The Timed Failures-Stability Model for CSP

G.M. Reed and A.W. Roscoe[1]

Oxford University Computing Laboratory
11 Keble Road, Oxford OX1 3QD, U.K.

Abstract. We present a mathematical model which is the most abstract allowing (i) a fully compositional semantics for timed CSP and (ii) a natural abstraction map into the standard failures/divergences model of untimed CSP. We discuss the construction and properties of this model and ways in which it can be used to give semantics to CSP reflecting several implementation strategies.

1 Introduction

Although widely used throughout the world in such critical applications as aviation and nuclear power, real-time programming is a poorly understood discipline. There are severe problems which arise in understanding the behaviour of real-time sequential code, for example relating to scheduling policies. The complexity of these problems will only intensify as we increasingly implement distributed real-time systems with the consequent possibility of nondeterministic behaviour. It is imperative that we begin now to develop the formal models on which the eventual solutions must be based. The authors have been working in this area for several years now, and have devised a number of related models for Timed CSP, a straightforward extension of Hoare's CSP notation. This paper presents the model which is central to their work.

Theories of concurrency can be divided into 'untimed' ones, which ignore the precise times at which events occur, concentrating only upon their

[1]The authors gratefully acknowledge that the work reported in this paper was supported by the U.S. Office of Naval Research under grant xxxxxx.

relative order, and 'real-time' ones which do record these times. Untimed theories tend to be simpler to apply and are used when one is not concerned about the precise timing details of a system (or are leaving these for later) and when the system does not rely for its correct internal functioning upon time-dependent features such as timeouts. One of the major contributions of the CSP/CCS conceptual model of concurrency, with no shared memory and handshaken communication, is that it does have a rich and usable untimed theory and, until a few years ago, the literature concentrated on this side.

Nevertheless there are occasions where timed analysis is necessary, and so a number of models and methodologies have arisen for dealing with real time. The authors' philosophy in designing real-time models has always been that the timed theory should not be separate from the untimed one, but should be a natural extension of it where there are well-understood ways of using both theories in the same development. Thus one should be able to prove properties about the untimed behaviour of a system, and be able to use this information rigorously when later refining it to meet timing constraints. Equally, if one is building a large and complex system where one needs to rely on timing only for the correctness of a few components, then we should have ways of localising the more complex timed analysis to those components.

To this end we have developed a number of timed models at different levels of abstraction in such a way that they and the untimed models form a natural hierarchy, with abstraction maps between them. Aspects of this work have already been reported and applied in a number of references, for example []. The key to getting the connection with the untimed models has been our use of the concept of *stability* (a form of observation dual to divergence) together with more obvious ones such as timed analogues of traces and refusals.

The purpose of this paper is to set down in definitive form the construction, philosophy and properties of the model that plays the central rôle in our theory, the *timed failures-stablilty model*. Though relatively complex, it turns out to be the simplest model which both gives a fully compositional congruence for Timed CSP and which extends the standard untimed failures/divergences model of CSP. The model presented here is somewhat more refined than the earlier version which we presented in [].

Having constructed the model we then seek to understand it, and also the nature of nondeterminism in real-time concurrency, by carrying out an in-depth study of the forms of nondeterministic behaviour it predicts.

We will show how the model can be used to give semantics to CSP. It can be argued that it is wrong to settle on a single semantics for a real-time language such as Timed CSP, since to do so constrains the implementor too much. And it is true that if one were implementing the constructs of CSP there would be a wide range of possible timed behaviours possible, no single one of which we could say is 'right'. CSP is, however, essentially a theoretical and specification language rather than one in which implementations are built directly. Therefore we will argue that it is sensible to have a single standard semantics for Timed CSP with as clean and elegant a semantics as possible. The timing details of implementations can then be built up from its constituent parts.

Finally, we will survey the growing body of work that is developing around this and our other models. This includes extensions to our basic theory, connections with other strands of work such as temporal logic, the development of methods to make the application of this work easier, and a number of real applications.

2 Time and topology: the construction of the model

2.1 The syntax of Timed CSP

At this stage it is appropriate to define the language we will use, in order that we can discuss it properly.

The version of untimed CSP we use is essentially that of [BHR,1984], [BR,1985] and [H]. Additionally we will denote by $\perp$ the *diverging* process which performs an infinite sequence of internal actions without communicating. Further, we will allow infinite nondeterministic choices $\sqcap S$ and the hiding of infinite sets of events $P \setminus X$.

One might think that a wide range of additional operators would be required to reflect timed behaviour (e.g., timeouts and interrupts). But in fact, under the standard semantics which we shall see later, it is possible to produce all of the commonly needed ones as *derived* operators (i.e., combinations of standard ones) if we introduce a single extra primitive: $WAIT\ t$ for each real number $t \geq 0$ is the process which for t units of time engages in no event visible to the environment and which then becomes able to terminate successfully. Intuitively, *SKIP* should coincide with $WAIT\ 0$. Therefore, for now at least, we will only add this one construct to the untimed language.

In constructing the language, we assume we are given an alphabet Σ from which all communications are drawn. In the syntax below, a ranges over Σ; X, Y over subsets of Σ; f over the set of functions from Σ to Σ; and F over 'appropriate' compositions of our syntactic operators. $P(a)$ denotes a function from the given X to the space of processes and S ranges over nonempty subsets of the set of processes. p ranges over process variables (needed to define recursions).

$$\begin{aligned} P ::= \;& \perp \mid \mathit{STOP} \mid \mathit{SKIP} \mid \mathit{WAIT}\, t \mid a \to P \mid a : X \to P(a) \mid \\ & P_1 \Box P_2 \mid P_1 \sqcap P_2 \mid \textstyle\bigsqcap S \mid P_1 \parallel P_2 \mid P_1 \, {}_X\|_Y \, P_2 \mid P_1 \,|||\, P_2 \mid \\ & P_1 ; P_2 \mid P \setminus X \mid f^{-1}(P) \mid f(P) \mid p \mid \mu p.P \end{aligned}$$

Technical notes. In order that the above syntax is properly defined we need to place a bound on the size of sets over which we allow ourselves to take nondeterministic choices. This bound can be any cardinal. We will find later that we need additional restrictions on the range of the function $P(a)$ and the members of each infinite set over which we apply $\bigsqcap$. These additional restrictions will be described and discussed later.

2.2 Postulates

Timed CSP inherits more than its syntax from the untimed version of the language. Our basic understanding of what a CSP process is stays the same. It is an entity which communicates in some *alphabet* of atomic events. These communications are still thought of as instantaneous: the moment when an event occurs is the time when the handshake which is 'its essence' takes place. The fact that each sequential process performing an event actually takes some time to perform it is reflected in a delay between the instantaneous occurrence and the time when the sequential process is able to do anything else. Timed CSP also retains the postulate that any event that is observable by the environment can only occur when the environment offers it: a handshake between the process and the environment. This means that the view the environment has of a process is essentially the same as that of a another process with which it might be combined in parallel.

We now state and discuss a number of assumptions we make which are specific to the way we view time. Some we would regard as obvious and others as ones which could have been varied. Yet others turn out to be necessary for subtle reasons we seek to explain.

(1) **Continuous time domain.** The time domain consists of the nonnegative real numbers $\mathbf{R}^+$, and there is no lower bound on the time difference between consecutive observable events. The other plausible general-purpose time domain would be the natural numbers $\mathbf{N}$ (i.e., nonnegative integers). We choose $\mathbf{R}^+$ rather than $\mathbf{N}$ because the latter implies a granularity which might be appropriate in modelling a synchronous system, but we wish to model processes running asynchronously in parallel. Using $\mathbf{N}$ in the latter case would sometimes force us to regard two events as happening simultaneously even when they do not, which might lead us into errors when reasoning about the system where this occurred. It will, nevertheless, be necessary to allow several events to happen at the same time since there is nothing to stop a pair of unsynchronised parallel processes communicating simultaneously.

We do not specify the units being used to model time: they might be nanoseconds, seconds or years so far as the theory is concerned. However in describing examples it is useful to follow the convention that the time consumed by the completion of an event as described above is generally much less than 1.

(2) **A global clock.** We assume that all events recorded by processes within the system relate to a *conceptual* global clock. This is time as recorded by some notional environment which interacts with the process and observes what happens and when. The environment's clock is not available in any sense to the processes comprising a network. This single thread of observed time leads to greater simplicity and abstraction.

When an application requires a clock which processes can refer to, then we must model the clock directly in Timed CSP, probably as a process that runs in parallel with the ones which use it. We might well build some nondeterminism into the definition of such a clock to allow for the fact that it does not keep perfect time, and if there were more than one such clock then this nondeterminism would allow for them drifting apart.

(3) **Realism.** We postulate that no process can perform infinitely many actions in a finite time. It is necessary to build this postulate into any semantics we build for CSP. Given the language described above, one would expect it to be maintained under the condition that any unwinding of a recursion is assumed to take time bounded below by some positive constant δ. It also turns out to be necessary, because of the expressive power of our model, to impose constraints on the domains of infinitary operators such as $\sqcap$ and $a : X \rightarrow P(a)$. This will be discussed more later.

(4) **Hiding and termination.** We wish $(a \to P)$ to denote the process that is willing at *any* time to engage in the event a and then to behave like the process P. Clearly, if $P = a \to P$,we then wish $P \setminus a = \perp$. However, consider $P = a \to STOP$ (the process that is willing to engage in a at any time ≥ 0 and then to deadlock). What do we wish $P \setminus a$ to denote?

We have already discussed the principle that, in CSP, observability is equated with external control. Given the process $a \to P$ and an environment eager to perform an a immediately, we would expect that a would indeed occur at time 0. By hiding, we remove control over the event(s) hidden. Hence, any time a process is willing to engage in a hidden action, it is permitted to do so and we would expect the hidden event to occur if no other event did. Thus, we assume that each hidden event takes place as soon as such an event becomes possible.

Our intuitive model of hiding is that of placing a given process within a box in which all the events to be hidden are constantly on offer, and then concealing all the hidden events within the box from the environment.

In the above example, we would wish:

$$(a \to STOP) \setminus a = WAIT\,\xi; STOP$$

where ξ is the time (assumed here to be deterministic) for the completion of the event a.

In order to model this idea of an event occurring as soon as it becomes available, we will need to record (either explicitly or implicitly) not only those times at which events are available, but also those at which they can *become* available.

Exactly the same argument applies to occurrence of the termination event $\surd$ in the sequential composition $P; Q$. The effect of this composition is to make such an event invisible and automatic, exactly as in hiding. Therefore we make the same assumption, namely that the hidden $\surd$ will occur and enable Q as soon as it becomes available in P.

(5) **Stability and the treatment of divergence.** As indicated earlier, stability plays the same rôle in the timed models as its dual, divergence does in the untimed ones. All behaviours we record in the timed models will come from observations we can make up to some finite time. This means that they are qualitively very different from the two types of observation recorded in the failures/divergences model for untimed CSP. There, the failure (s, X) meant that, after the trace s, the process would refuse X even if

it were offered for ever after; s being a divergence means that we can watch the process performing internal actions, once again *for ever*. A process becomes *stable* when it loses the capacity to make any further progress without making some externalcommunication. Importantly we can record the time at which stability occurs.

What does a stable process look like? For consistency with the untimed models and in order to be able to make useful deductions about the behaviour of a stable process, we take a rather severe view. We assume that once it becomes stable a process' available actions remain constant until one of them occurs, and furthermore that its subsequent behaviour does not depend on the time when the event occurred. In other words, given the initially stable process $a \to P$, the ways P can behave if we accept a at time 100 will be exactly the same as those which could have arisen if a happened at time 0, only with 100 added on the time when everything occurs. (The effect will be like 'shifting' the behaviours of P by 100 time units.)

This view of a stable process is closely related to the principle stated earlier that no process has access to the global clock of the observer: if P did in the example above then it could 'know' it was being used at different absolute times and so behave accordingly. It also means that the activity of any internal clock which a process may start counts as internal actions.

Notice that if we have observed of a process (a) that it has become stable and (b) is refusing some set of events, then we know it will refuse this set for ever.

If, informally, we think of a process as having a red light on the side which stays on as long as the process is making any internal progress, then it becomes stable at the time when the light goes out.

The untimed theory takes a very uncharitable view of processes which could perform an infinite sequence of internal actions. All processes with this potential immediately were identified with the most nondeterministic one, and considered useless. There are two distinct places in which a timed theory can be less severe. Consider the process

$$(\mu p.((\mathit{WAIT}\, x; a \to p)\Box(\mathit{WAIT}\, y; b \to \mathit{STOP}))) \setminus a\ .$$

If $x < y$ then (assuming a symmetric implementation of $\Box$) we would expect this process to diverge in the timed theory and offer no communications to the environment. If, on the other hand, $y < x$ then we would expect there to be intervals where the event b is enabled and the hidden a is not, meaning that, if we offer this version of the process a b, then – using timing information – we can guarantee that it will be accepted eventually. Since an untimed theory cannot make this type of distinction it has to identify any potential for an infinite sequence of internal actions with divergence. But as the above example, as well as various others, show, it is possible and desirable to distinguish the two when we have time to play with.

The second place where untimed theories are severe on divergence is in the way they treat processes like $\perp \sqcap (a \to P)$ which can diverge but can also perform some action – either instead of or interrupting the divergence. For various technical reasons which we will not repeat here the untimed theories often do not allow us to reason about these actions or what might happen after them, because they identify any divergent process with the most nondeterministic one. As soon as a process has the potential to diverge, these theories treat them as irrecoverably undefined. It will turn out that it is not necessary to make this type of identification in the timed theory.

In summary, stability will give us the ability to relate the timed theory with the untimed one because the untimed theory is really a theory of non-divergent processes, in the sense that it treats any process which can perform an infinite sequence of internal actions as useless. The refusals which the failures/divergences model records are those *after stability.* The timed theory should also let us reason about processes which can perform these sequences of internal actions, because time gives us the ability to analyse their behaviour with sufficient precision.

2.3 The metric space approach

In the models for untimed CSP, it is usual to use complete partial orders with continuous or monotone functions as the basis for defining the meanings of recursions. Various workers have defined complete metrics over these and similar models of concurrency, usually based on the number of steps over which a pair of processes behave indistinguishably. Of course when a recursion represents a contraction mapping with respect to such a metric, it has a unique fixed point which must be the same as the one predicted by a partial order theory. The problem which attaches to this approach is that not all recursions give rise to contraction maps. Consider the recursion

$$P = a \to (P \setminus a)$$

which fails to be a contraction in the number-of-steps metric because the hiding operator can actually push points further apart by concealing a's which guard their differences. Indeed, over untimed models this recursion has as a fixed point any process of the form $a \to Q$, for Q a process which cannot communicate an a until it has the possibility of divergence.

Over the timed models the cpo approach leads immediately to problems[2]. The most obvious of these comes from the need for a least or bottom element. Experience would suggest that this should be the most nondeterministic process, but it turns out that there is no such element in the models we use since it would violate the assumption about only finitely many events occurring in a finite time. We will see later that a requirement for increasing sequences to have least upper bounds would also cause problems.

Fortunately the problems which appeared in the untimed models with the metric approach now disappear. The reason for this is that we now have a different and more natural criterion for judging the distance between two processes: the length of time for which they behave indistinguishably.

Consider an implementable operator F acting on a process P. (It is convenient to restrict attention to the case of a unary operator, but of course the situation is no different in the more general case.) Provided we assume that the observable behaviour of $F(P)$ depends only on what F can observe of P (rather than seeing into the structure of P in some way that the environment cannot – an ability which would probably make the definition of a denotational semantics based on the observations impossible) and that it cannot somehow speed up P to observe it faster than we could, then the possible behaviours of $F(P)$ up to a given time must only depend on the behaviours of P up to the same time. This means that, under the metric based on the time-of-indistinguishability, every operator will be nonexpanding. Consider the case of the hiding operator which caused us problems above. Although some of the communications of P are hidden in $P \setminus a$, the length of time which it takes P to complete them is not: every event which $P \setminus a$ performs is attributable to one that P could have performed at least as soon. We might christen this healthiness condition of operators as 'absence of clairvoyance'.

As a curiosity, suppose we *could* build an operator $F(P)$ which speeded up its argument by a factor of $\eta > 1$. Then, assuming that recursive unfolding and the completion of the communication a both take time δ exactly,

[2]These problems are not always insuperable. In one recent case – the infinite timed failures model [] – the metric space approach was no longer usable but it prove possible, with considerable effort, to get a (non-complete) partial-order based theory to work.

the process

$$\mu p.a \to F(p)$$

could perform infinitely many actions in time $\frac{2\delta\eta}{\eta-1}$.

Given that all operators are nonexpanding, and that we have assumed earlier that every recursive unfolding takes some time bounded below by a positive constant δ, it turns out that all recursions represent contraction mappings and so have unique fixed points. For example, the behaviour of a system up to time $n\delta$ will be determined by an n-fold unwinding of the recursion. (Of course, it may take substantially less unwinding than this for a given recursion; what we have here is a global upper bound.)

2.4 Exploring compositionality: in search of the right congruence

We have previously mentioned that the model we are developing is rather complex and the semantics of processes in it can be somewhat intricate. Before we construct the model it is helpful to review the reasons for this complexity. Readers who are familiar with the theory of untimed CSP will know that it is possible to give natural and compositional semantics to it in reasonably simple models, for example the traces model and the failures/divergences model.

At this point it is worth discussing just what 'compositional' means in this context. A *compositional* semantics is one where it is possible to determine the natural semantic value of any combination, using standard operators, of processes from the values of the individual processes. Here, 'natural' might either relate to some operational or other intuition about what the semantic values 'ought' to be, or more formally could be defined relative to some operational semantics and an abstraction map which tells us exactly what the semantic values ought to be. A simple example of a non-compositional semantics for untimed CSP which lies between the two compositional ones mentioned above would be to model each process by a set of traces and an indication of which traces it could deadlock on. The two processes

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

would have the same value in this semantic model, but if we were to combine each process in parallel ($\|$) with itself, the first would be able to deadlock on the first step while the other would not. Thus it simply is not possible to give an accurate denotational semantics to untimed CSP in this model. From this simple example (as well as our discussion later) the reader will see that finding compositional congruences is not always easy.

Given the similarities between untimed CSP and Timed CSP, one would expect that they could be modelled by similar congruences. In fact this turns out not to be the case, as we shall shortly see. The essential reason for this is the same as the theme underlying the discussion of the differences in the treatment of internal actions and divergence above. This is that in a real-time theory we have to reason about what a process will do from moment to moment in response to various stimuli, and thus have the ability to resolve a lot of the nondeterminism that cannot be avoided when we deliberately abstract away from time in the untimed models. When modelling real time we are always concerned about what a process can do at a particular moment, while the untimed models have to be concerned about possibilities over all future times.

Before proceeding with this discussion it is helpful to introduce some of the notation we will be using to describe timed behaviour: the notation of timed traces.

Notation

A *timed event* is an ordered pair (t, a), where a is a communication and $t \in \mathbf{R}^+$ is the time at which it occurs. The set $\mathbf{R}^+ \times \Sigma$ of all timed events is denoted $T\Sigma$. The set of all *timed traces* is

$$(T\Sigma)^*_{\leq} \ = \ \{s \in T\Sigma^* \mid \text{if } (t,a) \text{ precedes } (t',a') \text{ in } s, \text{ then } t \leq t'\}.$$

If $s \in (T\Sigma)^*_{\leq}$, we define $\#s$ to be the length (i.e., number of events) of s and $\Sigma(s)$ to be the set of communications appearing in s (i.e., the second components of all its timed communications). $begin(s)$ and $end(s)$ are respectively the earliest and latest times of any of the timed events in s. (For completeness we define $begin(\langle\rangle) = \infty$ and $end(\langle\rangle) = 0$.)

If $X \subseteq \Sigma$, $s \upharpoonright X$ is the maximal subsequence w of s such that $\Sigma(w) \subseteq X$; $s \setminus X = s \upharpoonright (\Sigma - X)$. If $t \in [0, \infty)$, $s \upharpoonright t$ is the subsequence of s consisting of all those events which occur no later than t, while $s \,|\!\!\upharpoonright t$ is the subsequence containing the events which occur *before* t. If $t \in [-begin(s), \infty)$ and $s =$

$\langle(t_0, a_0), (t_1, a_1), \ldots, (t_n, a_n)\rangle$,

$$s + t \;=\; \langle(t_0 + t, a_0), (t_1 + t, a_1), \ldots, (t_n + t, a_n)\rangle\ .$$

If $s, t \in (T\Sigma)^*_{\leq}$, we define $s \cong t$ if, and only if, t is a permutation of s (i.e., events that happen at the same time can be re-ordered). We will regard timed traces which are thus congruent as equivalent, simply different ways of writing down the same observation.[3]

If $s, w \in (T\Sigma)^*_{\leq}$, $\mathit{Tmerge}(s, w)$ is defined to be the set of all traces in $(T\Sigma)^*_{\leq}$ obtained by interleaving s and w. Note that this is a far more restricted set than in the untimed case, as the times of events must increase through the trace. In fact, $\mathit{Tmerge}(s, w)$ only contains more than one element when s and w record a pair of events at exactly the same time, and even these two traces will be equivalent ($\cong$).

Given a Timed CSP process P, $\mathit{Traces}(P)$ will denote the set of all timed traces which are possible for P.

Suppose P_1 and P_2 are both CSP processes that both perform some number of internal actions before terminating successfully. Perhaps one is SKIP and the other is $(a \rightarrow a \rightarrow \mathit{SKIP}) \setminus a$. Now consider the process $(P_1; a \rightarrow \mathit{STOP})\Box(P_2; b \rightarrow \mathit{STOP})$.

In the untimed theory we do not know how long P_1 and P_2 take to run and do not wish to specify this time. Also we do not know whether the implementation of the $\Box$ operator runs both its arguments at once, gives the left one priority, or does something else. Thus, in the the untimed theory, there is absolutely no way we can tell which of a and b becomes available first. Thus, in calculating the value of

$$((P_1; a \rightarrow \mathit{STOP})\Box(P_2; b \rightarrow \mathit{STOP})) \setminus a$$

the untimed theory cannot exclude the possibility that the b will occur (presumably accepted by the environment at any moment up to the one where the hidden a becomes available). However, if we now set $P_1 = \mathit{SKIP}$ and

[3]Notice that, given this assumption, the order of events in a timed trace carries no information that is not also contained in the times of the events. At first sight it might seem more natural to record process histories as sets of timed events; but the problem with this is that it is possible in CSP to have a parallel process which performs two copies of the same event at the same time. One could use multisets instead, the effect of which would be the same as our timed traces under the above congruence relation. The choice is very much a matter of taste.

$P_2 = \mathit{WAIT}\,1$ and put a timed semantics on $\Box$ in which its arguments are allowed to proceed together until a communication takes place, it becomes certain that the a will be available – and hence occur because of our assumptions about hiding – before b is possible. Thus, in this case, the timed theory would tell us that b cannot occur. Real-time analysis lets us make precise assumptions about how long various aspects of CSP will take to execute and to draw the appropriate conclusions. (Notice the similarity between this example and the one used earlier in the discussion of divergence.)

The above example is actually very telling. We knew that the event b was not possible because we knew that the process before hiding could only perform a b after being unable to refuse an a. This suggests that in order to know what traces are possible in $P \setminus a$ we need to know something about the pattern of refusals in P. To confirm this suspicion all we have to do is consider the same process only with nondeterministic choice replacing external choice.

$$((a \to \mathit{STOP}) \sqcap (\mathit{WAIT}\,1; b \to \mathit{STOP})) \setminus a$$

We can reasonably expect the process before hiding to have exactly the same timed traces of observable actions as the corresponding part of the original version. But this one is not obliged to offer an a before offering a b, and so it can perform a b even after hiding a. This example means that timed traces without refusal information cannot give us a compositional congruence.

In the failures/divergences model we only have to give information about what a process can refuse at the end of a trace. This turns out not to be sufficient in the real-time case. The reasons for this begin to be apparent from the arguments in the last paragraph, where we knew b was not possible because of what was refusable before it happened. In fact this example is not quite good enough, since we can tell, by looking at what refusals are possible on the empty trace and the times when b is possible, that the b cannot occur after hiding. Consider, though, what would happen if we composed the $\Box$ version of the above process (before hiding) nondeterministically with STOP:

$$((a \to \mathit{STOP}) \Box (\mathit{WAIT}\,1; b \to \mathit{STOP})) \sqcap \mathit{STOP}$$

On the empty trace this can refuse anything, and can perform exactly the same events, at exactly the same times as

$$((a \to \mathit{STOP}) \sqcap (\mathit{WAIT}\,1; b \to \mathit{STOP})) \sqcap \mathit{STOP}$$

and behave the same way after each such event. Thus in a congruence based on timed traces and refusals after last communication, we could not tell these two processes apart. Nevertheless, if we hid a in them, the first could not perform a b for exactly the same reason as above, while the second one clearly could. The congruence could not therefore be compositional.

As a further example, consider

$$\begin{aligned} P_1 &= ((a \to STOP) \Box (b \to STOP)) \sqcap (a \to c \to STOP) \\ P_2 &= ((a \to c \to STOP) \Box (b \to STOP)) \sqcap (a \to STOP) \end{aligned}$$

On the basis of their timed traces and refusals after traces, P_1 and P_2 are indistinguishable – and note that neither uses $WAIT\, t$ or hiding in its definition. However, let

$$Q = (WAIT\ 1 \Box (b \to STOP))\ ;\ a \to c \to STOP$$

Operationally we would expect:

$$(P_1 \parallel Q) \setminus b \neq (P_2 \parallel Q) \setminus b$$

In particular, we would expect (on the assumption that the event a takes, or might take, time ξ to complete):

$$\begin{array}{lll} \langle (1,a)(1+\xi,c) \rangle & \in & Traces((P_1 \| Q) \setminus b) \qquad \text{but} \\ \langle (1,a)(1+\xi,c) \rangle & \notin & Traces((P_2 \| Q) \setminus b) \end{array}$$

The essential reason for the sorts of behaviour seen in these examples can be traced to our earlier discussion of the hiding operator. We said there that all hidden events take place 'as soon as such an event becomes possible'. This means that any behaviour of P in which a hidden event has been possible for a non-zero time does not give rise to any behaviour in $P \setminus X$. In particular, any non-hidden event which is only possible in such circumstances is always, in $P \setminus X$, *pre-empted* by hidden events.

We infer that we need to know what a process can refuse during its trace, not only after it or upon achieving stability. We will have to know what the process could refuse at each time during the trace, these refusals potentially changing due to internal state changes as well as visible actions. This is a *crucial* issue in achieving a successful semantics for real-time parallel languages.

Algebraic properties give a useful test of a mathematical model and the definition of a semantics over it, which is related to the discussion above and yet, in a sense, more concrete. For our discussion above has been based on an intuitive feel for how processes ought to behave operationally. We cannot turn this inuition into rigorous mathematical arguments without defining an operational semantics formally. Though this has now been done [], in a way fully congruent with the semantics of this paper, the necessary arguments are both very complex and are tied to one specific operational viewpoint.

An intermediate standpoint is to use one's intuition to write down a number of algebraic identities which are 'clearly true' in any reasonable implementation and then to use these as healthiness criteria for one's model and semantics. Examples we might use are the *distributivity* of any operator that only uses (at most) one copy of its arguments over nondeterministic choice, for example

$$\begin{aligned} P\Box(Q \sqcap R) &= (P\Box Q) \sqcap (P\Box R) \\ P;(Q \sqcap R) &= (P;Q) \sqcap (P;R) \\ (Q \sqcap R) \setminus X &= Q \setminus X \sqcap R \setminus X \end{aligned}$$

The argument for these is that $P \sqcap Q$ is intended to represent a process which can behave like P or like Q, and that therefore the behaviours of $(P \sqcap Q) \setminus a$ (for example) are precisely those possible for $P \setminus a$ and those possible for $Q \setminus a$. Another example would be the 'commutativity' of hiding:

$$P \setminus X \setminus Y = P \setminus Y \setminus X$$

which one would expect to hold under most realistic implementations of hiding. It would probably be impossible to come up with a 'complete' set of such laws for testing a semantics which was uncontroversial. Indeed to have a complete set of such laws, in the most obvious sense, would imply that we had fallen into the same trap of being over-specific that we mentioned above in connection with a specific operational semantics. Nevertheless, such laws as these provide a valuable and more tangible supplement to the intuition used earlier. The failure of such a law or the impossibility of producing a reasonable semantics in which they hold will tell us that something is wrong.

2.5 The Timed Failures-Stability Model (TM_{FS})

We are now ready to build a mathematical model based on the intuition developed in the last few sections. It will model each process by its set of observable behaviours – timed traces with refusals throughout, up to some finite time – and will match each with the associated stability time. We will put a metric on it based on the time for which it is impossible to tell two processes apart.

The main thing which it remains for us to decide is the way in which stability values are tied in with the timed traces and timed refusals which we have already discovered we need.

Consider the processes:

$$\begin{array}{rcl} P & = & WAIT\ 1; a \rightarrow STOP \\ \\ Q & = & (b \rightarrow STOP \\ & & \Box \\ & & WAIT\ 1; a \rightarrow (WAIT\ 1; STOP)) \end{array}$$

If we once again assume that $\Box$ runs its arguments in parallel until a visible action, we see that both can perform the timed trace $\langle(1,a)\rangle$ but Q (i) cannot perform the a without having made b available first and (ii) becomes stable later. If we were to associate stability values with timed traces alone then we could not tell that the late stability of $P \sqcap Q$ on trace $\langle(1,a)\rangle$ only happens when b is offered first, which would mean that we would be forced to predict the same late stability on $\langle(1,a)\rangle$ for $(P \sqcap Q) \setminus b$. But $Q \setminus b$ cannot perform the event a because it is pre-empted, and it would follow that

$$(P \sqcap Q) \setminus b \neq P \setminus b \sqcap Q \setminus b$$

in contradiction to the principle stated earlier. We can deduce from this that we must associate stability values with trace/timed refusal pairs – namely the whole observation we are making of a process.

We still have a number of choices: do we record for each trace/refusal pair the set of all possible observed stabilities for it? And need we record not only the stability observed at the end of a trace but also those that may have been observed at intermediate points along the way?

It would be inappropriate only to record the behaviours of a process which happened to lead to stability in a finite time. Therefore if we were recording all times at which a process could become stable we would also have to include a special value, say ∞, representing the fact that the process happens not to become stable. This would lead to problems related to our metric space approach, since one could not tell at any finite time between the process that could become stable at any natural number time and the one which could also remain unstable.

Although we are assuming our ability to see stability, it is not something which an implemented operator will usually need to observe of a process in order to determine the timed traces or refusals of its result. Provided this property holds of all operators – and we will assume it does – we fortunately do not need to know the set of times when a process might become stable. Indeed, we will assume that the implementation itself has no way of observing stability: it is simply a tool that we use to reason about processes externally. Another consequence of this assumption is that the stability of any construct $F(P)$ at a given time depends only on F and whether P happens to be stable at the time which F has observed it up to (which, given our earlier assumptions, is no later than the current time). So in particular the current presence or absence of stability does not depend upon whether it happens to have been observed earlier.

The previous two paragraphs together suggest that it is desirable and sufficient to associate each trace/refusal pair with a single stability time: the least upper bound of all the times (including ∞) when it might become stable, given that the trace and refusal have been observed. This is what we will do.[4]

Notation

The following are some more components from which our model will be built. Stability values are as described above. The time intervals we use are finite nonempty, closed at the left and open at the right. Not only do intervals of this form the most natural ones for partitioning the interval $[0, \infty) = \mathbf{R}^+$, but this shape of interval also turns out too be the correct choice for modelling process behaviour: it reflects the idea that an event which is offered might be accepted immediately, and allows us to reason correctly about events which happen at the same time.

Notice that although refusal sets may contain infinitely many different members of Σ, they can only change finitely often in a finite time. This is essentially an assumption that processes and the environment only undergo finitely many state-changes in a finite time.

[4] An equivalent approach is to associate each pair with the set of all times after the end of the trace when *instability* might be observed. This has been suggested by Blamey [].

$$
\begin{array}{rcll}
\alpha\colon TSTAB & = & \mathbf{R}^+ \cup \{\infty\} & \text{(Stability values)} \\
I\colon TINT & = & \{\ [l(I), r(I)) \mid 0 \le l(I) < r(I) < \infty\} & \text{(Time Intervals)} \\
T\colon RTOK & = & \{I \times X \mid I \in TINT \wedge X \in P(\Sigma)\} & \text{(Refusal Tokens)} \\
\aleph\colon RSET & = & \{\bigcup Z \mid Z \subseteq RTOK \wedge Z \text{ finite}\} & \text{(Refusal Sets)}
\end{array}
$$

We define various functions over $RSET$, to extract the set of communications used, times used, beginning and end, shifting and restriction.

$$
\begin{array}{rcl}
\Sigma(\aleph) & = & \{a \in \Sigma \mid \exists t.\, (t, a) \in \aleph\} \\
I(\aleph) & = & \{t \in [0, \infty) \mid \exists a.\, (t, a) \in \aleph\} \\
begin(\aleph) & = & inf(I(\aleph)),\ \forall \aleph \neq \emptyset \\
end(\aleph) & = & sup(I(\aleph)),\ \forall \aleph \neq \emptyset \\
begin(\aleph) & = & \infty,\ \text{for } \aleph = \emptyset \\
end(\aleph) & = & 0,\ \text{for } \aleph = \emptyset \\
\forall t \ge -begin(\aleph),\ \aleph + t & = & \{(t' + t, a) \mid (t', a) \in \aleph\} \\
\forall t \in [0, \infty),\ \aleph \restriction t & = & \aleph \cap ([0, t) \times \Sigma) \\
\forall I \subseteq \mathbf{R}^+\ \aleph \uparrow I & = & \aleph \cap (I \times \Sigma) \\
\forall a \in \Sigma,\ \aleph \downarrow a & = & \{t \mid (t, a) \in \aleph\}.
\end{array}
$$

Each process will be modelled as a set of triples $(s, \alpha, \aleph)$, with $s \in (T\Sigma)^*_{\le}$, $\alpha \in TSTAB$ and $\aleph \in RSET$. The following functions are natural projections of such sets and operations to ensure (i) that there is one stability value for each trace/refusal pair and (ii) that all equivalent traces are treated the same.

$$
\begin{array}{rcl}
Traces(S) & = & \{s \mid \exists \alpha, \aleph.\, (s, \alpha, \aleph) \in S\} \\
Stab(S) & = & \{(s, \alpha) \mid \exists \aleph.\, (s, \alpha, \aleph) \in S\} \\
Fail(S) & = & \{(s, \aleph) \mid \exists \alpha.\, (s, \alpha, \aleph) \in S\} \\
\underline{SUP}(S) & = & \{(s, \alpha, \aleph) \mid (s, \aleph) \in Fail(S) \\
 & & \qquad \wedge\ \alpha = sup\{\beta \mid (s, \beta, \aleph) \in S\}\} \\
CL_{\cong}(S) & = & \{(s, \alpha, \aleph) \mid \exists w.\, (w, \alpha, \aleph) \in S \wedge s \cong w\}
\end{array}
$$

The evaluation domain TM_{FS}

We formally define TM_{FS} to be those subsets S of $(T\Sigma)^*_{\le} \times TSTAB \times RSET$ satisfying:

1. $\langle\rangle \in Traces(S)$

2. $(s.w, \aleph) \in \mathit{Fail}(S) \;\Rightarrow\; (s, \aleph \upharpoonright \mathit{begin}(w)) \in \mathit{Fail}(S)$

3. $(s, \alpha, \aleph) \in S \;\wedge\; s \cong w \;\Rightarrow\; (w, \alpha, \aleph) \in S$

4. $t \in [0, \infty) \Rightarrow\; \exists n(t) \in \mathbf{N}. \forall s \in \mathit{Traces}(S). \mathit{end}(s) \leq t \Rightarrow \#s \leq n(t)$

5. $(s, \alpha, \aleph), (s, \beta, \aleph) \in S \Rightarrow\; \alpha = \beta$

6. $(s, \alpha, \aleph) \in S \;\Rightarrow\; \mathit{end}(s) \leq \alpha$

7. $\begin{pmatrix} (s, \alpha, \aleph) \in S \;\wedge\; (s.\langle (t, a) \rangle, \aleph) \in \mathit{Fail}(S) \;\wedge \\ t > t' \geq \alpha \;\wedge\; t \geq \mathit{end}(\aleph) \end{pmatrix} \Rightarrow (t', a) \notin \aleph$

8. $(s, \alpha, \aleph) \in S \;\Rightarrow\;$ if $t > \alpha$, $t' \geq \alpha$, $a \in \Sigma$ and
   $w \in (T\Sigma)^*_{\leq}$ is such that $w = \langle (t, a) \rangle . w'$, then
   $(s.w, \alpha', \aleph') \in S \;\wedge\; \aleph \subseteq \aleph' \upharpoonright t \;\Rightarrow$
   $\exists \gamma \geq \alpha' + (t' - t)$ such that
   $(s.(w + (t' - t)), \gamma, \aleph_1 \cup \aleph_2 \cup (\aleph_3 + (t' - t))) \in S$,
   where $\aleph_1 = \aleph' \upharpoonright \alpha$, $\aleph_2 = [\alpha, t') \times \Sigma(\aleph' \cap ([\alpha, t) \times \Sigma))$,
   and $\aleph_3 = \aleph' \uparrow [t, \infty)$.

9. $(s, \alpha, \aleph) \in S \wedge \aleph' \in \mathit{RSET}$ such that $\aleph' \subseteq \aleph$
   $\Rightarrow\; \exists \alpha' \geq \alpha$ such that $(s, \alpha', \aleph') \in S$

10. $(s, \alpha, \aleph) \in S \;\wedge\; t_1 < \alpha \;\wedge\; t_2 \geq 0 \;\Rightarrow$
    $\exists \aleph', \beta. \aleph \subseteq \aleph' \;\wedge\; (s, \beta, \aleph') \in S \;\wedge\; \beta \geq t_1 \;\wedge$
    $(t' \leq t_2 \;\wedge\; (t', a) \notin \aleph) \Rightarrow (s \upharpoonright t'.\langle (t', a) \rangle, \aleph' \upharpoonright t') \in \mathit{Fail}(S) \;\wedge$
    $((0 < t' \leq t_2 \wedge \neg \exists \epsilon > 0. ((t' - \epsilon, t') \times \{a\}) \subseteq \aleph') \Rightarrow$
    $(s \Vert\!\!\upharpoonright t'.\langle t', a \rangle), \aleph' \upharpoonright t') \in \mathit{Fail}(S))$

11. $(s, \alpha, \aleph) \in S \wedge I \in \mathit{TINT}$ such that $I \subset [\alpha, \infty)$
    $\Rightarrow (s, \alpha, \aleph \cup (I \times \Sigma(\aleph \cap ([\alpha, \infty) \times \Sigma)))) \in S)$

Although some of these axioms appear complex, each reflects one or more simple healthiness properties. We will now give an intuitive explanation of each.

1. Every process has initially done nothing at all.

2. If a process has been observed to communicate $s.w$ while refusing $\aleph$ then, at the time when the first event of w occurred, the pair $(s, \aleph \upharpoonright \mathit{begin}(w))$ had been observed.

3. Traces which are equivalent (i.e., are the same except for the permutation of events happening at the same times) are interchangeable. Essentially, this postulates that there can be no causal dependance between simultaneous events: notice that if a process has trace $\langle (t, a), (t, b) \rangle$ then this axiom and axiom 2 show it has trace $\langle (t, b) \rangle$.

4. The process cannot perform an infinite number of visible events in a finite time.

5. There is only one stability value for each trace/refusal pair: the least time by which we can guarantee stability after the given observation.

6. The time of stability is not before the end of the trace.

7. A stable process cannot communicate an event which it has been seen to refuse since stability.

8. After stability the same set of events is available at all times. Furthermore the behaviour of a process after such an event does not depend on the exact time at which it was executed. Thus the trace w and the corresponding part of the refusal may be translated so as to make the first event of w now occur at time t'.

   The stability value γ corresponding to the translated behaviour may, in general, be greater than the obvious value because the translated behaviour may in some circumstances be possible for other reasons. Note, however, that if stability is still inferable in the new behaviour before time t', then the axiom may be used in reverse to translate the tail of the behaviour so that the beginning of w occurs back at t. This, in combination with axiom 9, can often be used to prove that the γ appearing on the right hand side of axiom 8 does equal $\alpha + t' - t$.

   There is a phenomenon related to this last discussion which it is worth pointing out. One can think of the way we record stability values as giving a record of by when, given the timed trace and refusal observed so far, can we guarantee that the process attains stability. Subconsciously one might think that things observed after stability give no information in this regard, but this would be wrong. It is in fact possible, by making some observation, to realise that the process must already have been stable for some time. A good example of this is provided by the process

   $$(\mathit{WAIT}\ 1; (a \to \mathit{STOP}) \sqcap \mathit{STOP}$$

Depending on which nondeterministic choice is made, this process either will or will not stabilise immediately. But we can only tell from refusals that it was stable at time 0 when a is refused at time 1 or later.

9. If a process has been observed to communicate s while refusing $\aleph$ then it can communicate the same trace while refusing any subset of $\aleph$. This simply reflects the fact that the environment might offer it less and so have less refused. However, because less has been observed, the stability value can, in general, be greater.

10. Given a triple $(s, \alpha, \aleph)$ and times $t_1 < \alpha$ and $t_2 \geq 0$, there exists a single refusal $\aleph'$ in $RSET$ containing $\aleph$ and stability value $\beta > t_1$ such that $(s, \beta, \aleph') \in \mathit{Fail}(P)$ and it is consistent to believe that the (finitely many) changes in the refusals of $\aleph'$ give complete information about what the process could have refused – because it can accept anything not in the refusal set. The events in s and the changes in $\aleph$ can be thought of as the process' state changes. Notice that axiom 9 ensures that $\beta \leq \alpha$.

    The construction involving t_1 and β ensures that the stability value α of $(s, \alpha, \aleph)$ is the supremum of stability values corresponding to such 'complete' behaviours – or in other words the time of stability is not increased simply though the environment failing to observe what would have been refused anyway.

    The last clause of the axiom states that, if an event was not refusable up to a given time t', then it was still possible at time t'. This means that we are assuming that any event which was on offer up to a change of state is also available at the instant of the state-change. Note that in the previous clause we state that $(s \upharpoonright t', \aleph' \upharpoonright t') \in \mathit{Fail}(S)$ whereas in this last one we vary this to $(s \| \!\upharpoonright t', \aleph' \upharpoonright t') \in \mathit{Fail}(S)$. Of course these two say the same in the case where s has no event happening at time t. But if there are one or more, the refusals *at* time t' refer to what the process can do *after* the event(s) at the given time, while the refusals just *before* t' allow us to reason about what it might have done *instead* of them.

    This last assumption could be dropped if we wanted to consider operators which could cause a 'clean' withdrawal of an offer to communicate. It is included in our presentation because none of the CSP operators can cause such a withdrawal and because we consider it to be a property which is operationally reasonable. We will also discuss in Section 2.6 below another small modification to this axiom.

The concept of a complete behaviour, introduced here, will be very important later.

11. Something that is refused at one time on or after stability is refused at all such times. This axiom says the same about the end of traces that part of axiom 8 says about other points in them. Notice that these extra refusals tell us nothing more about stability time.

Note 1. In both axioms 7 and 8, we carefully distinguish (via t' and t) between events *at* stability and events *after* stability. This is a necessary distinction. For example, the process $P = (a \to STOP \,\Box\, WAIT\ 1); b \to STOP$ will, in the standard semantics, become stable on the pair $(\langle\rangle, [0,1) \times (\Sigma - \{a\}))$ at time 1; however $\langle(1,a)\rangle \in \mathit{Traces}(P)$ but $\forall t > 1, \langle(t,a)\rangle \notin \mathit{Traces}(P)$. Events which are possible at the very moment of stability might, as in this example, result from alternatives to the stable behaviour rather than from the behaviour itself. This possibility of nondeterminism at the point of stability will cause us various difficulties later.

Note 2. The axioms above are (when taken together) strictly stronger than those we have presented in earlier papers, in the sense that they restrict further the class of processes. The difference between this set and the axioms of [...] is that axiom 10 above has replaced both axioms 4 (which it obviously strengthens) and 11 of the earlier paper. A discussion of this point and of our reasons for strengthening of the axioms will be found in section 2.6 below. The numbering of the axioms has also changed from earlier papers.

The complete metric on TM_{FS}

As described earlier, the metric on our model will be based on the length of time for which it is impossible to tell a pair of processes apart. To define it we need a function which gives a standard representation of a process' behaviour up to time t. If $S \subseteq (T\Sigma)^*_{\leq} \times TSTAB \times RSET$ and $t \in [0,\infty)$, we define

$$\begin{aligned} S(t) \;=\; & \{(s,\alpha,\aleph) \in S \mid \alpha < t \;\wedge\; end(\aleph) < t\} \\ & \cup \{(s,\infty,\aleph) \mid end(s) < t \;\wedge\; end(\aleph) < t \;\wedge\; \exists \alpha \geq t.\, (s,\alpha,\aleph) \in S\}. \end{aligned}$$

$S(t)$ has a representative of each timed failure $(s, \aleph)$ which ends before t. Where the stability value is also less than t, it is included, and otherwise it is replaced by the standard value ∞. It is worth noting that any pair S_1 and S_2 of distinct sets of triples satisfyingaxiom 5 have a time t such that $S_1(t) \neq S_2(t)$: if $Fail(S_1)$ and $Fail(S_2)$ were unequal then we need only pick t after the end of some element of the symmetric difference. If $(s, \alpha, \aleph) \in S_1$ and $(s, \beta, \aleph) \in S_2$ where $\alpha \neq \beta$, then t can be any time greater than both $end(\aleph)$ and the lesser of α and β (which must be finite).

The complete metric on TM_{FS} is now defined:

$$d(S_1, S_2) \ = \ inf\{2^{-t} \mid S_1(t) = S_2(t)\}$$

Given the observation we made above about being able to distinguish S_1 and S_2, it is easy to show that this function defines an ultrametric, namely a metric satisfying the strong triangle inequality

$$d(P, R) \leq max\{d(P, Q), d(Q, R)\}.$$

The completeness of this metric can be demonstrated as follows. First, the set of all $S \subseteq (T\Sigma)^*_\leq \times TSTAB \times RSET$ satisfying axiom 5 alone is a complete metric space under this metric: if S_n is a Cauchy sequence we know that, for each t, there is $n = n(t)$ with $m \geq n$ implying $S_n(t) = S_m(t)$, it is easy to see that the limit of the sequence S_n is the set of all triples $(s, \alpha, \aleph)$ with $\alpha < \infty$ contained in any such $S_{n(t)}(t)$ plus all those of the form $(s, \infty, \aleph)$ contained in all $S_{n(t)}(t)$ for sufficiently large t. Second, the set of all S is this space satisfying axiom 9 is closed within this space, since any failure of this axiom becomes apparent in a finite time (i.e., if S fails it, then there is a time t such that $S'(t) = S(t)$ implies S' fails it), which means that the set of all S not satisfying it is closed. Finally, within the set of all sets satisfying 5 and 9, the set of all S satisfying any one of the other axioms is closed. Since the intersection of closed sets is closed, the model TM_{FS} is thus a closed (and hence complete) subset of a complete metric space.

2.6 More properties of the model

The most subtle – and most powerful – of our axioms is axiom 10. This says that each observed behaviour of a process can be interpreted in terms of some 'complete' description of how it might behave. If we define a t_2-complete behaviour[5] to be one satisfying the conditions on the right hand side of the implication, then this axiom together with axiom 9 says that the stability value associated with any timed failure $(s, \aleph)$ is the supremum of all those associated with its $t-$complete extensions (i.e., $t-$complete behaviours with the same timed trace, and larger timed refusal). Recall axiom 11 of []:

$$\begin{array}{l}(s.w, \alpha, \aleph) \in S \wedge \aleph' \in RSET \text{ is such that } end(s) \leq begin(\aleph') \ \wedge \\ end(\aleph') \leq begin(w) \ \wedge \ (\forall (t, a) \in \aleph', (s.\langle (t, a)\rangle, \aleph \restriction t) \notin Fail(S)) \\ \qquad\qquad \Rightarrow \ (s.w, \alpha, \aleph \cup \aleph') \in S\end{array}$$

This says that that, if the timed failure $(s.w, \aleph)$ is observable, and if $\aleph'$ contains events between the end of s and the beginning of w which were impossible, then the process would also have refused $\aleph'$ if the environment had offered it. Since this must be true in every run of the process which exhibits $(s.w, \aleph)$, no further information about stability is gained from observing the refusal of $\aleph'$, so the observed stability time is the same. (Note axiom 9.)

It is a consequence of our new axioms. Assuming the conditions on the left-hand-side then, if $t > end(s.w, \aleph \cup \aleph')$, obviously any $t-$complete extension $(s.w, \aleph^*)$ of $(s.w, \aleph)$ must have $\aleph^* \supseteq \aleph \cup \aleph'$, which means (by axiom 9) that $(s.w, \aleph \cup \aleph') \in Fail(S)$. It is then easy to see that any complete extension of $(s.w, \aleph)$ is one of $(s.w, \aleph \cup \aleph')$, and vice-versa. The sup property of stability values discussed above then ensures that the stability values associated with $(s.w, \aleph)$ and $(s.w, \aleph)$ are the same.

For reasons discussed earlier we have not based our fixed point theory on a partial order. Nevertheless there are other reasons for wanting to have an order over TM_{FS} based (as with many of the orders over untimed CSP) on the notion of nondeterminism: $P \sqsubseteq Q$ should mean that Q is more predictable than P – any observation of Q could be taken for one of P. Such an order will turn out to be useful for understanding the structure of our model, understanding the way it treats nondeterminism, and for developing a notion of refinement. Recalling that the triple $(s, \alpha, \aleph)$ means that the timed failure $(s, \aleph)$ can be observed and that α is the supremum of the resulting stability values (i.e., any stability value less-than-or equal to α might occur), the order is best defined as follows. $P \sqsubseteq Q$ if and only if

$$\forall (s, \alpha, \aleph) \in Q. \, \exists \alpha' \geq \alpha.(s, \alpha', \aleph) \in P$$

[5]Depending on the circumstances, we will refer both to timed failures $(s, \aleph)$ of P and triples $(s, \alpha, \aleph) \in P$ as $t-$complete behaviours if they satisfy this condition.

or, in other words, if every member of $\mathit{Fail}(Q)$ is in $\mathit{Fail}(P)$ but with a possibly greater associated stability value.

We have already observed that the partial order cannot have a least element because of axiom 4 – the least one could have no bound on the number of events which can occur up to time t. It also fails to be closed under the limits of increasing sequences. In the case of Σ infinite this is easy to demonstrate, using the same examples which work for untimed CSP with unbounded nondeterminism, for example

$$P_n = \sqcap \{m \rightarrow STOP \mid m \geq m\}$$

is a sequence of processes, ordered under $\sqsubseteq$, with no upper bound – any upper bound could neither communicate nor refuse the whole of Σ in contradiction to the axioms.

It also fails to be closed under limits when Σ is finite, though here the examples are a little more subtle and rely upon time-specific arguments. It turns out that no upper bound of certain sequences of well-formed processes can satisfy axiom 10, either because they must have an infinite number of state-changes or because they fail to leave events available at the instant when they are withdrawn. As an example of the first, suppose Q_n is the process that makes the event a available during the intervals $[0, 1 - 2^{-1}]$, $[1-2^{-2}, 1-2^{-3}], \ldots, [1-2^{-2n}, 1-2^{2n+1}]$ and refuses it in the appropriate half-open intervals interleaving and following these. Let $P_n = \sqcap \{Q_m \mid m \geq n\}$. A little thought will reveal that any upper bound of the ordered sequence P_n would be obliged to change state infinitely often in the time interval $[0, 1]$ (when no communication has taken place) and that there is no 1-complete extension of the timed failure $(\langle\rangle, \emptyset)$.

One could plausibly argue for a strengthening of axiom 10 that would ban this counterexample. One of the things this axiom does is to assert that, at least as far as one can detect in some sense, processes only change state finitely often in a finite time. When we asserted in axiom 4 that processes could only *communicate* finitely often in a finite time it was done by postulating the existence, for each process, of a uniform bound function $n(t)$ on the number of events the process could perform up to time t. We could have taken this approach with axiom 10 and also postulated that each of the t-complete behaviours for a process has its number of state-changes bounded by $n(t)$ (using the process' bound function from axiom 4). The reader should be able to see that this would ban the processes P_n of the previous paragraph, since the number of state-changes they make up to time 1 is not bounded (though, for any nondeterministic choice they might make, it is finite).

The strengthening of axiom 10 would, however, neither solve the incompleteness problem with infinite alphabets, and nor would it remove the following example. Let t_n be any strictly increasing sequence converging to 1 from below, and let $Q_n = ((a \rightarrow STOP) \Box WAIT\, t_n); STOP$. Under the standard semantics, the process Q_n offers a until time t_n, whereupon the $WAIT\, t_n$ process terminates and removes the possibility of the a. If a is offered at exactly t_n, it may occur or may not – the $\Box$ operator has to arbitrate between two events which become ready simultaneously. This is precisely the situation covered by our discussion of part of axiom 10 – events which are offered are still possible at the instant from which they are refusable when withdrawn. Now consider the processes $P_n = \sqcap \{Q_m \mid m \geq n\}$. P_n may withdraw the offer of an a at any sufficiently large t_m, but note that it cannot communicate a at time 1. It is however obliged to offer a up to t_n, and as n increases this value increases to 1. Any upper bound would be obliged to offer a up to time 1 without the possibility of performing it *at* time 1, in violation of the same aspect of axiom 10.

Infinite complete behaviours

Axiom 10 gave us the notion of a $t-$complete behaviour. This gives us a 'convincing explanation' of how a process might have behaved up to time t, and the axiom tells us that we can find one of these extending any given timed failure $(s, \aleph)$ with a stability value as close as we please to that associated with $(s, \aleph)$. In technical manipulations we will be doing later it will be useful to be able to extend this to an infinite complete behaviour, which gives us a convincing explanation of how the process might behave over all time. This will be a triple $(s, \alpha, \aleph^*)$, where s is still a finite timed trace, $\alpha \in \mathbf{R}^+ \cup \{\infty\}$, but now $\aleph$ is allowed to extend to infinity: it is a set of pairs (t, a) such that each of its restrictions $\aleph^* \upharpoonright t$ is in $RSET$ (i.e., it only changes finitely often in a finite time). Informally this triple means that the process might be observed to perform the trace s and be observed though all time to refuse $\aleph^*$, and that given this we know that it became stable at time α at the latest. Such a triple (or, where appropriate, the pair $(s, \aleph^*)$) can be said to be a complete infinite behaviour of a process P if $(s, \aleph \upharpoonright t) \in Fail(P)$ for all t, $\alpha = inf\{\alpha' \mid (s, \alpha', \aleph^* \upharpoonright t) \in P\}$ and the same conditions applied as for a t-complete behaviour, namely

$$(t,a) \notin \aleph^* \Rightarrow (s|\!\!\backslash t.\langle(t,a)\rangle, \aleph^*|\!\!\backslash t) \in \mathit{Fail}(P)$$

for each $t \in [0,\infty)$ and $a \in \Sigma$, and

$$(\neg\exists\epsilon > 0.(t-\epsilon,t) \times \{a\} \subseteq \aleph^*) \Rightarrow (s\|\!\!\backslash t.\langle(t,a)\rangle, \aleph^*|\!\!\backslash t) \in \mathit{Fail}(P)$$

when $t \in (0,\infty)$ and $a \in \Sigma$.

This gives an obvious extension to all time of what axiom 10 provides us with up to any finite time – a plausible explanation of the state-changes the process went through in getting to the trace s and those which might happen after the end of s on the assumption that no event subsequently occurs. The following lemma shows that these always exist, and that it is (as we might have hoped) consistent to believe that a stable process does not change state.

Lemma 1 If $P \in TM_{FS}$, $(s,\alpha,\aleph) \in P$, and $t < \alpha$ then there is an infinite complete behaviour $(s,\beta,\aleph^*)$ of P such that $\aleph \subseteq \aleph^*$ and $t < \beta \leq \alpha$. Furthermore, if $\beta < \infty$, we can assume

$$\aleph^*\!\uparrow\![\beta,\infty) = [\beta,\infty) \times \Sigma(\aleph^*\!\uparrow\![\beta,\infty))$$

Proof We will first give one construction that works for the main statement above in all cases, and then give a different one which works for the second statement in its restricted case. Pick a value t' such that $t < t' < \alpha$. Starting with $(\beta_0, \aleph_0) = (\alpha, \aleph)$, we use axiom 10 iteratively, on the nth iteration starting from $(s, \beta_{n-1}, \aleph_{n-1})$ with $t_2 = t'$ and $t_1 = T + n$ where $T = \mathit{end}(s,\aleph)$, thereby obtaining $(s,\beta_n,\aleph_n)$. Necessarily, the β_n form a (not necessarily strictly) decreasing sequence of values between t' and α, and $\aleph_n \subseteq \aleph_{n+1}$ for all n. And $(s,\beta_n,\aleph_n)$ is a $(T+n)-$complete behaviour for $n > 0$. Now, set

$$\aleph^* = \aleph_1\!\uparrow\![0, T+1) \cup \bigcup_{n=2}^{\infty} \aleph_n\!\uparrow\![T+n-1, T+n)$$

Notice that, by construction, each $\aleph^*|\!\!\backslash t$ for $t \in [0,\infty)$ belongs to $RSET$. Since $\aleph \subseteq \aleph^*|\!\!\backslash T+n \subseteq \aleph_n$, it follows (using axiom 9) that there is some γ_n with $\beta_n \leq \gamma_n \leq \alpha$ such that $(s,\gamma_n,\aleph^*|\!\!\backslash T+n) \in P$. Clearly the γ_n form a decreasing sequence with a limit β^* satisfying $t < t' \leq \beta^* \leq \alpha$. Claim $(s, \beta^*, \aleph^*)$ is a complete infinite behaviour of P.

If $(t, a) \notin \aleph^*$, then choose $n = 1$ if $t < T+1$ or otherwise let n be such that $T+n-1 \leq t < T+n$. By definition of $\aleph^*$, we then know that $(t, a) \notin \aleph_n$ and hence $(s \upharpoonright t.\langle(t, a)\rangle, \aleph_n \upharpoonright t) \in \mathit{Fail}(P)$. Since $\aleph^* \upharpoonright t \subseteq \aleph_n \upharpoonright t$ it follows by axiom 9 that $(s \upharpoonright t.\langle(t, a)\rangle, \aleph^* \upharpoonright t) \in \mathit{Fail}(P)$. If $t > 0$ and $\neg \exists \epsilon > 0.(t-\epsilon, t) \times \{a\} \subseteq \aleph^*$ we choose $n = 1$ if $t \leq T+1$ and otherwise n is such that $T+n-1 < t \leq T_n$. A similar argument to the above then shows that $(s \Vert\!\upharpoonright t.\langle(t, a)\rangle, \aleph_n \upharpoonright t)$ and hence $(s \Vert\!\upharpoonright t.\langle(t, a)\rangle, \aleph^* \upharpoonright t)$ belong to $\mathit{Fail}(P)$. This completes the proof of the first statement.

Suppose $(s, \aleph')$ is a $t-$complete behaviour of P and that $(s, \beta, \aleph' \upharpoonright t) \in P$ for some $\beta < t$. Let $A = \Sigma(\aleph' \uparrow [\beta, t))$. If $a \notin A$ we know by completeness of $(s, \aleph')$ that $(s.\langle(t, a)\rangle, \aleph' \upharpoonright t) \in \mathit{Fail}(P)$. Axiom 8 (with $w = \langle(t, a)\rangle$) then tells us that, for all $t' \in [\beta, \infty)$, $(s.\langle(t', a)\rangle, \aleph' \upharpoonright \beta \cup [\beta, t') \times A) \in \mathit{Fail}(P)$. It follows easily that, for all $t'' < t'$, the failure $(s, \aleph' \upharpoonright \beta \cup [\beta, t') \times A)$ is $t''-$complete. Axiom 11 tells us, that for $t' \geq t$, the stability value associated with this failure is β. It follows that $(s, \beta, \aleph' \upharpoonright \beta \cup [\beta, \infty) \times A)$ is a complete infinite behaviour of P.

Suppose that the stability value β^* produced by the first part of this result was finite. This must have been because one of the sequence γ_n which converged down to it was finite. Clearly there then exists n such that $\gamma_n < T + n$. Thus the preconditions of the previous paragraph are satisfied by the failure $(s, \aleph^* \upharpoonright T + n + 1)$, $t = T + n$ and $\beta = \gamma_n$. The conclusions of that paragraph then give exactly what is required for the second part of the lemma. $\Box$

One immediate corollary of this result is that, for any $(s, \alpha, \aleph) \in P$, α is the supremum of all the stability values α^* associated with the complete infinite extensions of $(s, \alpha, \aleph)$.

2.7 A study of nondeterminism in TM_{FS}

Nondeterminism is a well-known consequence of concurrency. In this section we will use the tightly-defined model we have created to study just how, and in what forms, nondeterminism appears in real-time concurrent systems. We will find that the subtleties of real-time behaviour – in particular issues relating to instants when a process can arbitrate between some internal action and an external communication – make it a rather harder subject than for untimed CSP.

One of the features of all the widely used models of untimed CSP is the way in which any process P can be identified with the set of all *deterministic*, or sometimes *pre-deterministic* processes processes Q which 'implement' it, namely $P \sqsubseteq Q$. (Where the general nondeterministic choice operator $\sqsubseteq$ was defined, this 'identification' simply amounted to saying that the set $imp(P)$ of implementations was nonempty and $\bigsqcap imp(P) = P$.)

A deterministic process was there one which never had the choice of accepting or refusing any action, which was equivalent to being maximal in the partial order. In the models with divergence this notion had to be weakened to say that a pre-deterministic process was one which was deterministic until it diverged. Blamey [] has written on this phenomenon and has argued that, since the correct structure of deterministic or pre-deterministic processes is generally easier to establish and justify than that of general ones, we can say that the axioms of a CSP model are complete if we have such a property. The rationale behind this term is that, given we know what the set of 'deterministic' ones is, and what the definition of general nondeterministic composition is, we can tell exactly which objects are the nondeterministic compositions of sets of 'deterministic' ones. Thus Blamey calls a set of axioms sound if they allow all such objects, and complete if they allow no others.

Certainly this form of completeness gives powerful evidence that the way the axioms extend the notion of (pre)-deterministic processes to nondeterministic ones is correct. It also gives us a much greater level of understanding of how the model fits together and how it treats nondeterminism.

A taxonomy of nondeterminism

We will find in this section that it is not altogether straightforward to construct an appropriate notion corresponding to deterministic processes and which is sufficient to give us a completeness result. The resulting investigations will, however, give us a much deeper understanding of the model and of the varieties of nondeterminism it encompasses.

Fully predictable deterministic processes are sufficient for completeness in models of untimed CSP. Essentially this is because it turns out that, given any behaviour of such a process (even though that process might be genuinely nondeterministic) it is always possible to find a complete deterministic process which 'sits inside' the given one and which exhibits the given behaviour. Things turn out not to be quite as simple in the case of real-time CSP. The following list enumerates various types of 'unpredictability' which cannot, for one reason or another, be factored out in this way.

1. The first is connected with axiom 10, which specifies that a withdrawn event is still possible at the instant of withdrawal. This is a form of nondeterminism which we are specifying *must* be present in any process which can retract an offer of communication – and we could not hope to get a completeness result of the type above unless the class of 'deterministic' processes contained retracting ones.

2. The second concerns processes which have an event possible at an isolated time, for example

$$(a \to STOP \Box b \to STOP) \setminus b$$

   which, in the standard semantics, can do a at time 0 but at no other. Since all refusals are over intervals, there is no process which can offer such a point event without also being able to refuse it if offered. We might term such an isolated event a *transient event*. No fully predictable implementation of the above process would be able to communicate an a.

3. Transient events can manifest themselves in another, yet more subtle, form at the very moment when a process is becoming stable. Up to the time when a process stabilises, our axioms allow it, for example, to make a single event available continuously but have its subsequent behaviour vary quite arbitrarily depending on when the event happens. Provided that each of these different behaviours is deterministic then the whole process is. However, once it has stabilised, axiom 8 tells us that the process' subsequent behaviour does not depend on when the event did. The problem with an event happening at the instant of stability is that it might be an *alternative* to stability rather than a manifestation of the stable configuration.

   This situation is actually rather similar to the one which led us to postulate that events are still possible at the moment when they are withdrawn, in that at the moment when a process would otherwise become stable it may be possible for it to do something else. The following example illustrates this. Consider the process

$$((a \rightarrow STOP)\Box SKIP); (a \rightarrow a \rightarrow STOP)$$

In the standard semantics, the first a is possible 'transiently' at time 0; otherwise the *SKIP* terminates immediately and the second a is also available at time 0, with the process being stable at once. The result of all this is that if an a is accepted at time 0 we cannot be sure whether or not the second will occur, while if we wait beyond this time we can be sure it will. Any predictable (even modulo the questions above) implementation of this process is forced, by axiom 8, to make the second a available following one at time 0. It follows that it cannot refuse a after the time taken to complete the first. Therefore the behaviour subsequent to the 'transient' a is never reflected by any such implementation. It should be clear that we could have varied the above example so that the different behaviour introduced by the transient was delayed an arbitrary number of communications beyond it, or could have been of a different sort such as a larger stability time.

In summary, when an event happens at a time *after* stability the same subsequent behaviours are possible at whatever times the event happens *at or after* stability. But subsequent behaviours which are enabled when an event happens *at* stability need not manifest themselves when the same event happens *after* it. This type of transient is more subtle than the last because they are not apparent when they happen, only in the effects they leave behind.

4. A final source of difficulties can be found in axiom 3. Recall that this states that events which happen at the same time can be re-ordered in a trace without changing behaviour. While one order in which a set of simultaneous events occurs may be totally consistent with what is refusable on the traces where they happen, this need not be the case with another ordering. This can either be because the occurrence of one event in the set coincides with the disabling of another, or (and this causes more problems) with the enabling of another. We will see examples of these phenomena later.

There is a sense in which difficulties 1 and 4 are more pervasive than 2 and 3. If we had a notion of 'implementation' which did not allow the forms of nondeterminism which arise under these headings, there would be processes with no implementations at all. This is not the case with the transient events of 2 and 3, which arise as alternatives 'grafted on' to otherwise well-behaved processes. One consequence of this is that forms 1 and 4 must be allowed throughout an implementation, while, if we are seeking an implementation of a process P which manifests one of P's behaviours $(s, \aleph)$, it is reasonable to restrict its transient events to ones in s.

Quasi-deterministic processes

To approach the definition we need for a completeness result, we will start out with one that is too strong for all the reasons set out above. We define a *fully deterministic* process to be one for which we can always tell whether a given (instantaneous) offer of an event will be accepted. Namely, for all timed traces s, $t \geq end(s)$ and $a \in \Sigma$, we never both have $s.\langle(t,a)\rangle \in Traces(P)$ and $(t,a) \in \aleph$ for which $(s, \aleph) \in \mathit{Fail}(P)$. This definition ignores stability values, though for some purposes one might wish to strengthen it accordingly.

To deal with the first, and part of the fourth, problem mentioned above we must allow an event to occur if the process was unable to refuse it in some half-open interval ending at the given time. We can define a process to be quasi-deterministic if, and only if, under the same circumstances as above, we never both have $s.\langle(t,a)\rangle \in Traces(P)$ and that there exists $\epsilon > 0$ with

$$(s', [max\{0, t-\epsilon\}, t+\epsilon) \times \{a\}) \in \mathit{Fail}(P)$$

If P has just started (i.e., $t = 0$), then it cannot both accept and reject a at time t. Otherwise, it must not accept a if it is able to reject it in some interval up to and including the the current time t.

If we had just wished to deal with problem 1 then we would have altered the above definition to

$$(s, [max\{end(s), t-\epsilon\}, t+\epsilon) \times \{a\}) \in \mathit{Fail}(P)$$

The difference between these appears in a process which has just been offering an event a, but has started to refuse it at the same moment when it has accepted an event b (there being no reason why a and b must be different). Axiom 10 does not *force* the process to be able to accept a at the same time as b – after all they may have been offered as alternatives. However there are circumstances where we would expect an a to be possible, and can deduce this from axiom 3. Consider, for example, the process

$$(((a \rightarrow STOP) \Box \mathit{WAIT}\, 1); STOP) \mid\mid\mid (b \rightarrow STOP)$$

which is forced, by axiom 10, to have the trace $\langle(1, a), (1, b)\rangle$ and hence, by axiom 3, has the trace $\langle(1, b), (1, a)\rangle$. The second and stronger definition above would have disallowed the a after the b, whereas the first allows it. A subtle variation on this example appears if a and b are replaced by the same event.

This still does not deal with the second and third problems discussed above of transient events. In dealing with these there are two things to notice. Firstly, a given recorded trace might have a number of transients in it. Thus we need to allow for at least any finite number of transients being possible for a given process. Second, it is quite possible for a transient of either sort to appear after the process has previously been stable (i.e., on a proper prefix of the current trace), as occurs in the process

$$a \rightarrow (a \rightarrow STOP \Box b \rightarrow STOP) \setminus a$$

Since we know that the behaviour of a stable process does not depend on the time at which the next event happens (axiom 8), it follows that if a transient is possible at some later time if the next event happens at one time then it must also be possible at suitably shifted later times when the next event occurs at some other time. Of course this means that sometimes, though in rather special circumstances, a process must have an uncountable infinity of transients if it has one. Given this discussion and the existence of the 'at stability' type of transient it is obviously important for us to understand the nature of stability in the class of quasi-deterministic processes. It will also allow us to find an appropriate strengthening of the definition to deal with stability.

The following result shows that quasi-deterministic processes actually have much in common with the deterministic and pre-deterministic processes of untimed CSP.

Lemma 2 Suppose $P \in TM_{FS}$ is quasi-deterministic. Then

1. If $(s, \alpha^*, \aleph^*)$ is a complete infinite extension of $(s, \emptyset)$, for $s \in Traces(P)$, then $(s, \aleph) \in Fail(P)$ if and only if $\aleph \subseteq \aleph^*$ (for all $\aleph \in RSET$).

2. The complete infinite extension $(s, \alpha^*, \aleph^*)$ of any $(s, \alpha, \aleph) \in P$ is unique.

3. If $(s, \alpha, \aleph)$ and $(s, \alpha', \aleph')$ are both in P then $\alpha = \alpha'$.

4. P is the only quasi-deterministic process with its trace/stability set $Stab(P)$.

5. If $Q \sqsupseteq P$ then Q is quasi-deterministic.

6. Over quasi-deterministic processes, the inequality in axiom 8 becomes an equality (i.e., the stability value of the shifted behaviour is also shifted by the same amount) provided that the time t' is strictly greater than α.

Proof Suppose s is any trace of P, and that $(s, \alpha, \aleph^*)$ is a complete infinite extension of the timed failure $(s, \emptyset)$. Now suppose $(s \upharpoonright t, \aleph) \in \mathit{Fail}(P)$ is such that $\mathit{end}(\aleph) \leq t)$. Claim that $\aleph \subseteq \aleph^*$. If not, there would be times $t_1 < t_2 < t$ such that no event of s occurs in $[t_1, t_2]$ and an event a such that $[t_1, t_2) \times \{a\} \subseteq \aleph$ and $[t_1, t_2) \times \{a\} \cap \aleph^* = \emptyset$. For all $t_1 < t' < t_2$ there thus exists ϵ with $(s \upharpoonright t_1, [t' - \epsilon, t' + \epsilon) \times \{a\}) \in \mathit{Fail}(P)$ though the completeness of $(s, \alpha, \aleph^*)$ ensures that $s \upharpoonright t_1.\langle(t', a)\rangle \in \mathit{Traces}(P)$. This contradicts our assumption of quasi-determinacy, and so the claim is established, proving part 1.

Part 2 follows easily from part 1, since if $(s, \alpha_1, \aleph_1)$ and $(s, \alpha_2, \aleph_2)$ were different complete extensions of $(s, \aleph)$ (and hence of $(s, \emptyset)$) there would be a time t such that either $\aleph_1 \upharpoonright t \not\subseteq \aleph_2$ or $\aleph_2 \upharpoonright t \not\subseteq \aleph_1$. We know that the stability value associated with any failure of the form $(s, \aleph)$ is the supremum of those associated with its complete infinite extensions. It follows that that stability value is the one belonging to the *only* complete infinite extension. Since this complete infinite extension is common to all failures $(s, \aleph')$ with the given trace, we have proved part 3.

In order to prove part 4 it is enough, by part 3, to prove that if P and Q are quasi-deterministic and $\mathit{Traces}(P) = \mathit{Traces}(Q)$ then $\mathit{Fail}(P) = \mathit{Fail}(Q)$. If not then, without loss of generality we may assume that there is $(s, \aleph) \in \mathit{Fail}(P)$ but not in $\mathit{Fail}(Q)$. Since $s \in \mathit{Traces}(Q)$ we can extend $(s, \emptyset)$ to a (unique) complete infinite extension $(s, \aleph^*)$. Necessarily, as in the proof of part 1, there are times $t_1 < t_2$ such that no event of s occurs in $[t_1, t_2]$ and an event a such that $[t_1, t_2) \times \{a\} \subseteq \aleph$ and $[t_1, t_2) \times \{a\} \cap \aleph^* = \emptyset$. For all $t_1 < t' < t_2$ there thus exists ϵ with $(s \upharpoonright t_1, [t' - \epsilon, t' + \epsilon) \times \{a\}) \in \mathit{Fail}(P)$ though the completeness of $(s, \aleph^*)$ ensures that $s \upharpoonright t_1.\langle(t', a)\rangle \in \mathit{Traces}(Q) = \mathit{Traces}(P)$. Thus part 4 is proved.

The proof of part 5 is completely elementary. It is worth noting that quasi-deterministic processes are not maximal under the order. In general we can 'improve' a quasi-deterministic process P either by decreasing its stability values or by exploiting the fact that we took the definition above which was weaker on what could happen at the same time as another event.

Axiom 8 says that, if $(s, \alpha, \aleph) \in P$ and if $t > \alpha$, $t \geq end(\aleph)$ (so that by time t we can be sure the process has been stable since time α, then any behaviour of P starting from t can be shifted back to any time $t' \geq t$. If $t' > \alpha$ and we could, in fact, have deduced that the process had been stable since α at time t', then the same axiom can be used to shift the behaviour back the other way; since the inequality then works both ways between the shifted stability values, the shift must be exact. Since, by part 3, stability values in quasi-deterministic processes depend only on the trace, this deduction can always be made for them. $\Box$

Part 4 is obviously very like the result which says that, in untimed CSP, a deterministic process is determined by its set of traces or a predeterministic process is determined by its sets of traces and divergences. One significant difference is that, in the timed case, by no means every plausible set of traces gives rise to a quasi-deterministic process. An example of this is provided by the traces of the process we used to illustrate the first type of transient above.

It is interesting what part 6 does not say – it does not say that the inequality of axiom 8 becomes an equality for $t' = \alpha$. This is because the definition of quasi-deterministic processes allows a limited form of the 'at stability' type of transient discussed earlier. Consider, for example, the process

$$(a \rightarrow \bot)\Box SKIP); (a \rightarrow STOP)$$

which, under the standard semantics is immediately stable and offers a, after which it can do nothing. If the a occurs at time $t > 0$ the subsequent stability time varies linearly with t. But an initial transient destroys this relationship for $t = 0$.

2.7.1 Adding transients to quasi-deterministic processes

We are now going to tackle the question of how one might add a transient event (and its subsequent consequences) to a quasi-deterministic process P. Lemma 2 and the above discussion give us a clear indication of how to deduce, once we have been told to place a transient at one point (a particular time in the closed interval between the end of a trace s and the stability time associated with s), where else it must be possible because of earlier stability. (We can ignore refusal information because of what we know from the lemma.)

Suppose $s = v.\langle(t,a)\rangle.w$, $t > \alpha$ where α is the stability time associated with v and $t' \geq \alpha$. Then we will write $s \overset{t'-t}{\Longrightarrow}_1 v.\langle(t',a)\rangle.(w + (t'-t))$ $(= s')$ and observe that any transient added after s must be added after s', shifted through $t'-t$. (The time of the shifted transient is guaranteed to be in range by axiom 8.) If $t' > t$, we will write $s \overset{t'-t}{\Longleftrightarrow}_1 s'$ and observe that if $s_1 \overset{t}{\Longleftrightarrow}_1 s_2$ then (i) $s_2 \overset{-t}{\Longleftrightarrow}_1 s_1$ and (ii) (by Lemma 2 (6)) the stability times of all traces beyond the shifted event in s_1 and s_2 are also shifted by precisely the same amount t. Since, if $s_1 \overset{t}{\Longrightarrow}_1 s_2$, the first shifted event of s_1 occurs strictly later than any predecessor, if $s_1' \cong s_1$ then there is $s_2' \cong s_2$ such that $s_1' \overset{t}{\Longrightarrow}_1 s_2'$.

We can form transitive closures of these relations to take account of the fact that several events in a trace might happen after stability, adding in the re-ordering congruence, as follows:

- If $s \cong s'$ then $s \overset{0}{\Longrightarrow} s'$ and $s \overset{0}{\Longleftrightarrow} s'$.
- If $s \overset{t}{\Longrightarrow}_1 s'$ and $s' \overset{t'}{\Longrightarrow} s''$, then $s \overset{t+t'}{\Longrightarrow} s''$.
- If $s \overset{t}{\Longleftrightarrow}_1 s'$ and $s' \overset{t'}{\Longleftrightarrow} s''$, then $s \overset{t+t'}{\Longleftrightarrow} s''$.
- $\overset{t}{\Longleftrightarrow}$ and $\overset{t}{\Longrightarrow}$ are the smallest relations consistent with the above.

Both these relations are transitive (adding the times) and reflexive (with time 0). We also have that $s \overset{t}{\Longleftrightarrow} s'$ implies $s' \overset{-t}{\Longleftrightarrow} s$ and that, in this case, the order in which the various shifts are carried out to get from s to s' is irrelevant. In relation to $\cong$, it is easy to see that if $s \overset{t}{\Longrightarrow} s'$ then the groups of simultaneous events in s remain together in s' and keep their relative order except that some might be amalgamated (in a process which can become stable instantly after some communication), and that if $s \overset{t}{\Longleftrightarrow} s'$ then the integrity and order of these groups is preserved completely.

Suppose P is a quasi-deterministic process, that s is one of its traces with associated stability value α, $a \in \Sigma$ and $t \in [end(s), \alpha]$. Let us consider what the version of P would look like which had the additional (and nondeterministic) possibility of communicating the initial events of a process Q at time t, and then continuing to behave like Q. For various reasons it appears to be sufficient to consider only cases where no events have happened already in s at time t, namely when $s = \langle\rangle$ or $t > end(s)$. The various varieties of transient which might co-incide with events at the end of s either cannot arise at all because of the axioms, become duplicated by stability so that they are not transients at all, or can be dealt with by including the events of s with which they co-incide as transients as well (essentially by absorbing part of P into Q). So let Q be any element of TM_{FS} which can communicate at time 0. We can construct the element of TM_{FS} which behaves as indicated above:

$$\underline{SUP}(P \cup \{(s'.w + t' + t, \alpha'' + t' + t, (\aleph' \upharpoonright t + t') \cup (\aleph'' + t' + t)) \mid \\ (s', \aleph') \in \mathit{Fail}(P) \wedge s \stackrel{t'}{\Longrightarrow} s' \wedge (w, \alpha'', \aleph'') \in Q \wedge \mathit{begin}(w) = 0\})$$

We can denote this combination by $P \xrightarrow{s,t} Q$.

Instant enabling

In our definition of quasi-determinism we only claimed to have dealt with one aspect of the difficulties arising from axiom 3. The concept of one event instantly enabling another, so that a process becomes unable to refuse one event *because* it has performed another *at the same time*, is another source of problems relating to the interplay of that axiom with the others. At first sight it is difficult to see how one might realise such a situation, especially if we assume that all events take non-zero time to complete. It is interesting to note that axioms 2 and 3 together state that if two events are possible at one time then either may appear without the other – meaning that any absolute causal dependance between two simultaneous events is impossible. But in fact it turns out that we can get close enough to this instant enabling to have problems with our definition of quasi-determinism.

Consider the process

$$P = ((a \to STOP \,|||\, b \to STOP) \Box SKIP); STOP$$

Here, the occurrence of either a or b at time 0 instantly enables the other, in the sense that the process cannot then refuse the other event – even though the original process could refuse both events at time 0 (and all later times) on the empty trace. Though the a or b which appear here are transients (of the first type discussed earlier) this is not the case if we offer the choice between this way of offering a and another:

$$P \Box (a \to c \to STOP) \Box \bot$$

Here, things become rather difficult to disentangle. It gets worse if we replace P by the process Q which works in essentially the same way except that it cannot perform an a after time 0:

$$Q = ((((a \to STOP)\Box SKIP); STOP) \,|||\, b \to STOP)\Box SKIP); STOP$$

The process

$$R = Q\Box(a \to c \to STOP)\Box\bot$$

cannot refuse a on its first step but may, when it performs $(0, a)$, instantly lose the ability to refuse b[6]. If, on the other hand, it performs $(0, b)$ then it can and must instantly begin refusing a. Thus the trace $\langle(0, a), (0, b)\rangle$ is very much allowed by our definition of quasi-determinacy, while the equivalent trace $\langle(0, b), (0, a)\rangle$ is not. We probably would not want to consider R quasi-deterministic, since it has a definite choice of what to do at time 0. Consider, however, the process which behaves like R except that when $(0, a)$ occurs it must pick the Q behaviour rather than the one with the following c. This is an element of TM_{FS} (though seemingly not one expressible in Timed CSP under its standard semantics) which would have no quasi-deterministic implementations under the current definition. The most troublesome trace any implementation must have is $\langle(0, b)\rangle$, since it is both refusable and carries with it no explanation of why it is there (i.e., the forceable event $(0, a)$).

Rather than attempt to get around this technical difficulty, we choose to simply note it and necessarily restrict the set of processes which can expect to be determined by their implementations. Define a process to be *free of instant enabling* if, whenever $t' > t$ and $(s\langle(t, a)\rangle, \aleph)$ is $t'-$complete, then there is $\epsilon > 0$ and $\aleph' \subseteq \aleph$ such that $(s, \aleph')$ is $(t + \epsilon)-$complete. This simply means that any events which might become enabled (i.e., unrefusable in a complete behaviour) instantly after a could have become enabled at that moment event if a had not occurred. Thus there is no causal relationship between the occurrence of a and the enabling of other events. Clearly the various examples in the discussion above fail to have this property.

It is interesting to note that, while 'instant enabling' seemingly describes the behaviour of the examples discussed above on an abstract level – the communication of a at time 0 instantly enables b – in fact the CSP defined examples worked by a preventing an internal action that would have stopped

[6]The purpose of the c in the definition of R is to ensure that we cannot ignore that it might lose the ability to refuse b – since along with this it also loses the ability to perform c later.

the b from being enabled. Although, on the surface, this might seem a very fine distinction it is in fact significant when we come to consider stability. For in the mechanism which we described second there is the implication that, when the enabling a occurred, the process had not already become stable. There is no such implication with the simple idea of instant enabling – as might for example appear in the prefixing operation $a \rightarrow P$, were it definable for actions a that take no time and P which can communicate at time 0.

In fact, our axioms prohibit instant enabling after stability as is shown by the following argument. Suppose $(s.\langle(t,a)\rangle, \aleph)$ is $t'-$complete, where $t' > t$, that $(s, \alpha, \aleph|\!\!\setminus t) \in P$ for $\alpha < t$ but that there is no $\aleph' \subseteq \aleph$ and $\epsilon > 0$ with $(s, \aleph')$ $(t+\epsilon)-$complete. If $(t,b) \notin \aleph$ then $(s.\langle(t,a),(t,b)\rangle, \aleph|\!\!\setminus t) \in \mathit{Fail}(P)$ and hence, by axioms 2 and 3, $(s.\langle(t,b)\rangle, \aleph|\!\!\setminus t) \in \mathit{Fail}(P)$. Axiom 7 then tells us that $(t',b) \notin \aleph$ for any $\alpha \le t' < t$. In other words

$$\{b \mid (t,b) \in \aleph\} \supseteq \Sigma(\aleph{\uparrow}[\alpha,t))$$

By the structure of RSET we then know that there is $\epsilon > 0$ such that

$$\aleph{\uparrow}[t, t+\epsilon) \supseteq [t, t+\epsilon) \times (\Sigma(\aleph{\uparrow}[\alpha,t)))$$

But exactly the same arguments and constructions used in the proof of Lemma 1 show that, for any $t' \ge t$ the triple $(s, \alpha, \aleph|\!\!\setminus t \cup ([t,t') \times \Sigma(\aleph{\uparrow}[\aleph, t)))$ is a $t'-$complete behaviour of P. This is exactly what we require to establish our claim.

Towards a completeness theorem

So far in this section we have presented a taxonomy of nondeterminism in our model, the class of quasi-deterministic processes which are perhaps those most analogous to the pre-deterministic ones of untimed CSP, an operator for introducing transient events into them, and discussed the phenomenon of instant enabling. In this final subsection we bring all of these things together by conjecturing a completeness theorem of the type discussed earlier, and by providing some evidence for this conjecture.

Define the class of *almost deterministic* processes to be the smallest one which contains the quasi-determinstic ones and which, whenever P is quasi-deterministic, Q is almost deterministic with communications at time 0,

$(s,\alpha) \in Stab(P)$ and $t \in [end(s),\alpha]$, $P \overset{s,t}{\longrightarrow} Q$ is almost deterministic. In other words an almost deterministic process is quasi-deterministic except for a finite number of occasions where transients are possible, which are arranged in a single unbranching sequence. We will take these as the class of processes which will form the basis of our completeness conjecture.[7]

We define an *implementation* of $P \in TM_{FS}$ to be any almost deterministic Q such that $P \sqsubseteq Q$. Let $imp(P)$ be the set of all its implementations.

We have already said that the general nondeterministic construct $\bigsqcap S$ would be allowed, subject to restrictions, for nonempty sets S of processes. In order to discuss completeness we need its definition and details of the restrictions. The nondeterministic composition of a set of processes can behave like any one of them – therefore the set of its observable behaviours should be the union of those of the processes over which we are taking the choice. Since, in TM_{FS}, we associate with each timed failure $(s,\aleph)$ only one stability value – the supremum of those times at which stability can actually occur – we form $\bigsqcap S$ as follows for a nonempty subset S of TM_{FS}:

$$\bigsqcap S = \underline{SUP}(\bigcup S)$$

where the $\underline{SUP}$ operator is as defined earlier. The restriction we need derives from axiom 4: if the elements of S have functions $n(t)$ bounding the numbers of events up to given times which are not bounded by some fixed function, then $\bigsqcap S$ would violate the axiom. Hence we assume that there is a fixed function $n^*(t)$ such that, for each $P \in S$, the number of events up to t in P is bounded by $n^*(t)$. The $\bigsqcap$ operator can only be used in such cases.

Notice that the functions $n(t)$ which exist for P by axiom 4 also work for every $Q \in imp(P)$, so that providing $imp(P)$ is nonempty, the nondeterministic composition $\bigsqcap(imp(P))$ is well-defined. We can thus state our conjecture:

Conjecture If $P \in TM_{FS}$ is free of instant enabling, then $imp(P)$ is nonempty and $\bigsqcap(imp(P)) = P$. $\Box$

If S is a set of processes Q such that $P \sqsubseteq Q$ (for fixed P), then it is easy to show that $P \sqsubseteq \bigsqcap S$. In order to prove the conjectured result it would thus be sufficient to find, for each $(s,\alpha,\aleph) \in P$ and $t < \alpha$, an element Q of $imp(P)$ which contains $(s,\beta,\aleph)$ for some $\beta > t$.

[7]If desired, this class could probably be tightened somewhat. For example one could attempt to restrict the class of transients introduced to the two specific classes identified earlier.

We expect the proof of this conjecture to consist of a construction of these Q's. Such a construction will necessarily be detailed and require careful checking of the axioms. In its essence we expect it to revolve around manipulations of complete infinite behaviours of the types constructed in Lemma 1. Starting with a complete infinite extension of the target behaviour, we would pad this out to a complete description of what the implementation Q could do after every timed trace and, where this is necessary detail (after at-stability transients), timed refusal. The only events of Q which could be transients would be ones of the target behaviour.

The following result will probably be important in this construction since it says that, if in the complete infinite behaviour $(s.w, \aleph^*)$ the first events of w apparently occurred at or before stability (because $(s, \alpha, \aleph^* |\!\backslash (\mathit{begin}(w))) \in P$ where $\mathit{begin}(w) \leq \alpha$), then we can extend the initial segment of the behaviour to infinity in such a way that we can still believe this. The importance of this is that events which happen after stability need to be treated differently from ones which happen at or before it.

Lemma 3 Suppose that $(s, \alpha, \aleph) \in P$ is $t-$complete where $\mathit{end}(s) \leq t \leq \alpha$. Then it has a complete infinite extension $(s, \beta, \aleph^*)$ with $t \leq \beta \leq \alpha$ and $\aleph^* |\!\backslash t = \aleph |\!\backslash t$.

Proof First suppose $t < \alpha$. Then, by Lemma 1, there is a complete infinite extension $(s, \alpha', \aleph')$ of $(s, \aleph)$ such that $t < \alpha'$. Let $\aleph^* = \aleph \uparrow [0, t) \cup \aleph' \uparrow [t, \infty)$. The same arguments which were applied in the proof of Lemma 1 show that there is β such that $(s, \beta, \aleph^*)$ is a complete infinite behaviour of P. Axiom 9 (applied to the finite restrictions of $\aleph'$ and $\aleph^*$) shows that $\alpha' \leq \beta \leq \alpha$, as required.

More care is required when $t = \alpha$. We know that there is a sequence of complete infinite extensions $(s, \alpha_n, \aleph_n)$ of $(s, \aleph)$ such that α_n is an increasing sequence converging on α from below. If any of them equal α then the same construction used in the last paragraph applies, so we could assume that all α_n are strictly less than α. We can also assume, thanks to the second part of Lemma 1, that the $\aleph_n$ are all constant after the point of stability. If we set $\aleph'_n = \aleph |\!\backslash \alpha \cup \aleph_n \uparrow [\alpha, \infty)$, it is easy to see that there is some $\beta_n \in [\alpha_n, \alpha]$ such that $(s, \beta_n, \aleph'_n)$ is a complete infinite behaviour. Now let

$$\aleph^* = \bigcap \{\aleph'_n \mid n \in \mathbf{N}\} \, .$$

Clearly $\aleph^* |\!\backslash \alpha = \aleph |\!\backslash \alpha$, $\aleph \subseteq \aleph^*$ and $\aleph^*$ is constant after α. (The fact that all the $\aleph'_n$, and hence, $\aleph^*$, are constant after α, is necessary to ensure that

$\aleph^*$ has the finite variability property – $\aleph^* \restriction t \in RSET$ – we require of complete infinite behaviours.) If we can show that $(s, \aleph^*)$ isa complete infinite behaviour then, since $\aleph \subseteq \aleph^* \subseteq \aleph_n$, its associated stability value must be α.

Completeness up to time α is a straightforward consequence of the α-completeness of $(s, \aleph)$. Beyond α it follows because, if $t' \geq \alpha$ and $(t', a) \notin \aleph^*$, there is some n with $(t', a) \notin \aleph'_n$. We then know that $(s.\langle(t', a)\rangle, \aleph'_n \restriction t')$ belongs to $\mathit{Fail}(P)$ by completeness of $(s, \aleph'_n)$, and hence so does $(s.\langle(t', a)\rangle, \aleph^* \restriction t')$ by axiom 9.

We have thus shown that $(s, \alpha, \aleph^*)$ is a complete infinite behaviour of P, which completes the proof. $\Box$

3 The semantics of Timed CSP

One might argue that Timed CSP should be supplied with a number of different semantics which differ in how the various operators deal with time. Thus an implementor would not be forced to give all constructs exactly the same timing characteristics. He could reason about processes in his implementation by giving a semantics for Timed CSP which accurately reflected how it worked.

If there is a falacy here it is that CSP and Timed CSP are not usually thought of as languages which are directly implemented in the usual sense. They are used to specify intended behaviour, or to reason about implementations at a level a little more abstract than code. We feel that it is better to have a standard semantics for Timed CSP in which the great majority of reasoning is done. This has the obvious advantage of not having to parameterise every result about the language with the semantics used to prove it, and that each term in Timed CSP will have the same meaning to everybody. We imagine that it will also be rather easier for someone working with an implementor to follow the principles set out below than to construct his own semantics for Timed CSP – an activity that would carry an extensive burden of proof to ensure it was a reasonable one.

In constructing the standard semantics we should aim for a combination of elegance – maintaining as many of the appealing algebraic properties of the untimed semantics as possible – with expressive power. For provided we can express a wide range of behaviours in our language, it should be possible to capture the essence of the majority of implementations by representing whatever constructs they contain as hybrids of several Timed CSP constructs.

Without further ado we will now define the standard semantic function $\mathcal{E}_T : TCSP \rightarrow TM_{FS}$.

$$
\begin{aligned}
\mathcal{E}_T[\![\bot]\!] &= \{(\langle\rangle, \infty, \aleph) \mid \aleph \in RSET\} \\
\mathcal{E}_T[\![STOP]\!] &= \{(\langle\rangle, 0, \aleph) \mid \aleph \in RSET\} \\
\mathcal{E}_T[\![SKIP]\!] &= \{(\langle\rangle, 0, \aleph) \mid \surd \notin \Sigma(\aleph)\} \cup \\
&\quad\ \{(\langle(t, \surd)\rangle, t, \aleph_1 \cup \aleph_2) \mid t \geq 0 \wedge (I(\aleph_1) \subseteq [0, t) \wedge \\
&\qquad\qquad\qquad\qquad\qquad \surd \notin \Sigma(\aleph_1)) \wedge\ I(\aleph_2) \subseteq [t, \infty)\} \\
\mathcal{E}_T[\![WAIT\ t]\!] &= \{(\langle\rangle, t, \aleph) \mid \aleph \cap ([t, \infty) \times \{\surd\}) = \emptyset\} \\
&\quad \cup \{(\langle(t', \surd)\rangle, t', \aleph_1 \cup \aleph_2 \cup \aleph_3) \mid t' \geq t \wedge I(\aleph_1) \subseteq [0, t) \\
&\qquad \wedge (I(\aleph_2) \subseteq [t, t') \wedge \surd \notin \Sigma(\aleph_2)) \wedge I(\aleph_3) \subseteq [t', \infty)\} \\
\mathcal{E}_T[\![a \rightarrow P]\!] &= \{(\langle\rangle, 0, \aleph) \mid a \notin \Sigma(\aleph)\} \cup \\
&\quad\ \{(\langle(t, a)\rangle.(s{+}(t{+}\delta)), \alpha{+}t{+}\delta, \aleph_1{\cup}\aleph_2{\cup}(\aleph_3{+}(t{+}\delta))) \mid \\
&\quad\ t \geq 0 \wedge (I(\aleph_1) \subseteq [0, t) \wedge a \notin \Sigma(\aleph_1)) \wedge I(\aleph_2) \subseteq [t, t{+}\delta) \\
&\quad\ \wedge (s, \alpha, \aleph_3) \in \mathcal{E}_T[\![P]\!]\} \\
\mathcal{E}_T[\![a : A \rightarrow P(a)]\!] &= \{(\langle\rangle, 0, \aleph) \mid A \cap \Sigma(\aleph) = \emptyset\} \cup \\
&\quad\ \{(\langle(t, a)\rangle.(s{+}(t{+}\delta)), \alpha{+}t{+}\delta, \aleph_1{\cup}\aleph_2{\cup}(\aleph_3{+}(t{+}\delta))) \mid \\
&\quad\ a \in A \wedge t \geq 0 \wedge (I(\aleph_1) \subseteq [0, t) \wedge A \cap \Sigma(\aleph_1) = \emptyset) \wedge \\
&\quad\ I(\aleph_2) \subseteq [t, t + \delta) \wedge (s, \alpha, \aleph_3) \in \mathcal{E}_T[\![P(a)]\!]\} \\
\mathcal{E}_T[\![P \Box Q]\!] &= \underline{SUP}(\{(\langle\rangle, max\{\alpha_P, \alpha_Q\}, \aleph) \mid (\langle\rangle, \alpha_P, \aleph) \in \mathcal{E}_T[\![P]\!] \\
&\quad\ \wedge\ (\langle\rangle, \alpha_Q, \aleph) \in \mathcal{E}_T[\![Q]\!]\} \\
&\quad\ \cup \{(s, \alpha, \aleph) \mid s \neq \langle\rangle\ \wedge\ (s, \alpha, \aleph) \in \mathcal{E}_T[\![P]\!] \cup \mathcal{E}_T[\![Q]\!] \\
&\quad\ \wedge\ (\langle\rangle, \aleph \upharpoonright begin(s)) \in Fail(\mathcal{E}_T[\![P]\!]) \cap Fail(\mathcal{E}_T[\![Q]\!])\}) \\
\mathcal{E}_T[\![P \sqcap Q]\!] &= \underline{SUP}(\mathcal{E}_T[\![P]\!] \cup \mathcal{E}_T[\![Q]\!]) \\
\mathcal{E}_T[\![\textstyle\bigsqcap S]\!] &= \underline{SUP}(\textstyle\bigcup S) \qquad (S \neq \emptyset) \\
\mathcal{E}_T[\![P \| Q]\!] &= \underline{SUP}(\{(s, max\{\alpha_P, \alpha_Q\}, \aleph_P \cup \aleph_Q) \mid \\
&\quad\ (s, \alpha_P, \aleph_P) \in \mathcal{E}_T[\![P]\!] \wedge (s, \alpha_Q, \aleph_Q) \in \mathcal{E}_T[\![Q]\!]\}) \\
\mathcal{E}_T[\![P\ {}_X\|_Y Q]\!] &= \underline{SUP}(\{(s, max\{\alpha_P, \alpha_Q\}, \aleph_P \cup \aleph_Q \cup \aleph_Z) \mid \\
&\quad\ \exists (s_P, \alpha_P, \aleph_P) \in \mathcal{E}_T[\![P]\!], (s_Q, \alpha_Q, \aleph_Q) \in \mathcal{E}_T[\![Q]\!] \\
&\quad\ \text{ with } \Sigma(\aleph_P) \subseteq X \text{ and } \Sigma(\aleph_Q) \subseteq Y \text{ such that} \\
&\quad\ s \in (s_P\ {}_X\|_Y s_Q) \wedge \Sigma(\aleph_Z) \subseteq (\Sigma - (X \cup Y))\}) \\
&\quad\ \text{where } v\ {}_X\|_Y w = \\
&\quad\ \{s \in (T\Sigma)^*_\leq \,|\, s \upharpoonright (X \cup Y) = s\ \wedge\ s \upharpoonright X = v \wedge s \upharpoonright Y = w\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{E}_T[\![P \,|||\, Q]\!] &= \underline{SUP}(\{(s, max\{\alpha_P, \alpha_Q\}, \aleph) \mid \exists (u, \alpha_P, \aleph) \in \mathcal{E}_T[\![P]\!] \\
&\quad \wedge\ (v, \alpha_Q, \aleph) \in \mathcal{E}_T[\![Q]\!] \text{ such that } s \in Tmerge(u, v)\}) \\
\mathcal{E}_T[\![P; Q]\!] &= CL_{\cong}(\underline{SUP}(\{(s, \alpha, \aleph) \mid \surd \notin \Sigma(s)\ \wedge\ \forall I \in TINT \\
&\quad (s, \alpha, \aleph \cup (I \times \{\surd\})) \in \mathcal{E}_T[\![P]\!]\} \\
&\quad \cup \{(s.(w + t), \alpha + t, \aleph_1 \cup (\aleph_2 + t)) \mid\ \surd \notin \Sigma(s) \\
&\quad \wedge\ end(\aleph_1) \leq t \\
&\quad \wedge\ (s.\langle (t, \surd) \rangle, \aleph_1 \cup ([0, t) \times \{\surd\})) \in Fail(\mathcal{E}_T[\![P]\!]) \\
&\quad \wedge\ (w, \alpha, \aleph_2) \in \mathcal{E}_T[\![Q]\!]\})) \\
\mathcal{E}_T[\![P \setminus X]\!] &= \underline{SUP}(\{s \setminus X, \beta, \aleph) \mid\ \exists \alpha \geq \beta \geq end(s). \\
&\quad (s, \alpha, \aleph \cup ([0, max\{\beta, end(\aleph)\}) \times X)) \in \mathcal{E}_T[\![P]\!]\}) \\
\mathcal{E}_T[\![f^{-1}(P)]\!] &= \{(s, \alpha, \aleph) \mid (f(s), \alpha, f(\aleph)) \in \mathcal{E}_T[\![P]\!]\} \\
\mathcal{E}_T[\![f(P)]\!] &= \underline{SUP}(\{(f(s), \alpha, \aleph) \mid (s, \alpha, f^{-1}(\aleph)) \in \mathcal{E}_T[\![P]\!]\}) \\
\mathcal{E}_T[\![\mu p.F(p)]\!] &= \text{The unique fixed point of the contraction mapping} \\
&\quad \hat{C}(Q) = C(\mathit{WAIT}\,\delta; Q)\text{, where } C \text{ is the mapping on} \\
&\quad TM_{FS} \text{ represented by } F.
\end{aligned}
$$

We now discuss the construction of the above semantics and the assumptions that are implicit in them. Where we discuss the difficulty or otherwise of implementing a particular operator, the reader should bear in mind that CSP is not primarily intended as an implementable language and that it deliberately (in the untimed version as well) contains a number of features which are useful in specification and reasoning but are impractical to implement. Part of the idea here is that one should be able to use the full language at the specification stage, but be forced to be more selective when we refine our specification to an eventual implementation. Thus, in practice, while we are likely to be concerned that some subset of the language (possibly an occam-like one) accurately reflects an implementation, we are unlikely to have this worry about the whole language.

- Note that $\perp$ and $STOP$ have exactly the same traces and refusals, but that, while $STOP$ becomes stable immediately, $\perp$ never becomes stable. These definitions will almost certainly remain unaltered in all semantics for Timed CSP.

- $SKIP$ is immediately stable, and is willing to terminate at any time. In practice a process will probably be 'switched off' as soon as it terminates, which corresponds in a sense to the assumption in the semantics that $SKIP$ is stable as soon as it terminates. In another sense it makes this decision on stability relatively unimportant.

- $WAIT\,t$ behaves like $SKIP$ except that it only becomes stable and able to terminate at time t.

- The prefixing constructs $a \to P$ and $a : A \to P$ are both assumed to take no time to set up (they are immediately stable and willing to commit themselves to communicate) and furthermore assume that each event takes exactly the same deterministic time $\delta > 0$ to complete. Notice that a typical history of one of these processes has three phases, reflected in the refusal component written $\aleph_1 \cup \aleph_2 \cup \aleph_3 + (t + \delta)$, the first when the initial event(s) are on offer, the second when such an event has occurred and is being completed, and the final one being a behaviour of P or a $P(a)$ shifted by an appropriate delay. One will frequently want to change some of the assumptions made here, and we will discuss this issue later.

- The external choice operator described here runs its two arguments together – at their natural speeds – until the environment accepts a choice from one of them, which commits the choice. Thus the process can, on the empty trace, only refuse sets offered by both its arguments and becomes stable when both its arguments do. If the first timed event chosen was possible for both, then the choice may subsequently behave like either – this potential ambiguity explains the use of the $\underline{SUP}$ operator. We have assumed that the operator has taken no time to set up, has the resources to run its arguments in parallel, and does not delay in transmitting their communications to the environment. It seems unlikely that, except perhaps in the case of self-timed circuitry, one would normally expect to implement an unrestricted operator with these characteristics. Either one would severely restrict the class of processes to which $\Box$ can be applied (note the remark about implementing only subsets above), or change one or more of these assumptions.

  We need to place a restriction on the applicability of the general prefixing operator $a : A \to P(a)$ when a is infinite. For it would not in general be true that if all the $\mathcal{E}_T[\![P(a)]\!]$ belonged to TM_{FS} then so does $\mathcal{E}_T[\![a : A \to P(a)]\!]$. Specifically we have to assume that the functions $n(t)$ which exist for all the $\mathcal{E}_T[\![P(a)]\!]$ by axiom ?? are bounded above by some function $m(t)$. This excludes examples such as

$$n : \mathbf{N} \to P_{n+1}$$

where $P_n = (a \rightarrow STOP) \,|||\, \ldots \,|||\, (a \rightarrow STOP)$ (n copies).

- The two nondeterministic choice operators $\sqcap$ and $\bigsqcap$ can behave like any of their arguments. The reason for the $\underline{SUP}$ operators is again the ambiguity this causes. Since these operators are unlikely to play much of a direct part in an implementation, there are really no operational assumptions here.

  As in the case of general choice, we have to restrict the application of $\bigsqcap$ to sets of processes where the number of events possible up to any given time is uniformly bounded, once again to protect axiom ??.

- The two synchronised parallel operators $\|$ and ${}_X\|_Y$ are closely related. The first expects its arguments to synchronise on all communications, which means that it can always refuse any communication that either refuses. The second constrains its arguments only to communicate in the sets X and Y respectively, and makes them synchronise on all communications in $X \cap Y$. The resulting process can thus always refuse anything outside $X \cap Y$, can refuse anything in X that its left-hand argument can, and can refuse anything in Y that its right-hand one can. Notice that $\|$ means the same as ${}_\Sigma\|_\Sigma$. One again we are assuming that the operation takes no time to set up, and we are assuming that there are enough resources to run each argument at its natural speed – i.e., this is an operator which runs its arguments genuinely in parallel rather than timeslicing them on a single processor.

- The interleaving parallel operator $\|$ also runs its arguments at their natural speeds with no setup time. This time, however, there is no synchronisation between the processes.

- The sequential composition of two processes behaves like the first them until it terminates (by communicating $\surd$) and then starts up the second. In this semantics we assume that this operation takes no time to set up and, more controversially, that the hand-over happens instantly.

  The first component of the definition takes account of the behaviours of P in $P;Q$ which have not terminated, have not been prepared to terminate so far, and have the abililty to refuse to terminate indefinitely. These are behaviours of $P;Q$. The reason why this part of the definition includes the condition that the process should continue to be able to terminate is to get the stability value right: if we had a behaviour of the form $(s, \alpha, \aleph)$ with $(s, \aleph \cup [0, end(s, \aleph)) \times \{\surd\}) \in Fail(P)$, then we would know that $(s, \aleph) \in \mathit{Fail}(P;Q)$ but, since it is possible that $\surd$ might become available before stability, the stability value α might actually be an over-estimate of that of the failure in $P;Q$. There are two points one should note about this, first that if the failure $(s, \aleph)$ is excluded from this clause because of this indefinite refusal requirement, then it will be included in the second component with $(w, \aleph_2) = (\langle\rangle, \emptyset)$. Secondly, if a process is stable and refusing $\surd$, then it will go on refusing it.

The second component deals with the case where P has terminated and Q has started. Note that P must have refused to terminate up to the moment when it did. The $\underline{\mathit{SUP}}$ operator is once again present to deal with ambiguity in the ways in which failures can be put together. We need the $CL_{\cong}$ operator to deal with a slightly uncomfortable side-effect of our assumption that the hand-over takes no time. For it is possible in our model that P might terminate at the same moment as it is performing some communication, a say, and that Q might itself communicate, say b the same moment when it is started up. The net effect is that $P;Q$ might perform two communications, one from each argument, at the same time. In order to satisfy axiom ? we must include the trace with them reordered with b before a.

- The definition of the hiding operator is one of the shortest – in contrast to the usual situation with untimed CSP – and yet it is actually extremely subtle. The ideas behind the hiding definition are similar in some ways to those behind sequential composition, where the first $\surd$ is a hidden event. Recall the postulate made earlier that hidden events happen as soon as they can. This means that any behaviour of P which could not have been extended though its length by the refusal of the whole of X cannot be a behaviour P within $P \setminus X$, since P would actually have accepted such an event, on offer continuously from the environment, and so not have reached this point. We have the same problem as in sequential composition with processes which are unable to continue refusing X until they are stable. This is the reason for using $end(s) \leq \beta \leq \alpha$ with

  $$(s, \alpha, \aleph \cup ([0, max\{\beta, end(\aleph)\}) \times X) \in \mathcal{E}_T[\![P]\!]$$

  Consider, for example, the process

  $$(\bot \Box b \to STOP) \setminus \{b\}$$

which becomes stable at time δ, even though $(\bot \Box b \rightarrow STOP)$ never becomes stable on the empty trace. The $(s \setminus X, \beta, X)$ recorded in the definition are just timed failures of $P \setminus X$ together with a time β which the process can reach without previously having become stable. (Note that, in the case where P becomes stable still refusing all of X, β might be less than $end(\aleph)$.)

We are again assuming that the hiding operator requires no time to set up and imposes no time overhead on the running of a process.

- The inverse image of a process P under a function f can perform an event a whenever P could have performed $f(a)$. The important points to note about this operator are that there may be several different events mapping to the same image $f(a)$, but that each behaviour of $f^{-1}(P)$ results from a unique one of P.

- On the other had the direct image operator can perform $f(a)$ whenever P could have performed a. This is, in some sense, a more obvious relationship between events but, in the case where f is not injective, can map many behaviours of P onto a single one of $f(P)$. Note that $f(P)$ can only refuse an event a when all of the events which map to a under f are refused. Again the ambiguity requires the use of the $\underline{SUP}$ operators.

- All the operators above are nonexpanding in the metric space (for the reasons discussed earlier). Thus any function we can define by combining them is also nonexpanding. It follows that, when composed with the contraction mapping sending Q to $WAIT\,\delta; Q$, it gives a contraction and hence has a unique fixed point. We should note that the recursive construct is itself nonexpanding, since if $F(P,Q)$ is a contraction in its first argument and nonexpanding in its second then $\mu.P.F(P,Q)$, considered as a function of Q, is also nonexpanding. (The proof of this may be found, for example, in [R82].) The assumption implicit in the definition given here is that making a recursive call takes time δ deterministically. This is another assumption that one might very will wish to alter, and which will be discussed later.

  It is often useful to use several, or even infinite vectors of, processes defined by mutual recursion. We have not included this possibility explicitly in our syntax simply because it is hard to give a reasonably concise, but sufficiently general, description of their syntax. Nevertheless it is easy to give semantics to mutual recursions (and actually rather more important to than in the untimed cases where, without details of timing, one can simulate arbitrary mutual recursions as single ones). Given a mutual recursive definition of the form $\underline{P} \Leftarrow F(\underline{P})$, where $\underline{P}$ is a vector of process variables and F represents the same type of vector of process terms involving them, the standard semantics would associate $\underline{P}$ with the unique fixed point of the contraction mapping on the product space whose λ-component is the semantic mapping associated with the λ-component of F, except that δ-delay is put on each recursive call of any P_μ as above. Of course this would be subject to alteration of assumptions about timing just as in the single recursion case.

Broadly speaking, the standard semantics assume that the completion of all events and the unwinding of any recursion take the same non-zero time δ, and that otherwise each of the operators (i) consumes no time itself and (ii) treats its operands, at each moment, like the corresponding untimed CSP operator treat its 'in the large'. Each of the operators preserves the axioms of TM_{FS}, and is monotone with respect to the nondeterminism order $\sqsubseteq$. In relation to the various discussions we had earlier when constructing TM_{FS}, we should perhaps now note the way in which the hiding operator essentially uses the fact that refusals are recorded thoughout a trace. The reader might wish now to go back and re-examine some of the earlier examples in the context of the semantics we have now defined.