Timed CSP: A Retrospective

Joël Ouaknine¹

Oxford University Computing Laboratory, UK

Steve Schneider²

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.

Keywords: 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.

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

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.
(essentially dropping all $\text{WAIT} \ t$ 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.

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

$$\left( a \rightarrow \text{STOP} \right) || \left( \text{WAIT} 1 \rightarrow b \rightarrow \text{STOP} \right)$$

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.\(^4\)

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 \textit{error} by stipulating that it should refine the specification process \(\text{RUN}_{\Sigma - \{\text{error}\}}\), which itself is capable of \textit{any} behaviour other than performing \textit{error}. Unfortunately, the ultrametric-based semantics for recursion in Timed CSP requires every recursion to be \textit{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, timestep-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


\(^4\) A nonstandard timeout operator was introduced in [12], which does allow the elimination of parallel operators in a discrete-time context, however at the expense of some standard Timed CSP axioms and laws.


