Revivals, stuckness and the hierarchy of CSP models

A.W. Roscoe

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

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 Reed *et al* (2007, FAC, 19, 3). 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 that paper. 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 set out a number of conjectures and challenges for future workers in this area.

Key words: Concurrency, CSP models, Full Abstraction

1 Introduction

The author has long worked on mathematical models for concurrent systems, in particular Hoare's CSP [13]. It therefore came as something of 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

Email address: `Bill.Roscoe@comlab.ox.ac.uk` (A.W. Roscoe).

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 the 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 established 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 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. In Section 7 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 models that are not finite traces, stable failures or 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, as well as setting out a programme for future work on the algebraic semantics of CSP.

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 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.

This paper has been available in draft form since 2005. During the intervening period, several papers have been published that developed ideas presented here and answered questions posed in earlier versions. The first of the two most notable is [30], which reports the embedding of the stable revivals model and its CSP semantics and laws into a theorem proving environment. This verified many of the claims made in the present paper as well as revealing several places where our assumptions were not stated as clearly as they should have been. The present version therefore benefits in two ways from that paper. The second paper, [28], is by the author and shows how the structural results of Section 7 can also be proved for divergence-strict models of CSP over a subtly extended language.

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 any 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 ($\triangleright$ and $\triangle$) for reasons that will become clear 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 that 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:

- $a \to P$ communicates the event $a \in \Sigma$ before behaving like P. This is *prefixing.*
- $?x : A \to P(x)$ 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.
- $P \triangleright Q$ 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 $(P \,\Box\, a \to Q) \setminus \{a\}$ 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 representation: 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 indexed families of processes: the fact that $\sqcap$ and $\Box$ are symmetric and associative makes this unambiguous. Since $\sqcap$ is also idempotent, it makes sense to apply its generalisation $\bigsqcap$ over any nonempty set or indexed family, while $\Box$ can only be applied to finite indexed families as there are CSP models (those involving acceptance sets) where $\Box$ is not idempotent. The reason $\Box$ is restricted to *finite* families is to avoid problems with internal actions. If $\bigsqcap$ 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 used within CSP, but these are resolved by non-process identifiers and are not CSP operators in the same sense as the above, particularly since non-process identifiers are given a declarative semantics (i.e., there is no assignment).

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].

- $P \underset{X}{\parallel} Q$ 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. $dom(R)$ must include all visible events used by P.

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.*
- $P \bigtriangleup Q$ 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].[1] By varying the type(s) of behaviour observed we get different models.

While any particular CSP model is based on a particular alphabet Σ, we only consider those models for which there are analogues for every size of Σ (perhaps bounded above by some infinite cardinal in size). If $\Sigma \subset \Sigma'$ then the congruence implied by the corresponding models for processes using events in Σ should be identical. The

[1] 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.

fact that models are congruences under CSP operators such as renaming, prefixing and hiding *implies* that they have strong symmetry properties and must, for example, treat all members of Σ alike. It is possible that for some small Σ two different families of model might co-incide.

We will frequently want to extend the alphabet under consideration to allow for CSP constructs over events not in a particular Σ. What we will generally do then is to rename the "old" Σ as Σ_0 and to re-base ourselves in some new $\Sigma_0 \cup \Sigma_1$. The assumptions of the previous paragraph then mean that any equalities proved over this larger alphabet still hold over Σ_0.

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.

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 of that book introduces the traces ($\mathcal{T}$), stable failures ($\mathcal{F}$), and failures-divergences ($\mathcal{N}$) models of CSP and the denotational semantics of 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 *stable revivals* model we will be studying also falls into this category, alongside a number of others[2] :

[2] The forms quoted here are in some cases not precisely those in which they were originally

- In the *stable failures model* $\mathcal{F}$ [23] 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.

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*, and
(iii) that 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. In place of such an acceptance set we may instead choose to observe things like refusals that are deducible from it.

presented. What we do here is strip them down to finite observations only.

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

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

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.[3] 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 that some unlikely pairs of processes would have to be distinguished. Thus

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

$$(SKIP;\ STOP) \,\square\, (a \to STOP) \not\equiv (STOP \,\square\, a \to STOP)$$

since in each case the left hand side can perform a from an unstable state, unlike the right hand side. We can conclude from the second of these that $SKIP;\ STOP \neq STOP$.

Another argument 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 postulates 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, then so is the same one with any selection of the A_is replaced by $\bullet$. The observations possible of each of the pair of processes displayed above will then be the same: sequences $\langle X\rangle$ and $\langle X, a, Y\rangle$, 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 nonempty and closed under prefix (initial subsequence), and whenever $\langle A_0, a_1, A_1, \ldots, a_r, A_r, a_{r+1}, A_{r+1}\rangle$ belongs to a process and $A_r \neq \bullet$ then $\langle A_0, a_1, A_1, \ldots, a_r, A_r, b, \bullet\rangle$ also belongs for all elements b of A_r. Here, $\mathcal{FL}$ stands for "finite linear".

If β is such a behaviour, and 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 $\beta' < \beta$. Our postulate about the non-observability of instability then implies that if β can be observed of a process, so can all $\beta' < \beta$.

We thus have the representation of an arbitrary CSP process in a new model $\mathcal{FL}$, whose healthiness conditions are those implied in the previous two paragraphs. We

[3] 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.

leave the presentation of the full details of this new compositional model for a later paper. ¶

Definition 3.1 *A 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. $\mathcal{M}$ must be compositional under every CSP operator, with the semantics of recursion being given by component-wise subset-least fixed point.*

The relations generating M_i must vary homogeneously as Σ varies: if $\Sigma \subseteq \Sigma'$ then the relations for Σ are the same as those for Σ' restricted to the domain of $\mathcal{FL}$-behaviours over Σ, possibly discarding some members of the range where this does not change the equivalence generated.

An example of discarding some members of the range is removing events in those refusal sets of processes defined over the alphabet Σ that intersect with $\Sigma' - \Sigma$.

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 from 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 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 occurring 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 the 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 P's 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 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 a congruence 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[4], and calculating the 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 extensions $s\hat{}t$. Thus, there are healthiness conditions that say that if s is a divergence then so is $s\hat{}t$, and any $(s\hat{}t, X)$ 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 is completely independent of the failures *per se* of its arguments, depending only on 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. Similarly there is a divergence-strict extension of $\mathcal{T}$ that represents a process as its sets of finite traces and divergence traces. Naturally, the divergences

[4] 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 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 $(s \setminus A)\hat{}\langle x\rangle$ implies it on this trace itself. But in a model where we have to know if $P \setminus A$ can diverge after precisely $(s\hat{}\langle x\rangle) \setminus A$, it is serious.

and finite traces predicted by $\mathcal{N}$ for any process are always the same as those predicted by this second model. 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 stipulation 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 appropriate to use more complex "divergences" than just traces. The model $\mathcal{FL}^{\Downarrow}$ is discussed in [28].

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

The solution to the problem of coping with unbounded nondeterminsm 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, but can also choose to perform an infinite number. Note that the the former process, hiding a, should become $STOP$ since it cannot diverge, while the latter one can diverge whne the infinite seuqanece of as is hidden.

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 $\mathcal{RT}^{\Downarrow\omega}$, 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 this model, and we will briefly discuss it again there. ¶

A way of dispensing with 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]. Finding the correct fixed point to model recursions is now a problem, requiring the following pair of facts to be reconciled and allowed for:

(A) With finitary models like $\mathcal{T}$ and $\mathcal{F}$, 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 the subset 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.

(A) and (B) need to be reconciled because together they 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: wherever a process has not reached divergence, the two are equal.

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 diverges on the empty trace) is neither refinement minimal nor subset minimal.

As shown in [26], it is possible to calculate the correct value by a non-standard fixed point process. This is over 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 $\mathcal{RT}$ and $\mathcal{FL}$ 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 of the four equivalences, $\mathcal{M}^{\Downarrow,\omega}$ is finer than $\mathcal{M}^{\Downarrow}$, but both of these last two are incomparable with $\mathcal{M}$.

As CSP congruences, $\mathcal{M}^{\Downarrow,\omega}$ and $\mathcal{M}^{\Downarrow}$ are the same, but the richer model allows us to consider processes with infinite nondeterminism. Thus, for finitely nondeterministic terms and finitely branching LTSs, the equivalence induced by $\mathcal{M}^{\Downarrow,\omega}$ is exactly the same as that induced by $\mathcal{M}^{\Downarrow}$: every such process takes a value where the infinite traces etc are *all* the limits of the finite behaviours.

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[5]), the stable revivals model identifies each process P with a a triple $(\mathit{Tr}, \mathit{Dead}, \mathit{Rev})$, where

- $\mathit{Tr} \subseteq \Sigma^{*\checkmark}$ consists of all P's finite traces (perhaps ending in $\checkmark$).
- $\mathit{Dead} \subseteq \Sigma^*$ consists of all traces (other than terminated ones) after which P can deadlock (reach a state from which no further action is possible).
- $\mathit{Rev} \subseteq \Sigma^* \times \mathcal{P}(\Sigma) \times \Sigma$ consists of all P's revivals. (s, X, a) means that P can perform s, stably refuse X, and then perform a: therefore we restrict revivals to those triples such that $a \notin X$, since plainly it is impossible for a process to refuse X in a given stable state and then have the same state perform a member of X. It is important to note that we have allowed $\checkmark$ neither 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, this 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 need to have observations recorded that tell us these things.[6] Thus the revivals we record 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 new model has the separate representation of deadlock. Without this, the only way of distinguishing between SKIP and $\mathit{SKIP} \sqcap \mathit{STOP}$ is to observe the refusal of sets including $\checkmark$ in the latter process.

[5] $\checkmark$ 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 our motivations was to permit the development of the ideas in [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.

[6] 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.

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.

We quote three versions of the next healthiness condition. The first one is only sufficient when the overall alphabet Σ is finite:

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

With a finite alphabet, one can look at all the members of Σ not in $X \cup \{a\}$, one after another, to extend X in (s, X, a) to Z such that $a \notin Z$ and $(s, Z, b) \in R$ for all $b \in \Sigma - Z$.[7]

With an infinite alphabet one cannot step through the whole of it in this way: we need to adopt the principle we just proved in this restricted case as the healthiness condition itself. Since earlier drafts of the present paper were restricted to the case of finite Σ, Samuel *et al* in [30] proposed their own version of **Rev3** to handle infinite alphabets. This is

Rev3 $(s, X, a) \in Rev$ and $\forall\, b \in Y.(s, X, b) \notin Rev$ implies $(s, X \cup Y, a) \in Rev$.

In other words, any revival can be extended by all the events $b \in Y$ such that (s, X, b) is not a revival: the argument for this is that the same state that witnesses the revival (s, X, a) cannot have any action of Y available.

Setting $Z = \{b \mid (s, X, b) \notin Rev\}$, we see that $X \subseteq Z$ and $(s, Z, a) \in Rev$ by **Rev3**.

In fact one can deduce [30] that not only is $(s, Z, a) \in Rev$, but actually $(s, Z, b) \in Rev$ for every $b \in \Sigma - Z$. For by construction $(s, X, b) \in Rev$ and $(s, X, b') \notin Rev$ for any $b' \in Z$. It follows that $(s, X \cup Z, b) = (s, Z, b) \in Rev$ by **Rev3**.

We can thus deduce that **Rev3** proves the following alternative version of the axiom, which itself easily proves **Rev3**.

Rev3′ $(s, X, a) \in Rev \Rightarrow \exists\, Z \subseteq \Sigma - \{a\}.X \subseteq Z \wedge \forall\, b \in \Sigma - Z.(s, Z, b) \in Rev$.

As they are equivalent the reader may select either **Rev3** or **Rev3′**.

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).

[7] To demonstrate this, set $Z_0 = X$ and $B_0 = \{b\}$ and $\Sigma - (Z_0 \cup B_0) = \{c_1, \ldots c_n\}$. For each $0 \leq m < n$ we set $Z_{m+1} = Z_m$ and $B_{m+1} = B_m \cup \{c_{m+1}\}$ if $(s, Z_m, c_{m+1}) \in Rev$; otherwise $Z_{m+1} = Z_m \cup \{c_{m+1}\}$ and $B_{m+1} = B_m$. We can prove that if $b \notin \{c_{m+1}, \ldots, c_n\}$ then $b \notin Z_m$ implies $(s, Z_m, b) \in Rev$ by induction: in the second case we know that $(s, Z_{m+1}, b) \in Rev$ for all $b \in B_m$ by **Rev3**^{fin}.

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

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

Replacing the last clause with $?x : \Sigma \to CHAOS$ (which works for $\mathcal{F}$) does not work in this model, because it would not have any revival of the form (s, X, a) 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 $a \to STOP$, but this is not true in models with representations of divergence and deadlock, 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 (component-wise union) 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 $Q = b \to STOP$ and $P = \mathbf{div} \mathbin{\Box} a \to P$, 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 basic CSP 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 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), P_n has no actions from X, 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 $\Phi(N) = (\mathit{Tr}_N, \mathit{Dead}_N, \mathit{Rev}_N)$ 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).

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

Fig. 1. Trace semantics

The semantic clauses for *traces* are, of course, identical to those for the traces model (and for $\mathcal{F}$): see Figure 1.

The calculation of most cases of $deadlocks(P)$ can be presented as typical denotational semantic clauses: see Figure 2. But this breaks down for parallel operators involving synchronisation, and specifically $\underset{X}{\|}$. If, for example $R = (a \to R) \sqcap (b \to R)$, 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.

$$
\begin{aligned}
deadlocks(STOP) &= \{\langle\rangle\} \\
deadlocks(SKIP) &= \emptyset \\
deadlocks(\mathbf{div}) &= \emptyset \\
deadlocks(a \to P) &= \{\langle a\rangle^\frown s \mid s \in deadlocks(P)\} \\
deadlocks(?x : A \to P) &= \{\langle\rangle \mid A = \emptyset\} \\
&\quad \cup \{\langle a\rangle^\frown s \mid a \in A \land s \in deadlocks(P[a/x])\} \\
deadlocks(P \mathrel{\Box} Q) &= ((deadlocks(P) \cup deadlocks(Q)) \cap \{s \mid s \neq \langle\rangle\}) \\
&\quad \cup (deadlocks(P) \cap deadlocks(Q)) \\
deadlocks(P \sqcap Q) &= deadlocks(P) \cup deadlocks(Q) \\
deadlocks(\bigsqcap S) &= \bigcup\{deadlocks(P) \mid P \in S\} \\
deadlocks(P \rhd Q) &= deadlocks(Q) \cup \{s \in deadlocks(P) \mid s \neq \langle\rangle\} \\
deadlocks(P;\ Q) &= deadlocks(P) \\
&\quad \cup \{s^\frown t \mid s^\frown\langle\checkmark\rangle \in traces(P) \land t \in deadlocks(Q)\} \\
deadlocks(P \bigtriangleup Q) &= \{s^\frown t \mid s \in traces(P) \cap \Sigma^* \land t \neq \langle\rangle \\
&\qquad \land t \in deadlocks(Q)\} \\
&\quad \cup \{s \mid s \in deadlocks(P) \land \langle\rangle \in deadlocks(Q)\} \\
deadlocks(P[\![R]\!]) &= \{t \mid \exists\, s \in deadlocks(P) . s \mathrel{R} t\} \\
deadlocks(P \setminus X) &= \{s \setminus X \mid s \in deadlocks(P)\}
\end{aligned}
$$

Fig. 2. Deadlock clauses other than for parallel

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

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

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.

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

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 of subsets of Σ from stable (as opposed to $\checkmark$-stable) states:

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

The only things which remain to be constructed for our semantics are the clauses for *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. We make use of the fact that *failures*(P) can be derived from the value in $\mathcal{R}$ of P.

$$\mathit{revivals}(\mathit{STOP}) = \emptyset$$
$$\mathit{revivals}(\mathit{SKIP}) = \emptyset$$
$$\mathit{revivals}(\mathbf{div}) = \emptyset$$

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

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

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

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

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

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

$$\begin{aligned}
revivals(P \underset{X}{\parallel} Q) = \{(u,\, Y \cup Z,\, a) \mid\; & \\
& \exists\, s, t.(s,\, Y) \in failures^\flat(P) \wedge (t, Z) \in failures^\flat(Q) \\
& \wedge\, u \in s \underset{X}{\parallel} t \wedge Y - X = Z - X \\
& \wedge\, ((a \in X \wedge (s,\, Y,\, a) \in revivals(P) \\
& \qquad\qquad\qquad \wedge\, (t,\, Z,\, a) \in revivals(Q)) \\
& \quad \vee\, a \notin X \wedge ((s,\, Y,\, a) \in revivals(P) \\
& \quad \vee\, a \notin X \wedge (t, Z, a) \in revivals(Q))))\} \\
& \cup\, \{(u,\, Y \cup Z,\, a) \mid \\
& \exists\, s, t.(s,\, Y,\, a) \in revivals(P) \wedge t\hat{\ }\langle\checkmark\rangle \in traces(Q). \\
& Z \subseteq X \wedge a \notin X \wedge u \in s \underset{X}{\parallel} t\} \\
& \cup\, \{(u,\, Y \cup Z,\, a) \mid \\
& \exists\, s, t.(t,\, Z,\, a) \in revivals(Q) \wedge s\hat{\ }\langle\checkmark\rangle \in traces(P). \\
& Y \subseteq X \wedge a \notin X \wedge u \in s \underset{X}{\parallel} t\}
\end{aligned}$$

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

$$\begin{aligned}
revivals(P[\![R]\!]) =\; & \{(s', X, a') \mid \exists\, s, a.s \mathrel{R} s' \wedge a \mathrel{R} a' \\
& \quad \wedge\, (s, R^{-1}(X), a) \in revivals(P)\}
\end{aligned}$$

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

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

Notice that the clause for the parallel operator has become more complicated because we have to deal with the cases in which the final event of the revival is, and is not, synchronised, and with the cases where only one of P and Q has terminated. 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 (see [30]). 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 $a \to P$.
- 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 $s\hat{\ }t$ where $s\hat{\ }\langle\checkmark\rangle \in \mathit{traces}(P)$ 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

[8] To amplify this: 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 $(a \to b \to a \to \mathit{STOP}) \setminus \{b\}$ from that of $a \to b \to a \to \mathit{STOP}$: a step of the extended revival is lost to hiding, as explained by the following example. The process $(a \to b \to a \to \mathit{STOP}) \rhd (a \to b \to \mathbf{div})$ has exactly the same representation as $a \to b \to a \to \mathit{STOP}$, but hiding b in the more complex process does not give the extended revival $(\langle\rangle, \emptyset, \langle a, a\rangle)$.

process variable. We explained informally in Section 3 why this is the correct fixed point to choose. As an example, consider the definition $P = a \to (P \setminus \{a\})$. 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. Even if $\Sigma = \{a\}$, this fixed point incorrectly predicts deadlock after $\langle a\rangle$.

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; 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 *Let* **CSP** *be the LTS of closed CSP terms under its operational semantics and V be the set of process variables. Then for each CSP term P and each $\sigma : V \to$* **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 structure 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 representing one iteration of the unwinding over 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: $\Phi(\sigma'[p]) = \Phi(Q)$. $F_O(Q)$ 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 $\alpha \in \mathcal{R}$ 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 rather than $\mathbf{div}$ to make a clear distinction between syntax and $\mathcal{R}$-semantics.)
- The structural induction over terms means we may assume that 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. 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 $\Phi(F^{k+1}(\mathbf{div}))$, which we know equals $F_D^{k+1}(\Phi(\mathbf{div}))$, so that all such 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 of the recursive case.

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 $\mathcal{R}$.

PROOF 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.

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

We know that the trajectories (operational sequences of states and actions) 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. N_n is stable and refuses $Y \cup X$, and so 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 similar to this one.

The combination of these lemmas and the one for recursion completes 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 **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 **L** are *dense* (a topological or order-theoretic concept) in $\mathcal{M}$. Because CSP has infinitary syntax if we wish, we can 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 does not have the trace $\langle \mathit{fail} \rangle$" for any fixed event *fail*; the stable failures model is fully abstract with respect to the test "P cannot deadlock on $\langle\rangle$"; and the failures/divergences model $\mathcal{N}$ is fully abstract with respect to the test "P can neither deadlock nor diverge on $\langle\rangle$" for finitely nondeterministic CSP. For details see [23].

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

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

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

PROOF We will represent 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:

$$\begin{aligned} T_{\langle\rangle} &= \mathbf{div} \\ T_{\langle a\rangle^\frown s} &= \mathbf{div} \,\Box\, a \to T_s \end{aligned}$$

  Note that this has no deadlocks or revivals, only the traces which are prefixes of s and are therefore implied by the healthiness conditions given that s is present.
- We can extend the above with the rule $T_{\langle\checkmark\rangle} = \mathit{SKIP}$ to deal with traces of the form $s^\frown\langle\checkmark\rangle$. $T_{s^\frown\langle\checkmark\rangle}$ has no deadlocks or revivals.
- For a deadlock $s \in \Sigma^*$ we define

$$\begin{aligned} D_{\langle\rangle} &= \mathit{STOP} \\ D_{\langle a\rangle^\frown s} &= \mathbf{div} \,\Box\, a \to D_s \end{aligned}$$

  This has no revivals, but it has 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 we can use **Rev3′** to pick Z such that $a \notin Z$, $X \subseteq Z$, and $(s, Z, b) \in \mathit{revivals}(V)$ for all $b \notin Z$. It follows that all the traces and revivals of the process $R_{(s,Z)}$ defined

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

  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, in all the cases above, 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! This is necessary sometimes since $\mathcal{R}$ does contain elements which look a little bizarre (like the individual

[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.

constructions above) thanks to unseen divergence. On the other hand, our constructions create odd-looking implementations of values that seem well behaved such as processes that would be deterministic if they had empty sets of divergences.

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 the stable revivals model 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 $(P \mid Q) \setminus X$ (where X is the set of labels on which P and Q interact) is stuck if the unrestricted process $P \mid Q$ can perform a trace of non-X events, and then be stable and able to perform an event in X but none outside X. 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 $(s, \Sigma - \{\mathit{request}\}, \mathit{request})$ for any $s \in (\Sigma - \{\mathit{request}\})^*$.

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 composition $P \underset{J}{\|} Q$ it can be expressed formally as follows. There are no revival (s, X, a) of Q and failure (t, Y) of P with $s \upharpoonright J = t \upharpoonright J$, such that $a \in J$

[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.

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 $P \underset{\{a\}}{\|} STOP$ 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 stuck-freeness 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.*

PROOF 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 $\|$ will mean $\underset{\Sigma}{\|}$, parallel with all events synchronised.

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

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

  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 $P \bigtriangleup SKIP$ 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

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

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

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

  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 ; $\mathbf{div}$ is included to prevent deadlock occurring when P terminates after s.) Therefore the context $C_{dead}(s)[P] = ((P;\ \mathbf{div}) \parallel O_s) \setminus \Sigma) \,|||\, a \to \mathbf{div}$ fails $\mathbf{T}_{\mathcal{R}}$ if and only if $s \in D$.
- Finally, consider a revival (s, X, b). Define

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

  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), because if P offered any element of X along with b, $C_1[P]$ 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

$$C_2[U] = U[\![^{a,\, d}/_{a,\, a}]\!] \parallel Reg_{\#s} \qquad \text{where}$$

$$\begin{aligned} Reg_0 &= a \to Reg_0 \,\Box\, c \to Reg_0 \\ Reg_{n+1} &= d \to Reg_n \,\Box\, c \to Reg_n \end{aligned}$$

  $C_2[U]$ 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 reaches 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. ∎

Aside from determining stuckness and *RespondsTo*, revivals have a role in specifying certain other sorts of property that cannot be expressed using failures alone.

Revivals allow us to insist that *whenever* some event a is offered, the total offer must satisfy some condition. Thus we get conditional control over what is offered from stable states. The way that both *RespondsTo* and stuck-freeness ban processes from making certain sorts of stable offer can be viewed as extreme forms of this.

A less extreme example is the statement that whenever P offers the event a it must also offer the event b. One cannot make this statement in failures, but it is easy in revivals: $\forall\, s.(s, \{b\}, a) \notin \mathit{Rev}$. Similarly one can state that whenever an event from the set A is offered, then so must be some event from B: $\forall\, s.\forall\, a \in A.(s, B, a) \notin \mathit{Rev}$. And $\forall\, s.\forall\, a \in A.\forall\, b \in B.(s, \{b\}, a) \notin \mathit{Rev}$ says that whenever any member of A is offered then so must the *whole* of B.

Here, B might consist of actions like $\{\mathit{back}, \mathit{cancel}, \mathit{home}\}$.

Thus revivals allow us to make certain types of useful specification that are not expressible over $\mathcal{F}$. This model has been implemented in version 2.90 of FDR to enable these extra types of specification to be checked. On the other hand, when checking any specification, it is always a good idea to check it in whichever model is the simplest that can express it. Not only is it likely to be algorithmically more efficient, but the errors reported by a tool like FDR in the event that a specification fails will be easier to interpret.

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 $\checkmark$-stable and therefore exhibit a refusal. The situation is not quite so easy with $\mathcal{R}$, since now our process can diverge, deadlock, terminate ($\checkmark$) 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.

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 $traces_\bot(P)$, $deadlocks_\bot(P)$ and $revivals_\bot(P)$ are: the ones from $\mathcal{R}$ plus all those associated with strict divergences. It follows that we can calculate the value of any CSP process in $\mathcal{R}^\Downarrow$, $\mathcal{R}^{\Downarrow\omega}$ and $\mathcal{R}^\#$ from its values in known models.

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 revivals 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 by:

DS1 $t \in Div \Rightarrow t\hat{\ }s \in Div$

DS2 $Div \subseteq Dead$

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

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 $t \in Div \Rightarrow t\hat{\ }u \in Inf$

Inf1 $t\hat{\ }u \in Inf \Rightarrow t \in Tr$

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 divergence strictness property

WDS $\overline{Div} \cap \Sigma^\omega \subseteq Inf$

where $\overline{X}$, for trace set X, is the union of X and 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 in slightly more general languagea principle discussed on page 257 of [23]:

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 divergence 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.*

PROOF This is analogous to that of Lemma 1.1 in [25]. All one has to show is that, if $P = (\mathit{Tr}, \mathit{Dead}, \mathit{Rev}, \mathit{Div}, \mathit{Inf})$ 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 $u \in \mathit{Inf}$, then there is a closed process P_u containing u such that $P \sqsubseteq P_u$. This is because we can then define

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

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 (through 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 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 number 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 a closed process would then be modified so that every infinite behaviour, that is a limit of finite ones of the process, is present.

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 as over $\mathcal{N}$, $\mathcal{U}$ (both [23]), and for $\mathcal{R}^{\#}$ they are the same as over $\mathcal{SBD}$ [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 they are closed up under divergences, for example

$$\begin{aligned} deadlocks_{\perp}(P \setminus X) = {} & \{s \setminus X \mid s \in deadlocks_{\perp}(P)\} \\ & \cup\, divergences(P \setminus X) \end{aligned}$$

$$\begin{aligned} revivals_{\perp}(P \,\Box\, Q) = {} & \{(\langle\rangle, X, a) \mid (\langle\rangle, X) \in failures_{\perp}(P) \cap failures_{\perp}(Q) \\ & \qquad \wedge (\langle\rangle, X, a) \in revivals_{\perp}(P) \cup revivals_{\perp}(Q)\} \\ & \cup \{(s, X, a) \mid (s, X, a) \in revivals(P) \cup revivals_{\perp}(Q) \\ & \quad \wedge s \neq \langle\rangle\} \\ & \cup \{(s, X, a) \mid s \in divergences(P \,\Box\, Q) \wedge a \notin X\} \end{aligned}$$

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

6.3 Full abstraction

Any closed process can be expressed in (infinitary) CSP. This is easier to prove in the case where the alphabet Σ is finite, so we address that first:

THEOREM 6.2 *When Σ is finite, 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.*

PROOF 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 $(Tr, Dead, Rev, Div) \in \mathcal{R}^{\Downarrow}$.

$$RSD(\xi) = \begin{cases} \mathbf{div} \quad \text{if } \langle\rangle \in Div, \text{ and otherwise} \\ (?x : (initials(\xi) - \{\checkmark\}) \to RSD(\xi/\langle x\rangle)\}) \\ \triangleright \\ \sqcap(\{STOP \mid \langle\rangle \in Dead\} \\ \quad \cup \{SKIP \mid \langle\checkmark\rangle \in Tr\} \\ \quad \cup \{?x : (\Sigma - Z) \to RSD(\xi/\langle x\rangle) \mid (\langle\rangle, Z, a) \in Rev, \\ \forall\, b \in \Sigma - Z.(\langle\rangle, Z, b) \in Rev\}) \end{cases}$$

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, $\langle\checkmark\rangle$ as a trace, or a revival whose trace is $\langle\rangle$, so the nondeterministic choice above is nonempty (noting that every revival $(\langle\rangle, X, b)$ can be extended to $(\langle\rangle, Z, b)$ for some suitable Z by **Rev3′**. 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 accordingly.

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 are 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, $\triangleright$ 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 $\triangleright$ 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.

$$RNSD(\xi) = \begin{cases} (?x : (\mathit{initials}(\xi) - \{\checkmark\}) \to RNSD(\xi/\langle x\rangle)\}) \\ \rhd \\ \bigsqcap(\{\mathbf{div} \mid \langle\rangle \in \ \mathit{div}\} \\ \quad \cup\{STOP \mid \langle\rangle \in \mathit{Dead}\} \\ \quad \cup \{SKIP \mid (\langle\checkmark\rangle \in \mathit{Tr}\} \\ \quad \cup \{?x : (\Sigma - Z) \to RNSD(\xi/\langle x\rangle) \mid \forall\, b \in \Sigma - Z.(\langle\rangle, Z, b) \in \mathit{Rev}, \\ \qquad \nexists\, X' \supset X.(\langle\rangle, X', a) \in \mathit{Rev}\}) \end{cases}$$

This completes the proof of Theorem 6.2. ∎

We cannot simply remove the assumption of finite Σ from the above theorem, since it is not then true: there are closed processes such as the most nondeterministic deadlock free process

$$DF = \bigsqcap\{a \to DF \mid a \in DF\} \sqcap SKIP$$

that cannot be expressed without infinite nondeterminism. What we can say, however, is that for every closed process P and every finite set F of P's revivals there is a finitely nondeterministic $P' \sqsupseteq_R P$ that has all of P's traces, divergences and deadlocks, and has the whole of F. To prove this one can use exactly the same constructions as above, but only use the revivals (Z) clause for revivals in F together with an arbitrary single revival for each trace not represented any of F, or the deadlocks, termination traces and divergences of P.

Theorem 6.2 and the modified argument in the last paragraph, together with the closure principle and the use of nondeterministic choice, easily give us the following:

THEOREM 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;

$\mathbf{T}_{\mathcal{F}}^{\Downarrow}$ is failed by P if 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

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

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 P_1. That is entirely possible in itself, since we could use the context

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

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 CSP context can be devised that overcomes this problem. [11]

What we seem to need for the CSP language of this paper is 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 $\mathbf{T}^{\Downarrow}_{\mathcal{F}}$.*

PROOF 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

[11] In the author's sequel to this paper [28] he describes how related problems to this one suggest the addition of a new operator Θ_b (throw) to CSP: $P \,\Theta_b\, Q$ behaves like P until it performs the action b, after which it starts the process Q. He terms the extended language CSP+, which has the qualities required in [28], and which also solves the problem mentioned here. One can now replace $F(P)$ in the definition of $C'[P]$ by $F(P)\,\Theta_b\, STOP$, thus creating a CSP+ context that maps precisely those P that can deadlock but not diverge on $\langle\rangle$ to a process ($a \to STOP$ or $STOP \sqcap a \to STOP$) that fails $\mathbf{T}^{\Downarrow}_{\mathcal{R}}$.

not have, even though it has both the failure (s, X) and the trace $s\hat{}\langle b\rangle$. 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 $s \notin divergences_{\perp}(P)$, $C_{rev}(s, X, a)[P]$ 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 $\mathbf{T}_{\mathcal{R}}$, $\mathbf{T}_{\mathcal{F}}$ and $\mathbf{T}_{Div}$.*

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 from 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.

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 be because of the underlying properties of the implementation as in Statemate Statecharts [11,29] and (for the *tock* event) in discrete timed models of CSP [17]. Alternatively, an observer might choose only to communicate with the process once stability is observed, perhaps out of caution. To model these things 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

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

stable ought to give, respectively, $a \to STOP$ and $a \to a \to STOP$ 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 should be used 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 omitting 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 that 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, $\mathcal{N} = \mathcal{F}^{\Downarrow}$ (as opposed to $\mathcal{F}^{\#}$) is normally adequate for most reasoning purposes for finitely nondeterministic processes, we expect that normally the right model to use in these circumstances 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 the 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 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. The assumptions we made earlier about the nature of models imply that if $\mathcal{M}$ is one of our models defined over an alphabet Σ large enough to contain all the events used by processes P and Q, then P and Q are equivalent in $\mathcal{M}$ if and only if they are equal over the corresponding model defined over a larger $\Sigma' \supseteq \Sigma$. In this section we will frequently need to extend the basic alphabet $\Sigma = \Sigma_0$ by further events, but this argument shows that the equivalence is not affected by doing so.

We assume similar properties for *congruences*: below, we only consider ocngruences that make sense for CSP processes defined over any size of alphabet (again, conceivably, with some infinite upper bound), and where the equivalence of otherwise of two processes is independent of alphabets large enough to express the processes.

If $\mathcal{X}$ identifies every pair of processes identified by $\mathcal{Y}$, we write $\mathcal{X} \preceq \mathcal{Y}$. If $\mathbf{S}$ is any set of congruences, and $\mathcal{X} \preceq \mathcal{Y}$ for all $\mathcal{Y} \in \mathbf{S}$ we will say that $\mathcal{X}$ is *sub*-$\mathbf{S}$.

Notice that every finite observation model is, by definition, a sub-$\mathcal{FL}$ congruence. It is unclear to the author whether there are any such congruences for which there is no congruent finite observation model. ¶

The following is the main theorem of this section.

THEOREM 7.1 *For the dialect of CSP set out in Section 2, the* only *sub-$\{\mathcal{RT}, \mathcal{A}\}$ models are $\mathcal{R}$, $\mathcal{F}$, $\mathcal{T}$ and the trivial congruence $\mathcal{NULL}$ that identifies all processes. Furthermore, every finite observation model $\mathcal{X}$ that is not $\mathcal{NULL}$, $\mathcal{T}$ or $\mathcal{F}$ satisfies $\mathcal{R} \preceq \mathcal{X}$.*

In other words, every non-trivial model that is not a member of the initial sequence $\mathcal{T} \prec \mathcal{F} \prec \mathcal{R}$ is strictly finer than $\mathcal{R}$, and furthermore this linearly ordered sequence cannot be extended since $\mathcal{A}$ and $\mathcal{RT}$ have no other congruence more abstract than them both.

The proof of this theorem consists of four main 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 to each other and establish successively that any sub-$\mathcal{FL}$ model 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.

In the whole of the proof we only use the assumption that we are reasoning about *models* rather than the more liberal idea of a *congruence* once (in the proof of Lemma 7.5). The author does not know if this assumption can be dropped. ¶

LEMMA 7.2 *If $\mathcal{M}$ is any sub-$\{\mathcal{RT}, \mathcal{A}\}$ congruence then $\mathcal{M} \preceq \mathcal{R}$.*

PROOF 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:

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

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 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 follows that the only behaviours recorded for $\mathcal{RT}$ of the processes $P_{\mathcal{A}}$ and $P_{\mathcal{R}}$ are of the three forms

(i) $\langle \bullet, a_1, \bullet, \ldots, \bullet, a_n, \bullet \rangle$ A
(ii) $\langle \bullet, a_1, \bullet, \ldots, a_n, X \rangle$
(iii) $\langle \bullet, a_1, \bullet, \ldots, X, a_n, \bullet \rangle$ where $a_n \notin X$,

where every behaviour of form (ii) is extendible either to one of form (iii) or to one of the form $\langle \bullet, a_1, \ldots, a_n, \Sigma \cup \checkmark \rangle$. Thus every behaviour recorded in the $\mathcal{RT}$ representation of $P_{\mathcal{A}}$, namely $(P_{\mathcal{A}})_{\mathcal{RT}}$ is deducible from one of P's $\mathcal{R}$ behaviours: traces, revivals and deadlocks, and indeed $P_{\mathcal{A}} =_{RT} P_{\mathcal{R}}$.

Thus, since $\mathcal{M} \preceq \mathcal{RT}$, we have $P_{\mathcal{A}} =_M P_{\mathcal{R}}$.

Since $P =_A P_{\mathcal{A}}$ by construction, and $\mathcal{M} \preceq \mathcal{A}$, we also have $P =_M P_{\mathcal{A}}$. Putting these two things together gives us $P =_M P_{\mathcal{R}}$. If $\mathcal{M}$ distinguishes any pair of processes identified by $\mathcal{R}$ this could not be true in general, so we may deduce that $\mathcal{M} \preceq \mathcal{R}$. ▮

Given this result, what we must prove to establish Theorem 7.1 is that the four models listed above are more abstract than all other finite behaviour models. 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 $\mathcal{FL}$ 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 7.4 below.

LEMMA 7.3 *Every sub-$\mathcal{FL}$ congruence $\mathcal{M}$ for CSP that distinguishes at least two processes satisfies $\mathcal{T} \preceq \mathcal{M}$.*

PROOF We can assume there are processes $P \sqsubset_{FL} Q$ in any sub-$\mathcal{FL}$ 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 $K \sqcap L \sqsubset_{FL} K$ since $M \preceq \mathcal{FL}$).

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 a 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 extend the alphabet Σ in which we are modelling processes to contain an element e that is not used in the particular P and Q chosen.

Let $C_{\mathcal{T}}(s)[\cdot]$ be the context that, as 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

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

This has the very useful property that its $\mathcal{FL}$ 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 $\mathcal{FL}$ value of $W \bigtriangleup \mathit{SKIP}$, because ✓ 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 $(e \to \mathit{SKIP}) \rhd \mathit{STOP}$ if it does.

Now consider

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

If W does not have the trace s this equals

$$((\mathit{STOP};\ P) \mathrel{\Box} e \to Q) \setminus \{e\} =_{FL} (\mathit{STOP} \mathrel{\Box} e \to Q) \setminus \{e\} = Q$$

If W does have the trace then it equals

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

by various standard CSP laws and inspection of the operational semantics; the last line following from $P \sqsubseteq_{FL} Q$. ∎

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 5, are incomparable since neither is weaker than the other.

LEMMA 7.4 *Every sub-$\mathcal{FL}$ congruence $\mathcal{M}$ for CSP that distinguishes at least two processes not identified by $\mathcal{T}$ satisfies $\mathcal{F} \preceq \mathcal{M}$.*

PROOF We can, in the same way as for the previous lemma, assume that there are $P \sqsubseteq_{FL} Q$ identified by $\mathcal{T}$ but not by $\mathcal{M}$. By Lemma 7.3 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 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 $\checkmark$-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 $\checkmark$-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 $C_{\mathcal{F}}(s,X)[U] \;\square\; \mathit{SKIP}$ does not work since the SKIP can resolve $\square$. This can be overcome by using an event e, not used in P and Q, to guard SKIP.

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

Thus context maps U to P and V to Q, completing the proof of this lemma. ∎

Both of the above proofs were reasonably straightforward in that they were able to map U and V to an arbitrary pair of processes such that $P \sqsubseteq_{FL} Q$ (and $P =_T Q$ in the second case). The author has not managed to find such a proof for the final lemma (Lemma 7.6), so its proof is more technical. Specifically, the author has found no way of handing general P and Q as above, but rather the P' and Q' shown to exist by the following preliminary lemma.

LEMMA 7.5 *Suppose $P \sqsubset_{FL} Q$ and $P \neq_M Q$ for some sub-$\mathcal{FL}$ model $\mathcal{M}$. Then there exist P' and Q' such that*

- $P \sqsubseteq_{FL} P' \sqsubset_{FL} \sqsubset Q' \sqsubseteq_{FL} Q$

- $P' \neq_M Q'$
- *There is a single $\mathcal{FL}$-behaviour β such that $P' = Q' \cup \{\beta\}$ (in their $\mathcal{FL}$-representations).*

PROOF To prove this we use an enumeration of $P - Q$ using some ordinal α: $\{\beta_i \mid u \in \alpha\}$. Since, for all β there are only finitely many $\beta' < \beta$, we can assume that $\beta_i < \beta_j \Rightarrow i < j$.

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:

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

We define:

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

We can then define a process Q_i for all $i \leq \alpha$:

- $Q_0 = Q$
- $Q_{i+1} = Q_i \cup FLB(\beta_i)$
- $Q_\lambda = \bigsqcap\{Q_i \mid i < \lambda\}$ for λ a limit ordinal.

By construction we have $Q_i \sqsubseteq_{FL} Q_j$ for $i < j$, and $Q_\alpha = P$, and by our choice of enumeration we have that $Q_{i+1} = Q_i \cup \{\beta_i\}$ for all i.

It is evident that there must be some least ordinal j such that $Q_j \neq_M Q$. Clearly $j \neq 0$, and j cannot be a limit ordinal. If it were then, as $Q_j \neq_M Q$, there must be some member ξ of one of the components of the $\mathcal{M}$ representation of Q_j that does not belong to the $\mathcal{M}$ representation of Q. As each such component is, by Definition 3.1, a relational image of the $\mathcal{FL}$ representation, it follows that there is some $\beta \in Q_j$ that maps to ξ. Since Q_j is just the union of $\{Q_i \mid i < j\}$ it follows that there is such a Q_i with β. Therefore Q_i could not be $\mathcal{M}$-equivalent to Q, contradicting the fact that j is minimal.

It follows that $j = i + 1$ for some i. Setting $Q' = Q_i$ and $P' = Q_{i+1}$ we have proved our lemma. ▮

LEMMA 7.6 *If $\mathcal{M}$ is any non-trivial sub-$\mathcal{FL}$ CSP model other than $\mathcal{T}$ or $\mathcal{F}$, then $\mathcal{R} \preceq \mathcal{M}$.*

PROOF Let $\mathcal{M}$ be such a model. By our earlier result we know that $\mathcal{F} \preceq \mathcal{M}$. We can assume that there are $P \sqsubseteq_{FL} Q$ such that $P =_F Q$ and $P \neq_M Q$. Applying Lemma 7.5 to P and Q we see that necessarily $P \sqsubseteq_F P' \sqsubseteq_F Q' \sqsubseteq_F Q$ and therefore $P' =_F Q'$. Let β be the single $\mathcal{FL}$ behaviour that represents the difference between P' and Q'. Since P' and Q' are trace equivalent, β must have some, and therefore a first, proper acceptance: write

$$\beta = \langle \bullet, a_1, \bullet, \ldots, a_s, A_s\rangle\hat{\ }\beta'$$

where $s > 0$ is the index of the first proper acceptance and β' may or may not be the empty sequence $\langle\rangle$.

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_0 used in P, and a second disjoint set of the same size Σ_1. We will assume that the operation a' represents a bijection from Σ_0 to Σ_1.

Now let $W(\beta_m, \beta^*)$ be the process that behaves identically to $FLB(\beta)$ 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 $t\hat{\ }\langle b\rangle$ 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

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

If β 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. In either case, $T_s[U]$ performs exactly as many as as there are events in β 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 ✓ so that the construct $E[U]$ defined below does not interfere with the rest of β. 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

$$E[U] = (W(\beta_m, \beta^*) \underset{\Sigma_1}{\|} T_s[U][\![a \mapsto \Sigma_0]\!])[\![\mathit{Unprime}]\!]$$

where *Unprime* is the renaming that maps all events c' in Σ_1 to c as well as being the identity function on Σ_0. Except for what happens immediately after s events, this process behaves identically to $W(\beta, \beta^*)$ 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_1 events of $W(\beta, \beta^*)$ 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 with A_s replaced by $\bullet$, or one implied by it. 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 $W(\beta, \beta^*)$ 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.

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 $(t, Y, b) \in$ $\mathit{revivals}(U) - \mathit{revivals}(V)$

$$D[V] =_M = Q' \neq_M P' =_M D[U]$$

which proves our result. ∎

This completes the proof of Theorem 7.1.

The state of knowledge it establishes about the finite observation models is illustrated in Figure 3. The clouds illustrate the regions where there are models we have not completely classified: note that Lemma 7.2 demonstrates that the parts of the main cloud that lie beneath $\mathcal{A}$ and $\mathcal{RT}$ are disjoint: they are represented by the small clouds.

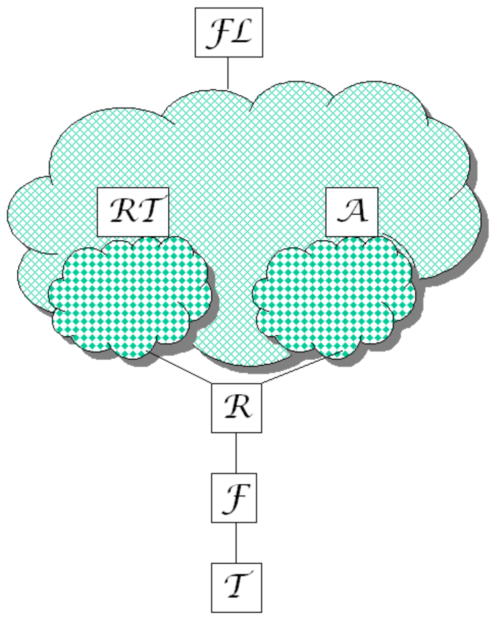

Fig. 3. The hierarchy as revealed by Theorem 7.1

This result raises the question of how it might be extended, for example by refining our knowledge of the clouds or extending this result from finite observation models ¶ to the other classes discussed earlier. Since this paper was submitted for publication, the author has answered the second question for models in the classes $\mathcal{M}^{\Downarrow}$ and $\mathcal{M}^{\Downarrow\omega}$: namely, we get an exact analogy of Theorem 7.1, but only with the addition of the Θ_a operator referred to in a footnote earlier.

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

- Firstly, the three other families of congruences are themselves more abstract than ones such as $\mathcal{R}^{\#}$, so there will certainly be more than four congruences below than this model under $\prec$.
- Secondly, there seems to be nothing to constrain the 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.

It is unclear to the author how much practical benefit there will be from discovering exactly what lies behind the clouds. The author believes, for example, that each of the two small clouds contains an infinite set of models. For example, between $\mathcal{R}$ and $\mathcal{A}$ we can create generalised revivals which, instead of recording only one event that can happen after a refusal is observed, record *up to* K for some fixed $K > 1$. Thus (s, X, Y) says that P can perform s, and stably refuse X from some state that can communicate each event (as alternatives) from the set Y where $|Y| \leq K$. The equivalent model for $K = 1$ is just $\mathcal{R}$, and for $K = 0$ is $\mathcal{F}$. If no limit is placed on K this model is just $\mathcal{A}$.

8 Conclusions

In this paper we have shown how the conformance equivalence defined in [8] and developed as revivals 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 7.1 we showed that $\mathcal{R}$ has a special place as the "biggest of the small models" in rather a striking sense.

As stated at the end of the last section, the author believes that there are many other models of CSP sitting undiscovered behind the clouds in Figure 3, but that the main motivation for discovering them would be academic rather than practical.

He believes that failures models remain the most important ones of CSP, in part thanks to 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 11 where we can guarantee or wish to ensure that observable actions only happen in $\checkmark$-stable states.

In this paper we have concentrated on the denotational semantics of CSP over behavioural models such as $\mathcal{R}$ as well as demonstrating congruence of these with operational semantics. In [13] CSP is given a wide range of algebraic laws, which were codified and supplemented in [23] to give an *algebraic* semantics. One of the main objectives of an algebraic semantics is to characterise the same equivalence as a chosen behavioural equivalence, and it is clear that moving from one behavioural equivalence to another will involve a change in the set of laws. In one respect the move from $\mathcal{F}$ to $\mathcal{R}$ is easy, since only one of the basic laws for $\mathcal{F}$ set out in [23] proves to be false, namely the distribution of internal choice over external choice:

$$P \sqcap (Q \,\Box\, R) = (P \sqcap R) \,\Box\, (P \sqcap R)$$

To see that this is false over $\mathcal{R}$ consider $P = STOP$, $Q = a \to Q$ and $R = b \to B$. The process on the right-hand side above has the revival $(\langle\rangle, \{a\}, b)$, but the one on the left-hand side does not.

It is therefore tempting to hope that the algebraic semantics for CSP over $\mathcal{R}$ will be a small step from that over $\mathcal{F}$ implied [12] in [23]. Unfortunately this is not so. In any model that is at least as rich as $\mathcal{R}$ one can make a distinction that is impossible in models based on failures. That is, we can see, for each trace of the form $s\hat{\ }\langle a\rangle$, whether the final a could have occurred from a stable state or not: the alternative is that it could only have occurred from an unstable state. This can be decided by seeing whether our process has the revival $(s, \emptyset, a)$. The ability to see when events can not occur stably means that the shape of the *step* laws of [23], which demonstrate how each operator behaves when its operands take the form $?x : A \to P(x)$ is no longer sufficiently general to cover all cases, simply because it does not cover the case of events that can only occur unstably. The author believes that the way around this problem is to include laws that show how the various operators respond to processes of the form $(?x : A \to P(x)) \rhd Q$, since the events in A are now offered unstably. This is a substantial task beyond the scope of the present paper. ¶

The investigation of further models for concurrency 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. The model introduced in this paper turned out to be important, in the author's opinion, primarily because of the (literally!) pivotal position it turns out to have in the hierarchy of models.

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

[12] The algebraic semantics in [23] is chiefly for $\mathcal{N}$, but there is an exercise on adapting it to $\mathcal{F}$.

of practically-relevant property. However, for most purposes, the distinctions between

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

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

$$(a \to STOP) \sqcap (b \to STOP) \sqcap STOP$$

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

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

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

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

namely, the seven different $\mathcal{R}$-values of divergence-free processes whose traces are $\{\langle\rangle, \langle a\rangle, \langle b\rangle\}$ 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 (version 2.90). The author was pleasantly surprised to discover, at the same time as he was trying out this functionality for the first time, an industrially relevant application of $\mathcal{R}$. This was of the type discussed in Section 11, namely showing that offers of one class of events implied offers of another.

As reported in [30], Markus Roggenbach's group at Swansea University have recently modelled $\mathcal{R}$ within their CSP Prover technology (based on the theorem prover Isabelle) and thereby verified some of the properties of this model claimed in the present paper. They also discovered an error in the semantics of prefix choice ($?x : A \to P(x)$) given in an earlier draft, as well as discovering some aspects of that version that were open to misinterpretation. Ideally, in future, every paper introducing new theories of CSP should be subjected to such "testing".

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, 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 7.1.

I would like to thank Markus Roggenbach and his group for their embedding of this model in CSP-Prover, as described above.

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 7.1. My work on this topic was funded by US ONR and by a grant from EPSRC, and I completed this version in the congenial surroundings of the UN International Institute for Software Technology in Macao, China.

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
$P \rhd Q$	"time-out" operator (sliding choice)
$P \bigtriangleup Q$	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 Glabbeek, *The linear time - Branching time spectrum I* The handbook of process algebra, Elsevier 2001.

[10] R.J. van Glabbeek, *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), Springer LNCS 1664, Springer-Verlag 1999.

[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 as number 68 at: `web.comlab.ox.ac.uk/oucl/work/bill.roscoe/publications`

[24] A.W. Roscoe, *An alternative order for the failures model*, Journal of Logic and Computation **2**, 5 pp557-577, 1992.

[25] A.W. Roscoe, *Unbounded nondeterminism in CSP*, 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 ENTCS **162**, pp 305-309 (2006).

[28] A.W. Roscoe, *The three Platonic models of divergence-strict CSP*, Proceedings of ICTAC 2008, LNCS 5160.

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

[30] D.G. Samuel, Yoshinao Isobe and M. Roggenbach, *The stable revivals model in CSP-Prover*, Proceedings of AVoCS 2008.