A Timed Model for Communicating Sequential Processes

G. M. Reed and A. W. Roscoe

Programming Research Group, Oxford University

I. Introduction.

The parallel language CSP [H,1985], an earlier version of which was described in [H,1978], 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 evolutionary development in [H,1980], [HBR,1981], [Ros,1982], [B,1983], [OH,1983], [BHR,1984], and [BR,1985], 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 nonnegative 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 [BHR,1984], 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.

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 [BHR,1984], and [BR,1985].

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 VII, previous constructions of timed CSP models have either relied on unrealistic (in the sense defined above) processes to make this distinction [J,1982], or else by design have

---

1The authors gratefully acknowledge that this research was funded under grant N0014-85-G-0134 from the U.S. Office of Naval Research.
not distinguished between these two concepts [KSdRGA-K,1985]. 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 [H,1980]) allows a complete treatment of deadlock and divergence for deterministic timed processes. At a later date, we shall present a hierarchy of timed models analogous to the existing range of untimed models, including our Timed Failures-Stability Model, which meets all the above requirements over the full range of CSP processes.

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.

II. A Review of CSP.

Throughout this paper, we will assume the background material of [BHR,1984], [BR,1985], and [H,1985]. A brief 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 \( \Sigma \) 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 [BHR,1984] and [BR,1985] with the addition of a process \( \text{WAIT} t \) for each \( t \geq 0 \): the process that terminates successfully after \( t \) units of time, and \( \bot \), 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 \( \Sigma \); \( X, Y \) over subsets of \( \Sigma \); \( f \) over the set of finite-to-one functions from \( \Sigma \) to \( \Sigma \); and \( F \) over "appropriate" compositions of our syntactic operators.

\[
P ::= \bot \mid \text{STOP} \mid \text{SKIP} \mid \text{WAIT} t \mid (a \rightarrow P) \mid P \Box Q \mid P \Join Q \mid P \parallel Q \mid P_X \parallel Q \mid P \parallel Q \mid P \setminus X \mid f^{-1}(P) \mid f(P) \mid \mu P.F(P)
\]

We will generally write \( P \setminus a \) rather than \( P \setminus \{a\} \) when hiding the single event \( a \).

We assume (from [BHR,1984], 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