# Reasoning About Timed CSP Models

Jin Song Dong, Xian Zhang*, Jun Sun, Ping Hao

School of Computing,
National University of Singapore
{dongjs,zhangxi5,sunj,haoping}@comp.nus.edu.sg

**Abstract. HORAE** is an interactive system which supports composing and reasoning of Timed CSP process descriptions. Timed CSP extends CSP by introducing the capability to quantify temporal aspects of sequencing and synchronization. It is a powerful language to model real-time reactive systems. However, there is no verification tool support for proving critical properties over systems modeled using Timed CSP. **HORAE**, using Constraint Logic Programming (CLP) as the underlying reasoning support, is capable to prove traditional safety properties and beyond, such as the reachability, deadlock-freeness, timewise refinement relationship, lower or upper bound of variables, and etc.

## 1 Overview of HORAE

**HORAE** is an interactive tool which provides composing and reasoning of Timed CSP process descriptions. Event-based specification languages like the classic Communicating Sequential Process (CSP) of Hoare's [3] and its timed extension Timed CSP [7] have been widely accepted and applied to a wide range of systems, including communication protocols, embedded systems, etc [8]. It is important that system specified using CSP or Timed CSP can be proved formally and even better if the proving is fully automated.

**HORAE** uses Constraint Logic Programming (CLP [4]) as underlying reasoning mechanism. CLP is designed for mechanized proving based on constraint solving. It has been successfully applied to model programs and transition systems for the purpose of verification. The main advantage of using CLP pertains to expressiveness. For example, [2] demonstrated the proof of some standard properties, as well as properties such as time bounds between important events, on a CLP representation of Timed Safety Automata. [6] showed that CLP can be used to specify and verify properties like whether systems modelled using Timed Automata are symmetric.

**HORAE** is built on the powerful constraint solver CLP($\mathcal{R}$) [5]. Both operational and denotational semantics of Timed CSP processes are encoded in CLP($\mathcal{R}$) [1]. The front-end of the tool is implemented in Java. The main features include:

- building Timed CSP models,
- specifying properties in a systematic way,
- verifying various kinds of properties with counterexamples provided if any, and
- generating LaTeX presentation of the Timed CSP models.

* Author for correspondence, phone: +65 65162834

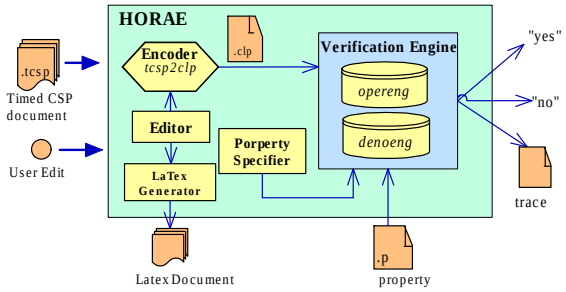


**Fig. 1.** Overview of HORAE

An overview of **HORAE** is shown in Figure 1. It mainly consists of five components, i.e., a powerful GUI editor to compose Timed CSP models, an encoder which translates Timed CSP processes to CLP models, a property specifier which is used to specify properties in a systematic way, a verification engine which is used to verify properties and a LaTeX code generator.

## 2 Building Timed CSP Models

In our system, Timed CSP processes are in ASCII form, i.e., machine readable Timed CSP. **HORAE** has a user-friendly editor to build Timed CSP models. The encoder **tcsp2clp** can automatically transform Timed CSP model in *.tcsp*-format into the CLP($\mathcal{R}$) *.clpr*-format by syntax rewriting.

## 3 Verifying Models

The verification engine is the core component. The verification is performed by the engine which takes a Timed CSP model in *.clpr*-format from the encoder and a property as input. This engine has two modules, which are **opereng** and **denoeng**, building on the operational model and the denotational model respectively. Different kinds of properties are checked by different modules. For instance, timewise refinement properties would be checked by **denoeng**, the variable bounds properties would be checked by **opereng**, and the safety and liveness properties can be checked by both modules.

The operational semantics is modelled by capturing both evolution relations and timed event transition relations of a process. In the denotational semantics, we encoded the timed traces and timed failures model of Timed CSP, where a Timed CSP process is represented by a set of timed traces or a set of timed failures. Detailed theory and encoding of both semantics can be found in [1].

### 3.1 Property Specification and Verification

The module *Property Specifier* is used to specify three kinds of properties in a systematic way, which can later be verified by the verification engine.

**Safety and Liveness** Reachability properties are specified as: $Reachable(P, Q, \Psi) = N$ which tests whether $N$ is a possible trace from process $Q$ to $P$, with some constraint $\Psi$. Deadlock freeness of process $P$ can be checked in $Deadlock(P)$.
**Timewise Refinement** Both trace timewise refinement and failures timewise refinement relationship can be checked between two Timed CSP processes. The refinement properties are specified in the form of $P\ T[TF=\ Q$ and $P\ SF[TF=\ Q$ respectively, where $P$ and $Q$ are two processes.
**Variable Bounds** We can identify the lower or upper bound of a variable. For example, we can identify whether a given value of a time variable $T$ is an upper or lower bound of the execution time for process $P$ to evolve into process $Q$. These properties are specified as: $Upper(P,\ Q,\ time) = T$ or $Lower(P,\ Q,\ time) = T$. Similarly, the lower or upper bound of the duration between two events $A$ and $B$ in a process $P$ can also be specified as $P :: Upper(A,\ B,\ time) = T$ or $P :: Lower(A,\ B,\ time) = T$.

## 4 Experiment

In this section, we compare our tool with the mature model checker for CSP, namely FDR (version 2.78), in terms of flexibility as well as efficiency. In the following, we demonstrate our experiment with three examples on a Unix system located at a Sunfire server with 1GB user memory. Because FDR is designed for CSP, the quantitative timing aspects of the examples have been abstracted before FDR verification.
**Timed Vending Machine (TVM)** The specification of the TVM is presented in [1]. The following properties are verified:

- *tvm-1* Deadlock-freeness,
- *tvm-2* Trace timewise refinement,
- *tvm-3* Whether it is possible that coffee is selected while tea is dispatched.

**Dining Philosopher** The classic dining philosopher specification is available in [3]. We implemented this example with $N$ philosophers. Three properties are verified:

- *philosN-1* Deadlock-freeness
- *philosN-2* No more than N+1/2 philosophers can eat at the same time.
- *philosN-3* Whether it is possible that one philosopher eat all the time with the others starving.

**The Railway Crossing** The railway crossing system is originally presented in [9]. The three properties selected for comparison are:

- *railway-1* Deadlock-freeness
- *railway-2* Legal trace check
- *railway-3* Whether the lower bound for a train passes the crossing is 320*s*.

The experiment results are summarized in Table 1, where - indicates that the property cannot be checked. We run the examples in both **HORAE** and FDR and calculate the execution time of each property if it can be checked in that system. From the table, we can see that most of our timings are competitive or even better, while in some cases, we are not so competitive. The important metric of our experiment is that our tool can handle a wider range of properties than FDR, including the timed-related properties, bounds of variables, event specified properties, and etc.

| **Assertion** | **CLP($\mathcal{R}$)** *(secs.)* | **FDR** *(secs.)* |
|---|---|---|
| *tvm-1* | 0.00 | 0.23 |
| *tvm-2* | 0.03 | 0.27 |
| *tvm-3* | 0.01 | – |
| *railway-1* | 0.25 | 0.25 |
| *railway-2* | 0.02 | 0.26 |
| *railway-3* | 0.32 | – |

| **Assertion** | **CLP($\mathcal{R}$)** *(secs.)* | **FDR** *(secs.)* |
|---|---|---|
| *philos3-1* | 0.12 | 0.25 |
| *philos3-2* | 0.22 | – |
| *philos3-3* | 0.04 | 0.17 |
| *philos4-1* | 0.84 | 0.28 |
| *philos4-2* | 2.5 | – |
| *philos4-3* | 0.1 | 0.3 |

**Table 1.** Experiment Results

## 5 Conclusion

In this work, we developed an interactive tool for modelling and reasoning of Timed CSP models. To our knowledge, it is the first mechanized reasoning support for Timed CSP. **HORAE** is a distinguished tool support for modelling and verifying complex real-time systems. It uses CLP as underlying reasoning support. Both operational and denotational semantics of Timed CSP are encoded so that a wide range of properties can be specified and verified soundly.

## References


1. J. S. Dong, J. Sun, P. Hao, and X. Zhang. A Reasoning Method for Timed CSP based on Constraint Solving. Technical report TRC6/06, School of Computing, National University of Singapore, 2006.
   http://www.comp.nus.edu.sg/∼zhangxi5/horae.ps.
2. Gopal Gupta and Enrico Pontelli. A constraint-based approach for specification and verification of real-time systems. In *IEEE Real-Time Systems Symposium*, pages 230–239, 1997.
3. C. A. R. Hoare. *Communicating Sequential Processes*. International Series in Computer Science. Prentice-Hall, 1985.
4. Joxan Jaffar and Michael J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.
5. Joxan Jaffar, Spiro Michaylov, Peter J. Stuckey, and Roland H. C. Yap. The clp(r) language and system. *ACM Trans. Program. Lang. Syst.*, 14(3):339–395, 1992.
6. Joxan Jaffar, Andrew E. Santosa, and Razvan Voicu. A clp proof method for timed automata. In *RTSS*, pages 175–186, 2004.
7. George M. Reed and A. W. Roscoe. A Timed Model for Communicating Sequential Processes. In *Proceedings of ICALP*, pages 314–323, 1986.
8. A.W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.
9. Steve Schneider. *Concurrent and Real-time System: The CSP approach*. JOHN WILEY & SONS, LTD, 2000.