Timed CSP: A Retrospective

Joël Ouaknine [1]

Oxford University Computing Laboratory, UK

Steve Schneider [2]

University of Surrey, UK

Abstract

We review the development of the process algebra Timed CSP, from its inception nearly twenty years ago to very recent semantical and algorithmic developments.

Key words: Process algebra, Timed CSP.

Timed CSP was first proposed in 1986 by Reed and Roscoe [25] as a real-time extension of the process algebra CSP. A front-runner amongst *timed* process algebras, it was quickly followed by a number of other dense-time and discrete-time process algebras, such as those appearing in [9,19,17,18,4,32,15,6,20,10], to name a few. The field continued to develop and expand into new directions (e.g., adding *probability* to time) and now constitutes a rich body of knowledge. [3]

Rather than aim at exhaustiveness, this paper retraces some of the milestones in the development of Timed CSP, and records some of its interesting features.

Reed and Roscoe's original model [25] was predicated on complete ultrametric spaces, and up to quite recently no significantly different other denotational semantics was known. Initially Timed CSP added a single primitive to the language CSP—*WAIT t*, for any time *t*—yet differed substantially at the denotational level from the cpo-based CSP. The resulting *Timed Failures* model nevertheless enjoyed natural projections to (untimed) CSP, later exploited by Schneider, Reed, and Roscoe in the form of *timewise refinement* [28,27,30]. The idea is simple, yet quite powerful: by syntactically transforming a Timed CSP process into a CSP one (essentially dropping all *WAIT t* terms), much information is preserved, and under appropriate conditions a number of properties can be formally established of the original Timed CSP process by studying its untimed counterpart.

[1] Email: joel@comlab.ox.ac.uk

[2] Email: s.schneider@surrey.ac.uk

[3] The papers concerned with process algebra and time number in the thousands according to http://scholar.google.com.

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

The semantics of Timed CSP is easily understood in relation to that of CSP: *timed failures* consist in traces and refusals (events that cannot be performed), but with every event performed or refused accompanied by a real-valued timestamp. In common with CSP, the refusal element of a timed failure embodies a *branching-time* aspect which is usually absent from other linear-time trace-based frameworks: the notion of *liveness* in Timed CSP, for example, consists in asserting that an event is never blocked, rather than postulate its eventual occurrence (certainly a reassurance to, say, jet fighter pilots relying on the '*eject*' button in case of emergency!).

In their respective doctoral theses, Schneider [28] and Davies [11] developed complete proof systems for Timed CSP. They also introduced a number of additional features, such as infinite choice, infinite observations, timeouts and interrupts, signals, and the removal of the requirement that every action and recursive call be preceded by a strictly positive amount of time—see [7] for a detailed account of these changes.

Jackson [16] was the first to look into model checking for Timed CSP. To this end, he defined a "finite-state" version of the language, together with a suitable temporal logic, and applied regions-based algorithms [1] to solve the model checking problem.

In 2001, Ouaknine [21] undertook a systematic study of the relationship between (dense-time) Timed CSP and a discrete-time version of it. This led him to extend Henzinger, Manna, and Pnueli's *digitization* techniques [14] to liveness properties, which resulted in a model checking algorithm for very a wide class of specifications that could be verified on the CSP model checker FDR. This work was later refined and extended in [22,23].

While most of the semantical developments of Timed CSP have tended to focus on the denotational side, Schneider equipped Timed CSP with a congruent operational semantics in [29], later slightly extended by Ouaknine in [21]. Full abstraction results of various kinds (with respect to may-testing, must-testing, and logical characterisations) can also be found in [29,22,12].

Perhaps surprising is the lack of work on *algebraic* semantics. This may be related to the fact that, unlike the case for (untimed) CSP (and indeed most process algebras), the parallel operators in Timed CSP cannot be reduced to other primitives. This observation was first recorded in [26], although in that instance it arose out of a rather circumstantial peculiarity of the semantic model. An interesting example is the following, taken from [21]: the process

$$(a \longrightarrow \mathit{STOP}) \,|||\, (\mathit{WAIT}\ 1 \fatsemi b \longrightarrow \mathit{STOP})$$

consisting of two interleaved components, the first of which offers an a immediately, and the second of which waits one time unit then offers a b, cannot be re-written in standard Timed CSP without some form of parallel composition. In other words, one cannot in general sequentially simulate the concurrent passage of time in Timed CSP, even if one includes timeouts. [4]

[4] A nonstandard timeout operator was introduced in [12], which does allow the elimination of

Although Timed CSP as described above has proved to be very successful, and indeed has been used in numerous case studies—see [31] for more details on the subject—some of its semantic requirements sit uneasily with the traditional style of "specification-as-refinement" usually advocated in CSP. For example, in untimed CSP, one specifies that a given process should not perform the event *error* by stipulating that it should refine the specification process $RUN_{\Sigma - \{error\}}$, which itself is capable of *any* behaviour other than performing *error*. Unfortunately, the ultrametric-based semantics for recursion in Timed CSP requires every recursion to be *time-guarded*—there should be some positive amount of time between two consecutive unwindings of a recursion. In [23], a root-and-branch review of the denotational semantics of Timed CSP was undertaken in order to allow such *Zeno* processes, and resulted in a substantially more expressive framework (predicated on cpo's rather than ultrametrics), in which processes could exhibit hitherto forbidden behaviours. As a result, many common specifications on Timed CSP processes (liveness, deadlock-freedom, timestop-freedom, . . .) have natural representations as refinements in this new model. Moreover, thanks to digitization techniques, the extra generality comes at no extra cost and Timed CSP processes can in fact be model-checked using an (untimed) CSP model checker such as FDR. It is perhaps worth noting that this new framework achieves its heightened expressiveness partly thanks to a restricted form of unbounded nondeterminism, which nonetheless does not destroy the formalism's valuable algorithmic properties.

These recent developments seem to indicate that Timed CSP remains an active research area, and progress is likely to continue for some time to come.

References

[1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS 90)*, pages 414–425. IEEE Computer Society Press, 1990.

[2] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] R. Alur and R. P. Kurshan. Timing analysis in COSPAN. In *Proceedings of Hybrid Systems III*, volume 1066, pages 220–231. Springer LNCS, 1996.

[4] J. C. M. Baeten and J. A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3:142–188, 1991.

[5] J. Bengtsson, K. G. Larsen, F. Larsen, P. Pettersson, and W. Yi. UPPAAL: A tool-suite for automatic verification of real-time systems. In *Proceedings of Hybrid Systems III*, volume 1066, pages 232–243. Springer LNCS, 1996.

[6] L. Chen. *Timed Processes: Models, Axioms and Decidability*. PhD thesis, University of Edinburgh, 1992.

parallel operators in a discrete-time context, however at the expense of some standard Timed CSP axioms and laws.

[7] J. Davies and S. A. Schneider. A brief history of Timed CSP. *Theoretical Computer Science*, 138(2):243–271, 1995.

[8] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Proceedings of Hybrid Systems III*, volume 1066, pages 208–219. Springer LNCS, 1996.

[9] R. Gerth and A. Boucher. A timed failures model for extended communicating processes. In *Proceedings of the Fourteenth International Colloquium on Automata, Languages and Programming (ICALP 87)*, volume 267, pages 95–114. Springer LNCS, 1987.

[10] R. Cleaveland, G. Lüttgen, and M. Mendler. An algebraic theory of multiple clocks. In *Proceedings of the Eighth International Conference on Concurrency Theory (CONCUR 97)*, volume 1243, pages 166–180. Springer LNCS, 1997.

[11] J. Davies. *Specification and Proof in Real-Time Systems*. PhD thesis, Oxford University, 1991.

[12] G. Lowe and J. Ouaknine. On timed models and full abstraction. In *Proceedings of the Twenty-first Conference on the Mathematical Foundations of Programming Semantics (MFPS 05)*, ENTCS, 2005.

[13] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. In *Proceedings of the Ninth International Conference on Computer-Aided Verification (CAV 97)*, volume 1254, pages 460–463. Springer LNCS, 1997.

[14] T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proceedings of the Nineteenth International Colloquium on Automata, Languages, and Programming (ICALP 92)*, volume 623, pages 545–558. Springer LNCS, 1992.

[15] M. Hennessy and T. Regan. A temporal process algebra. In *Proceedings of the Third International Conference on Formal Description Techniques for Distributed Systems and Communications Protocols (FORTE 90)*, pages 33–48. North-Holland, 1991.

[16] D. M. Jackson. *Logical Verification of Reactive Software Systems*. PhD thesis, Oxford University, 1992.

[17] A. Jeffrey. Abstract timed observation and process algebra. In *Proceedings of the Second International Conference on Concurrency Theory (CONCUR 91)*, volume 527, pages 332–345. Springer LNCS, 1991.

[18] A. Jeffrey. Discrete timed CSP. Programming Methodology Group Memo 78, Department of Computer Sciences, Chalmers University, 1991.

[19] F. Moller and C. Tofts. A temporal calculus of communicating systems. In *Proceedings of the First International Conference on Concurrency Theory (CONCUR 90)*, volume 458, pages 401–415. Springer LNCS, 1990.

[20] X. Nicollin and J. Sifakis. The algebra of timed processes, ATP: Theory and application. *Information and Computation*, 114:131–178, 1994.

[21] J. Ouaknine. *Discrete Analysis of Continuous Behaviour in Real-Time Concurrent Systems*. PhD thesis, Oxford University, 2001. Technical report PRG-RR-01-06.

[22] J. Ouaknine. Digitisation and full abstraction for dense-time model checking. In *Proceedings of the 8th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 02)*, volume 2280. Springer LNCS, 2002.

[23] J. Ouaknine and J. Worrell. Timed CSP = closed timed epsilon-automata. *Nordic Journal of Computing*, 10, 2003.

[24] G. M. Reed. *A Mathematical Theory for Real-Time Distributed Computing*. PhD thesis, Oxford University, 1988.

[25] G. M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. In *Proceedings of the Thirteenth International Colloquium on Automata, Languages, and Programming (ICALP 86)*, pages 314–323. Springer LNCS, 1986. *Theoretical Computer Science*, 58:249–261.

[26] G. M. Reed and A. W. Roscoe. The timed failures-stability model for CSP. *Theoretical Computer Science*, 211:85–127, 1999.

[27] G. M. Reed, A. W. Roscoe, and S. A. Schneider. CSP and timewise refinement. In *Proceedings of the Fourth BCS-FACS Refinement Workshop*, Cambridge, 1991. Springer WIC.

[28] S. A. Schneider. *Correctness and Communication in Real-Time Systems*. PhD thesis, Oxford University, 1989.

[29] S. A. Schneider. An operational semantics for Timed CSP. *Information and Computation*, 116:193–213, 1995.

[30] S. A. Schneider. Timewise refinement for communicating processes. *Science of Computer Programming*, 28:43–90, 1997.

[31] S. A. Schneider. *Concurrent and Real Time Systems: the CSP approach*. John Wiley, 2000.

[32] Y. Wang. *A Calculus of Real-Time Systems*. PhD thesis, Chalmers University of Technology, 1991.