REAL-TIME CSP

G M REED

Rapporteur: A Saeed

Mathematical foundations for real-time distributed computing

G.M. Reed

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

Together with A.W. Roscoe, the author has earlier presented two models (the Timed Stability Model and the Timed Failures-Stability Model) offering timed versions of Hoare's CSP. In this lecture, the author outlines a hierarchy of untimed and timed models for CSP which includes the two above, and which allows one to reason about concurrent processes in a uniform fashion with the minimum of complexity. This hierarchy supports timewise refinement of specifications and the development of powerful proof rules for verification.

The objective of the lecture is to relate the above mathematical theory to the current state of the art, to indicate the nature of further development, and to discuss the inclusion of appropriate material on this topic into the computing science curricula.

Extended abstract

Programming languages which involve some form of parallelism are becoming prominent as computer science seeks to take advantage of the opportunities of distributed computing. At present, there remain serious difficulties with our efforts to reason effectively about the behaviour of such systems. Concurrent execution introduces non-determinism and such undesirable behaviour as deadlock and starvation, and it creates the crucial need for formal proof systems to understand such pathological behaviour. These proof systems require in turn the development of formal semantic models to establish their consistency and completeness, and to assist in achieving correct designs and implementations. Among the major obstacles to the full exploitation of parallel languages are the absence of standard semantic models, and consequently the lack of a consistent calculus for the rigorous specification and verification of concurrent programs.

In the case of sequential programming languages, it is well understood that programs can be taken to denote input-output functions or state-transformations and that the logical systems for specification and verification can be transparently based on the standard denotational semantics for the language in question. However, there is no agreement on an accepted method for assigning meanings to concurrent programs. Many different semantic models have been proposed. Each of the models attempted to describe effectively a particular aspect of the complex behaviour of concurrent programs. Hence, in a given model, it may be relatively easy to reason about one type of semantic property, but difficult or impossible to reason about others. Furthermore, it is difficult to establish the relative consistency between the various existing models of a given language, since each model may be

based on different mathematical or operational structures. Thus, it would be extremely beneficial at this point to isolate a single structure on which to base a uniform theory for generating a hierarchy of models for a common language capable of expressing the full complexity of distributed computing.

One major goal of such a theory would be the generation of successful models for real-time distributed computing. Although widely used throughout the world in such critical applications as aviation and nuclear power, real-time programming is a poorly understood discipline. The solutions to current problems involving sequential real-time systems will be most difficult and will take many years of work. Furthermore, the complexity of these problems will only intensify as we implement distributed real-time systems with nondeterministic behaviour. It is imperative that we begin now to develop the formal models on which the eventual solutions must be based.

Here, we outline a uniform mathematical theory of real-time distributed computing as described above within the context of the parallel language CSP (Communicating Sequential Processes) [H,1985]. This language, which was initially described by Hoare in [H,1978] has become a major tool for the analysis of structuring methods and proof systems involving parallelism. For example, the Ada programming language can be said to be "CSP-like" in the sense that concurrently active processes interact by some form of synchronized communication: the so-called "handshake" in CSP and the "rendezvous" in Ada. Furthermore, Occam [Occ,1984], the parallel language designed for the parallel commercial computer, the INMOS Transputer, was specifically based on CSP concepts. The theoretical parallel languages CCS and SCCS ([M,1980] and [M,1983]) are even more closely related to CSP. There are, by now, hundreds of research papers in the literature concerning this family of languages. Hence a uniform theory for the definition, specification, and verification of CSP processes would be a valuable contribution towards a consensus on the formal foundations of concurrency.

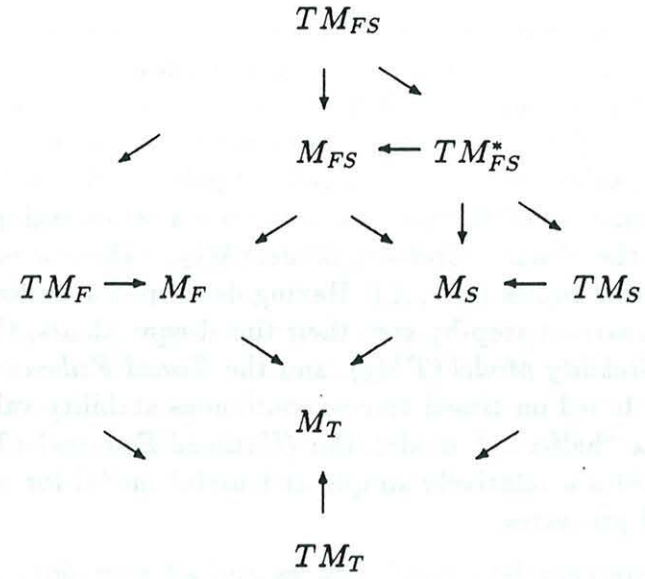
The unifying mathematical structure used in our hierarchy is that of complete metric spaces. Real time gives a particularly natural measure for comparing processes: we can think of two processes as being t-alike if they are indistinguishable up to time t. This notion is easily formalised as a metric over the space of processes which provides a natural fixed point theory, seemingly with few of the disadvantages of the traditional ways of defining fixed points in untimed models. In particular, we are able to deal effectively for the first time with the problems of unbounded nondeterminism, infinite hiding, and the subtle relationship between divergence and deadlock.

The relationships between the models in our hierarchy are based on the key concept of stability. 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 ([Ros,1982], [B,1983], [OH,1983], and [BR,1985]) as providing discrete information for a given trace s ("0" cannot diverge, "∞" can diverge), and our corresponding timed models 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 rely on this notion of **stability**, which is the dual of divergence.

We begin our hierarchy of CSP models with the topological *Trace Model* (M_T) from [Ros,1982]. We then reformulate the untimed Divergence and Failures Models into topological terms. First, we convert the Divergence Model from [OH,1983] into the *Stability Model* (M_S). This model distinguishes between deadlock and divergence in a manner that can be readily extended to the consideration of timed processes. Processes are identified with a set of ordered pairs (s, α), where $\alpha = 0$, if the process cannot diverge after engaging in the trace s and (s, ∞) otherwise. Next, we construct a topological *Failures Model* (M_F), where a process is identified with a set of ordered pairs (s, X) such that X represents the set of alphabet events which the process can refuse after engaging in s. We then merge the two models into the *Failures-Stability Model* (M_{FS}), where a process is identified with a set of ordered three tuples (s, α, X). Having developed a uniform hierarchy of untimed models, we then construct step-by-step their timed equivalents, the *Timed Trace Model* (TM_T), the *Timed Stability Model* (TM_S), and the *Timed Failures-Stability Model* (TM_{FS}). These models are based on timed traces, continuous stability values, and timed refusals. We also consider a "half-way" model, the *(Untimed Failures)-(Timed Stability) Model* (TM^*_{FS}), which provides a relatively simple and useful model for reasoning about an important class of timed processes.

The fact that our models are complete metric spaces and all recursions are contraction mappings make them natural vehicles for correctness proofs using the form of recursion induction described in [Ros,1982]. (A predicate that represents a non-empty closed subset and which is preserved by a recursion must contain the unique fixed point.) The introduction of stability seems to enhance the range of useful predicates which represent closed sets, since it (to a limited extent) allows us to look into the future. It is also our topological structure which lets us have the choice of infinite hiding, infinite alphabet renaming, unbounded non-determinism, or the non-equivalence of deadlock and divergence as we wish. Our basic models are flexible with respect to these and many other issues which are predetermined in the partial order models.

Finally, we formulate a hierarchical structure via projection mappings from the more complex models to the simpler ones. It is the uniformity of behaviour after a process has become stable that allows these projection mappings to preserve information. In particular, since the liveness properties predicted by the Timed Failures-Stability Model for a given process can be inferred from the time of stability on, we can often exploit this fact by reasoning in the simpler (Untimed-Failures)-(Timed Stability) Model. Indeed we have given case studies whereby the design of quite complicated timed processes can be started in the simple Traces Model and then moved gradually up the hierarchy to the Timed Failures-Stability Model, where at each step the specification and verification techniques of the relevant model are appropriate to the complexity of the design decision.



References.

[B,1983] S.D. Brookes, *A model for communicating sequential processes,* Oxford University D.Phil. thesis 1983.

[BR,1985] 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).

[H,1978] C.A.R. Hoare, *Communicating sequential processes*, CACM 21 (1978), 666-677.

[H,1980] C.A.R. Hoare, *A model for communicating sequential processes*, On the construction of programs CUP (1980), 229-248.

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

[M,1980] R. Milner, *A calculus of communicating systems*, Springer LNCS 92 (1980).

[M,1983] R. Milner, *Calculi for synchrony and asynchrony*, Theoretical Computer Science 25 (1983), 267-310.

[Occ,1984] *The occam programming manual*, (Inmos Ltd.) Prentince-Hall (1984).

[OH,1983] E.R. Olderog and C.A.R. Hoare, *Specification-oriented semantics for communicating processes*, Springer LNCS 154 (1983), 561-572. (Also, Acta Informatica 23 (1986), 9-66.)

[Re,1989] 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, to appear.

[RR,1986] 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 (1988) 249-261 .

[RR,1987] 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).

[Ros,1982] A.W. Roscoe, *A mathematical theory of communicating processes*, Oxford University D.Phil. thesis 1982.

DISCUSSION

Rapporteur: A Saeed

The discussion was opened by Professor Joseph asking what is the result of sequentially composing a divergent process with skip and then hiding the events of the divergent process. Dr. Reed replied that the resulting process diverges. It is in order to deal correctly with these kind of situations that it is necessary to retain so much information in the semantics.

Professor Mok then enquired as to how deadlines were specified, to which Dr. Reed replied that the specification would require that within a certain deadline period the event required must not be in the refusal set. He also commented that the wait construct is used in programs, not in specifications.

Professor Randell recalled that the speaker had mentioned that part of this work is being done within an ESPRIT project, and wondered what his experiences of this had been. Dr. Reed commented that the ESPRIT project involves some 10 partners, mainly working in temporal logic. He felt the collaboration to be very useful in that one often finds that someone working in one framework is struggling with a problem which in a different guise had been solved in another framework. Bringing the discussion to the teaching of Computing Science, Professor Randell said he had the impression that what is being taught at a particular institution is whatever the researchers there happen to be working on. Dr. Reed agreed that this seems to be the case.

Professor Bron observed that for sequential programs the unusual approach is to try to systematically derive programs from specifications, and wondered whether the CSP work was going in this direction? Dr. Reed replied that system design methodologies and tools were being developed for this.