

Model Checking for the Concurrent Constraint Paradigm

Alicia Villanueva,

*Dep. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n, E-46022 Valencia, Spain
E-mail: villanue@dsic.upv.es*

This paper abstracts the contents of the Ph.D. dissertation which has been recently defended by the author. Although *model checking* was defined to automatically verify hardware, in the last decades it has been showed that it is possible to apply the technique also to software. The *concurrent constraint paradigm* is a simple but powerful computational model which we can use to specify reactive and hybrid systems. The thesis considers three of the timed languages of this paradigm. It presents two methods to apply the model checking technique to two different timed concurrent constraint languages, and it is also defined a denotational semantics which is fully abstract w.r.t. the operational behavior of another timed concurrent constraint language. This new semantics allows one to perform useful static analysis of programs.

Keywords: Timed Concurrent Constraint languages, Model Checking, Denotational Semantics

1. Extended Abstract of the Ph.D. Dissertation

Formal verification of temporal properties is necessary in many real applications. We can find in the literature many case studies which show us how formal verification techniques allowed scientists to find errors in systems that were thought to be correct [8]. *Model checking* [5,6] is an automatic formal verification technique that, given a model of a system and a temporal formula, determines whether the model satisfies the formula. This technique is based on the exhaustive analysis of the state-space of the system, therefore the main drawback of model checking software is the state explosion problem. Many approaches such as the symbolic approach [4], the abstract interpre-

tation approach [7], the on-the-fly approach [10], etc. have been defined in the last years to mitigate this problem. Moreover, although model checking was defined for verifying finite-state systems, many researchers try to extend it to handle infinite-state systems [11].

In the presented dissertation we consider the Concurrent Constraint (cc) paradigm [12] to specify reactive and hybrid systems. Then, we provide the necessary formalism to verify such kind of systems. In particular, we handle three of the extended languages defined from the cc paradigm: the tcc [13], tccp [2] and hcc [9] languages. The first two languages have a discrete notion of time allowing the programmer to model reactive systems. The last one introduces a continuous notion of time which allows us to model hybrid systems.

It is well known that an appropriate denotational semantics allows us to perform very interesting analysis over languages in a simple way. In that sense, in the thesis we show that, although both denotational and operational semantics were given when tcc was defined, they do not always coincide. We define a fully abstract denotational semantics (w.r.t. the operational semantics) for the tcc model. We also describe an application of our new semantics to the analysis of the expressive power of the new construct introduced in tcc to model the timeout or preemption behaviors. We show that the new construct makes the tcc language a more powerful language than the cc model.

The main result of this thesis is the definition of a model checking algorithm for tccp programs. The idea is to exploit the good features of the cc paradigm to mitigate the state explosion problem of model checking. We take advantage of the *constraint* nature of the language in order to redefine the three phases of the model checking technique. First of all we use constraints to define what a state is in the model of the system. A state of the model can be seen as a conjunction of constraints.

This means that a state of our model represents a set of states of a classical *Kripke Structure* [8].

Furthermore, constraints are directly used in the logic that we consider [3], thus it is not necessary to transform our model into a Kripke Structure. If we had use classical temporal logics, this transformation would be necessary since they could not handle our model directly. Note that we have only partial information while classical logics need full information about values of variables.

We also define a method to verify hcc programs. We show that the hcc language allows the programmer to specify hybrid systems in general and linear hybrid systems in particular [1]. The key idea in this case is also to take advantage of the nature of the cc paradigm. The approach presented in this thesis is the first attempt to apply the model checking technique to the hcc language.

To summarize, in this thesis we have tackled the verification problem for reactive and hybrid systems specified in languages from the cc paradigm. We think that constraints provide us a tool for defining compact and realistic methods for the automatic verification of systems.

2. Formal Information on the Ph.D. Dissertation

Author: Alicia Villanueva García.

Title: Model Checking for the Concurrent Constraint Paradigm.

Language of Presentation: English.

Advisors: Professors María Alpuente and Moreno Falaschi.

Date of Defense: May 24, 2003.

Institution granting degree: Universidad Politécnica de Valencia, Spain *and* Università degli Studi di Udine, Italy.

Further information on the thesis can be found at the URL:

<http://www.dsic.upv.es/~villanue>

Acknowledgements

The author wants to thank her advisors María Alpuente and Moreno Falaschi for their dedication and support during the last years. She wants also thanks their reviewers Professor Gabbrielli, Professor Podelsky and Professor Dovier for their comments, which were an important help to improve the quality of the dissertation.

References

- [1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.
- [2] F. S. de Boer, M. Gabbrielli, and M. C. Meo. A Timed Concurrent Constraint Language. *Information and Computation*, 161:45–83, 2000.
- [3] F. S. de Boer, M. Gabbrielli, and M. C. Meo. A Temporal Logic for reasoning about Timed Concurrent Constraint Programs. In G. Smolka, editor, *Proceedings of 8th International Symposium on Temporal Representation and Reasoning*, pages 227–233. IEEE Computer Society Press, 2001.
- [4] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} and beyond. *Information and Computation*, 98(2):142–170, June 1992.
- [5] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of Workshop on Logic of programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, Berlin, 1981. Springer-Verlag.
- [6] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. In J. R. Wright, L. Landweber, A. Demers, and T. Teitelbaum, editors, *Proceedings of the 10th Annual ACM Symposium on Principles of Programming Languages*, pages 117–126. ACM Press, 1983.
- [7] E. M. Clarke, O. Grumberg, and D. E. Long. Model Checking and Abstraction. *ACM Transactions on Programming Languages and Systems*, 16:1512–1542, 1994.
- [8] E. M. Clarke, O. Grumberg, and D. E. Long. Model Checking. *Nato ASI Series F*, 152, 1996.
- [9] V. Gupta, R. Jagadeesan, and V. A. Saraswat. Hybrid cc, hybrid automata and program verification. In Alur, Henzinger, and Sontag, editors, *Hybrid Systems III*, volume 4 of *Lecture Notes in Computer Science*, pages 52–75. Springer-Verlag, 1996.
- [10] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997.
- [11] O. H. Ibarra, T. Bultan, and J. Su. On Reachability and Safety in Infinite-State Systems. *International Journal of Foundations of Computer Science*, 12(6):821–836, 2001.
- [12] V. A. Saraswat. *Concurrent Constraint Programming Languages*. The MIT Press, Cambridge, MA, 1993.
- [13] V. A. Saraswat, R. Jagadeesan, and V. Gupta. Foundations of Timed Concurrent Constraint Programming. In *Proceedings of 9th Annual IEEE Symposium on Logic in Computer Science*, pages 71–80, New York, 1994. IEEE.