$

$$\mathit{revivals}(P[\![R]\!]) = \{(s', X, a') \mid \exists\, s, a.s\ R\ s' \wedge a\ R\ a' \wedge (s, R^{-1}(X), a) \in \mathit{revivals}(P)\}$$

$$\begin{array}{lcl} \mathit{revivals}(P;\ Q) & = & \{(s, X, a) \mid (s, X, a) \in \mathit{revivals}(P)\} \\ & & \cup \{(s\hat{}t, X, a) \mid s\hat{}\langle\checkmark\rangle \in \mathit{traces}(P) \wedge (t, X, a) \in \mathit{revivals}(Q)\} \end{array}$$

$$\begin{array}{lcl} \mathit{revivals}(P \,\triangle\, Q) & = & \{(s, X, a) \in \mathit{revivals}(P) \mid (\langle\rangle, X) \in \mathit{failures}^\flat(Q)\} \\ & & \cup \{(s, X, a) \mid (s, X) \in \mathit{failures}^\flat(P) \wedge (\langle\rangle, X, a) \in \mathit{revivals}(Q)\} \\ & & \cup \{(s\hat{}t, X, a) \mid s \in \mathit{traces}(P) \cap \Sigma^* \wedge t \neq \langle\rangle \wedge (t, X, a) \in \mathit{revivals}(Q)\} \end{array}$$

Notice that the clause for the parallel operator has become a little more complicated because we have to deal with the cases in which the final event of the revival is, and is not, synchronised. The most interesting clause is that for hiding: a state in $P \setminus X$ is only stable if the corresponding state of P refuses the whole of X. Since the successor event is never in the refusal of a revival, it follows that in the clause above it is never in X, and so never gets hidden for any refusal that remains valid. Similarly in $P[\![R]\!]$, the corresponding failure of P has to refuse every single event which renames to the set X, meaning that the relational image a' of P's successor event a is certainly not in X.[8]

As usual, it is a mechanical calculation that all of these clauses preserve $\mathcal{R}$'s healthiness conditions. We will however show in Section 5 below how, as an alternative, this fact can be a corollary to other results.

In Section 8.2 of [23], the author showed how definitions like the ones above, in which the behaviours of each operator $P \oplus Q$ (not necessarily binary) are formed from relations applied to those of subsets of the arguments, are always distributive with respect to finite and infinite nondeterministic choice (i.e. component-wise union). For this to apply, the reason for any behaviour being present in the binary construct $P \oplus Q$ must be one of the following:

- It is present independently of P and Q, such as the revival $(\langle\rangle, \Sigma - \{a\}, a)$ in the (unary) construct $a \rightarrow P$.
- It is there simply because of some behaviour of P, such as any trace $t \in \mathit{traces}(P) \cap \Sigma^*$ in $P;\ Q$.
- It is there simply because of some behaviour of Q, such as any behaviour b of Q in $P \rhd Q$.
- It is there because of a behaviour of P and a behaviour of Q, such as any deadlock $s^\frown t$ where $s^\frown\langle\checkmark\rangle \in \mathit{traces}(P)$ and $t \in \mathit{deadlocks}(Q)$ in $P;\ Q$.

No behaviour of $P \oplus Q$ should depend on anything more complex than this. Since the derivation of $\mathit{failures}(P)$ above requires only one behaviour of P to create each member, it follows by inspection that all the non-recursive operators over $\mathcal{R}$ meet this condition. Following the reasoning behind Theorem 8.2.1 of [23], we therefore have the following:

PROPOSITION 4.2 *All the individual non-recursive operators of CSP are distributive over finite and infinite nondeterministic choice for $\mathcal{R}$, and hence all CSP-definable operators are monotonic with respect to refinement ($\sqsubseteq$) and continuous with respect to subset ($\subseteq \equiv \sqsupseteq$).*

We can therefore compute the $\subseteq$-least fixed point of any CSP term $F(p)$ with a free variable via the formula $\bigcup\{F^n(\mathbf{div}) \mid n \in \mathbb{N}\}$ and this is the semantic value given to the recursive term $\mu\, p.F(p)$. By a standard argument, this fixed point is itself monotonic and continuous (though not necessarily distributive) in any other free process variable. We explained informally in Section 3 why this is the correct fixed point to choose. As an

[8] To amplify this further: suppose we wanted to refine $\mathcal{R}$, and so proposed an extended revivals model which contains behaviours of the form (s, X, t) in which t is a trace of length 1 or 2, rather than just the singleton event used in $\mathcal{R}$. We would then discover that it is impossible to calculate the semantic value of $(a \rightarrow b \rightarrow a \rightarrow \mathit{STOP}) \setminus \{b\}$ from that of $a \rightarrow b \rightarrow a \rightarrow \mathit{STOP}$: a step of the extended revival is lost to hiding, as explained by the follwing example. The process $(a \rightarrow b \rightarrow a \rightarrow \mathit{STOP}) \rhd (a \rightarrow b \rightarrow \mathbf{div})$ has exactly the same representation as $a \rightarrow b \rightarrow a \rightarrow \mathit{STOP}$, but hiding b in the more complex process does not give the extended revival $(\langle\rangle, \emptyset, \langle a, a\rangle)$.

example consider the definition $P = a \rightarrow (P \setminus \{a\})$. The $\subseteq$-least fixed point gives this traces $\{\langle\rangle, \langle a\rangle\}$, revivals $\{(\langle\rangle, X, a) \mid a \notin X\}$ and no deadlocks, which is operationally correct since our process is initially stable offering a, but after that diverges. If Σ is a proper superset of $\{a\}$, the $\sqsubseteq$-least fixed point has many more behaviours which would not correspond to reality.

This $\subseteq$-least fixed point definition of recursion can, of course, be extended to mutual recursion in the standard way.

5 Congruence and full abstraction for $\mathcal{R}$

For the above semantics to make sense we need to establish equivalence of the values in $\mathcal{R}$ predicted by observation of the operational semantics and the denotational semantic clauses. This will be a classic operational congruence result in the style of those established in, for example, [23, 25, 26] for other models of CSP. We will also develop full abstraction results for $\mathcal{R}$.

5.1 Congruence

We will only give the operational semantic rules here that we use directly, but the rest can be found in Section 7.3 of [23].

The proofs of such results always fall into three parts: setting up an appropriate inductive framework for the proof, establishing that the semantic clauses of non-recursive operators are correct, and then demonstrating that the chosen fixed point theory matches that produced by the operational semantics (where the rule for recursion is that, unconditionally, $\mu\, p.F(p) \xrightarrow{\tau} F(\mu\, p.F(p))$ – in other words straightforward unwinding).

The first of these is always essentially the same and involves establishing the congruence result by structural induction on CSP terms in which free process variables are mapped to arbitrary LTS nodes by a function σ. One proves that each term P with its free variables substituted by σ yields an LTS which, when the natural abstraction mapping Φ from LTS nodes to the semantic model is applied, gives the same result as when the denotational semantics is computed with $\Phi(\sigma(p))$ substituted for every free variable p. We can frame this as a formal result:

THEOREM 5.1 *Suppose* **CSP** *the LTS of closed CSP terms under its operational semantics and V is the set of process variables. Then for each CSP term P and each $\sigma : V \rightarrow$ **CSP**, we have $\Phi(\sigma(P)) = \mathcal{R}[\![P]\!](\overline{\sigma})$, where*

- *$\sigma(P)$ means the term P with all its variables p substituted by the appropriate $\sigma[p]$.*
- *$\mathcal{R}[\![P]\!]$ is the denotational meaning of the term P: a mapping that takes an* environment *ρ (a function from P's free process variables to $\mathcal{R}$) and gives a value in $\mathcal{R}$.*
- *$\overline{\sigma}$ is the environment that maps p to $\Phi(\sigma[p])$.*

This result is proved by structural induction over the term P, for all σ simultaneously.

The case where the term P is a process variable is trivial. The other cases constitute the other two parts of the proof discussed above. One part always consists of a series of usually straightforward results: one for each non-recursive operator. We will see an example below.

The recursion case can be challenging in cases where the model contains infinite behaviours (see [25, 26]), but is always essentially straightforward in cases like $\mathcal{R}$ where only

finitary behaviours are used. The essence of the argument is easy to understand, and we have already alluded to it:

- Suppose we have a recursion $\mu\, p.P$. We must think of P as a function from potential values x of p to the value it represents when all p's in P take value x. This applies both in the operational semantics (where the values are LTS nodes) and in the denotational one.

  - For a given σ, the function giving its operational semantics is F_O that maps a given node $Q \in \mathbf{CSP}$ to the term $\sigma'(P)$, where $\sigma'[p] = (SKIP;\ Q)$ and $\sigma'[q] = \sigma[q]$ for $p \neq q$. The reason for using $SKIP;\ Q$ it that it creates the same initial τ as the unwinding rule of recursion. This does not change the $\mathcal{R}$ semantics of a term: $\Phi(\sigma'[p]) = \Phi(Q)$. $F_O(P)$ is then interpreted under the operational semantics of CSP, with the closed terms that substitute process variables being interpreted as themselves.
  - For the same σ, the denotational function F_D maps a member α of $\mathcal{R}$ to $\mathcal{R}[\![P]\!](\overline{\sigma}[\alpha/p])$.

  The unwinding rule for recursion automatically makes the operational semantics of $\mu\, p.F(p)$ a fixed point (in the sense of strong bisimulation) of the operational function F_O. We know that the denotational fixed point δ is $\bigcup\{F_D^n(\Phi(\mathbf{div})) \mid n \in \mathbb{N}\}$. (We have written $\Phi(\mathbf{div})$ here to make a clear distinction between syntax and $\mathcal{R}$-semantics.)

- The structural induction over terms means the operational and denotational semantics of the body P of $\mu\, p.P$ are congruent, or in other words that F_O and F_D are congruent: for any **CSP** node Q we have $\Phi(F_O(Q)) = F_D(\Phi(Q))$.

  This tells us that

  $$F_D(\Phi(\sigma(\mu\, p.P))) = \Phi(F_O(\sigma(\mu\, p.P))) = \Phi(\sigma(\mu\, p.P))$$

  In other words the abstraction under Φ of the operational semantics of the recursion is a fixed point of F_D. the function from which we compute the denotational value δ. We still have to prove it is the right one.

- Because δ is the *least* fixed point of F_D, we know by the above that it is contained in $\Phi(\sigma(\mu\, p.P))$. The reverse containment follows because the unwinding rule of CSP's operational semantics adds a τ step to the computation. This means that every behaviour of $\sigma(\mu\, p.P)$ observable in k actions is also observable of $F_O^{k+1}(Q)$ for any LTS node Q at all, since the first $k+1$ steps of that process are completely independent of Q. ($k+1$ is used rather than k to allow for observing deadlock after a trace with length k, and revivals with a trace this long.)

  This observation of $\Phi(\sigma(\mu\, p.P))$ is therefore present in $\Phi(F^{k+1}(\mathbf{div})) = F_D^{k+1}(\Phi(\mathbf{div}))$, so that all observations are present in $\Phi(F^k(\mathbf{div}))$ for sufficiently large k. Hence

  $$\Phi(\sigma(\mu\, p.P)) \subseteq \bigcup\{F_D^k(\Phi(\mathbf{div})) \mid k \in \mathbb{N}\}$$

  completing the proof.

The lemmas for the individual operators all take the following form. It is only necessary to prove that the revivals components of the left- and right-hand sides are equal, since the traces and deadlocks cases can be deduced from the corresponding congruence theorem for $\mathcal{F}$.

LEMMA 5.2 *Suppose N is any node in the LTS CSP$^+$ of CSP terms where free variables have been instantiated to nodes in another LTS. (So typically $N = \sigma(P)$ for some CSP term P and mapping σ of free variables to the other LTS.) Then*

$$\Phi(N \setminus X) = \Phi(N) \setminus X$$

where the hiding operator on the right is the one defined over $\mathcal{R}$.

We recall the operational semantic clauses for hiding: whenever P can perform an action not in X (including τ), so can $P \setminus X$, and whenever P can perform $a \in X$, $P \setminus X$ can perform τ.

$$\frac{P \xrightarrow{x} P'}{P \setminus X \xrightarrow{x} P' \setminus X} \quad (x \notin X) \qquad \frac{P \xrightarrow{a} P'}{P \setminus X \xrightarrow{\tau} P' \setminus X} \quad (a \in X)$$

We know that the trajectories (operational sequences of states and actions) of of a state $N \setminus X$ are exactly those of N in which all X-actions have been converted to τ and all processes have the operator "$\setminus X$" applied. We also know that the state $M \setminus X$ (the form of all those reachable from $(N \setminus X)$ is stable if and only if N is stable and has no X-action. Now any revival of $N \setminus X$ is of the form (s, Y, a) $(a \notin \{\tau, \checkmark\})$ where there is a trajectory

$$N \setminus X \xrightarrow{x_1} N_1 \setminus X \xrightarrow{x_2} \ldots \xrightarrow{x_n} N_n \setminus X \xrightarrow{a} N_{n+1} \setminus X$$

in which $\langle x_1, \ldots, x_n \rangle \upharpoonright \Sigma^\checkmark = s$ and $N_n \setminus X$ is stable. It easily follows that there are actions y_i such that $y_i = x_i$ if $x_i \neq \tau$ and $y_i \in X \cup \{\tau\}$ if $x_i = \tau$ such that

$$N \xrightarrow{y_1} N_1 \xrightarrow{y_2} \ldots \xrightarrow{y_n} N_n \xrightarrow{a} N_{n+1}$$

is a trajectory of N. Since N_n is known to be stable and refuse $Y \cup X$ is follows that N has a revival of the form $(t, Y \cup X, a)$ where $t \setminus X = s$. This shows that there is a revival of N which maps to (s, Y, a) under the denotational model of $\setminus X$. The argument also works in reverse, which completes the proof of our lemma. ∎

The arguments for all the other operators are very similar to this one.

The combination of these lemmas and the one for recursion constitutes the proof of Theorem 5.1.

5.2 Full abstraction

The classic concept of full abstraction compares an abstract semantics $\mathcal{M}$ of a programming language $\mathbf{L}$ against an underlying operational semantics. It asserts two things:

- $\mathcal{M}$ contains no "junk", namely regions which do not correspond to anything in the operational semantics. Formally this is often stated as saying that the denotations of $\mathbf{L}$ are *dense* (a topological or order-theoretic concept) in $\mathcal{M}$. Because CSP has infinitary syntax if we wish, we can go usually go one better there and show that *every* member of one of its models is denoted by some program. For these CSP models, this part of the result essentially says that the healthiness conditions on the model are strong enough.

- $\mathcal{M}$ distinguishes two objects P and Q if and only if this is necessary to be able to decide if programs built from them do or do not satisfy some simple test or tests. In other words $P \equiv_{\mathcal{M}} Q$ if and only if for all program contexts $C[\cdot]$ we have that $C[P]$ passes these test(s) if and only if $C[Q]$ does.

Thus the traces model is fully abstract with respect to the test "P has the trace $\langle pass \rangle$" for any fixed event *pass*; the stable failures model is fully abstract with respect to the test "P can deadlock immediately"; and the failures/divergences model $\mathcal{N}$ is fully abstract with respect to the test "P can deadlock or diverge immediately" for finitely nondeterministic CSP. For details see [23]. There is further discussion of full abstraction for $\mathcal{N}$ in Section **??** below.

In this section we cover this issue for $\mathcal{R}$.

The first thing we will do is to show that the entire $\mathcal{R}$ model is denotable.

THEOREM 5.3 *For each member $V = (Tr_V, Dead_V, Rev_V)$ of $\mathcal{R}$ there is a closed CSP term (one without free variables) P_V such $\Phi(P_V)$ and the denotational value of P_V (now known to be equal thanks to congruence) are both V.*

We will prove this by representing V as a large nondeterministic composition[9], with one option for each behaviour in its representation.

- If s is a trace in Σ^*, then T_s is defined:

$$\begin{array}{rcl} T_{\langle\rangle} & = & \mathbf{div} \\ T_{\langle a\rangle\hat{}s} & = & \mathbf{div} \,\Box\, a \rightarrow T_s \end{array}$$

  Note that this has no deadlocks or revivals, only the traces which are prefixes of s and therefore implied by the healthiness conditions given that s is present.

- We can extend the above with the rule $T_{\langle\checkmark\rangle} = SKIP$ to deal with traces of the form $s\hat{}\langle\checkmark\rangle$. $T_{s\hat{}\langle\checkmark\rangle}$ then it has no deadlocks no revivals.

- For a deadlock s we define

$$\begin{array}{rcl} D_{\langle\rangle} & = & STOP \\ D_{\langle a\rangle\hat{}s} & = & \mathbf{div} \,\Box\, a \rightarrow D_s \end{array}$$

  This has no revivals, the deadlock s, and those traces implied (**Tr1**) by s being a trace (itself a consequence of **Dead1**).

- The processes defined above are the greatest (under the refinement order) that contain the respective trace or deadlock. There is no such greatest process, for a revival (s, X, a) unless $X \cup \{a\} = \Sigma$, but there is always a maximal such process that refines any V that has (s, X, a). We can extend (s, X, a) to a maximal revival within V as follows:

  - Enumerate the members of $\Sigma - (X \cup \{a\})$ as $\{b_1, \ldots, b_m\}$, and set $X_0 = X$, $Y_0 = \{a\}$.
  - Assuming that (s, X_r, y) is a revival of V for each $y \in Y_r$ (noting this is true for $r = 0$), thanks to healthiness condition **Rev3** we have that *either* $(s, X_r \cup \{b_{r+1}\}, y) \in V_R$ for all $y \in Y_r$, *or* $(s, X_r, b_{r+1}) \in V_R$. In the first case set $X_{r+1} = X_r \cup \{b_{r+1}\}$ and $Y_{r+1} = Y_r$, and in the second set $X_{r+1} = X_r$ and $Y_{r+1} = Y_r \cup \{b_{r+1}\}$.

[9] It should be noted that though this obviously has the potential to yield an infinitary term, it does not go beyond the reaches of the structural induction used to prove congruence, since the syntax tree will have no infinite descending sequence of simpler terms.

– We then have that $(s, X_m, y) \in V_R$ for all $y \in Y_m = \Sigma - X_m$. It follows that all the traces and revivals of the process $R_{(s,X_m)}$ defined

$$\begin{array}{rcl} R_{(\langle\rangle,X)} & = & ?y : \Sigma - X \rightarrow \mathbf{div} \\ R_{(\langle a\rangle^\frown s,X)} & = & \mathbf{div} \,\Box\, a \rightarrow R_{(s,X)} \end{array}$$

belong to V and include the original (s, X, a).

Now define P_V to be the nondeterministic composition of the processes described above for every behaviour in V. It follows from the above that P_V equals V over our chosen model. This establishes that every member of our model is representable in CSP. This completes the proof of Theorem 5.3. ∎

We have taken advantage here of the fact that **div** has as few behaviours as possible. This made our life relatively easy at the expense of creating a process that, objectively speaking, behaves very bizarrely! Something like this is necessary here since $\mathcal{R}$ does contain elements which look a little bizarre (like the individual constructions above) thanks to unseen divergence.

We remarked earlier that the preservation of healthiness conditions could be derived as a corollary. It is the above result which permits this, since it implies that, were there to be a case in which a (binary, say) operator did not preserve them, there would be a CSP term $P \oplus Q$ (with P and Q both closed terms with semantics in the model) which did not satisfy the conditions. However we know that the operational and denotational semantics of this term are congruent, and it is easy to check that the abstraction mapping Φ only creates values in $\mathcal{R}$, so we would have a contradiction.

We now turn to the subject of the sort of test that $\mathcal{R}$ characterises, before moving on to consider its extensions.

The idea of this congruence arose in [8] to model behaviour called *stuckness* (in order to achieve its complement, *stuck-freedom*). In [22] we stated that stable revivals is fully abstract with respect to determining these conditions, and also that it was fully abstract with respect to a related condition *RespondsTo*.

The nature of CCS[10] (the language used in [8]) parallel makes it straightforward to define stuckness there. The parallel composition $(P \mid Q) \setminus X$ is stuck if the unrestricted process $P \mid Q$ can perform a trace of non-X events, and be stable, able to perform an event in X, but no others. For that represents a deadlock state where one of the participants wants to communicate with another.

Clearly that sort of behaviour is instantly recognisable from the revivals of $P \mid Q$. The same effect can be achieved in CSP by renaming all the processes in a network so that every synchronised event is mapped to both itself and a new, special event, say *request*, that is not synchronised. The network is then stuck-free if this renamed version does not have the revival $(s, \Sigma - \{request\}, request)$ for any $s \in (\Sigma - \{request\})^*$.

The *RespondsTo* condition is similar, but with a different motivation relating to the proper behaviour of plug-in components. P *Respondsto* Q if and only if P cannot cause Q to deadlock: Q cannot get into a state where it is not deadlocked, but is waiting solely for P which refuses to respond. This is clearly an asymmetric condition. In the parallel

[10] In CCS, $P \mid Q$ allows P and Q either to synchronise on events they agree on – in which case they are hidden and become τs – or perform them unsynchronised. The same effect as the combination of parallel and hiding internal events in CSP is then achieved by applying the *restriction* operator $\setminus X$ which stops the unsynchronised versions of the common events from happening.

composition $P \underset{J}{\parallel} Q$ it can be expressed formally as follows. There are no revival (s, X, a) of Q and failure (t, Y) of P with $s \restriction J = t \restriction J$, such that $a \in J$ and $(X \cap J) \cup Y = \Sigma^{\checkmark}$. A pair of processes which synchronise on their entire alphabets satisfy *RespondsTo* in both directions if and only if they are stuck-free.

The test we will choose to characterise revivals is the following one:

$\mathbf{T}_{\mathcal{R}}$ P satisfies this test if the parallel composition $P \underset{\{a\}}{\parallel} STOP$ is stuck-free, where we will assume that P uses no event other than a. In other words P fails the test if and only if it has the revival $(\langle\rangle, \emptyset, a)$.

Another way of describing $\mathbf{T}_{\mathcal{R}}$ is to say that P fails it if, on the empty trace, P can stably offer a.

This is in fact precisely equivalent to $STOP$ *RespondsTo* P. It follows that if $\mathcal{R}$ is fully abstract with respect to $\mathbf{T}_{\mathcal{R}}$ then it is with respect to each of stuckness and *RespondsTo* in general.

We said above that it is traditional to judge tests such as this over the operational semantics. Note, however, that we have established a congruence between operational and denotational semantics and this means that it is equivalent to judge it in terms of $revivals(P)$ as described above.

THEOREM 5.4 *Two processes are equivalent over $\mathcal{R}$ if and only if, for all CSP contexts $C[\cdot]$ which restrict the visible events to be within $\{a\}$, $C[P]$ passes $\mathbf{T}_{\mathcal{R}}$ if and only if $C[Q]$ does.*

We know from what we have already done that, for arbitrary P and Q, we can calculate the $\mathcal{R}$-semantics of $C[P]$ and $C[Q]$ from those of P and Q. It follows that if P and Q are equivalent in $\mathcal{R}$ then $(\langle\rangle, \emptyset, a) \in revivals(C[P])$ if and only if $(\langle\rangle, \emptyset, a) \in revivals(C[Q])$. Thus the "only if" half of the theorem is true.

For the "if" half we will show that, for every behaviour that we record in a process's representation in $\mathcal{R}$, there is a context $C(b)[\cdot]$ such that $C(b)[P]$ fails $\mathbf{T}_{\mathcal{R}}$ if and only if P has b. We will write $C(b)$ as $C_{tr}(b)$,$C_{dead}(b)$ or $C_{rev}(b)$ depending on whether b is a trace, deadlock or revival. Below $\parallel$ will mean $\underset{\Sigma}{\parallel}$, parallel with all events synchronised.

- First consider traces of the form $s\hat{}\langle\checkmark\rangle$. Let

  $$C_{tr}(s\hat{}\langle\checkmark\rangle)[P] = ((T_{s\hat{}\langle\checkmark\rangle} \parallel P) \setminus \Sigma);\ a \rightarrow STOP$$

  where $T_{s\hat{}\langle\checkmark\rangle}$ is as defined earlier. This has both the trace $\langle a\rangle$ and our chosen revival precisely when P has $s\hat{}\langle\checkmark\rangle$ as a trace.

- Apparently the simplest type of behaviour is a trace $s \in \Sigma^*$. But, as indicated earlier, the only way we can deal with these is via the interrupt operator. The process $P \triangle SKIP$ has the trace s if and only if it has $s\hat{}\langle\checkmark\rangle$: we then use the method for terminating traces above. So

  $$C_{tr}(s)[P] = C_{tr}(s\hat{}\langle\checkmark\rangle)[P \triangle SKIP]$$

- For a deadlock trace $s \in \Sigma^*$, consider the processes

  $$\begin{aligned} O_{\langle\rangle} &= ?x : \Sigma \rightarrow \mathbf{div} \\ O_{\langle a\rangle\hat{}s} &= (a \rightarrow O_s) \,\Box\, \mathbf{div} \end{aligned}$$

The only point at which O_s is stable is after the trace s. It follows that $(P;\ \mathbf{div}) \parallel O_s$ can deadlock at all only when P can deadlock after s. (The ; **div** is included to prevent deadlock occurring when P terminates after s.) Therefore the context $C_{dead}(s)[P] = ((P;\ \mathbf{div}) \parallel O_s) \setminus \Sigma) \mid\mid\mid a \rightarrow \mathbf{div}$ fails $\mathbf{T}_{\mathcal{R}}$ if and only if $s \in D$.

- Finally consider a revival (s, X, b). Define

$$\begin{aligned} Q^Y_{\langle\rangle} &= ?x : Y \rightarrow STOP \\ Q^Y_{\langle a\rangle\hat{}s} &= b \rightarrow Q^Y_s \end{aligned}$$

This simply steps through the trace s and offers the whole of Y after it. Let R be the renaming that maps all events other than b to a fixed event $c \neq a$, and maps b to a. Then the process

$$C_1[P] = (P \parallel Q_s^{X \cup \{a\}})[\![R]\!]$$

can perform a trace of $\#s$ events (all cs and as) and then offer *only* a from a stable state exactly when P has the revival (s, X, b) – if P offered any element of X along with b, then $C_1[P]$ would offer c as well as a.

If U is any process using only the events a and c, then choose a further event d (so this proof relies on $\mid \Sigma \mid \geq 3$) and consider

$$\begin{aligned} C_2[U] &= U[\![{}^{a,\,d}/_{a,\,a}]\!] \parallel Reg_{\#s} \qquad \text{where} \\ Reg_0 &= a \rightarrow Reg_0 \ \Box\ c \rightarrow Reg_0 \\ Reg_{n+1} &= d \rightarrow Reg_n \ \Box\ c \rightarrow Reg_n \end{aligned}$$

$C_2[U]$ renames all a's that occur before and including the $\#s$th event to d. It follows that

$$C_{rev}(s, X, b)[P] = (C_2[C_1[P]]) \setminus \{d, c\}$$

has the revival $(\langle\rangle, \emptyset, a)$ if and only if P has the trace s (all of whose events are, thanks to the renaming, hidden as d's or c's) and then reach a state where it offers b and no event of X (otherwise the hiding of c would mean the state where a is offered is not stable).

This completes the proof of full abstraction. ∎

6 Revivals and divergence

As indicated in Section 3, we can extend $\mathcal{R}$ in three ways to encompass divergence and infinite traces. Since deadlocks are final and revivals are only intended to reflect a single step of behaviour after a stable state, it is not necessary – and almost certainly impossible, in a congruence – to introduce any richer types of infinite observation into the models. When extending $\mathcal{F}$ to $\mathcal{N}$ it is not necessary to carry the component of finite traces from $\mathcal{N}$ across, because any process which is observed for long enough will always either diverge or become ✓-stable and therefore exhibit a refusal. The situation is not quite so easy with $\mathcal{R}$, since now our process can diverge, deadlock, terminate (✓) or exhibit a revival by moving to a non-deadlocked stable state.

For convenience – and to have an easy representation of the trace set – we choose, for $\mathcal{R}^{\Downarrow}$ etc., to retain the finite trace component.

6.1 Healthiness conditions

Thanks to the existing structures and CSP semantics of $\mathcal{R}$, $\mathcal{U}$ [25] and $\mathcal{SBD}$ [26], we know exactly what the sets of traces, deadlocks, revivals, strict and non-strict divergences and strict and weakly strict infinite traces of any CSP process are. We also know what the strict sets $traces_\perp(P)$, $deadlocks_\perp(P)$ and $revivals_\perp(P)$ are: the ones from $\mathcal{R}$ plus all those associated with strict divergences.

Each of $\mathcal{R}$'s extensions can, however, be regarded as a self-contained model with its own healthiness conditions and CSP semantics. The healthiness conditions for the finite traces, deadlocks and divergences are exactly the same as in $\mathcal{R}$.

- Each member of $\mathcal{R}^{\Downarrow}$ has components $(Tr, Dead, Rev, Div)$ of finite traces, deadlocks, revivals and divergences. The divergences are governed simply by the strictness healthiness conditions:

  DS1 $t \in Div \Rightarrow t^\frown s \in Div$

  DS2 $Div \subseteq Dead$

  DS3 $t \in Div \Rightarrow (s, X, a) \in Rev$

  We do not need a condition stating $Div \subseteq Tr$ because this is implied by **DS2** and **Dead1**.

- Each member of $\mathcal{R}^{\Downarrow,\omega}$ has components $(Tr, Dead, Rev, Div, Inf)$, the same ones plus one of infinite traces, governed by the additional properties

  DS4 $t \in Div \Rightarrow t^\frown u \in Inf$

  Inf1 $t^\frown u \in Inf \Rightarrow t \in Tr$

  plus a closure property we discuss below.

- $\mathcal{R}^{\#}$ has the same components as $\mathcal{R}^{\Downarrow,\omega}$ but different healthiness properties: since it is not divergence strict it replaces **DS1-4** by the weak strictness property

  WS $\overline{Div} \subseteq Inf$

  where $\overline{X}$, for trace set X, is the set of infinite traces with infinitely many prefixes in X. This model also uses the closure property discussed below.

We need a closure property to ensure that the set of infinite traces is consistent with what we know must be possible from the finite information available – broadly speaking, there are enough infinite traces to be consistent with what the revival information allows us to force. This subject was discussed at considerable length, for example in [25, 2], when $\mathcal{U}$ was initially described. There were many formulations of the required property. Most of these were specific to the language (namely failures) of $\mathcal{U}$, however there is one that is more general. We say a process P is *closed* if $infinites(P) = \overline{traces(P)}$. The following re-states a principle discussed on page 257 of [23] in slightly more general language:

Closure Each process is the nondeterministic choice of closed processes.

Over $\mathcal{R}^{\Downarrow,\omega}$, nondeterministic choice is simple component-wise union. Over $\mathcal{R}^{\#}$ we need, in general, to close up under weak strictness after taking the union. However we establish the following result:

THEOREM 6.1 *Every member of $\mathcal{R}^{\#}$ is the component-wise union of closed processes.*

The proof is analogous to that of Lemma 1.1 in [25]. All one has to show is that, if $P = (Tr, Dead, Rev, Div, Inf)$ is the WS-closure of the component-wise union (which we will write $\bigcup \mathbf{X}$) of a nonempty set $\mathbf{X}$ of closed processes, and $u \in Inf$, then there is a closed process P_u containing u such that $P \sqsubseteq P_u$. This is because we can then define

$$\mathbf{X}_u = \mathbf{X} \cup \{P_u \mid u \in Inf\}$$

and clearly the component-wise union of the $\mathbf{X}_u$ is P.

We construct P_u via a series of closed processes Q_i. We know that, for every finite prefix t of u, there is a process $P_t \in \mathbf{X}$ that has trace t. (In fact we know that there is a P'_t that has an extension of t as a divergence.)

Define Q_0 to be any member of $\mathbf{X}$.

Suppose we have defined Q_r. Then either Q_r contains u or, because it is closed, it contains a longest prefix t_r of u. In the first case we can set $P_u = Q_r$. In the second case, let a be the next element of u after t_r. We now define Q_{r+1} to have all behaviours of Q_r together with just some of those of $P_{t_r \hat{} \langle a \rangle}$, namely those that begin with the trace $t_r \hat{} \langle a \rangle$. Q_{r+1} is closed as the union of a pair of closed sets, and a little analysis shows it satisfies the other healthiness properties.

If we have not already defined P_u (though the case where $u \in Q_r$) then u is not a behaviour of any member of the Q_r. In that case Q^*, their component-wise union, is closed except that it does not contain u. This is because all prefixes of an infinite trace $w \neq u$ must all be contained in Q_r, where r is minimal such that $t_r \not\leq w$. It follows that Q^* with the addition of u is closed, and that u is the only behaviour it has that is not in $\bigcup \mathbf{X}$, which shows that $P \sqsubseteq P_u$. This completes the proof of Theorem 6.1. ∎

We might note that all finite-state processes (ones with only a finite numbers of states in the LTS/operational semantics) and ones built from finitely nondeterministic CSP, as discussed earlier, are closed.

The author is confident that using the **Closure** principle in the way formulated here will be the key to the problem of formulating more difficult models where we need richer infinite structures than infinite traces, such as $\mathcal{R}^{\Downarrow,\omega}$. The definition of closed process would then be modified so that every infinite behaviour, that is a limit of finite ones of the process, is present.

6.2 CSP semantics

There is no need to give three more complete semantics for CSP here. The semantic clauses for divergences and infinite traces are – for $\mathcal{R}^{\Downarrow}$ and $\mathcal{R}^{\Downarrow,\omega}$ – exactly the same[11] as over $\mathcal{N}$, $\mathcal{U}$ (both [23]) and for $\mathcal{R}^{\#}$ they are the same as $\mathcal{SBD}$ [26]. The clauses for $traces(P)$ $deadlocks(P)$ and $revivals(P)$ for $\mathcal{R}^{\#}$ are identical to those given above, and the clauses for $traces_{\perp}(P)$, $deadlocks_{\perp}(P)$ and $revivals_{\perp}(P)$ for $\mathcal{R}^{\Downarrow}$ and $\mathcal{R}^{\Downarrow,\omega}$ are identical except that

[11] In all cases where $traces(P)$ is used in one of these definitions, we use the definition in terms of other components given above.

they are closed up under divergences, for example

$$\begin{aligned}
\mathit{deadlocks}_\perp(P \setminus X) &= \{s \setminus X \mid s \in \mathit{deadlocks}_\perp(P)\} \\
&\quad \cup \mathit{divergences}(P \setminus X) \\
\\
\mathit{revivals}_\perp(P \,\Box\, Q) &= \{(\langle\rangle, X, a) \mid (\langle\rangle, X) \in \mathit{failures}_\perp(P) \cap \mathit{failures}_\perp(Q) \\
&\qquad \wedge (\langle\rangle, X, a) \in \mathit{revivals}_\perp(P) \cup \mathit{revivals}_\perp(Q)\} \\
&\quad \cup \{(s, X, a) \mid (s, X, a) \in \mathit{revivals}(P) \cup \mathit{revivals}_\perp(Q) \wedge s \neq \langle\rangle\} \\
&\quad \cup \{(\langle\rangle, X, \checkmark) \mid X \subseteq \Sigma \wedge \langle\checkmark\rangle \in \mathit{traces}(P) \cup \mathit{traces}(Q)\} \\
&\quad \cup \{(s, X, a) \mid s \in \mathit{divergences}(P \,\Box\, Q) \wedge a \notin X\}
\end{aligned}$$

The fixed points for calculating recursion exactly parallel the ones use in the respective failures-based model.

6.3 Full abstraction

Any closed process can be expressed in (infinitary) CSP in a style based on that used for $\mathcal{N}$ in Section 9.3 of [23] over either of $\mathcal{R}^{\Downarrow,\omega}$ and $\mathcal{R}^{\#}$. This is key to the expressibility result for these models, which is an analogue of Theorem 5.3.

THEOREM 6.2 *Every element of $\mathcal{R}^{\Downarrow}$, and every closed element of $\mathcal{R}^{\Downarrow,\omega}$ and $\mathcal{R}^{\#}$ is expressible in finitely nondeterministic (though infinitary syntax) CSP.*

The proof of this follows the same pattern as that given on page 235 of [23] for $\mathcal{N}$. We build a mutual recursion indexed, essentially, by members of the semantic model we are expressing. We can deal with the first two models at one stroke, as there is a natural 1–1 correspondence between the closed elements of $\mathcal{R}^{\Downarrow,\omega}$ and the whole of $\mathcal{R}^{\Downarrow}$. We therefore only give a single definition for these two classes of process. In the definition below, ξ abbreviates an arbitrary $(\mathit{Tr}, \mathit{Dead}, \mathit{Rev}, \mathit{Div}) \in \mathcal{R}^{\Downarrow}$.

$$\mathit{RSD}(\xi) = \begin{cases}
\textbf{div} \quad \text{if } \langle\rangle \in \mathit{Div}, \text{ and otherwise} \\
(?x : (\mathit{initials}(\xi) - \{\checkmark\}) \to \mathit{RSD}(\xi/\langle x\rangle)\}) \\
\rhd \\
\sqcap(\{\mathit{STOP} \mid \langle\rangle \in \mathit{Dead}\} \\
\quad \cup \{\mathit{SKIP} \mid \langle\checkmark\rangle) \in \mathit{Trv}\} \\
\quad \cup \{?x : (\Sigma - X) \to \mathit{RSD}(\xi/\langle x\rangle) \mid (\langle\rangle, X, a) \in \mathit{Rev}, a \neq \checkmark, \\
\qquad \nexists X' \supset X.(\langle\rangle, X', a) \in \mathit{Rev}\})
\end{cases}$$

We can paraphrase this definition as follows. We are creating an interpreter for the revivals-and-strict-divergences model. The immediate behaviour of any process is determined by what it does on the empty trace, and after any initial event a we change the parameter to $\xi/\langle a\rangle$, the process consisting of all ξ's behaviours on traces starting with a, but deleting the initial a.

If the process diverges immediately, then thanks to strict divergence we can just set the process equal to **div**. Otherwise, it certainly has $\langle\rangle$ as a deadlock or a revival whose trace is $\langle\rangle$, so the nondeterministic choice above is nonempty. In that case the definition looks to see whether our process can deadlock, terminate or perform events from stable states, and reproduces the corresponding behaviour.

There are three important things to notice about this definition:

- Firstly, despite the use of $\sqcap$, it only uses finite nondeterminism. This is because, since Σ is finite, there only finitely many revivals possible on $\langle\rangle$.

- Secondly, interpreted over $\mathcal{R}^{\Downarrow,\omega}$, it naturally creates a closed process. This is because there is only one basic state it can reach on any finite trace s, namely $RSD(\xi/s)$, and so, if u is any infinite trace all of whose finite prefixes are traces, the definition $RSD(\xi)$ has a trajectory that reaches each of these states in turn – meaning that it can perform u.

- Thirdly, $\rhd$ is used in a fundamental way. We need to be able to express the fact that all of the events in $initials(\xi) - \{\checkmark\}$ can occur, but not in a way that implies that any of them can occur in a stable state, since those a which lack the revival $(\langle\rangle, \emptyset, a)$ cannot. This distinction cannot be made over failures-based models, which explains why $\rhd$ is treated as primitive in this paper.

The same structure (and argument for closure) works for closed elements of $\mathcal{R}^{\#}$, where again infinite traces tell us nothing and so can be disregarded. The only difference is that now divergence becomes one of the standard options, rather than a special case. The following is the corresponding "interpreter" for revivals with non-strict divergences.

$$RNSD(\xi) \quad = \quad \left\{ \begin{array}{l} (?x : (initials(\xi) - \{\checkmark\}) \to RNSD(\xi/\langle x\rangle)\}) \\ \rhd \\ \sqcap(\{\mathbf{div} \mid \langle\rangle \in \ div\} \\ \quad \cup \{STOP \mid \langle\rangle \in Dead\} \\ \quad \cup \{SKIP \mid (\langle\checkmark\rangle \in Tr\} \\ \quad \cup \{?x : (\Sigma - X) \to RNSD(\xi/\langle x\rangle) \mid (\langle\rangle, X, a) \in Rev, \\ \qquad \nexists\, X' \supset X.(\langle\rangle, X', a) \in Rev\}) \end{array} \right.$$

This completes the proof of Theorem 6.2. ∎

This result, together with the closure principle and the use of nondeterministic choice, trivially give us the following:

COROLLARY 6.3 *Every member of $\mathcal{R}^{\Downarrow,\omega}$ and $\mathcal{R}^{\#}$ is expressible in infinitary CSP.*

These expressibility results naturally lead us to ask with respect to what tests $\mathcal{R}^{\Downarrow}$, $\mathcal{R}^{\Downarrow,\omega}$ and $\mathcal{R}^{\#}$ are fully abstract. Just as with the pair $\mathcal{F}^{\Downarrow} = \mathcal{N}$ and $\mathcal{F}^{\Downarrow,\omega} = \mathcal{U}$, we must expect that the first two have the same full abstraction property for boundedly and unboundedly nondeterministic CSP respectively.

We proceed by analogy with $\mathcal{F}$ and the pair $\mathcal{N}$ and $\mathcal{U}$, which are respectively fully abstract with respect to the tests:

$\mathbf{T}_{\mathcal{F}}$ is failed by P if P can deadlock immediately;

$\mathbf{T}^{\Downarrow}_{\mathcal{F}}$ is failed by P is P can either deadlock or diverge immediately.

It is therefore natural to speculate that $\mathcal{R}^{\Downarrow}$ and $\mathcal{R}^{\Downarrow,\omega}$ are fully abstract (for their respective languages) with respect to the test

$\mathbf{T}^{\Downarrow}_{\mathcal{R}}$ is failed by P if P can either diverge or stably offer a on the empty trace.

Since the models do yield this information and are congruences for their respective CSP's, they are certainly capable of determining from P's value whether $C[P]$ satisfies this test for an arbitrary context. It turns out, however, that this test alone is not strong enough to distinguish between all pairs of processes, for example

$$P_1 = a \rightarrow \mathbf{div} \qquad \text{and} \qquad P_2 = STOP \sqcap (a \rightarrow \mathbf{div})$$

The problem here is that we would need $C[P_2]$ to make a stable offer because P_2 can deadlock immediately – the only behaviour that distinguishes it from the P_1. That is entirely possible in itself, since we could use the context

$$C'[P] = (F(P) \,\Box\, a \rightarrow STOP) \setminus \{b\}$$

where $F(P)$ renames all P's initial events to b and all subsequent ones to a – achievable using double renaming (such as used in the definition of C_2 in the proof of Theorem 5.4). This process has the stable revival $(\langle\rangle, \emptyset, a)$ if and only if P can deadlock on $\langle\rangle$. However, $C'[P]$ fails $\mathbf{T}^{\Downarrow}_{\mathcal{R}}$ whenever P can diverge on its second step, meaning that $C'[\cdot]$ fails to distinguish P_1 and P_2 since $C'[P_1]$ and $C'[P_2]$ both fail $T^{\Downarrow}_{\mathcal{R}}$. Note that anything like the test for a deadlock trace given for $\mathcal{R}$ in the proof of Theorem 5.4 will also create this difficulty. It does not seem likely to the author that any other context can be devised that overcomes this problem.

What we seem to need to do is use a pair of tests:

THEOREM 6.4 *$\mathcal{R}^{\Downarrow}$ and $\mathcal{R}^{\Downarrow,\omega}$ are, with respect to their versions of CSP, fully abstract with respect to the pair of tests $\mathbf{T}^{\Downarrow}_{\mathcal{R}}$ and $\mathbf{T}^{\Downarrow}_{\mathcal{F}}$.*

We know that our models have sufficient information in the semantics of P to allow us to determine whether an arbitrary CSP context $C[P]$ meets each of these tests. We also know that $T^{\Downarrow}_{\mathcal{F}}$ allows us to distinguish any pair of processes (for example (P_1, P_2) from above that confound $T^{\Downarrow}_{\mathcal{R}}$) that are distinguished in $\mathcal{N}$ or $\mathcal{U}$ as appropriate. We can therefore restrict our attention to a pair of processes $P \neq Q$ with identical behaviour in failures, divergences and finite/infinite traces. Without loss of generality we can therefore assume that P has a revival (s, X, b) that Q does not have, even though it has both the failure (s, X) and the trace $s^\frown\langle b\rangle$. We can also assume that s is not a divergence. We can now use the same context $C_{rev}(s, X, b)$ devised in the proof of Theorem 5.3 with the property that, for any process P such that $s \notin \mathit{divergences}_{\perp}(P)$, $C_{rev}(s, X, a)[P]$ has the revival $(\langle\rangle, \emptyset, a)$ if and only if P has (s, X, b). This completes the proof of Theorem 6.4. ∎

We finally move on to the model $\mathcal{R}^{\#}$ that does not have strict divergence. The corresponding failures-based model, $\mathcal{F}^{\#}$ is as shown in [20], fully abstract with respect to the pair of tests $\mathbf{T}_{\mathcal{F}}$ and $\mathbf{T}_{Div}$ (which is failed if P diverges immediately). Given this, and the reasoning above, it is straightforward to deduce the following:

THEOREM 6.5 *$\mathcal{R}^{\#}$ is fully abstract with respect to the three tests $T_{\mathcal{R}}$, $T_{\mathcal{F}}$ and T_{Div}.*

The divergence test is or-ed into the finite tests for the models with strict divergence because the strictness principle means it is impossible to tell, for a divergence trace, what a process can do from its value in the abstract model. When we drop this principle we can see other behaviours separately from divergence.

The full abstraction results quoted so far are all in terms of what tests a given process *may* fail. There is no guarantee that a process that *can* fail a given test will actually do

so. We can also look at models from the point of view of tests that we definitely want to succeed; this is actually what we are likely to want of a process. A process is guaranteed to pass the test $T_{\mathcal{F}}^{\Downarrow}$ when it can neither deadlock nor diverge immediately. This corresponds to one reason why $\mathcal{N}$ is fundamental: it is the weakest congruence where, for any trace s and nonempty set of events X, we can tell from a process's value whether it is *guaranteed* to accept an event from X.

We can clearly tell the same things from a process's value in $\mathcal{R}^{\Downarrow}$, but we can additionally guarantee that, if left to become stable, it *will* deadlock or *will not* offer some banned event, without banning it communicating such events from unstable states. It is clear how this corresponds to detecting the stuckness and *RespondsTo* conditions we discussed earlier – each of these is something that happens because of an offer from a stable state.

There seems to be a strong sense in which revivals models are primarily useful in cases where we are particularly interested in what it offers on stable states as opposed to what it can communicate in general.

We can also make an interesting distinction here with refusal testing models. Revivals allow us to see *static* offer behaviour as required for the applications discussed above. Imagine, however, that we want to run a process in such a way that all visible events come from ✓-stable states. This might either be because of the underlying properties of the implementation or because the environment chooses only to communicate once stability is observed, perhaps out of caution. Examples of the former behaviour are Statemate Statecharts [11, 28] and discrete timed models of CSP such as [17]. We can regard this requirement as the need to see stability *dynamically*, and perhaps encapsulate it in a new operator $stable(P)$ that only allows communications other than ✓ from stable states. We would usually expect to need divergence information when considering this operator, for a divergent process may never reach stability.

We certainly cannot model this operator in failures-based models, but it is tempting to think we ought to be able to do so in revivals since it allows us to observe what offers are made stably. This is, however, deceptive since revivals cannot distinguish behaviour that follows stable and unstable instances of an event if both are possible. Consider, for example

$$(a \rightarrow a \rightarrow STOP) \rhd (a \rightarrow STOP) \quad \text{and} \quad (a \rightarrow STOP) \rhd (a \rightarrow a \rightarrow STOP)$$

stable ought to give, respectively, $a \rightarrow STOP$ and $a \rightarrow a \rightarrow STOP$ when applied to these processes, but actually they are indistinguishable in revivals models. We can, however, compute *stable* over refusal testing models – simply retain only those behaviours

$$\langle X_0, a_0, X_1, a_1, \ldots, a_n, X_{n+1} \rangle$$

such that all X_i except X_{n+1} are restricted to be not equal to $\bullet$. The author conjectures that $\mathcal{RT}$ and its extensions are, in useful senses, fully abstract with respect to computing $stable(P)$. ¶

6.4 Potential applications

We have seen that the stable revivals model $\mathcal{R}$ is the right one for reasoning precisely, but without extraneous detail, about the offers made and refused in individual stable states of processes or networks. It follows that the three models involving both revivals and infinite behaviours when we want to reason both about these things and to limit what infinite behaviours can arise.

Any model of a CSP-style system that is intended to provide a complete description must involve divergence, since the failure to involve it means that **div** is the most refined process – and we would hardly expect **div** to be adequate for any, let alone all, practical purposes. It follows that if we want a single model of a system to provide a comprehensive description from the point of view of correctness then either it must encompass divergence or we must have a separate proof the divergence is absent. In the first of these cases, if we want to analyse the system from this description for the type of property in which stable revivals are key, then the model used must encompass both revivals and divergence.

Just as, with failures, $\mathcal{N} = \mathcal{F}^{\Downarrow}$ is normally adequate for most reasoning purposes for finite-state processes, we expect that normally the right model to use for this purpose will be $\mathcal{R}^{\Downarrow}$. If it is necessary to reason about unboundedly nondeterministic processes, then normally $\mathcal{R}^{\Downarrow,\omega}$ will suffice.

There is, however, a potential application for $\mathcal{R}^{\#}$ in same sort of application from which the notation of stuck-freeness arose, namely operating system analysis. One can imagine that an operating system is, in a sense, a context $C[\cdot]$ in which its application programs run. If one is designing the part of the operating system devoted to closing down the system (i.e. what happens after one presses the "shut down" button) it might be of no importance what the system does, even having the possibility of diverging, before such a button is pressed. However after that button is pressed, you must guarantee that the system closes down cleanly without being able to diverge. There is a clear role for $\mathcal{R}^{\#}$ in such analyses where part of the proof of clean termination involves an analysis for stuckness.

7 Normal forms and algebraic semantics

In this section we study how the move to revivals models and others finer than failures affects the algebraic semantics of CSP. We discover that it implies a rather large-scale change to the structure of the semantics, and set out a programme for developing an algebraic framework that should cover all the models discussed in this paper.

The objective of an algebraic semantics is to characterise process equivalence as precisely the relation provable between process terms using a set of algebraic laws about the operators in a language, for example symmetric, associative, distributive and idempotence properties, together with one or more rules to handle recursive terms. For example, the idempotence law of $\sqcap$ readily proves that the terms $a \rightarrow STOP$ and $(a \rightarrow STOP) \sqcap (a \rightarrow STOP)$ are equal, but the similarly obvious equality between the recursions $\mu\, p.a \rightarrow p$ and $\mu\, p.a \rightarrow a \rightarrow p$ needs some rule other than a standard algebraic law to prove equality.

In traditional models of CSP, as described in Chapter 11 of [23], the algebraic semantics takes the following form:

- When each new operator is introduced, a number of algebraic laws are given for it to help describe it.
- Which laws are important for the algebraic semantics depends on whether the new operator is one that is used in the normal form, or one that has to be eliminated.
- For normal form operators it is vital that enough algebraic laws are given to enable all the nuances of whatever equivalence is being studied to be proved.
- For other operators what we need are the laws that enable the given operators to be eliminated from finite terms (ones with no recursion). We need to classify the different

types of immediate behaviours an operator might encounter, and how it handles them. One such form of behaviour is nondeterminism, which is handled using distributive laws such as

$$P \underset{X}{\parallel} (Q \sqcap R) = (P \underset{X}{\parallel} Q) \sqcap (P \underset{X}{\parallel} R) \qquad \langle \underset{X}{\parallel}\text{-dist} \rangle$$

We also need laws to handle termination and divergence (the latter, of course, being very different depending on whether the model we are studying is divergence strict or not).

Completing the set, for all the models studied in [23], is a a *step* law for each such operator, namely a description of how it transforms initial visible steps of the arguments to our operator into initial steps (visible or otherwise) of the compound, and how the successor states are computed. These laws have a close resemblance to some of the clauses of an operational semantics. The step law of hiding is as follows: the reader might like to compare it to the operational semantics given in Section 5.1.

$$(?x : A \to P) \setminus X = \begin{cases} ?x : A \to (P \setminus X) & \text{if } A \cap X = \emptyset \\ (?x : (A - X) \to (P \setminus X)) & \\ \quad \triangleright \sqcap \{(P[a/x]) \setminus X \mid a \in A \cap X\} & \text{if } A \cap X \neq \emptyset \end{cases} \qquad \langle \text{hide-step} \rangle$$

- A normal form is described in which there is a unique syntactic form representing each different semantic value, or at least each different semantic value of a finitary subclass of processes.
- Either a strategy is given for transforming *all* processes to normal form, or this is only done for finite processes and one or more rules are introduced to cover other processes in terms of finite approximations.

7.1 Normal forms

A normal form is a standardised way of expressing a process that depends only on its value in our chosen semantic model. In that sense all of the forms we have used earlier for *expressing* semantic values in CSP are normal forms – patently they depend only on a process's semantics. In this paper we have seen two distinct styles of structuring these forms: there is the style used for finite-behaviour models like $\mathcal{R}$ in which each individual behaviour is assigned its own process, and these are then combined by nondeterministic choice; and there is the sort we used for $\mathcal{R}^{\Downarrow}$ and other models with divergence, in which we define a sort of formal interpreter that only has one main state per trace. We might reasonably call these styles eager-choice and late-choice normal forms, since the former represents a process as the immediate nondeterministic choice of highly refined processes and the latter offers no meaningful nondeterministic choice at all except at the latest possible moment.

Usually, when constructing an algebraic semantics, the late-choice form is used. This approach has a great deal in common with the idea of *determinisation* much used in model checking: we take a process that may have many ways of performing a given trace and transform it into a structure where it can only reach a single state where all the ways it can choose to act after that state are recorded. Indeed, this type of normal form is vital to the

refinement checking of FDR: the specification process (P where we are checking $P \sqsubseteq Q$) is always normalised so that, for every trace of Q, we can readily check if its behaviours are possible for the single state representing that state in $normal(P)$.

We have therefore, in Section 6.2, already seen the normal form for $\mathcal{R}^{\Downarrow}$: it is the form used when describing how to express every term. Note how it

(i) Sets out the set of initial events of each process in a standard way.

(ii) Explicitly introduces deadlock or termination via $STOP$ or $SKIP$ when the semantics demands this.

(iii) Gives terms that express every revival after the empty trace in a standardised way.

(iv) After any particular initial event a, behaves in exactly the same way P_a no matter which part of the syntax a was chosen from. This is vital to ensure that only one main state can be reached in the normal form for any trace.

The natural late choice normal form for $\mathcal{R}$ is very similar. We describe it here in an inductive form:

- If I is any subset of Σ and P_x is in normal form for every $x \in I$ then both of

  $$(?x : I \to P_x) \rhd \mathbf{div} \qquad \text{and} \qquad (?x : I \to P_x) \rhd SKIP$$

  are in normal form.

- Suppose I and P_x are as above, and ACC is a nonempty subset of $\mathcal{P}(I)$ (the powerset of I) subject to the condition that for each nonempty $A \in ACC$ there must be some $x \in A$ that does not belong to any $B \in ACC$ such that $B \subset A$. (Thus each A either gives a representation of deadlock or some revival not captured by a smaller B.) Then both of

  $$(?x : I \to P_x) \rhd \sqcap\{?x : A \to P_x \mid A \in ACC\} \quad \text{and}$$
  $$(?x : I \to P_x) \rhd (SKIP \sqcap \sqcap\{?x : A \to P_x \mid A \in ACC\})$$

  are in normal form.

These are identical to the types of behaviour allowed by the $\mathcal{R}^{\Downarrow}$ normal form except that here **div** appears together with the initial events representing a node that cannot become $\checkmark$-stable, whereas in the strict divergence model it is simply a normal form in itself representing all processes that are able to diverge immediately.

While not all the non-revivals models described in this paper have published normal forms (as far as we are aware), we do not attempt to construct them here. One thing we ought to remark on is that the late-choice style of normal form can only work for models in the classes that do not need non-closed processes, for the simple reason that every late-choice normal form automatically constructs a closed process.

Using algebraic laws one can only guarantee to transform recursion-free finite terms to normal form in a finite time. There are several strategies for handling terms with recursion, all involving one or more infinitary rules. One is quoted in [23], another involves the idea of *syntactic approximations*: we obtain a syntactic approximation to a term by unwinding its recursions finitely and then replacing all remaining recursions by the bottom process. P then refines Q if and only if, for all syntactic approximations Q' of Q, there is one of P such that $P' \sqcap Q' = Q'$ (equivalent to $Q' \sqsubseteq P'$). Of course P and Q are equivalent ($P = Q$) if and only if they refine each other.

7.2 Transformation to pre-normal forms

In some cases, as in Chapter 11 of [23], the transformation to normal form is done in two or more stages, and we can term the intermediate forms *pre-normal forms*.

One would expect the set of detailed laws about choice operators to differ in moving from failures-style models to revivals-style ones. After all, there have to some equivalences between processes that are provable in the first case but not the other, such as between the terms following terms that are equivalent over $\mathcal{F}$ but not over $\mathcal{R}$.

$$(a \rightarrow STOP \,\Box\, b \rightarrow STOP) \setminus \{b\} \quad \text{and} \quad (a \rightarrow STOP) \sqcap STOP$$

We will discuss this issue in Section 7.3.

We might hope, however, that the reduction to pre-normal form goes through pretty much the same as in previously-studied models. On examining how the reduction strategy set out in [23] for $\mathcal{N}$ might work for $\mathcal{R}^{\Downarrow}$, however, it quickly becomes apparent that there is a major hole: the categories of law for a non-normal-form operator quoted above no longer cover all possible situations. This is because, as we have already noted several times, revivals-based models allow us to distinguish in general between events that happen from stable states and those that can only occur in non-stable states after a given trace. The reduction strategy for $\mathcal{N}$ in [23] involves expanding any non-divergent process out to the form of a nondeterministic choice of $\checkmark$-stable choices, using distributive laws to move the nondeterminism to the outside and then using termination and step laws on the $\checkmark$-stable choices.

This expansion is not possible in revivals model because a process P can have actions that (in all processes equivalent to P) can never occur from a stable state. Just as in the two revivals-based normal forms studied in the previous section, we can expect to see terms of the form

$$P = (?x : A \rightarrow P'_x) \rhd P''$$

appear in our reduction strategy. We will need new families of laws to handle these. The author conjectures that it will be possible to construct additional laws to do this. To be ¶ more precise, for any unary (one place) operator we will need a law which covers this form, such as

$$P \setminus X = \begin{cases} (?x : A \rightarrow (P'_x \setminus X)) \rhd (P'' \setminus X) & \text{if } A \cap X = \emptyset \\ (?x : (A - X) \rightarrow (P'_x \setminus X)) & \\ \rhd \ (P'' \setminus X \sqcap \bigsqcap \{P'_x \mid x \in A \cap X\}) & \text{otherwise} \end{cases}$$

where P is as above.

For any binary operator, we should need laws that show how this form P combines with each of its own form, $SKIP$, $?x : A \rightarrow Q_x$ and in some cases **div**. The objective should ¶ be a set of laws which, with different sets of **div** laws for the three families represented by $\mathcal{R}$, $\mathcal{R}^{\Downarrow}$ and $\mathcal{R}^{\Downarrow,\omega}$, $\mathcal{R}^{\#}$, are true in all linear observation models and can reduce any finite process to a pre-normal form constructed using only $SKIP$, **div**, prefix-choice, the form $?x : A \rightarrow P_x \rhd P'$, and nondeterminism.

It is reasonable to expect that such a set of laws exist because (i) conventional step laws, and the additional classes described above, are closely based on operational semantic rules, (ii) distributivity, as discussed earlier, is a natural consequence of using CSP-style

operators over models based on sets of linear observations, and (iii) the two separate styles of laws for **div** derive fundamentally from the ways in which divergence is represented (or not) in the two families of model.

The details of different styles of finite observation such as traces, revivals or refusal testing should then be captured using variations on the laws of these last five constructs.

7.3 Laws at the boundaries of models

Although we have seen that the original step laws of CSP are no longer general enough for normal form transformation, they and almost all of the standard algebraic laws of CSP (see [23]) hold over $\mathcal{R}$ and $\mathcal{R}^{\Downarrow}$. The only exception other than some of those laws which were already in [23] specific to certain model(s) is the one which says that internal choice distributes over external choice, namely

$$(P \,\Box\, Q) \sqcap R = (P \sqcap R) \,\Box\, (Q \sqcap R) \quad \langle \sqcap\text{-}\Box\text{-dist} \rangle$$

To see that this law fails over revivals consider the processes

$$(a \rightarrow STOP \,\Box\, b \rightarrow STOP) \sqcap STOP \quad \text{and}$$
$$((a \rightarrow STOP) \sqcap STOP) \,\Box\, ((b \rightarrow STOP) \sqcap STOP)$$

which would have to be equal if this law were true. The first, however, does not have the revival $(\langle\rangle, \{a\}, b)$ while the second does if its left-hand choice resolves to the left and its right-hand one resolves to the right.

In [23], the law $\langle \sqcap\text{-}\Box\text{-dist} \rangle$, played a vital role in a part of the normal form strategy that we still expect to be true for revivals, namely ensuring that the successor process for any given event does not depend on which part of the structure the event appears in.

So we cannot just discard $\langle \sqcap\text{-}\Box\text{-dist} \rangle$, rather we have to replace it with a law encapsulating a weaker principle. This essentially has to record that after an event our model has *no memory* of the acceptance set from which it was drawn.

It has been said that the two laws which most characterise failures-style equivalences are $\langle \sqcap\text{-}\Box\text{-dist} \rangle$ and the distributive law of prefixing, one version of which is

$$?x : A \rightarrow (P_x \sqcap Q_x) = (?x : A \rightarrow P_x) \sqcap (?x : A \rightarrow Q_x)$$

What this says is that one cannot tell whether an internal choice was made before or after a visible event. It is not true under bisimulation-like equivalences. The best form of no-memory law the author has found generalises this distributive law. It extends the right-hand side of the above to the case where the input sets may be different.

$$(?x : A \rightarrow P_x) \sqcap (?x : B \rightarrow Q_x) \;=\; (?x : A \rightarrow R_x) \sqcap (?x : B \rightarrow R_x)$$
$$\text{where} \qquad R_x = \begin{cases} P_x & \text{if } x \in A - B \\ Q_x & \text{if } x \in B - A \\ P_x \sqcap Q_x & \text{if } x \in A \cap B \end{cases}$$

This is true in all the CSP models mentioned in this paper except those based on refusal testing $\mathcal{RT}$.

The author conjectures that, modulo the required extensions to step laws (which will be true in all models), the replacement of $\langle \sqcap\text{-}\Box\text{-dist} \rangle$ by the above law exactly captures the difference between failures and revivals based equivalences. ¶

We might ask how how the other equivalences discussed at the beginning of this paper such as $\mathcal{RT}$ and $\mathcal{A}$ will differ. Since both of these equivalences are finer than $\mathcal{R}$, we know that $\langle \sqcap\text{-}\Box\text{-dist} \rangle$ is not true there either, but we must suspect that additional, and different, laws will fail for these two. The answer for $\mathcal{A}$ is a little unexpected. As a consequence of the idempotence and symmetry of $\Box$ and $\sqcap$, and the distributivity of external over internal choice – which all are true over $\mathcal{R}$ – we get

$$\begin{array}{rcl} P \sqcap Q & = & (P \sqcap Q) \,\Box\, (P \sqcap Q) \\ & = & (P \,\Box\, P) \sqcap (P \,\Box\, Q) \sqcap (Q \,\Box\, P) \sqcap (Q \,\Box\, Q) \\ & = & P \sqcap Q \sqcap (P \,\Box\, Q) \end{array}$$

This is obviously not true in $\mathcal{A}$, since if $P = a \rightarrow STOP$ and $Q = b \rightarrow STOP$, the right-hand side has a ready set $(\{a, b\})$ that the left-hand side does not have. It follows that (at least) one of the laws used must be false over $\mathcal{A}$. The law in question is the idempotence of $\Box$: this can fail for a nondeterministic argument.

The author conjectures that removing this law ($\langle \Box\text{-idem} \rangle$) from those of revivals models will capture acceptance-set equivalences. ¶

$\langle \Box\text{-idem} \rangle$ is true over refusal testing models, but the no-memory principle is not true in general. The author suspects that the characterisation of this model will either be standard laws minus $\langle \sqcap\text{-}\Box\text{-dist} \rangle$ or these plus a much weakened version of the no memory principle. The main point about this model is that it the more refusal an observer has observed during trace s, the more refined is his view of the process on all longer traces. ¶ ¶

8 The hierarchy revisited

Since introducing the hierarchy of CSP models in Section 3, we have learned a lot more about it and discovered new models. In this section we will see that, at least for the more abstract end of the spectrum of models, we have actually completed the picture.

Recall that a *congruence* is a notion of equivalence for processes that is compositional under all operators of a language. It does not have to provide a solution in itself for recursions, but must satisfy the unwinding rule for recursions. Each of our semantic models induces a congruence for CSP. We can extend the congruences to cover arbitrary processes drawn from all possible finite alphabets Σ by defining two processes to be equivalent if they are equated by the appropriate model constructed over any sufficiently large Σ: it is obvious that the equivalences induced by each type of model are the same for any alphabet that contains all the visible events of both processes.

If $\mathbf{S}$ is a nonempty collection of congruences, define a sub-$\mathbf{S}$ congruence to be one that does not distinguish any pair of processes equated by any member of $\mathbf{S}$. In other words, it is a congruence at least as coarse as the transitive closure of the union of the equivalence relations created by the members of $\mathbf{S}$. If $\mathbf{S}$ is a singleton we will just write, for example, "sub-$\mathcal{R}$". If $\mathcal{X}$ is sub-$\mathcal{Y}$ we will sometimes write "$\mathcal{X} \preceq \mathcal{Y}$".

With this nomenclature, we will first establish the following result, and then explore the extent to which it might be generalised.

THEOREM 8.1 *For the dialect of CSP set out in Section 2, the* only *sub-$\{\mathcal{RT}, \mathcal{A}\}$ congruences are $\mathcal{R}$, $\mathcal{F}$, $\mathcal{T}$ and the trivial congruence $\mathcal{NULL}$ that identifies all processes.*

In other words, the only non-trivial finite observation congruences more abstract than refusal testing and stable acceptances are the well-known ones $\mathcal{F}$ and $\mathcal{T}$, together with $\mathcal{R}$. The proof of this result proceeds in four lemmas.

The first lemma shows that any sub-$\{\mathcal{RT}, \mathcal{A}\}$ congruence $\mathcal{M}$ satisfies $\mathcal{M} \preceq \mathcal{R}$. The other three are similar and establish, successively that any sub-$\mathcal{R}$ congruence that is strictly less abstract than any non-final member of the sequence

$$\mathcal{NULL}, \quad \mathcal{T}, \quad \mathcal{F}, \quad \mathcal{R}$$

is no less abstract than the next member of this sequence. These four together clearly establish our result.

LEMMA 8.2 *If $\mathcal{M}$ is any sub-$\{\mathcal{RT}, \mathcal{A}\}$ congruence then $\mathcal{M} \preceq \mathcal{R}$.*

To prove this we need to show that any two processes that are $\mathcal{M}$-equivalent are $\mathcal{R}$-equivalent. Since $\mathcal{M} \preceq \mathcal{A}$ and $\mathcal{M} \preceq \mathcal{RT}$ we know that $(P =_{\mathcal{A}} Q \vee P =_{\mathcal{RT}} Q) \Rightarrow P =_{\mathcal{M}} Q$. Another way of viewing this is to say that the relation $=_{\mathcal{M}}$ is a transitive superset of $=_{\mathcal{A}} \cup =_{\mathcal{RT}}$, so we know that if $P =_{\mathcal{A}} R$ and $R =_{\mathcal{RT}} Q$, then $P =_{\mathcal{M}} Q$.

The same strategy used in Theorem 5.3 to show that all members of $\mathcal{R}$ are expressible also works for $\mathcal{A}$ with very little amendment – the only difference is that the range of acceptance sets after each trace is now unrestricted. To represent the acceptance pair (s, A) we can use the process $R_{(s,\Sigma-A)}$ defined as in the proof of Theorem 5.3. The structure of that process is important so we recall it here:

$$\begin{aligned} R_{(\langle\rangle,X)} &= ?y : (\Sigma - X) \rightarrow \mathbf{div} \\ R_{(\langle a\rangle^\frown s,X)} &= \mathbf{div} \,\Box\, a \rightarrow R_{(s,X)} \end{aligned}$$

It follows that we can create a process $P_{\mathcal{A}}$ that implements P's representation in $\mathcal{A}$ using the same structures: the crucial feature of *this implementation* is that (because of the use of **div** in the above definition) on no trajectory of any $R_{(s,X)}$, and hence $P_{\mathcal{A}}$, is there more than one stable state. Furthermore each stable state is followed (if anything) by a range of visible actions leading only to divergence.

$P_{\mathcal{R}}$, P's representation in $\mathcal{R}$ generated by the Theorem 5.3 shares this property of trajectories, since it is built from the same components.

Since all the refusal testing observations of these two processes are made of such trajectories, it follow that the only behaviours recorded for $\mathcal{RT}$ of the processes $P_{\mathcal{A}}$ and $P_{\mathcal{R}}$ are of the three forms

(i) $\langle \bullet, a_1, \bullet, \ldots, \bullet, a_n, \bullet \rangle$

(ii) $\langle \bullet, a_1, \bullet, \ldots, \Sigma \cup \{\checkmark\} \rangle$

(iii) $\langle \bullet, a_1, \bullet, \ldots, X, a_n, \bullet \rangle$ where $a_n \notin X$.

These three forms are a slightly elaborate way of expressing the behaviours recorded of P in $\mathcal{R}$: traces, deadlocks and revivals. $P_{\mathcal{A}}$ and $P_{\mathcal{R}}$ have exactly the same such behaviours – it does not matter whether we inspect P's revivals after first converting to $\mathcal{A}$ or directly – and so $P_{\mathcal{A}} =_{\mathcal{RT}} P_{\mathcal{R}}$.

It therefore follows that $P =_{\mathcal{M}} P_{\mathcal{R}}$. Since any other process Q with the same value in $\mathcal{R}$ has $P_{\mathcal{R}} = Q_{\mathcal{R}}$ (they are the same syntactic process, as $X_{\mathcal{R}}$ is built solely from the semantic value of X in $\mathcal{R}$), we have $P =_{\mathcal{M}} Q$. It follows that $\mathcal{M} \preceq \mathcal{R}$ as claimed. ∎

Given this result, what we must prove to establish Theorem 8.1 is that the four congruences listed above are the only ones such that $\mathcal{M} \preceq \mathcal{R}$. Obviously in discussing the relationship between different notions of equivalence we have to be careful what we mean by equivalence and equality between a particular pair of processes. In the arguments below we will sometimes claim that one process is equivalent to another: our default interpretation for this will be that the two processes are equivalent in $\mathcal{R}$ since that implies equivalence in the whole range of congruences under discussion.

The first two results below have probably been implicitly assumed for years: certainly the author had long made this assumption without formulating them. The only formal proof that the author is aware of in the literature is the result of Bolton and Lowe [3] that there is no congruence strictly between $\mathcal{T}$ and $\mathcal{F}$. That is a slightly weaker version of Lemma 8.4 below. Even though there are significant similarities between the three proofs, we give them in detail since we will want to analyse them later.

LEMMA 8.3 *Every sub-$\mathcal{R}$ congruence $\mathcal{M}$ for CSP that distinguishes at least two processes satisfies $\mathcal{T} \preceq \mathcal{M}$.*

We can assume there are processes $P \sqsubset_R Q$ in any sub-$\mathcal{R}$ congruence $\mathcal{M}$ stronger than the null one, that are not identified by $\mathcal{M}$. This is because there are certainly processes K and L such that $K \neq_{\mathcal{M}} L$. The third process $K \sqcap L$ cannot be $\mathcal{M}$-equivalent to both K and L, for otherwise they would be equivalent to each other, so without loss of generality we can assume $K \sqcap L \sqsubset_{\mathcal{M}} K$ (and hence $K \sqcap L \sqsubseteq_{\mathcal{R}} K$ since $M \preceq \mathcal{R}$).

Suppose $U \neq_{\mathcal{T}} V$. The lemma is proved if we can establish $U \neq_{\mathcal{M}} V$. Without loss of generality we can assume there is trace s belonging to U but not V. $U \neq_{\mathcal{M}} V$ is established if we can find a context $C[\cdot]$ such that $C[X] = P$ when $s \in \mathit{traces}(P)$ and $C[X] = Q$ otherwise, since $\mathcal{M}$ is assumed to be a congruence, and in a congruence no context can map two equivalent processes to two inequivalent ones.

By the way we have defined the notion of congruence above, we may assume that the alphabet Σ in which we are modelling processes contains an element e that is not used in the particular P and Q chosen.

Let $C_{\mathcal{T}}(s)[\cdot]$ be the context that, in the proof of full abstraction for $\mathcal{T}$, maps a process U to one with the traces $\{\langle\rangle, \langle e\rangle\}$ or $\{\langle\rangle\}$ depending on whether U has, or does not have, the trace s. Let

$$C_1(s)[U] = (C_{\mathcal{T}}(s)[U] \triangle \mathit{SKIP}) \underset{\{e\}}{\parallel} e \rightarrow \mathit{SKIP}$$

This has the very useful property that its $\mathcal{R}$ value depends only on the trace set of $C_{\mathcal{T}}(s)[U]$, not on any other aspect of that process's behaviour. In general, the revivals value of $W \triangle \mathit{SKIP}$, because $\checkmark$ cannot be refused until it occurs, depends only on the traces of W, not its deadlocks, failures and revivals.

The value of $C_1(s)[W]$ is STOP if W does not have s, and $(e \rightarrow \mathit{SKIP}) \rhd \mathit{STOP}$ if it does.

Now consider

$$C_2(s)[W] = ((C_1(s)[W];\ P) \Box e \rightarrow Q) \setminus \{e\}$$

If W does not have the trace s this equals

$$((\mathit{STOP};\ P) \Box e \rightarrow Q) \setminus \{e\} = (\mathit{STOP} \Box e \rightarrow Q) \setminus \{e\} = Q$$

If W does have the trace then it equals

$$\begin{aligned}
& ((((e \to SKIP) \rhd STOP);\ P) \Box e \to Q) \setminus \{e\} \\
=_R\ & ((((e \to P) \rhd STOP)) \Box e \to Q) \setminus \{e\} \\
=_R\ & ((e \to (P \sqcap Q)) \rhd (e \to Q)) \setminus \{e\} \\
=_R\ & (e \to ((P \sqcap Q) \sqcap Q)) \setminus \{e\} \\
=_R\ & P \sqcap Q \\
=_R\ & P
\end{aligned}$$

by various standard CSP laws and inspection of the operational semantics, the last line following from $P \sqsubseteq_R Q$. ∎

One of the things we should note about this proof is that the interrupt operator $\triangle$ played an important role. It would have been worrying had this not been the case since if we dropped that operator the result would not be true – $\mathcal{T}$ and the congruence, for the reduced language, of stable failures without a separate trace component, referred to in Section 4, are incomparable since neither is weaker than the other.

LEMMA 8.4 *Every sub-$\mathcal{R}$ congruence $\mathcal{M}$ for CSP that distinguishes at least two processes not identified by $\mathcal{T}$ satisfies $\mathcal{F} \preceq \mathcal{M}$.*

We can, in the same way as for the previous lemma, assume that there are $P \sqsubseteq_R Q$ identified by $\mathcal{T}$ but not by $\mathcal{M}$. By Lemma 8.3 we know that $\mathcal{M}$ is at least as strong as $\mathcal{T}$. What we therefore need to do is prove that any pair of processes U and V that are trace equivalent but $\mathcal{F}$-inequivalent are also $\mathcal{M}$-inequivalent. For such processes U, V we may assume without loss of generality that there is a failure (s, X) in $\mathit{failures}(U) - \mathit{failures}(V)$. Following the model above we will construct a context that maps processes to P or Q depending on whether or not they have (s, X). As before, we construct this from the context $C_{\mathcal{F}}(s, X)[\cdot]$ used in full abstraction, this time $C_{\mathcal{F}}(s, X)[\cdot]$, mapping a process U to $STOP$ if it has (s, X) and to **div** if not. If P is ✓-free it is easy: set

$$C(s, X)[U] = Q \sqcap (P \underset{\emptyset}{\parallel} C_{\mathcal{F}}(s, X)[U])$$

since $(P \underset{\emptyset}{\parallel} C_{\mathcal{F}}(s, X)[U])$ equals P if $C_{\mathcal{F}}(s, X)[U]$ is $STOP$, and otherwise refines Q since it has the same traces but is never stable. If P is not ✓-free the right hand term may be able to refuse to terminate when P cannot, and the obvious solution of replacing $C_{\mathcal{F}}(s, X)[U]$ by $C_{\mathcal{F}}(s, X)[U] \Box SKIP$ does not work since the $SKIP$ can resolve $\Box$. This can be overcome by using an event e, not used in P and Q, to guard $SKIP$.

$$C(s, X)[U] = Q \sqcap ((P;\ (e \to SKIP)) \underset{\{e\}}{\parallel} ((e \to SKIP) \Box C_{\mathcal{F}}(s, X)[U])) \setminus \{e\}$$

LEMMA 8.5 *Any sub-$\mathcal{R}$ congruence $\mathcal{M}$ for CSP that distinguishes a pair of processes not distinguished by $\mathcal{F}$ satisfies $\mathcal{R} \preceq M$ and is therefore equivalent to $\mathcal{R}$.*

Let us suppose there were such a congruence $\mathcal{M}$ and that the processes it distinguishes that $\mathcal{F}$ does not are P and Q. By the previous two lemmas we know that $\mathcal{F} \preceq \mathcal{M}$. As before we may assume that $P \sqsubset_R Q$.

Then we can be sure of the following things about P and Q.

- They have the same traces and stable failures (including deadlock traces).
- The only difference between P and Q is that P has some revivals (s, X, a) such that (a) Q does not have them and (b) since we know, by equivalence under $\mathcal{F}$, that in any such case either s is a deadlock of Q or there is some b distinct from a such that $(s, X, b) \in \mathit{revivals}(Q)$ we can guarantee (in the second case by healthiness condition **Rev3**) that $(s, X \cup \{a\}) \in \mathit{failures}(Q) = \mathit{failures}(P)$.

Now suppose that U and V are arbitrary processes identified by $\mathcal{F}$ but not by $\mathcal{R}$. Then without loss of generality we can assume that there is a revival $(t, Y, b) \in (\mathit{revivals}(U) - \mathit{revivals}(V))$. What we are going to set out to do is create a context $D[\cdot]$ so that $D[U] = P$ and $D[V] = Q$. If we can do this then we have established our result: once again, we know that $\mathcal{M}$ is a congruence and distinguishes P and Q, so to achieve this it *must* distinguish U and V.

We know, thanks to our proof of Theorem 5.4, that there is a context $C_{rev}(t, Y, b)[\cdot]$ such that $C_{rev}(t, Y, b)[U]$ is a refinement of $(a \rightarrow \mathit{STOP}) \sqcap \mathit{STOP}$ that has the revival $(\langle\rangle, \emptyset, a)$ and $C_{rev}(t, Y, b)[V]$ is a refinement of $(a \rightarrow \mathit{STOP}) \rhd \mathit{STOP}$. For every $(s, X, c) \in \mathit{revivals}(P) - \mathit{revivals}(Q)$ where X is maximal for this s and c, we can construct a new context as follows:

$$\begin{aligned} B_{(\langle\rangle, X, c)}[W] &= W[\![c/a]\!] \underset{\{c\}}{\parallel} ?x : (\Sigma - X) \rightarrow \mathbf{div} \\ B_{(\langle d\rangle^\frown s', X, c)}[W] &= (d \rightarrow B_{(s', X, c)}[W]) \,\Box\, \mathbf{div} \end{aligned}$$

Thanks to our observations above, in particular that $(s, X \cup \{c\})$ is a failure of P and Q, we get that every behaviour of $B_{(s,X,c)}[(a \rightarrow \mathit{STOP}) \rhd \mathit{STOP}]$ is one of Q, but $B_{(s,X,c)}[(a \rightarrow \mathit{STOP}) \sqcap \mathit{STOP}]$ additionally contains (s, X, c) from the revival $(\langle\rangle, \emptyset, a)$ being renamed and synchronised with $?x : (\Sigma - X) \rightarrow \mathbf{div}$. It follows that

$$B^*[W] = Q \sqcap \bigsqcap \{B_{(s,X,c)} \mid (s, X, c) \in \mathit{revivals}(P) - \mathit{revivals}(Q)\}$$

gives P when applied to $C_{rev}(t, Y, b)[U]$ and Q when applied to $C_{rev}(t, Y, b)[V]$ This means we can set

$$D[W] = B^*[C_{rev}(t, Y, b)[W]]$$

and achieve the goal set out above. ∎

This concludes the proof of Theorem 8.1. ∎

While this result is powerful it immediately begs the questions

- Can it be generalised within the finite observation class?
- Do similar results hold in the non-finite observation families of models such as $\{\mathcal{T}^{\Downarrow}, \mathcal{F}^{\Downarrow}, \mathcal{R}^{\Downarrow}, \ldots\}$?

This is too broad a subject to be addressed within the present paper. We will restrict ourselves to one further result, preceded by some conjectures and observations.

The author expects that similar results will hold in the two divergence-strict families of models discussed in this paper (which are really just two different faces of the same family). They will not parallel the ones in this paper precisely, since the author has discovered a strange model slightly more abstract than $\mathcal{N}$ while researching this issue: one can abstract away from whether or not an event that leads immediately to divergence is refused or not! ¶

The situation with the $\mathcal{M}^{\#}$ family will be more complex for two reasons:

- Firstly, all the families of congruences are themselves more abstract than ones such as $\mathcal{R}^{\#}$, so there will certainly be more than four congruences stronger than this model.
- Secondly, there seems to be nothing to constrain that level of detail recorded beyond the first divergence on a trace to be the same as that before. For example, representing a process in a pair of models, exactly one of which is divergence strict, will produce a congruence that is different from all our named ones, but which is still stronger than $\mathcal{R}^{\#}$. For example $(\mathcal{F}^{\Downarrow}, \mathcal{T})$ will tell you about failure information prior to the first possible divergence on a trace, but only trace information beyond this point.

The author asserts that the condition that our congruence has to be sub-$\mathcal{R}$ (equivalent to sub-$\{\mathcal{RT}, \mathcal{A}\}$ thanks to Lemma 8.2) in Lemmas 8.3, 8.4, and 8.5 can be weakened to saying that it has to be a finite observation congruence as defined in Section 3.1. This would imply that every non-trivial finite observation congruence $\mathcal{M}$ that is not $\mathcal{T}$, $\mathcal{F}$ satisfies $\mathcal{R} \preceq \mathcal{M}$, giving our initial sequence of models a very special role.

So we should examine how this condition was used in these three Lemmas. In the first two it was used weakly, simply to underpin our statement that ultimate contexts such as $C(s, X)[U]$ are equal to P or Q – the fact that these statements are true in $\mathcal{R}$ imply they are true in the hypothetical models $\mathcal{M}$. In fact, the author believes that these equivalences are true in all finite observation models, and in particular in the conjectured model $\mathcal{FL}$. If this is so then the generalisations of Lemmas 8.3 and 8.4 are true, and in any case the condition can be weakened to saying that these equivalences are true in $\mathcal{M}$ itself.

The third proof uses the sub-$\mathcal{R}$ condition in some detail because it analyses the differences between processes in $\mathcal{M}$ using revivals and conditionally, but separately, adds each revival from the difference of P and Q to Q. The author has not been able to find contexts that work as smoothly for this lemma as for the other two without this analysis.

He has, however, managed to generalise the point-by-point extension construction to cover arbitrary $\mathcal{FL}$ behaviours of the type described in Section 3.1 under the assumptions on the observability of (in)stability and the meaning of $\bullet$ set out there. We can therefore generalise Lemma 8.5 to the following:

THEOREM 8.6 *Any finite observation congruence $\mathcal{M}$ for CSP that distinguishes a pair of processes not distinguished by $\mathcal{F}$ satisfies $\mathcal{R} \preceq M$.*

In the the previous proof we took processes $P \sqsubset Q$ that our hypothetical model $\mathcal{M}$ distinguished and showed how we could add each revival in $\mathit{revivals}(P) - \mathit{revivals}(Q)$ to Q conditionally on the presence of some chosen revival (t, Y, b) in a process argument U.

This time all we know is that $\mathcal{M} \preceq \mathcal{FL}$, so to follow the same shape we would have to be able to add each $\mathcal{FL}$ behaviour in the difference of P and Q to Q, but still only conditional on a revival. This is much more challenging and the author has not found a way of solving this for an arbitrary behaviour. Instead we will study the difference in $\mathcal{FL}$ behaviour of P and Q carefully and show we can be more choosy about the behaviour we need to add.

Because Σ is finite we can enumerate the members of this difference $\beta_1, \beta_2, \ldots$ (the list may be finite or countably infinite). Furthermore we can choose to do this in such a way that

(a) if β_i has more events than β_j then $i > j$, and

(b) if β_i and β_j are the same except that some of the non-$\bullet$s in β_i are $\bullet$s in β_j then $i > j$.

For any $\mathcal{FL}$-behaviour at all, it is straightforward to define the most refined process that has it. There are two sorts of $\mathcal{FL}$ behaviours we need to consider:

$$(A_0, a_1, A_2, \ldots, A_{n-1}, a_n, A_n) \qquad \text{and} \qquad (A_0, a_1, A_2, \ldots, A_{n-1}, a_n, \bullet, \checkmark)$$

We define:

$$\begin{aligned}
FLB(\bullet) &= \mathbf{div} \\
FLB(A) &= ?x : A \to \mathbf{div} \\
FLB(\bullet, \checkmark) &= SKIP \\
FLB((\bullet, c)^\frown\beta) &= (c \to FLB(\beta)) \rhd \mathbf{div} \\
FLB((A, c)^\frown\beta) &= (c \to FLB(\beta)) \,\Box ?x : (A - \{c\}) \to \mathbf{div}
\end{aligned}$$

Let $Q_0 = Q$ and $Q_{n+1} = Q_n \sqcap FLB(\beta_{n+1})$. Not all the Q_i can be equivalent over $\mathcal{M}$ since we know there is an $\mathcal{M}$-behaviour that distinguishes P and Q, and that this is the relational image of one of the β_i. Since evidently $Q_{m+1} \sqsubseteq Q_m$, this means that without loss of generality we can set $Q' = Q_{m-1}$ and $P' = Q_m$ to be the first consecutive pair that differ. Furthermore we know

(i) That β_m is not a trace (i.e. all $A_i = \bullet$) because P' and Q' are, by construction and the original assumption that P and Q are equivalent over $\mathcal{F}$, trace-equivalent.

(ii) That all behaviours $\beta < \beta_m$ are also present in Q' thanks to our choice of enumeration of the β_i.

This means we can write β_m as $(\bullet, a_1, \bullet, \ldots, \bullet, a_s)^\frown\beta'$ for some $s \geq 0$ where the first element A_s of β' is not $\bullet$, and furthermore (by assumption (ii)) replacing A_s by $\bullet$ gives a behaviour of Q'.

Since P' and Q' are equivalent in $\mathcal{F}$, we know that Q' has an $\mathcal{FL}$-behaviour β^* that exhibits the failure $(\langle a_1, \ldots, a_s\rangle, \Sigma - A_s)$. This is because the presence of β_n and hence its prefix $(\bullet, a_1, \ldots, \bullet, a_s, A_s)$ shows P' has this failure. Without loss of generality we can assume $\beta^* = (\bullet, a_1, \ldots, \bullet, a_s, B)$ where $B \subseteq A_s$.

We can assume, in the same way as we have introduced other events before, that our alphabet Σ is sufficiently large that it contains all the members Σ_1 used in P, and a second disjoint set of the same size Σ_2. We will assume that the operation a' represents a bijection from Σ_1 to Σ_2.

Now let $W(\beta_m, \beta^*)$ be the process that behaves identically to $FLB(\beta_m)$ except that after $\langle a_1, \ldots, a_s\rangle$ it can communicate not only the members of A_s but also a' for each member a of B (leading to **div**). We will use this in creating a context $D[\cdot]$ such that, for a chosen revival (t, Y, b), $D[U] = P'$ or $D[U] = Q'$ depending on whether $(t, Y, b) \in \mathit{revivals}(U)$ or not.

In doing this we again appeal to the context $C_{rev}(t, Y, b)[U]$ that maps each process U with the trace $t^\frown\langle b\rangle$ to a process whose trace set is $\{\langle\rangle, \langle a\rangle\}$ and which has the revival $(\langle\rangle, \emptyset, a)$ if and only if $(t, Y, b) \in \mathit{revivals}(U)$. If β_m does not end in $\checkmark$, define

$$\begin{aligned}
T_0[U] &= STOP \sqcap ((C_{rev}(t, Y, b)[U] \triangle c \to AS) \underset{\{a,c\}}{\parallel} a \to c \to AS) \setminus \{c\} \\
T_{n+1}[U] &= a \to T_0[U] \\
AS &= a \to AS
\end{aligned}$$

If β_m does end in $\checkmark$, the definition can easily be adjusted so that this process terminates after exactly as many as as there are non-tick events in β_m. In either case, $T_s[U]$ performs exactly as many as as there are events in β_m that precede the crucial acceptance A_s. It then has the choice of deadlocking, performing a after $\bullet$ or, if (t, Y, b) is present in U, offering a stably. After that it performs enough stable offers of a and if necessary a $\checkmark$ so that the construct $E[U]$ defined below does not interfere with the rest of β_m. c has a twin role in this definition. Firstly it is an interlock: the parallel composition with $a \to c \to AS$ ensures that the interrupting c cannot happen until after the first step, and the hiding of c ensures that $T_s[U]$ can become stable after $s+1$ as. Now let

$$E[U] = (W(\beta_m, \beta^*) \underset{\Sigma_1}{\parallel} T_s[U][\![a \mapsto \Sigma_1]\!])[\![Unprime]\!]$$

where *Unprime* is the renaming that maps all events c' in Σ_2 to c. Except for what happens immediately after s events, this process behaves identically to $W(\beta_m, \beta^*)$ because the renamed $T_s[U]$ does not then block any event. After s events $T_s[U]$ can, whatever U is, choose to deadlock or perform a without the observation of stability. When $T_s[U]$ deadlocks, this has the effect of allowing only the Σ_2 events of $W(\beta_m, \beta^*)$ which, thanks to the *Unprime* remaining, appear as the offer of B and are followed immediately by certain divergence. In other words, deadlock by $T_s[U]$ here results in the behaviour β^*, which we know belongs to Q'. The unstable occurrence of a in $T_s[U]$ at this point leads to a behaviour that is either β_m with A_s replaced by $\bullet$, or one implied by this one. By our assumptions all such behaviours belong to Q'. It follows that, if U does not contain (t, Y, b), then $E[U]$ refines Q'.

If U does contain (t, Y, b) then since $T_s[U]$ can stably offer as throughout the length of the behaviour β and then terminate if appropriate, it follows that β is a behaviour of $E[U]$. Note that since, after s events $W(\beta_m, \beta^*)$ additionally offers the images of $B(\subseteq A_s)$ in Σ_2, $E[U]$ can perform each event from B in two ways, but the actual offer after s events is the same as in β_m.

The extra possibility of the revival in $T_s[U]$ after s events clearly cannot add any behaviours that are not automatically present in P' because it has β. Hence $E[U]$ refines P'. Now, defining

$$D[U] = E[U] \sqcap Q'$$

we have exactly what we wanted and, since $\mathcal{M}$ distinguishes Q' and P', if $(t, Y, b) \in revivals(U) - revivals(V)$

$$D[V] = Q' \neq P' = D[U]$$

which proves our result. ∎

These results beg the question of what congruences exist that are finer than the revivals one. The author believes that there is probably no nice finite classification of the same sort as we have seen above, and that for example there is an infinity of different finitary congruences between $\mathcal{R}$ and $\mathcal{RT}$ characterised *inter alia* by recording up to any fixed bound of stable refusals during a trace. ¶

9 Conclusions

In this paper we have shown how the conformance equivalence defined in [8] and developed in [22] can be turned into a full model of CSP which is fully abstract with respect to the

classes of property described in those two papers. We have also shown how it fits into the hierarchy of CSP models and how this hierarchy allows to create easily a number of extensions to include divergence and infinite traces. In Section 8.1 we showed that $\mathcal{R}$ has a special place as the "biggest of the small models" in rather a striking sense.

The author suspects there may be further models that could be discovered, for example ones between $\mathcal{A}$ and $\mathcal{R}$ which record traces and "hyper revivals": combinations of a trace s, a stable refusal X, and a set (of size less than of equal to than some constant h) of events that are all offered in the state exhibiting the failure (s, X). We would interpret a set of size exactly h as recording an offer of this size or bigger, and one of size less than s as a precise ready set. In this hierarchy, setting $h = 0$ would yield an equivalence the same as $\mathcal{F}$, $h = 1$ would yield $\mathcal{R}$ and $h =\mid \Sigma \mid$ would yield $\mathcal{A}$. observations in a trace is limited, but less so than in $\mathcal{R}$. This would be of interest to anyone studying the hierarchy for its own sake, but the author thinks it unlikely that either the new models suggested by this example or the one at the end of the previous section would be of much practical use. ¶

He believes that failures models remain the most important ones of CSP, and that their full abstraction properties such as the abilities to decide "can deadlock immediately" and "can deadlock or diverge immediately". Revivals models are, however, clearly important when one wants to examine stable configurations of networks such as those examined by the stuck-freeness and *RespondsTo* conditions we described earlier, in that they allow us to express conditions in terms of processes' individual offers. Another obvious application is in refining the concepts used in deadlock analysis such as the different types of *conflict* discussed in Chapter 13 of [23]. One fact that is worth bearing in mind, however, is that practical networks most often consist of deterministic processes, it being the hiding of their interactions that creates externally visible nondeterminism. In such cases the failures representations of processes convey all necessary information about revivals, so that definitions of stuck-freeness etc over failures models are in fact equivalent to revivals ones.

Revivals, particularly coupled with divergences, might well find uses in situations like those envisaged in Section 6.3 where we can guarantee or wish to ensure that observable actions only happen in $\checkmark$-stable states.

We were able, in Section 7, to lay out a plan for including all known CSP models within a single framework of algebraic semantics where we use a common set of step and distributive laws. The problem is that the step laws required are more general and more complex than those listed in [23] because of the need to handle combinations of stable and unstable actions. We also conjectured that once this was done, the differences between the known models could be captured in differences in very few laws.

The investigation of further models like those introduced in this paper has both positive and negative qualities. On the plus side it shows how one can sometimes get the exact model one wants for some purpose. It also sheds considerable light on the nature of CSP and its existing models, especially their algebraic properties. On the other hand, as the concurrency community has learned to its cost, the proliferation of models makes our work less accessible to potential users and can give the impression that we are more concerned with academic minutiae than with applications. We have seen that revivals are necessary to capture precisely a particular sort of practically-relevant property. However, for most

purposes, the distinctions between

$$(x?\{a,b\} \to STOP) \rhd STOP$$
$$(x?\{a,b\} \to STOP) \sqcap STOP$$
$$(a \to STOP) \sqcap (b \to STOP) \sqcap STOP$$
$$(x?\{a,b\} \to STOP) \rhd (a \to STOP \sqcap STOP)$$
$$(x?\{a,b\} \to STOP) \rhd (b \to STOP \sqcap STOP)$$
$$(x?\{a,b\} \to STOP) \sqcap (a \to STOP \sqcap STOP)$$
$$(x?\{a,b\} \to STOP) \sqcap (b \to STOP \sqcap STOP)$$

the seven different $\mathcal{R}$-values of divergence-free processes with traces $\{\langle\rangle, \langle a\rangle, \langle b\rangle\}$ that can deadlock immediately, are probably not that important. They are all equivalent in $\mathcal{F}$.

Both revivals and refusal-testing refinement have recently been implemented in FDR.

Acknowledgements

I would like to thank Tony Hoare, Jakob Rehof and Sriram Rajamani for discussions about Conformance, Joy Reed and Jane Sinclair for discussions about responsiveness, Gavin Lowe and Michael Goldsmith for correspondence about normal forms, and Michael Goldsmith and Phil Armstrong for implementing revivals refinement checking. I had useful discussions with Jakob and Antti Valmari about the material in Section 8.1.

The presentation of this paper has been much improved by the suggestions of anonymous referees. In particular these led indirectly to me writing the two sections on the CSP hierarchy and discovering the results in Section 8.1. My work on this topic was funded by US ONR.

Appendix: Notation

This paper follows the notation of [23], from which most of the following is taken.

Σ	(Sigma): alphabet of all communications
τ	(tau): the invisible action
Σ^τ	$\Sigma \cup \{\tau\}$
$\Sigma^\checkmark$	$\Sigma \cup \{\checkmark\}$
$\Sigma^{*\checkmark}$	$\{s, s\hat{}\langle\checkmark\rangle \mid s \in \Sigma^*\}$
A^*	set of all finite sequences over A
$\langle\rangle$	the empty sequence
$\langle a_1, \ldots, a_n\rangle$	the sequence containing $a_1, \ldots, a_n$ in that order
$s\hat{}t$	concatenation of two sequences
$s \setminus X$	hiding: all members of X deleted from s
$s \underset{X}{\parallel} t$	the set of traces composed from subsequences s and t which share members of X and are disjoint elsewhere.
$s \leq t$	$(\equiv \exists\, u.s\hat{}u = t)$ prefix order

Processes:

$\mu\, p.P$	recursion
$a \rightarrow P$	prefixing
$?x : A \rightarrow P$	prefix choice
$P \,\Box\, Q$	external choice
$P \sqcap Q, \quad \bigsqcap S$	nondeterministic choice
$P \underset{X}{\parallel} Q$	generalised parallel
$P \setminus X$	hiding
$P[\![R]\!]$	renaming (relational)
$P[\![a \mapsto A]\!]$	renaming in which a maps to every $b \in A$
$P[\![A \mapsto a]\!]$	renaming in which every member of A maps to a
$P \triangleright Q$	"time-out" operator (sliding choice)
$P \,\triangle\, Q$	interrupt
$P[x/y]$	substitution (for a free identifier x)
$P \xrightarrow{a} Q$	($a \in \Sigma \cup \{\tau\}$) single action transition in an LTS

Models:

$\mathcal{T}$	traces model
$\mathcal{N}$	failures/divergences model (divergence strict)
$\mathcal{F}$	stable failures model
$\mathcal{R}$	stable revivals model
$\mathcal{A}$	stable ready sets, or acceptances, model
$\mathcal{RT}$	stable refusal testing model
$\mathcal{FL}$	the finest finite observation model
$\mathcal{M}^{\Downarrow}$	the model $\mathcal{M}$ extended by strict divergence information
$\mathcal{M}^{\Downarrow,\omega}$	$\mathcal{M}$ extended by strict divergences and infinite traces or similar
$\mathcal{M}^{\#}$	$\mathcal{M}$ extended by non-strict divergences and infinite traces or similar
$\mathcal{U}$	failures/divergences/infinite traces model with divergence strictness
$\mathcal{SBD}$	finite and infinite traces/divergences model strict under ω-divergent infinite traces
$\mathcal{X} \preceq \mathcal{Y}$	X identifies all processes identified by $\mathcal{Y}$
$\bot_{\mathcal{N}}$ (etc.)	bottom elements of models
$\top_{\mathcal{F}}$ (etc.)	top elements of models
$\sqsubseteq$	refinement over whatever model is clear from the context

References

[1] G. Barrett, The fixed-point theory of unbounded nondeterminism, FAC **3** 110-128, 1991.

[2] S.R. Blamey, *The soundness and completeness of axioms for CSP processes*, Topology and category theory in computer science, OUP 1991.

[3] C. Bolton and G. Lowe, *A hierarchy of failures-based models*, ENTCS **96**, 129-152, 2004.

[4] S.D. Brookes and A.W. Roscoe, *An improved failures model for CSP*, Proceedings of the Pittsburgh seminar on concurrency, LNCS 197 (1985).

[5] M. Broy, *A theory for nondeterminism, parallelism, communication and concurrency*, Theoretical Computer Science **45**, pp1–61 (1986).

[6] R. de Nicola and M. Hennessy, *Testing equivalences for processes*, Theoretical Computer Science **34**, 1, 83–134, 1987.

[7] Formal Systems (Europe) Ltd, *Failures-Divergence Refinement: FDR2 Manual*, 1997.

[8] C. Fournet, C.A.R. Hoare, S.K. Rajamani and J. Rehof, *Stuck-free conformance*, Proceedings CAV 04, 16th International Conference on Computer Aided Verification, Boston, USA, July 2004.

[9] R.J. van Glabeek, *The linear time - Branching time spectrum I* The handbook of process algebra, Elsevier 2001.

[10] R.J. van Glabeek, *The linear time - Branching time spectrum II* Proceedings of CONCUR 1993.

[11] D. Harel, *Statecharts: A visual formalism for complex systems*, Science of Computer Programming **8**, 3, 231-274, 1987.

[12] C.A.R. Hoare, *A model for communicating sequential processes*, in 'On the construction of programs' (McKeag and MacNaughten, *eds*), Cambridge University Press, 1980.

[13] C.A.R. Hoare, *Communicating sequential processes*, Prentice Hall, 1985.

[14] P.B. Levy, *Infinite trace semantics*, Proc 2nd APPSEM Workshop, 2004 `www.cs.ioc.ee/appsem04/accepted.html`

[15] Abida Mukkaram, *A refusal testing model for CSP*, Oxford University D.Phil thesis, 1993.

[16] E.R. Olderog and C.A.R. Hoare, *Specification-oriented semantics for communicating processes*, *Acta Informatica*, **23**, 9–66, 1986.

[17] J. Ouaknine, Discrete analysis of continuous behaviour in real-time concurrent systems, Oxford University D.Phil thesis, 2000.

[18] I. Phillips, *Refusal testing*, Theoretical Computer Science **50** pp241-284 (1987).

[19] A. Puhakka, *Weakest congruence results concerning "any-lock"*, Proc TACAS 2001, Springer LNCS 2215 (2001).

[20] A. Puhakka, A and A. Valmari, *Weakest-Congruence Results for Livelock-Preserving Equivalences*, Proceedings of CONCUR '99 (Concurrency Theory), LNCS 1664, Springer-Verlag 1999, pp. 510-524.

[21] J.N. Reed, J. Sinclair and A.W. Roscoe, *Responsiveness of inter-operating components*, FAC **16** pp394–411 (2004).

[22] J.N. Reed, A.W. Roscoe and J. Sinclair, *Responsiveness and stable revivals*, FAC **19** 3, 303-319, 2007.

[23] A.W. Roscoe, *The theory and practice of concurrency*, Prentice-Hall International, 1998. Updated version available via `web.comlab.ox.ac.uk/oucl/work/bill.roscoe/publications` – number 68.

[24] A.W. Roscoe, *An alternative order for the failures model*, in 'Two papers on CSP', technical monograph PRG-67, Oxford University Computing Laboratory, July 1988. Also appeared in Journal of Logic and Computation **2**, 5 pp557-577.

[25] A.W. Roscoe, *Unbounded nondeterminism in CSP*, in 'Two papers on CSP', technical monograph PRG-67, Oxford University Computing Laboratory, July 1988. Also Journal of Logic and Computation, **3**, 2 131–172, 1993.

[26] A.W. Roscoe, *Seeing beyond divergence*, in Proceedings of "25 Years of CSP", LNCS3525 (2005).

[27] A.W. Roscoe, *Confluence through extensional determinism*, Proceedings of the Bertinoro meeting on concurrency, BRICS 2005 (and to appear in ENTCS).

[28] A.W. Roscoe and Zhenzhong Wu, *Verifying Statemate Statecharts Using CSP and FDR*, Proceedings of ICREM 2006, Springer LNCS 4260.