Electronic Communications of the EASST
Volume ?? (2011)

Proceedings of the
11th International Workshop on
Automated Verification of Critical Systems
(AVoCS 2011)

A Simulator for Timed CSP

Marc Dragon, Andy Gimblett, Markus Roggenbach[1]

2 pages

Guest Editors: Jens Bendisposto, Cliff Jones, Michael Leuschel, Alexander Romanovsky
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

[1] Acknowledging support by the SafeCap project, http://safecap.cs.ncl.ac.uk/index.php/Safecap_Project_Wiki.

A Simulator for Timed CSP

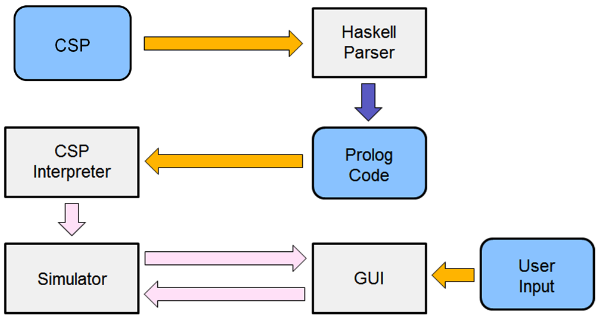
Marc Dragon, Andy Gimblett, Markus Roggenbach[†]

Swansea University, Wales, UK

Abstract: We present a simulator for Timed CSP based on the tool ProB.

Keywords: Timed CSP, ProB, Process Algebra, Real-Time, Simulation.

Time is an integral aspect of computer systems. It is essential for modelling a system's performance, but may also affect its safety or security. Timed CSP [Sch00] conservatively extends the process algebra CSP with timed primitives, where real numbers ≥ 0 model how time passes with reference to a single, conceptually global, clock. While there have been approaches for model checking Timed CSP ([Sch00, DHSZ06]), to the best of our knowledge we are the first to present a simulator for Timed CSP. Here, we restrict time to rational values only. Theoretically, this limits the expressibility of the language. Practically, this limitation turns out to be negligible (for instance all examples of Schneider's book [Sch00] can be dealt with in our simulator). The simulator is the outcome of an undergraduate project at Swansea University [Dra11].



Our Timed CSP simulator extends the open source tool ProB [Leu]. ProB's CSP simulator works as follows: The CSP specification is analyzed by a parser (written in Haskell) and translated to a representation in Prolog. A CSP Interpreter (in Prolog) stores the "firing rules" of CSP's operational semantics. The Simulator (also in Prolog) determines the actions available and the resultant states. A GUI (written in Tcl/Tk) allows the user to interact with the Simulator.

Timed CSP is closed under rational time [DNR11]. Consider, for example, the following firing rule ($\overset{t}{\rightsquigarrow}$ stands for a timed transition of duration t):

$$\frac{P \overset{d'}{\rightsquigarrow} P'}{(P \triangleright^{d} Q) \overset{d'}{\rightsquigarrow} (P' \triangleright^{d-d'} Q)} \quad [0 < d' \leq d]$$

Let $P \triangleright^{d} Q$ have rational times only (in particular, d is rational). Let d' be rational. Then $d - d'$ is rational and, by induction, P' has rational times only. Thus, $P' \triangleright^{d-d'} Q$ has rational times only.
Decision 1 *Our Timed CSP simulator deals with rational time only.*

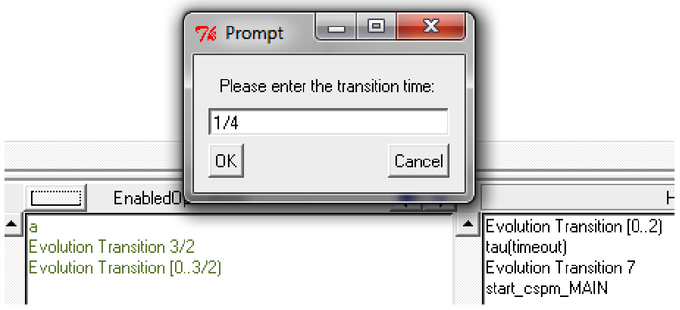
ProB also implements firing rules for those untimed CSP operators which usually are treated as syntactic sugar, e.g., the untimed timeout $P \triangleright Q = (P \,\square\, Q) \sqcap Q$. We follow ProB's design:
Decision 2 *All untimed and timed operators have their own timed firing rules.*

[†] Acknowledging support by the SafeCap project, http://safecap.cs.ncl.ac.uk/index.php/Safecap_Project_Wiki.

To this end, in [DNR11] we extend Timed CSP's operational semantics as given in [Sch00]: (1) In a definitional way, as e.g. for the untimed timeout, (2) in a conservative way, as e.g. for the replicated alphabetized parallel or for the conditional (for which [Sch00] gave no firing rules).

The core of our simulator is a rational arithmetic built on top of Prolog's built-in proper integers. Simulating Timed CSP provides two challenges: (1) In order to calculate the largest time step possible for a Timed CSP process, one has to analyze the process recursively. Consider, for instance, the process $T = (P \triangleright^e Q) \triangleright^f R$ with $0 < e < f$ and untimed processes P, Q and R. In T, the process P is enabled within the time interval $[0, e)$. A time step of length e (and a τ-transition) leads to the new state $Q \triangleright^{f-e} R$. Thus, the largest time step possible in T is e – see [DNR11] for details. (2) When the user chooses a timed transition of, say, d time units, the constant d needs to be propagated recursively along the term structure. Given, e.g., a time step $0 < d < e$ for the above term T, the resulting Timed CSP term is $(P \triangleright^{e-d} Q) \triangleright^{f-d} R$.

Currently, our simulator implements a slightly restricted sublanguage of Timed CSP: Processes can include only rational constants in timed operators; while most Timed CSP operators have been implemented, the operator $a@u \to P(u)$ (time of an action) is not supported yet.



Prompt

Please enter the transition time:

1/4

OK

Cancel

EnabledO

```
a
Evolution Transition 3/2
Evolution Transition [0..3/2]
```

```
Evolution Transition [0..2]
tau(timeout)
Evolution Transition 7
start_cspm_MAIN
```

The screen-shot shows a typical run of our simulator. Besides simulating examples given in [Sch00], we extensively use our tool within the SafeCap project in order to explore how the change of signalling rules affects railway capacity.

The ProB team has checked our implementation and intends to make it part of the next ProB distribution. This will require some minor changes to our code, mostly regarding syntax. It is future work to remedy the above mentioned, surmountable restrictions and to apply our tool within further application domains.

Acknowledgement We thank Erwin Catesbeiana (Jr.) for inspiring us to go the extra mile.

Bibliography

[DHSZ06] J.S. Dong, P. Hao, J. Sun, and X. Zhang. A Reasoning Method for Timed CSP Based on Constraint Solving. In LNCS 4260, Springer 2006.

[Dra11] M. Dragon. *A Timed CSP Simulator for Railway Systems*, BSc Dissertation, Swansea University, 2011.

[DNR11] M. Dragon, N. Nguyen, M. Roggenbach. *Theoretical foundations for simulating Timed CSP*, Technical report CSR 1-2011, Swansea University, 2011.

[Leu] M. Leuschel. The ProB Animator and Model Checker. Last accessed June 2011. http://www.stups.uni-duesseldorf.de/ProB/index.php5/Main_Page.

[Sch00] S. Schneider. *Concurrent and Real-time systems.* Wiley, 2000.