Theoretical Computer Science 58 (1988) 249-261
North-Holland

A TIMED MODEL FOR COMMUNICATING SEQUENTIAL PROCESSES

G.M. REED and A.W. ROSCOE*
Programming Research Group, Oxford University, Oxford OX1 2JD, United Kingdom

1. Introduction

The parallel language CSP [9], an earlier version of which was described in [7], has become a major tool for the analysis of structuring methods and proof systems involving parallelism. The significance of CSP is in the elegance by which a few simply stated constructs (e.g., sequential and parallel composition, nondeterministic choice, concealment, and recursion) lead to a language capable of expressing the full complexity of distributed computing. The difficulty in achieving satisfactory semantic models containing these constructs has been in providing an adequate treatment of nondeterminism, deadlock, and divergence. Fortunately, as a result of an evolutionay development in [8], [10], [15], [1], [14], [2], and [4] we now have several such models.

The purpose of this paper is to report the development of the first real-time models of CSP to be compatible with the properties and proof systems of the above-mentioned untimed models. Our objective in this development is the construction of a timed CSP model which satisfies the following:

(1) *Continuous with respect to time.* The time domain should consist of all non-negative real numbers, and there should be no lower bound on the time difference between consecutive observable events from two processes operating asynchronously in parallel.

(2) *Realistic.* A given process should engage in only finitely many events in a bounded period of time.

(3) *Continuous and distributive with respect to semantic operators.* All semantic operators should be continuous, and all the basic operators as defined in [2], except recursion, should distribute over nondeterministic choice.

(4) *Verifiable design.* The model should provide a basis for the definition, specification, and verification of time critical processes with an adequate treatment of nondeterminism, which assists in avoidance of deadlock and divergence.

* The authors gratefully acknowledge that this research was funded under Grant N0014-85-G-0134 from the U.S. Office of Naval Research.

0304-3975/88/$3.50 © 1988, Elsevier Science Publishers B.V. (North-Holland)

(5) *Compatible.* The model and its associated proof systems should be a "natural" extension of untimed models and proof systems, and the model should contain the timed equivalents of those CSP constructs modelled in [2, 4].

A crucial element in achieving a CSP model satisfying the above requirements proved to be in making the subtle distinction between deadlock and divergence in timed processes. As indicated in Section 7, previous constructions of timed CSP models have either relied on unrealistic (in the sense defined above) processes to make this distinction [11], or else by design have not distinguished between these two concepts [12]. In this paper, we present the resolution of this issue via concentration on a formal description of the *Timed Stability Model.* This model (with appropriate restrictions such as those utilised for the Deterministic Trace Model in [8] allows a complete treatment of deadlock and divergence for deterministic timed processes. In [17], we describe our *Timed Failures-Stability Model,* which meets all the above requirements over the full range of CSP processes. At a later date, we shall present a hierarchy of timed models analogous to the existing range of untimed models.

The paper is organised as follows: The second section contains a brief review of CSP. The third section discusses the rationale for basing our semantic domains on a complete metric structure. The fourth section discusses our timing assumptions. The fifth section describes the Timed Stability Model. The sixth section illustrates the applicability of the Timed Stability Model to the definition, specification, and verification of time-critical processes. Finally, the seventh section contains our conclusions and a comparison with other work.

2. A review of CSP

Throughout this paper, we will assume the background material of [2, 4, 9]. A breif reminder is given below.

Nondeterministic processes are those which make internal progress leading to arbitrary choices which cannot be observed from the outside; such choices serve to reduce the range of possible future behaviours of the processes. *Deadlock* occurs in a distributed system when each component process is prepared to engage in some further action; but since the processes involved cannot agree on what the next action will be, nothing further can happen. *Divergence* occurs when a process is engaged in an infinite unbroken sequence of internal actions invisible to the environment, and as a result leaves its environment waiting eternally for a response.

A CSP process communicates with its environment in some alphabet Σ of atomic communications or "events". Communications require the co-operation of all participants and are considered to be instantaneous. At each point in the history of a process, there is a finite sequence of elements of the alphabet which the process may have been "observed" to communicate with its environment. Such sequences are called *traces* of the process, and all our knowledge about a given process is limited to statements about such traces and about possibilities of future behaviour of the process after a particular trace has been observed. As indicated, there are now a variety of CSP models and associated proof systems.

We shall use essentially the same abstract syntax for CSP given in [2, 4] with the addition of a process WAIT t for each $t \geq 0$: the process that terminates successfully after t units of time, and $\perp$, the diverging process which engages in no event visible to the environment. We use P, Q, R to range over syntactic processes; a, b over the alphabet Σ, X, Y over subsets of Σ; f over the set of finite-to-one functions from Σ to Σ; and F over "appropriate" compositions of our syntactic operators.

$$P ::= \perp \mid \text{STOP} \mid \text{SKIP} \mid \text{WAIT } t \mid (a \to P) \mid P \Box Q \mid P \sqcap Q \mid \| Q \mid$$

$$P_X \|_Y Q \mid P \| Q \mid P; Q \mid P \backslash X \mid f^{-1}(P) \mid f(P) \mid \mu Q.F(Q)$$

We will generally write $P \backslash a$ rather than $P \backslash \{a\}$ when hiding the single event a.

We assume (from [2], for example) that the reader is familiar with recursive processes $\mu P.F(P)$, which are defined as the least fixed point of continuous mappings on a semantic domain structured as a complete partial order. We also assume familiarity with the concept of such recursive processes defined as the unique fixed point of contraction mappings on a semantic domain structured as a complete metric space (see [13, 15, 5, 6, 16]).

3. Time and topology

In this paper, we propose a theory of communicating sequential processes based on an underlying topological structure. Most of the above-mentioned untimed CSP models have been based on domains of complete partial orders, and different models often have incompatible orderings. Our reasons for choosing a topological approach based on domains of complete metric spaces are primarily that

(1) topological embeddings seem a natural method by which to induce a hierarchy on the various models, and

(2) topological domains appear more appropriaate for modelling continuous concepts such as real time.

To illustrate the intuitive appeal of topological limits in the analysis of CSP processes, consider the following example in the Trace Model [8]:

$$Q = b \to Q \qquad\qquad P_0 = Q$$

$$P = a \to P \qquad \forall n \geq 1, \quad P_n = a \to P_{n-1}$$

Recall that, by $P = a \to P$, we mean $P = \mu P.F(P)$ where $F(R) = a \to R$ is an appropriate mapping on our semantic domain. In the Trace Model semantics, we have

$$P_0 = \{\langle\ \rangle, \langle b\rangle, \langle bb\rangle, \langle bbb\rangle, \ldots\},$$

$$P_1 = \{\langle\ \rangle, \langle a\rangle, \langle ab\rangle, \langle abb\rangle, \langle abbb\rangle, \ldots\},$$

$$P_2 = \{\langle\ \rangle, \langle a\rangle, \langle aa\rangle, \langle aab\rangle, \langle aabb\rangle, \langle aabbb\rangle, \ldots\},$$

$$P_3 = \{\langle\ \rangle, \langle a\rangle, \langle aa\rangle, \langle aaa\rangle, \langle aaab\rangle, \langle aaabb\rangle, \langle aaabbb\rangle, \ldots\},$$

$$\vdots$$

$$P = \{\langle\ \rangle, \langle a\rangle, \langle aa\rangle, \langle aaa\rangle, \langle aaaa\rangle, \langle aaaaa\rangle, \langle aaaaaa\rangle, \ldots\}.$$

Clearly, an observer looking at a record of all traces of length $\leqslant n$ cannot distinguish between P and P_n. Hence, it seems intuitive that $\lim_{n\to\infty} P_n = P$. Indeed this is the case under a complete metric structure based on $d(R_1, R_2) = 1/2^n$, where n is the greatest integer such that R_1 and R_2 agree on all traces of length $\leqslant n$. However, based on the standard complete partial order structure under set inclusion for the Trace Model, $\lim_{n\to\infty} P_n$ does not exist. When we move to the timed CSP models, this situation becomes critical. If an observer looking at a record of all traces completed in n units of time cannot distinguish between P_n and P, we would certainly expect $\lim_{n\to\infty} P_n = P$.

Of course, the price paid for intuition in the topological Trace Model is the loss of continuity of the hiding operator and the restriction to guarded recursions to ensure contraction mappings. For example, note that

$$\lim_{n\to\infty} (P_n \backslash a) = Q \neq (P\backslash a) = \{\langle\ \rangle\} = \text{STOP},$$

$\mu P.P$ is undefined.

Surprisingly, we shall see that, as we move to topological timed models, we are able to retain intuition and to avoid such problems.

In constructing the topological timed models, we regard the treatment of divergence as the crucial issue. It would seem useful to distinguish between *deadlock*, a property which we can recognize in a finite period of time and *divergence*, a property which we cannot. In particular we wish $\lim_{n\to\infty}(\text{WAIT } n) = \bot$, where $\bot \neq \text{STOP}$.

In untimed CSP, it is only necessary to know that a given process can or cannot diverge after engaging in a trace s; in the timed models, it is necessary to know (if the process cannot diverge after s) *when* it will again be ready to respond to the environment. This analysis leads us to consider the untimed Divergence Models [15, 1, 14, 4] as providing discrete information for a given trace s ("0" cannot diverge, "∞" can diverge), and our corresponding timed model as providing continuous information ($\alpha \in [0, \infty]$ such that the process is guaranteed to be stable within α time after engaging in s). Our topological models will be based on this notion of *stability*, which is the dual of divergence.

In a forthcoming paper giving our hierarchy of timed CSP models, we shall use the above analysis to present the Timed Stability Model as a natural extension of an untimed, topological *Stability Model*. Due to space limitations here, we present only the former.

4. Timing postulates

Before giving a formal description of the Timed Stability Model, let us first present our basic assumptions about timing in a distributed system.

(1) *A global clock.* We assume that all events recorded by processes within the system relate to a *conceptual* global clock.

(2) *A system delay constant.* As previously indicated, we realistically postulate that a process can engage in only finitely many events in a bounded period of time. The structure of our timed models allows several parameters by which to ensure adherence with this postulate. In the current presentation, for simplicity we assume the existence of a single delay constant δ such that

(a) For each $a \in \Sigma$ and each process P, the process $(a \to P)$ is ready to engage in P only after a delay of time δ from participation in the event a.

(b) A given recursive process is only ready to engage in an observable event after a delay of δ time from making a recursive call.

(3) *Hiding.* We wish $(a \to P)$ to denote the process that is willing at *any* time to engage in the event a and then to behave like the process P. Clearly, if $P = a \to P$, we then wish $P \backslash a = \perp$. However, consider $P = a \to \mathrm{STOP}$ (the process that is willing to engage in a at any time $\geqslant 0$ and then to deadlock). What do we wish $P \backslash a$ to denote? By hiding, we remove external control. Hence, any time a process is willing to engage in an internal action, it is permitted to do so. Thus, we assume that each hidden event has taken place as soon as such event was possible. In the above example, we would wish

$$(a \to \mathrm{STOP}) \backslash a = \mathrm{WAIT}\ \delta ; \mathrm{STOP}$$

Our hiding assumption can be illustrated in terms of "Hoare's vending machines" from [9]. Suppose we have a timed vending machine TVM which is willing to make one transaction by first accepting a 5p coin, then after a wait of 2 seconds, becoming ready for a button to be pushed, which after an additional two seconds will result in a chocolate being offered to the customer.

$$\mathrm{TVM} = (5\mathrm{p} \to (\mathrm{WAIT}\ 2 ; (\mathrm{Push} \to (\mathrm{WAIT}\ 2 ; (\mathrm{Choc} \to \mathrm{STOP}))))).$$

Suppose now, we wish to automate the vending machine by hiding Push. Clearly, we would wish the button to be pushed internally as soon as possible.

$$\begin{aligned} \mathrm{ATVM} &= \mathrm{TVM} \backslash \mathrm{Push} \\ &= (5\mathrm{p} \to (\mathrm{WAIT}\ 2 ; \mathrm{WAIT}\ \delta ; \mathrm{WAIT}\ 2 ; (\mathrm{Choc} \to \mathrm{STOP}))) \\ &= 5\mathrm{p} \to (\mathrm{WAIT}\ 4 + \delta ; \mathrm{Choc} \to \mathrm{STOP}). \end{aligned}$$

In order to model this idea of an event occurring as soon as it becomes available, we shall record not only those times at which events are available, but also those at which they can become available.

(4) *Timed stability.* We shall model a timed CSP process as a specified set of ordered pairs (s, α), where s is a *timed trace* of the process and α is the time at which the process is guaranteed to be stable after engaging in s. If (s, α) is in the process P and $\alpha < \infty$, then the next observable event in the life of the process following s may occur at any time on or after time α at the discretion of the environment, and the set of possible next events must be the same at *all* such times. Clearly no event can become available after α. We think of timed stability as a red light on the outside of a process which goes off when the process can make no more internal progress.

5. The Timed Stability Model

As indicated above, we need to distinguish between the times when a process can communicate an event, and the times at which it can become ready to communicate it. Therefore we need two ways to record each event that might occur: for $a \in \Sigma$, a will denote the communication of a at any time, but we shall use the special notation $\hat{a}$ to denote communication of a the moment it becomes available We shall denote the set $\Sigma \cup \{\hat{a} \mid a \in \Sigma\}$ by $\hat{\Sigma}$; e will be a typical element of $\hat{\Sigma}$. A *timed event* is an ordered pair (t, e), where e is the communication and $t \in [0, \infty)$ is the time at which it occurs. The set $[0, \infty) \times \hat{\Sigma}$ of all timed events is denoted $\mathrm{T}\Sigma$. The set of all *timed traces* is

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

If $s \in (\mathrm{T}\Sigma)^*_{\leq}$, we define $\#s$ to be the length (i.e., the 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, with any ^'s removed). $\tilde{s}$ is the sequence where any ^'s have been removed from the communications in s: thus, if $s = \langle(t, a), (t', \hat{b})\rangle$, $\tilde{s} = \langle(t, a), (t', b)\rangle$.

begin(s) and end(s) are respectively the earliest and latest times of any of the timed events in s. (For completeness we define begin $(\langle\ \rangle) = \infty$ and end$(\langle\ \rangle) = 0$.)

If $X \subseteq \Sigma$, $s \upharpoonright X$ is the maximal subsequence w of s such that $\Sigma(\tilde{w}) \subseteq X$. $s \backslash X = s \upharpoonright (\Sigma - X)$. If $t \in [0, \infty)$, $s \upharpoonright t$ is the subsequence of s consisting of all those events which occur no later than t. If $t \in [-begin(s), \infty)$ and $s = \langle(t_0, e_0), (t_1, e_1), \ldots, (t_n, e_n)\rangle$,

$$s + t = \langle(t_0 + t, e_0), (t_1 + t, e_1), \ldots, (t_n + t, e_n)\rangle.$$

If $s, w \in (\mathrm{T}\Sigma)^*_{\leq}$, Tmerge$(s, w)$ is defined to be the set of all traces in $(\mathrm{T}\Sigma)^*_{\leq}$ obtained by interleaving s and w.

If, $s, w \in (\mathrm{T}\Sigma)^*_{\leq}$, we define $s \cong w$ if and only if s is a permutation of w.

Let $\mathrm{STAB} = [0, \infty] = [0, \infty) \cup \{\infty\}$. This is the set of all "stability values". Whenever $S \subseteq (\mathrm{T}\Sigma)^*_{\leq} \times \mathrm{STAB}$, we define

$$\mathrm{Traces}(S) = \{s \in (\mathrm{T}\Sigma)^*_{\leq} \mid \exists \alpha \in \mathrm{STAB}.(s, \alpha) \in S\},$$

$$\mathrm{CL}_{\cong}(S)=\{(s,\alpha)\mid \exists(w,\alpha)\in S \text{ such that } s\cong w\},$$

$$\mathrm{SUP}(S)=\{(s,\alpha)\mid s\in \mathrm{Traces}(S)\wedge \alpha=\sup\{\beta\mid \exists(w,\beta)\in S \text{ such that } \tilde{s}=\tilde{w}\}\}.$$

We are now in a position to define our evaluation domain. We formally define $\mathrm{TM_S}$ to be the set of all those subsets S of $(\mathrm{T}\Sigma)^*_{\leq}\times\mathrm{STAB}$ satisfying

(1) $(s,\alpha),(s,\alpha')\in S \Rightarrow \alpha=\alpha'$,

(2) $\langle\ \rangle\in \mathrm{Traces}(S)$,

(3) $s.w\in\mathrm{Traces}(S)\Rightarrow s\in\mathrm{Traces}(S)$,

(4) $(s,\alpha)\in S\Rightarrow(\tilde{s},\alpha)\in S$,

(5) $(s,\alpha)\in S\wedge s\cong w\Rightarrow(w,\alpha)\in S$,

(6) $s.\langle(t,a)\rangle\in\mathrm{Traces}(S)\Rightarrow \exists t'\leq t.\,(s\upharpoonright t').\langle(t',\hat{a})\rangle\in\mathrm{Traces}(S)$
$\wedge\,(t'\leq t''<t\Rightarrow(s\upharpoonright t'').\langle(t'',a)\rangle\in\mathrm{Traces}(S))$,

(7) $(s,\alpha)\in S\Rightarrow \mathrm{end}(s)\leq\alpha$,

(8) $(s,\alpha)\in S\wedge s.\langle(t,\hat{a})\rangle\in\mathrm{Traces}(S)\Rightarrow t\leq\alpha$,

(9) $(s,\alpha)\in S\Rightarrow$ if $t\geq\alpha$, $a\in\Sigma$ and $w\in(\mathrm{T}\Sigma)^*_{\leq}$ is such that $w=\langle(t,a)\rangle.w'$, then $(s.w,\alpha')\in S\Leftrightarrow(s.(w+(\alpha-t)),\alpha'+(\alpha-t))\in S$,

(10) $\forall t\in[0,\infty).\ \exists n(t)\in\mathbb{N}.\forall(s,\alpha)\in S.\ \mathrm{end}(s)\leq t\Rightarrow \#s\leq n(t)$.

Condition (1) states that each trace has a unique stability value (the time after which stability is guaranteed). Conditions (2) and (3) are inherited from the traces model of CSP. Condition (4) states that when an event becomes available ($\hat{a}$), it can also be communicated "normally" (a). Condition (5) states that the order of events which happen at the same time is irrelevant. Condition (6) states that when an event is available, it must have been continuously available since becoming available at some earlier time. Condition (7) states that the stability value for a trace cannot be less than the time of the last event in the trace. Conditions (8) and (9) assert that, after a process has become stable, the range of future behaviours that are available does not change (until some further communication). Condition (10) reflects our realism requirement that infinitely many events cannot occur in a finite period.

If $S\in\mathrm{TM}_S$ and $t\in[0,\infty)$, we define

$$S(t)=\{(s,\alpha)\in S\mid \alpha<t\}\cup\{(s,\infty)\mid \mathrm{end}(s)<t\wedge\exists\alpha\geq t.\,(s,\alpha)\in S\}.$$

This set, which is not necessarily in TM_S, records everything which can be observed about S before time t. Note that $S(0)=\emptyset$ for every S. The metric on TM_S is defined as

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

We now show how to define a semantic function $\varepsilon:\mathrm{CSP}\rightarrow\mathrm{TM}_S$. Notice how, in SKIP, WAIT and $a\rightarrow P$, the event becomes available once, but remains available forever after.

$$\varepsilon[\![\bot]\!]=\{(\langle\ \rangle,\infty)\},\qquad \varepsilon[\![\mathrm{STOP}]\!]=\{(\langle\ \rangle,0)\},$$

$$\varepsilon[\![\mathrm{SKIP}]\!]=\{(\langle\mathrm{x}\rangle,0)\}\cup\{(\langle(0,\hat{\surd})\rangle,0)\}\cup\{(\langle(t,\surd)\rangle,t)\mid 0\leq t\},$$

$$\varepsilon[\![\mathrm{WAIT}\ t]\!]=\{(\langle\ \rangle,t)\}\cup\{(\langle(t,\hat{\surd})\rangle,t)\}\cup\{(\langle(t',\surd)\rangle,t')\mid t\leq t'\},$$

$$\varepsilon[\![a\to P]\!]=\{(\langle\ \rangle,0)\}\cup\{(\langle(0,\hat{a})\rangle.(s+\delta),\alpha+\delta)\mid(s,\alpha)\in\varepsilon[\![P]\!]\},$$

$$\cup\{(\langle(t,a)\rangle.(s+(t+\delta)),\alpha+t+\delta)\mid(s,\alpha)\in\varepsilon[\![P]\!]\wedge t\geq 0\},$$

$$\varepsilon[\![P\,\square\,Q]\!]=\mathrm{SUP}(\varepsilon[\![P]\!]\cup\varepsilon[\![Q]\!]),$$

$$\varepsilon[\![P\sqcap Q]\!]=\mathrm{SUP}(\varepsilon[\![P]\!]\cup\varepsilon[\![Q]\!]).$$

When two processes are running synchronously in parallel, they co-operate on all events. An event can become available just when it becomes available in one of the partners and is already available in the other. When $v, w\in(\mathrm{T}\Sigma)^*_{\leq}$ have $\tilde{v}=\tilde{w}$, they can run in parallel yielding the trace $s=v\vee w$ with $\tilde{s}=\tilde{v}$, such that its nth element is "hatted" if and only if the nth element of v or w is.

$$\varepsilon[\![P\|Q]\!]=\mathrm{SUP}\{(s_P\vee s_Q,\max(\alpha_P,\alpha_Q\})\mid(s_P,\alpha_P)\in\varepsilon[\![P]\!]\vee(s_Q,\alpha_Q)$$
$$\in\varepsilon[\![Q]\!]\wedge\tilde{s}_P=\tilde{s}_Q\}.$$

Similar considerations apply to the alphabetised parallel operator.

$$\varepsilon[\![P_X\|_Y Q]\!]=\mathrm{SUP}\{(s,\max\{\alpha_P,\alpha_Q\})\mid\exists(s_P,\alpha_P)\in\varepsilon[\![P]\!],(s_Q,\alpha_Q)\in\varepsilon[\![Q]\!].$$
$$s\in(s_{P\,X}\|_Y s_Q)\},$$

where

$$v_X\|_Y w=\{s\in(\mathrm{T}\Sigma)^*_{\leq}\mid s\restriction(X\cup Y)=s\wedge\tilde{s}\restriction X=\tilde{v}\wedge\tilde{s}\restriction Y=\tilde{w}$$
$$\wedge s\restriction X-Y=v\restriction X-Y\wedge s\restriction Y-X=w\restriction Y-X$$
$$\wedge s\restriction X\cap Y=(v\restriction X\cap Y\vee w\restriction Y\cap X)\}.$$

$$\varepsilon[\![P\ |||\ Q]\!]=\mathrm{SUP}\{(s,\max(\alpha_P,\alpha_Q\})\mid\exists(u,\alpha_P)\in\varepsilon[\![P]\!],(v,\alpha_Q)\in\varepsilon[\![Q]\!].$$
$$s\in\mathrm{Tmerge}(u,v)\},$$

$$\varepsilon[\![P;Q]\!]=\mathrm{CL}_{\cong}(\mathrm{SUP}(\{(s,\alpha)\mid(s,\alpha)\in\varepsilon[\![P]\!]\wedge\surd\notin\Sigma(s)\}$$
$$\cup\{(s.(w+t),\alpha+t)\mid s.\langle(t,\hat{\surd})\rangle\in\mathrm{Traces}(\varepsilon[\![P]\!])$$
$$\wedge\surd\notin\Sigma(s)\wedge(w,\alpha)\in\varepsilon[\![Q]\!]\})),$$

$$\varepsilon[\![P\backslash X]\!]=\mathrm{SUP}\{(s\backslash X,\alpha)\mid(s,\alpha)\in\varepsilon[\![P]\!]\wedge s\text{ is }X\text{-active}\}$$

where s is X-active provided it contains no element of the form (t,a) for $a\in X$ (i.e., all communications in X are of the form $\hat{a}$).

$$\varepsilon[\![f^{-1}(P)]\!]=\{(s,\alpha)\mid(f(s),\alpha)\in\varepsilon[\![P]\!]\},$$

$$\varepsilon[\![f(P)]\!]=\mathrm{SUP}\{(f(s),\alpha)\mid(s,\alpha)\in[\![P]\!]\},$$

$$\varepsilon[\![\mu P.F(P)]\!]=\text{the unique fixed point of the contraction mapping }\hat{C}(Q)$$
$$=C(\mathrm{WAIT}\ \delta;Q),\text{ where }C\text{ is the mapping on }\mathrm{TM}_S\text{ represented by }F.$$

(1) *Hiding and recursion.* Again, consider P, Q, and P_n as defined in Section 3 (with the appropriate change in P_n to reflect the delay induced by each recursive

call in P).

$$Q = b \to Q \qquad\qquad P_0 = Q$$
$$P = a \to P \qquad \forall n \geq 1, \quad P_n = a \to (\text{WAIT } \delta\,; P_{n-1})$$

In the Timed Stability Model, $\lim_{n\to\infty} P_n = P$ and $\lim_{n\to\infty}(P_n \backslash a) = P\backslash a = \perp \neq \text{STOP}$. In fact, as desired, all operators are now continuous, and all, except recursion, distribute over $\sqcap$.

Also note that all syntactic recursions are now represented by contraction mappings, and hence are valid. For example,

$$\mu P.P = \text{fix}(\hat{C}), \quad \text{where } \hat{C}(Q) = \text{WAIT } \delta\,; Q$$
$$= \perp.$$

(2) *Compatibility with untimed models.* All but 3 of the 31 algebraic laws for semantics of the Failures-Divergence Model from [4, Table 1] hold in the Timed Stability Model (with the identification of $\sqcap$ with $\Box$). These three are:

$$P \,\|\, \text{STOP} = \begin{cases} \text{STOP} & \text{if } P \neq \perp \\ \perp & \text{if } P = \perp; \end{cases}$$

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

$$(a \to P)\backslash b = \begin{cases} (a \to P\backslash b) & \text{if } a \neq b, \\ P\backslash b & \text{if } a = b. \end{cases}$$

The failure of the first and third laws simply reflects the passage of time (for example, WAIT $n \,\|\, \text{STOP} = \text{WAIT } n\,; \text{STOP}$). The failure of the second law reflects our use of the delay constant δ to implement our view of realism: two processes in parallel can run faster than a sequential process. As indicated, in the complete version of this paper, we shall present a range of options whereby such parameters can be varied to suit the desired "view of the world".

Note that the Timed Stability Model does differ from the Failures-Divergence Model [4] in that $\Box$ is not strict with respect to $\perp$. In fact, it differs from all previously mentioned CSP models relevant to divergence in that $(s, \infty) \in P$ does not imply that $(s.w, \infty) \in P$ for all traces w. That is, just because a process may diverge after engaging in a given trace, it does not mean that some time later after extending the trace, the process might not again become stable. For example, let $P = a \to P$ and consider the process $R = (b \to (P\backslash a))\Box(b \to (b \to \text{STOP}))$. Both $(\langle(0, b)\rangle, \infty)$ and $(\langle(0, b)(\delta, b)\rangle, 2\delta) \in R$. If our only observation is $\langle(0, b)\rangle$, we must assume the worst; however, once we observe $\langle(0, b)(\delta, b)\rangle$, we know that we are safe. Although it is possible to modify our model to conform to the untimed models in this regard, we choose to allow the finer distinction of CSP processes made possible by the topological structure of our evaluation domain.

The Timed Stability Model is very much a partial correctness model. It gives an upper bound on the traces that *might* occur in a given process; it cannot guarantee that traces *will* occur, because of the potential nondeterminism of concurrent systems, and for reasons discussed below. The advantage of the model is in the simplicity by which it describes timing behaviour. Although we show in [17] how the present model can be extended to reflect total correctness, the resulting Timed Failures-Stability Model is necessarily much more complex. We are currently exploring the feasibility of establishing partial correctness in the Timed Stability Model and establishing total correctness in the corresponding untimed Failures-Stability Model, then bridging the two models through the concept of stability. If "total correctness" occurs in the timed model, it must have happened by the time of stability.

However, just as was the case with the untimed traces model, the present model is sufficient to describe the total correctness properties of *deterministic* processes (ones which cannot make internal decisions which materially alter their future behaviours). (The question of which untimed CSP programs are deterministic is discussed in the earlier literature, for example [9]. However, it is an interesting question exactly what is meant by a *timed* deterministic process, and which syntactic constructions preserve determinism.) It should be possible to refine the hiding operator $P\backslash X$ when P is deterministic, for if, after any trace s, a hidden event can become stably available in P, then it will. In these circumstances it is impossible for P to execute any nonhidden event after the time of stability, for the hidden one would have pre-empted it.

6. Specification and verification of timed processes

As an example of proof techniques in the Timed Stability Model, we consider a timed theory of recursion induction. This theory is essentially a generalisation of a similar theory for the topological Trace Model developed in [15, 2]. The meanings of any undefined terms in the presentation below can be inferred from similar untimed concepts in [9].

Basic Question. If we can prove some specification S true for all recursive calls, when can we infer the property true of the process?

In particular, if, given a specification S, we can show that, for each $Q \in \mathrm{TM}_S$, if Q satisfies S, then $\hat{C}(Q)$ satisfies S for the contraction mapping $\hat{C}$, when can we conclude that $P = \mathrm{fix}(\hat{C})$ satisfies S?

A *specification* S on TM_S is a mapping from the complete metric space TM_S to $\{\mathrm{T}, \mathrm{F}\}$. We say it is *continuous* if the set $\{P \mid S(P) = \mathrm{T}\}$ is closed. A specification S is *satisfiable* provided there exists $Q \in \mathrm{TM}_S$ such that $S(Q) = \mathrm{T}$ or (Q **sat** S).

Theorem. *If* $\hat{C} : \mathrm{TM}_S \to \mathrm{TM}_S$ *is a contraction mapping and S is a continuous, satisfiable specification, then if* ($\forall Q \in \mathrm{TM}_S$, Q **sat** $S \Rightarrow \hat{C}(Q)$ **sat** S), *then* $\mathrm{fix}(\hat{C})$ **sat** S.

Now suppose we wish to specify a timed vending machine which is capable of an unbounded number of transactions and which

(1) does not give out more chocolates than it receives payment for;

(2) on becoming stable, is able to offer a chocolate if it has received more payments than the number of chocolates it has given out;

(3) if it has given out exactly as many chocolates as it has received payment for it can accept a further coin when it becomes stable; and

(4) is initially stable, and never waits more than 4 seconds after any action before becoming stable again.

Because ours is a partial correctness model, we cannot completely specify (2) and (3). However, we can at least specify that the machine has the potential to do these things. (This is enough if, as is the case with the process below, the machine can be shown to be deterministic.)

The specification. $\forall Q \in \mathrm{TM}_S$, Q **sat** S provided

(S_1) $(s, \alpha) \in Q \Rightarrow \Sigma(s) \subseteq \{5\mathrm{p}, \mathrm{Choc}\}$ and $\#(s \restriction \{\mathrm{Choc}\}) \leq \#(s \restriction \{5\mathrm{p}\})$;

(S_2) $((s, \alpha) \in Q$ and $\alpha < \infty$ and $\#(s \restriction \{\mathrm{Choc}\}) < \#(s \restriction \{5\mathrm{p}\}))$

$\Rightarrow s.\langle(\alpha, \mathrm{Choc})\rangle \in \mathrm{Traces}(Q)$;

(S_3) $((s, \alpha) \in Q$ and $\alpha < \infty$ and $\#(s \restriction \{\mathrm{Choc}\}) = \#(\mathrm{s} \restriction \{5\mathrm{p}))$

$\Rightarrow s.\langle(\alpha, 5\mathrm{p})\rangle \in \mathrm{Traces}(Q)$;

(S_4) $(\langle\ \rangle, 0) \in Q$ and $(s.\langle(t, a)\rangle, \alpha) \in Q \Rightarrow \alpha \leq t + 4$.

Consider the timed vending machine TVM$*$, where

$$\mathrm{TVM}* = 5\mathrm{p} \rightarrow (\mathrm{WAIT}\ 2; (\mathrm{Choc} \rightarrow \mathrm{WAIT}\ 3; \mathrm{TVM}*)).$$

To show that TVM$*$ satisfies the above specification S, we need to show that $\mathrm{fix}(\hat{C})$ **sat** S, where

$$\hat{C}(Q) = 5\mathrm{p} \rightarrow (\mathrm{WAIT}\ 2; (\mathrm{Choc} \rightarrow (\mathrm{WAIT}\ 3; \mathrm{WAIT}\ \delta; Q))).$$

Each of (S_1), (S_2), and (S_4) is continuous and satisfiable by STOP, and (S_3) is continuous and satisfiable by ($5\mathrm{p} \rightarrow \mathrm{STOP}$). Hence, we need only to show for $i = 1, 4$ and $\forall Q$, (Q **sat** S_i) $\Rightarrow$ ($\hat{C}(Q)$ **sat** (S_i)). ((S_4) will require that $\delta \leq 1$.) We can conclude by the above Theorem that $\mathrm{TVM}* = \mathrm{fix}(\hat{C})$ **sat** S. Thus, we have reduced the verification of a recursive process to simple verification of each specification component for each recursive call.

7. Comparisons and conclusions

Pioneering work on the development of semantic models for timed versions of CSP was carried out in [11]. This work demonstrated the basic compatibility of timed CSP with the algebraic properties of the untimed language, though the hiding operator failed to distribute over $\sqcap$. Also, the distinction between deadlock and divergence relied on "unrealistic" processes; the existence of these unrealistic processes appears to have led to several problems.

Recently, in [12], the authors have presented a timed semantic model for a subset of the CSP of [7]. This model demonstrates the ability to describe many interesting aspects of real-time systems. However their work, though complementary, is largely independent of the aims of this paper since it is based on integer time and does not distinguish deadlock and divergence.

It is interesting to note that the authors of [3] have made essentially the same "realism" postulates as are made in the present paper. There they found that the use of the reals as the time domain allowed them to give a fully-abstract model using temporal logic.

We conclude by making three observations about our concept of timing.

(1) It might be argued that time in our model is too "discrete" in that limits only appear at infinity. For example, one might well wish $\lim_{n\to\infty} \mathrm{WAIT}\,(1-2^{-n}) = \mathrm{WAIT}\;1$, which is not true in our present model (the limit does not exist). We hope to investigate different metrics (probably not ultrametrics) which preserve limits already present in our model and add ones like the above.

(2) We regard our present model and semantics as a basis for developing more "realistic" semantics.

(3) The fact that a CSP process can sometimes communicate more that one copy of a single event at the same time complicates our model in several ways. (The $|||$ and $f(P)$ operators, f not injective, can introduce this phenomenon.) For example, if these were not possible we could achieve a considerable simplification by replacing timed traces by sets of timed events. (The order of communications would be recoverable from the times in the timed events.) In practice it may well prove desirable to remove $|||$ from the language and to restrict the use of $f(P)$, which would allow this and other simplifications.

Acknowledgment

We gratefully acknowledge many helpful discussions with C.A.R. Hoare. This research was undertaken at Professor Hoare's suggestion and with his encouragement. We also thank Michael Goldsmith and Geraint Jones for helpful suggestions.

References

[1] S.D. Brookes, A model for communicating sequential processes, Ph.D. Thesis, Oxford University, 1983.

[2] S.D. Brookes, C.A.R. Hoare and A.W. Roscoe, A theory of communicating sequential processes, *J. ACM* **31** (1984) 560-599.

[3] H. Barringer, R. Kuiper and A. Pnueli, A fully abstract concurrent model and its temporal logic, in: *Proc. 18th POPL* (1986) 173-183.

[4] S.D. Brookes and A.W. Roscoe, An improved failures model for communicating processes, in *Proc. Pittsburgh Seminar on Concurrency*, Lecture Notes in Computer Science **197** (Springer, Berlin, 1985) 281-305.

[5] J.W. de Bakker and J.I. Zucker, Processes and the denotational semantics of concurrency, *Inform. and Control* **54** (1982) 70-120.

[6] W.G. Golson and W.C. Rounds, Connections between two theories of concurrency: metric spaces and synchronisation trees, *Inform. and Control* **57** (1983) 102-124.

[7] C.A.R. Hoare, Communicating sequential processes, *Comm. ACM* **21** (1978) 666-677.

[8] C.A.R. Hoare, A model for communicating sequential processes, in: *On the Construction of Programs* (Cambridge University Press, London, 1980) 229-248.

[9] C.A.R. Hoare, *Communicating Sequential Processes* (Prentice-Hall, Englewood Cliffs, NJ, 1985).

[10] C.A.R. Hoare, S.D. Brookes and A.W. Roscoe, A theory of communicating sequential processes, Tech. Monograph PRG-16, Oxford University Computing Laboratory (1981).

[11] G. Jones, A timed model for communicating processes, Ph.D. Thesis, Oxford University, 1982.

[12] R. Koymans, R.K. Shyamasundar, W.P. de Roever, R. Gerth and S. Arun-Kumar, Compositional semantics for real-time distributed computing, Tech. Rept. 68, Faculteit der Wiskunde en Natuurwetenschappen, Katholieke Universiteit, Nijmegen, 1985.

[13] M. Nivat, Infinite words, infinite trees, infinite computations, in: *Foundations of Computer Science III*, (Mathematical Centre Tracts **109** (Mathematical Centre, Amsterdam, 1979) 3-52.

[14] E.R. Olderog and C.A.R. Hoare, Specification-oriented semantics for communicating processes, Lecture Notes in Computer Science **154** (Springer, Berlin, 1983) 561-572.

[15] A.W. Roscoe, A mathematical theory of communicating processes, Ph.D. Thesis, Oxford University, 1982.

[16] W.C. Rounds, Applications of topology to the semantics of communicating processes, in: *Proc. Pittsburgh Seminar on Concurrency*, Lecture Notes in Computer Science **197** (Springer, Berlin, 1985) 360-372.

[17] G.M. Reed and A.W. Roscoe, Metric spaces as models for real-time concurrency, in: *Proc. Third Workshop on the Mathematical Foundations of Programming Language Semantics* (1987), Lecture Notes in Computer Science **298** (Springer, Berlin, 1988) 331-343.