Metric spaces as models for real-time concurrency

G.M. Reed and A.W. Roscoe

Oxford University Computing Laboratory
7-11 Keble Road, Oxford OX1 3QD, U.K.

ABSTRACT. We propose a denotational model for real time concurrent systems, based on the failures model for CSP. The fixed point theory is based on the Banach fixed point theorem for complete metric spaces, since the introduction of time as a measure makes all recursive operators naturally contractive. This frees us from many of the constraints imposed by partial orders on the treatment of nondeterminism and divergence.

1 Introduction

Real time has generally been considered to be too much an implementation matter to include in abstract models of concurrency. Most existing theories view of time is restricted to the relative order of events and to high level concepts such as 'eventually' and 'forever'. Nevertheless there are several reasons why it is desirable to have the ability to reason about real times. Most obviously, it is likely that anyone specifying a real system will wish to impose constraints on its running speed (and perhaps more detailed timing matters concerning its external communications). But perhaps more importantly from a theoretical point of view, there are several concepts commonly used in concurrent languages, such as interrupts and priority, which do not fit easily or at all into untimed models.

We therefore believe that there is a need for models of real-time parallel computation. But since it is likely to remain easier and cleaner, where possible, to do analysis in an untimed framework, it is important that a real-time model has well-understood links with an untimed theory. We have chosen to base our work on (extensions of) the theoretical version of CSP and to try to discover models that have links with that language's untimed theory. In an earlier paper [RR] we showed how the traces model [H1] could be expanded to include time. In this paper we give a timed version of the 'failures' model (including divergence) described in [BR,H2].

One of the main purposes of this paper is to show how real time gives a particularly natural measure for comparing processes: we can think of two processes as being $t$-alike if they are indistinguishable up to time $t$. This notion is easily formalised as a metric over the space of processes which provides a natural fixed point theory, seemingly with few of the disadvantages of the traditional ways of defining fixpoints in untimed models. In particular, we are able to deal with the problems of unbounded nondeterminism.

In the next section we present the model and the semantics of CSP. The difficulties of time mean that the model is quite complex and some of the semantic operators quite subtle; unfortunately time constraints for publication mean we cannot motivate or explain our definitions as

1The work reported in this paper was supported by the U.S. Office of Naval Research under grant N0014-85-G-0123.
thoroughly as we would have liked. (We aim to give a fuller presentation of our work in the near future.) Section 3 shows how we have treated nondeterminism and divergence and how the introduction of time frees us from difficulties found in the construction of untimed models. Finally we present our conclusions and outline some ways in which our work can be extended.

2. The timed failures-stability model for CSP

2.1 Objectives of Timed CSP

Our objective is the construction of a timed CSP model which provides a basis for the definition, specification, and verification of real-time processes with an adequate treatment of divergence and deadlock. Furthermore, we wish the model to be a "natural" extension of existing untimed models, and in particular, it should contain the timed equivalents of those CSP constructs modelled in [BHR,BR].

2.2 Abstract syntax for TCSP (Timed Communicating Sequential Processes)

We shall essentially extend the abstract syntax for untimed CSP from [BHR,BR] (with the addition of \( \bot \), 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.

The basic requirement for analysing real-time programming languages is the ability to model time-outs and interrupts. This can be accomplished in CSP simply by the addition of a process \( \text{WAIT} \ t \) for each real number \( t > 0 \): the process which engages in no visible event to the environment and which terminates successfully after \( t \) units of time. Intuitively, \( \text{SKIP} \) should coincide with \( \text{WAIT} \ 0 \).

TCSP

\[
P ::= \bot | \text{STOP} | \text{SKIP} | \text{WAIT} \ t | (a \rightarrow P) | P \square Q | P \cap Q | P \parallel Q |
\]

\[
P_X \parallel_Y Q | P \parallel Q | P \cap Q | P \setminus X | f^{-1}(P) | f(P) | \mu_p.F(p)
\]

2.3 Timing Postulates

The following are 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.** 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 \( \delta \) such that:

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