Analysing TM_{FS}: a Study of Nondeterminism in Real-Time Concurrency

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

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

Abstract. It is well-known that nondeterminism is a consequence of concurrent computation. Modelling it correctly is one of the chief requirements of any theory. In this paper we study how nondeterminism arises in real-time concurrency. The vehicle used is TM_{FS}, the timed failures/stability model for CSP. We see both how this model is able to capture a wide range of subtle behaviours which have no direct untimed analogues, and how the axioms of the model can be justified by a detailed study of nondeterminism.

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 provides a detailed study of 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 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

[1]The authors gratefully acknowledge that the work reported in this paper was supported by the U.S. Office of Naval Research under grant N00014-87-G-0242. Auxiliary support was provided by the ESPRIT BRA SPEC project.

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 there should be 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 [RR86,RR87,Re88,Re90]. 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 central model in this hierarchy is the *timed failures/stability model*, TM_{FS}. 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. Versions of this model appeared in a number of references over the past few years, for example [RR87,Re90]. The process of development has taken the form of slight tightening of the axioms. This paper, together with a companion paper [RR90] presents what we consider to be the definitive version of the model. The other paper gives an extensive explanation of the philosophy of the model and its construction, and presents a detailed analysis of what it means to be a semantics for Timed CSP over it. The present paper concentrates more on the construction of the model, a detailed analysis of the types of nondeterminism it contains, and a justification in terms of this of why we can now claim to have the definitive model.

Comparable work on models of real-time concurrency may be found in [BG87,Z86], though generally the other authors have been concerned with a timed theory alone rather than trying to set up links with untimed ones. It might also be interesting to contrast their generally rather weaker axiomatisations with ours, particularly in the light of the work in this paper.

2 The construction of the model

2.1 The syntax of Timed CSP

It is impossible to discuss our models properly without a knowledge of Timed CSP notation, the assumptions which underlie it, and the way it is given a semantics in the model. In this section we introduce the notation and summarise the main ideas behind it. The complete standard semantics for Timed CSP is given in a later section. All of this, in particular the construction of the standard and other semantics, is discussed in more detail in the companion paper [RR90].

Timed CSP uses essentially the same notation as untimed CSP, extended as necessary to handle timing ideas. The version of untimed CSP we use is essentially that of [BHR84], [BR85] and [H85]. 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 $\bigsqcap S$ 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. But in fact, under the standard semantics, 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. $SKIP$ means the same as $WAIT\,0$. The reader should consult [S90] for details of how further operators such as timeout, tireout and interrupt can be defined in terms of existing ones. Therefore 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 and f over the set of functions from Σ to Σ. $P(a)$ 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).

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

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. 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.

(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.

(4) **Hiding and termination.** We wish $(a \rightarrow P)$ 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 $P = a \rightarrow P$,we then wish $P \setminus a = \perp$. However, consider $P = a \rightarrow (b \rightarrow STOP)$ (the process that is willing to engage in a at any time ≥ 0 and then to offer a b). What do we wish $P \setminus a$ to denote – in other words, at what time can we rely on the b becoming available?

We have already discussed the principle that, in CSP, observability is equated with external control. Given the process $a \rightarrow P$ 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. The process $(a \rightarrow (b \rightarrow STOP)) \setminus a$ above thus executes its hidden a immediately, and thus the b becomes available after the time taken to complete a.

(5) **Stability and the treatment of divergence.** As indicated earlier, stability plays the same role 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 qualitatevely 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 external communication. 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 $a \rightarrow P$, 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.)

Notice that if we have observed of a process (a) that it has become stable and (b) that it is refusing some set of events, then we know it will refuse this set for ever (until some event not in the set is accepted so the process can get out of the stable state).

For various reasons the timed models are not forced to model divergence (lack of finite stability) as severely as the untimed ones. In particular it is not necessary to treat any process which can diverge as one which is useless.

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

Although it is possible to place a natural refinement order on timed models (see later) it turns out that this order neither has a least element nor directed limits – both necessary for the cpo approach to fixed points to work. Fortunately, though, time gives a very natural measure for making the models into complete metric spaces: two processes are at distance 2^{-t} if t is the supremum of all times for which it is impossible to distinguish them.

Unlike the case with metrics over untimed models, it turns out that all CSP recursions over the timed models have a natural 'lack of clairvoyance' property which means that they can easily and naturally be made into contraction mappings.

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

The above discussion summarises the philosophy and framework within which we model timed CSP. The timed failures/stability model TM_{FS}, which we are analysing in this paper, models each process as the set of all observations of the form $(s, \alpha, \aleph)$ which can be made of it. Here s is a *timed trace* recording a finite sequence of events which have occurred and the times when they happened; $\aleph$ is a *timed refusal* which records the events which have been observed to be refused by the process up to some time. For reasons we have discussed extensively elsewhere, $\aleph$ records not only what is refused after the end of s but also what was refused during s. α (which may be ∞) is the supremum of all times when the process might become stable given the observation of $(s, \aleph)$. A process which diverges has stability value ∞.

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 $X \subseteq \Sigma$, $s \,|\!\setminus X$ is the maximal subsequence w of s such that $\Sigma(w) \subseteq X$; $s \setminus X = s \,|\!\setminus (\Sigma - X)$. If $t \in [0, \infty)$, $s \,|\!\setminus t$ is the subsequence of s consisting of all those events which occur no later than t, while $s \,\|\!\setminus t$ 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.[2]

If $s, w \in (T\Sigma)^*_{\leq}$, $Tmerge(s, w)$ 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, $Tmerge(s, w)$ 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, $Traces(P)$ will denote the set of all timed traces which are possible for P.

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

[2] 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.

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.

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

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

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

Each process will be modelled as a set of triples $(s, \alpha, \aleph)$, with $s \in (T\Sigma)^*_{\leq}$, $\alpha \in \mathit{TSTAB}$ and $\aleph \in \mathit{RSET}$. 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.

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

The evaluation domain TM_{FS}

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

1. $\langle\rangle \in \mathit{Traces}(S)$

2. $(s.w, \aleph) \in \mathit{Fail}(S) \ \Rightarrow\ (s, \aleph \,|\!\backslash\, \mathit{begin}(w)) \in \mathit{Fail}(S)$

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

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

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

6. $(s, \alpha, \aleph) \in S \Rightarrow end(s) \leq \alpha$

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

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

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

11. $(s, \alpha, \aleph) \in S \wedge I \in TINT$ 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. Further intuition can be gained about the axioms from the detailed mathematical use we will make of them later.

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 $(s, \aleph \upharpoonright begin(w))$ 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 dependence 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 $\alpha + t' - t$. This is true, for example, whenever $t' \geq t$.

   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

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

   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 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 up to time t_2 (i.e., an arbitrarily large finite time) – 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 refuseable 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 $(s|\!\backslash t', \aleph'|\!\backslash t') \in \mathit{Fail}(S)$ whereas in this last one we vary this to $(s\|\!\backslash t', \aleph'|\!\backslash t') \in \mathit{Fail}(S)$. 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 2.5 below another possible 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 the 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

$$P = (a \rightarrow STOP \Box WAIT\ \ 1); b \rightarrow STOP$$

will, in the standard semantics, become stable on the pair $(\langle\rangle, [0,1) \times (\Sigma - \{a\}))$ at time 1; however $\langle(1,a)\rangle \in \mathit{Traces}(P)$ (as, indeed, is forced by the last clause of axiom 10) 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 [Re88,Re90] 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 satisfying axiom 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 is demonstrated in [RR90].

2.5 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. Define a t_2-complete behaviour[3] to be one satisfying the conditions on the right hand side of the implication in axiom 10. In other words it is a behaviour which, up to time t_2, gives a convincing explanation of what the process might have refused in total up to time t_2. This axiom together with axiom 9 says that, for any time t, 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 [Re90]:

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

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

[3] 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.

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 $(s.w, \aleph \cup \aleph')$ 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

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

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

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

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

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 rather 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 $Q_n = ((a \rightarrow STOP) \Box \mathit{WAIT}\, t_n); STOP$. Under the standard semantics, the process Q_n offers a until time t_n, whereupon the $\mathit{WAIT}\, t_n$ 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 refuseable when withdrawn. Now consider the processes $P_n = \bigsqcap\{Q_m \mid m \geq n\}$. 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, for arbitrary t, 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 $(s, \aleph \upharpoonright t) \in \mathit{Fail}(P)$ for all t, $\alpha = \mathit{inf}\{\alpha' \mid (s, \alpha', \aleph^* \upharpoonright t) \in P\}$ and the same conditions applied as for a t-complete behaviour, namely

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

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

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

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 $\aleph^* \upharpoonright t$ for $t \in [0, \infty)$ belongs to $RSET$. Since $\aleph \subseteq \aleph^* \upharpoonright T + n \subseteq \aleph_n$, it follows (using axiom 9) that there is some γ_n with $\beta_n \leq \gamma_n \leq \alpha$ such that $(s, \gamma_n, \aleph^* \upharpoonright T + n) \in P$. 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 $(s \|\!\upharpoonright t.\langle(t, a)\rangle, \aleph_n \upharpoonright t)$ and hence $(s \|\!\upharpoonright t.\langle(t, a)\rangle, \aleph^* \upharpoonright t)$ 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 $(s, \alpha, \aleph)$.

3 The semantics of Timed CSP

The construction of TM_{FS} was strongly influenced by our intuition about the semantics of Timed CSP. In this section we show how the standard semantics are actually defined. They are called the 'standard' semantics because, while they are in some sense the most straightforward and natural, it would be wrong to prescribe a single real-time semantics in a world where, for perfectly good reasons, different implementations had different timing characteristics. This issue is covered in detail in [RR90].

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 \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\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]\!]\})
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{E}_T[\![P\ {}_X\|_Y Q]\!] = {} & \underline{SUP}(\{(s, max\{\alpha_P, \alpha_Q\}, \aleph_P \cup \aleph_Q \cup \aleph_Z) \mid \\
& \exists (s_P, \alpha_P, \aleph_P) \in \mathcal{E}_T[\![P]\!], (s_Q, \alpha_Q, \aleph_Q) \in \mathcal{E}_T[\![Q]\!] \\
& \text{ with } \Sigma(\aleph_P) \subseteq X \text{ and } \Sigma(\aleph_Q) \subseteq Y \text{ such that} \\
& s \in (s_P\ {}_X\|_Y s_Q) \wedge \Sigma(\aleph_Z) \subseteq (\Sigma - (X \cup Y))\}) \\
& \text{where } v\ {}_X\|_Y w = \\
& \{s \in (T\Sigma)^*_\leq \mid s \upharpoonright (X \cup Y) = s\ \wedge\ s \upharpoonright X = v\ \wedge\ s \upharpoonright Y = w\} \\
\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]\!] \\
& \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 \\
& (s, \alpha, \aleph \cup (I \times \{\surd\})) \in \mathcal{E}_T[\![P]\!]\} \\
& \cup \{(s.(w + t), \alpha + t, \aleph_1 \cup (\aleph_2 + t)) \mid\ \surd \notin \Sigma(s) \\
& \wedge\ end(\aleph_1) \leq t \\
& \wedge\ (s.\langle (t, \surd)\rangle, \aleph_1 \cup ([0, t) \times \{\surd\})) \in Fail(\mathcal{E}_T[\![P]\!]) \\
& \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). \\
& (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} \\
& \hat{C}(Q) = C(\mathit{WAIT}\ \delta; Q), \text{ where } C \text{ is the mapping on} \\
& TM_{FS} \text{ represented by } F.
\end{aligned}
$$

In order to preserve axiom 4 it is necessary to place a restriction on the use of the infinitary operators $\sqcap S$ and $x : A \to P(x)$. This is that there is a single function $n(t)$ which works (in axiom 4) for all of the processes being combined.

Provided this restriction is observed, all of the above operators preserve the axioms and are nonexpanding with respect to the metric. The first of these properties in particular requires a considerable amount of tedious checking: for a large number of the most interesting cases (for example hiding) the reader is referred to [Re88]. The fact that that reference uses a slightly weaker axiomatisation is irrelevant except when considering axiom 10 of this paper (replacing axioms 4 and 11 of the earlier work). However, just as occurred with the old axiom 4, a complete behaviour of any combination of processes is always constructed from suitable complete behaviours of an argument or arguments.

4 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 $\bigsqcap imp(P) = P$.) This idea has often been very important, notably in the development of the untimed infinite traces model [Ro88].

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 [Bl90] has written on the phenomenon of processes being determined by their (pre-)deterministic implementations, 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. The rest of this section investigates the question of a corresponding completeness result for Timed CSP, since this provides a valuable focus for our study of nondeterminism.

4.1 A taxonomy of nondeterminism

Unfortunately, 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. The reader may note that each of them relates, in one way or another, to singular behaviour at one instant.

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
$$((a \to STOP)\Box SKIP); (a \to a \to STOP)$$
   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 refuseable 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.

4.2 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

$$(((a \to STOP) \Box WAIT\,1); STOP) \,|||\, (b \to STOP)$$

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

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

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 $(s \upharpoonleft t, \aleph) \in Fail(P)$ is such that $end(\aleph) \le t$). Claim that $\aleph \subseteq \aleph^*$. If not, since the result is trivial if $t = 0$, 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 $(s, \aleph^*)$ for Q. 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 follows immediately from the relevant defintions. 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 $t > \alpha$, $t \geq \mathit{end}(\aleph)$ (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

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

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 $t = 0$.

4.3 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 Lemma 2.)

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 $s \stackrel{t'-t}{\Longrightarrow}_1 v.\langle(t', a)\rangle.(w + (t' - t))$ $(= s')$ and observe that any transient added after s must be added after s', shifted through $t' - t$. The subscript 1 here denotes the 'one-step' nature of the relation. (The time of the shifted transient is guaranteed to be in range by axiom 8.) If $t' > t$, we will write $s \stackrel{t'-t}{\Longleftrightarrow}_1 s'$ and observe that if $s_1 \stackrel{t}{\Longleftrightarrow}_1 s_2$ then (i) $s_2 \stackrel{-t}{\Longleftrightarrow}_1 s_1$ 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 $s_1 \stackrel{t}{\Longrightarrow}_1 s_2$, 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 $s_1' \stackrel{t}{\Longrightarrow}_1 s_2'$.

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 $s \stackrel{0}{\Longrightarrow} s'$ and $s \stackrel{0}{\Longleftrightarrow} s'$.
- If $s \stackrel{t}{\Longrightarrow}_1 s'$ and $s' \stackrel{t'}{\Longrightarrow} s''$, then $s \stackrel{t+t'}{\Longrightarrow} s''$.
- If $s \stackrel{t}{\Longleftrightarrow}_1 s'$ and $s' \stackrel{t'}{\Longleftrightarrow} s''$, then $s \stackrel{t+t'}{\Longleftrightarrow} s''$.
- $\stackrel{t}{\Longleftrightarrow}$ and $\stackrel{t}{\Longrightarrow}$ 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 $s \stackrel{t}{\Longleftrightarrow} s'$ implies $s' \stackrel{-t}{\Longleftrightarrow} s$ 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 $s \stackrel{t}{\Longrightarrow} s'$ 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 $s \stackrel{t}{\Longleftrightarrow} s'$ 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 sorts 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:

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

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

4.4 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

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

cannot refuse a on its first step but may, when it performs $(0, a)$, instantly lose the ability to refuse b[4]. 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 refuseable 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., unrefuseable 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 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 $(s, \alpha, \aleph|\!\backslash t) \in P$ 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 $(s.\langle(t, a), (t, b)\rangle, \aleph|\!\backslash t) \in Fail(P)$ and hence, by axioms 2 and 3, $(s.\langle(t, b)\rangle, \aleph|\!\backslash t) \in Fail(P)$. Axiom 7 then tells us that $(t', b) \notin \aleph$ for any $\alpha \leq t' < t$. 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

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

[4]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.

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

4.5 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-deterministic ones and which, whenever P is quasi-deterministic, Q is almost deterministic with communications at time 0, $(s, \alpha) \in Stab(P)$ and $t \in [end(s), \alpha]$ $(t = end(s) \Rightarrow s = \langle\,\rangle)$, $P \xrightarrow{s,t} Q$ 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.[5]

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.

Recall that general nondeterministic choice was defined

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

We required 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 $n^*(t)$.

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 $\sqcap(imp(P))$ 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 $\sqcap(imp(P)) = P$. $\Box$

If S is a set of processes Q such that $P \sqsubseteq Q$ (for fixed P), then it is easy to show that $P \sqsubseteq \sqcap S$. 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 $\beta > t$.

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

[5] 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.

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 $(s, \alpha, \aleph^* \mid\!\backslash (\mathit{begin}(w))) \in P$ 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 $\aleph^* \mid\!\backslash t = \aleph \mid\!\backslash t$.

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 $\aleph'_n = \aleph \mid\!\backslash \alpha \cup \aleph_n \uparrow [\alpha, \infty)$, 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 $\aleph^* \mid\!\backslash \alpha = \aleph \mid\!\backslash \alpha$, $\aleph \subseteq \aleph^*$ 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 – $\aleph^* \mid\!\backslash t \in \mathit{RSET}$ – we require of complete infinite behaviours.) If we can show that $(s, \aleph^*)$ is a 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 $(s.\langle (t', a) \rangle, \aleph'_n \mid\!\backslash t')$ belongs to $\mathit{Fail}(P)$ by completeness of $(s, \aleph'_n)$, and hence so does $(s.\langle (t', a) \rangle, \aleph^* \mid\!\backslash t')$ by axiom 9.

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

Assuming we can indeed establish the conjecture, it might be worthwhile broadening the definition of quasi-deterministic processes to encompass instant enabling. It is possible that one might be able to prove that for any *CSP-definable* process the result stands without assuming the lack of instant enabling assumption. On the other hand one could argue that by this result and the understanding already gained of instant enabling, we would already have gained sufficient insight into nondeterminism in TM_{FS} and confidence in the completeness of the axioms.

5 Conclusions

In this paper we have simultaneously provided a study of the detailed structure of our model TM_{FS} and of the types of nondeterminism which it can model. We provided individual explanations of its various axioms and also showed how these axioms fit together by proving a series of Lemmas and other useful results about the model. Finally, we conjectured a 'completeness' result which would allow us to argue that, at least in some sense, our axioms were definitive.

We hope that the rather detailed work in this paper will provide useful insight, and a source of potential hard cases, to those engaged in more practical work using Timed CSP.

We conclude by giving a brief survey of the literature of Timed CSP which defines the current state of the subject. As we have already stated, this paper is complementary to another one [RR90] which describes our philosophy and the rationale behind the congruence used in more detail, as well as defining just what it means to be a semantics for Timed CSP over TM_{FS}. The overall hierarchy of models is described in [Re88] and [Re90]. Proof systems derived from the semantic models and operators are described in [S90] and [DS90]. In [J90] it has been shown how a temporal logic compatible with timed CSP can be developed. The extremely useful technique of *timewise refinement* for lifting process developments and specifications from the untimed failures/divergence model to the timed models is described in [S90]. Significant case-studies in the application of Timed CSP can be found in [Re88], [J89], [S90], [Se90]and [St90].

Acknowledgements

We are very grateful to all those working in Oxford on Timed CSP for their help and enthusiasm. In particular, Steve Schneider has been of great assistance while we have been writing this paper.

6 References

[Bl90] S.R. Blamey, *The soundness and completeness of axioms for CSP processes*, Topology, Category Theory and Computer Science, (Oxford University Press, 1990), G.M. Reed, A.W. Roscoe, R.F. Wachter, editors.

[BG87] A. Boucher and R. Gerth, *A timed failures model for extended communicating sequential processes*, ICALP'87,Springer LNCS

[BHR84] S.D. Brookes, C.A.R. Hoare and A.W. Roscoe, *A theory of communicating sequential processes,* JACM 31 (1894), 560-599.

[BR85] S.D. Brookes and A.W. Roscoe, *An improved failures model for communicating processes,* Proceedings of the Pittsburgh Seminar on Concurrency, Springer LNCS 197 (1985).

[DS90] J.W. Davies and S.A. Schneider, *Factorising proofs in Timed CSP*, Proceedings of the Fifth Workshop on the Mathematical Foundations of Programming Language Semantics (April, 1989), LNCS 442 (1990), 129-159.

[H85] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice-Hall International, 1985.

[J89] D.M. Jackson, *The sepcification of aircraft engine control software using Timed CSP*, Oxford University M.Sc. differtation, 1989.

[J90] D.M. Jackson, *Specifying Timed Communicating Sequential Processes using temporal logic*, Technical Report PRG-90-5, Oxford university Computing Laboratory 1990.

[Re88] G.M. Reed, *A uniform mathematical theory for real-time distributed computing*, Oxford University D.Phil thesis 1988.

[Re90] G.M. Reed, *A hierarchy of models for real-time distributed computing*, Proceedings of the Fifth Workshop on the Mathematical Foundations of Programming Language Semantics (April,1989), LNCS 442 (1990), 80-128.

[RR86] G.M. Reed and A.W. Roscoe, *A timed model for communicating sequential processes*, Proceedings of ICALP'86, Springer LNCS 226 (1986), 314-323; Theoretical Computer Science 58, 249-261.

[RR87] G.M. Reed and A.W. Roscoe, *Metric spaces as models for real-time concurrency*, Proceedings of the Third Workshop on the Mathematical Foundations of Programming Language Semantics (April, 1987), LNCS 298 (1988), 331-343.

[RR90] G.M. Reed and A.W. Roscoe, *The Timed Failures-Stability Model for CSP*, submitted for publication.

[Ro88] A.W. Roscoe, *Unbounded nondeterminism in CSP*, in 'Two papers on CSP', Technical monograph PRG-67, Oxford University Computing Laboratory 1988.

[S90] S.A. Schneider, *Correctness and communication in real-time systems*, Oxford University D.Phil. thesis 1990. Published as technical monograph PRG-88.

[Sc90] B. Scattergood, *An application of Timed CSP to robot control software*, Oxford University MSc dissertation 1990.

[St90] R. Stamper, *The specification of AGV control software using Timed CSP*, Oxford University MSc dissertation 1990.

[Z86] A.E. Zwarico, *A formal model of real-time computing*, University of Pennsylvania technical report (1986)