SIMULATION AND VERIFICATION OF HYBRID SYSTEMS USING CHI [1]

D.A. van Beek, **J.E. Rooda**, **R.R.H. Schiffelers**

Department of Mechanical Engineering
Eindhoven University of Technology, P.O.Box 513
5600 MB Eindhoven, The Netherlands

INTRODUCTION

The hybrid χ (Chi) formalism (van Beek *et al.*, 2006*a*; Man and Schiffelers, 2006) integrates concepts from dynamics and control theory with concepts from computer science. It integrates ease of modeling with a straightforward, structured operational semantics. Ease of modeling is ensured by means of, among others, the following concepts: 1) different classes of variables: discrete, continuous and algebraic; 2) strong time determinism of alternative composition in combination with delayable guards; 3) integration of urgent and non-urgent actions; 4) differential algebraic equations as in mathematics; 5) concepts for complex system specification: 5a) process terms for scoping that integrate abstraction, local variables, local channels and local recursion definitions; 5b) process definition and instantiation that enable process re-use, encapsulation, hierarchical and/or modular composition of processes; and 5c) different interaction mechanisms: handshake synchronization and synchronous communication that allow interaction between processes without sharing variables, and shared variables that enable modular composition of continuous-time or hybrid processes.

The formal semantics of χ allows the definition of provably correct implementations for simulation and verification. *Copyright © 2006 IFAC*

SIMULATION OF χ MODELS

For simulation of hybrid χ models, two simulators are available: a symbolic simulator and a simulator based on S-functions (The MathWorks, Inc, 2005*b*). Both simulators are defined in terms of a so-called stepper, which computes the set of possible transitions for given a χ process. The stepper consists of three main functions: a function $\mathcal{S}_\mathrm{a}$ which returns the set of action steps for given a χ process, a function $\mathcal{S}_\mathrm{d}$ which returns the set of time steps for given a χ process, and a function Tr which returns the reduced set of transitions. Action steps and time steps can be seen as symbolic transitions. They contain all information that is needed to determine the transitions that the process from which they are derived can perform without solving predicates. An action step represents zero or more action transitions and a time step represents zero or more time transitions.

In general, the set of transitions of a χ specification is infinite. In particular, the number of time transitions of a χ process is usually infinite: if a process can delay for t time units, then, for every $0 \leq t' \leq t$, it can also delay for t' time units. To get rid of these additional time transitions, instead of returning all time transitions of a time step, for each trajectory only the time transition with longest duration is returned. Although this reduced set of transitions can still be infinite, in practice, this is rarely the case.

Note that an implementation of the stepper functions may impose additional restrictions on the χ syntax. For instance, for an implementation of function Tr, a (symbolic) solver is needed to compute the solutions of action predicates, the solution of delay predicates (differential algebraic equations), and the maximum duration of a time transition. Depending on the solver that is used, additional restrictions may be required.

The stepper functions are defined in such a way that it is easy to define different implementations. In case

[1] Work partially done in the framework of the HYCON Network of Excellence, contract number FP6-IST-511368

of the symbolic simulator, the symbolic solving capabilities of Maple (MapleSoft, n.d.) are used for implementing the Tr function. The simulator based on S-functions interacts with Matlab Simulink (The MathWorks, Inc, 2005*a*) via a so-called DE$^+$ simulator. The DE$^+$ simulator performs action transitions until the stepper returns a time step. This time step is then executed by Simulink by solving the delay predicates (ODEs) and monitoring the root functions (possibly resulting in state-events) that are specified in the time step. At the end of the time transition, the DE$^+$ simulator is called again.

VERIFICATION OF χ MODELS

One of the most successful formalisms for hybrid system verification is the theory of hybrid automata. In (Man and Schiffelers, 2006), formal translations between χ and hybrid automata (in both directions) have been defined. The translation from hybrid automata to χ aims to show that the χ formalism is at least as expressive as the theory of hybrid automata. The translation from (a subset of) χ to hybrid automata enables verification of χ specifications using existing hybrid automata based verification tools.

To the best of our knowledge, none of the hybrid automata definitions from literature is expressive enough to be used as the target for the translation of hybrid χ. Therefore, The translation uses a target hybrid automata definition, called HA_{u} automata, where the u stands for urgency, that uses features from different hybrid automata definitions. In particular, the definition of the jump predicate in combination with a set of changeable variables is based on (Alur *et al.*, 1996), the solution concept that allows piecewise differentiable functions is based on (van der Schaft and Schumacher, 2000), and the definition of urgent transitions was inspired by (Nicollin *et al.*, 1992).

In (Man and Schiffelers, 2006; van Beek *et al.*, 2006*b*), we use the verification tool PHAVer (Polyhedral Hybrid Automaton Verifyer) (Frehse, 2005) to show that it is indeed possible to verify properties of χ specifications using an existing hybrid automata based verification tool. Since a manual translation is very time consuming and error prone, the translation has been automated by implementing it using the programming language Python (Python website, 2005).

NORMALIZATION OF χ MODELS

The χ process algebra is a rich language that has strong support for modular composition by allowing unrestricted combination of operators such as sequential composition, parallel composition, and scoping of local variables and channels. The fact that the χ process algebra is such a rich language potentially complicates the development of tools for χ, since the implementations have to deal with all possible combinations of the χ atomic statements and the operators that are defined on them. This is where the process algebraic approach of equational reasoning, that allows rewriting models to a simpler form, is essential.

Instead of defining simulation and verification implementations on the full χ language, the process algebraic approach of equational reasoning makes it possible to transform χ models in a series of steps to a (much simpler) normal form (process algebraic linearization), and to define the implementations on the normal form. The original χ model and its normal form are bisimilar, which ensures that relevant model properties are preserved. The normal form has strong syntactical restrictions, no parallel composition operator, and is quite similar to a hybrid automaton. Partial normalization, keeping the top level parallelism intact, is also possible. Currently, generation of the normal form is automated, and correctness of the transformation to the normal form is proved.

REFERENCES

Alur, R., T. A. Henzinger and P. H. Ho (1996). Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering* **22**(3), 181–201.

Frehse, Goran (2005). PHAVer: Algorithmic verification of hybrid systems past HyTech. In: *Hybrid Systems: Computation and Control, 8th International Workshop* (Manfred Morari and Lothar Thiele, Eds.). Vol. 3414 of *Lecture Notes in Computer Science*. pp. 258–273. Springer-Verlag.

Man, K. L. and R. R. H. Schiffelers (2006). Formal Specification and Analysis of Hybrid Systems. PhD thesis. Eindhoven University of Technology.

MapleSoft (n.d.). Maple. http://www.maplesoft.com.

Nicollin, X., A. Olivero, J. Sifakis and S. Yovine (1992). An approach to the description and analysis of hybrid systems. In: *Workshop on Theory of Hybrid Systems*. pp. 149–178.

Python website (2005). http://www.python.org.

The MathWorks, Inc (2005*a*). *Using Simulink, version 6*. http://www.mathworks.com.

The MathWorks, Inc (2005*b*). *Writing S-functions, version 6*. http://www.mathworks.com.

van Beek, D. A., K. L. Man, M. A. Reniers, J. E. Rooda and R. R. H. Schiffelers (2006*a*). Syntax and consistent equation semantics of hybrid Chi. *Journal of Logic and Algebraic Programming* **68**(1-2), 129–210.

van Beek, D. A., K.L. Man, M. A. Reniers, J. E. Rooda and R. R. H. Schiffelers (2006*b*). Formal verification of hybrid Chi models using PHAVer. In: *Proceedings of the conference on Mathematical Modelling 2006*. Vienna, Austria.

van der Schaft, A. J. and J. M. Schumacher (2000). *An Introduction to Hybrid Dynamical Systems*. Vol. 251 of *Springer Lecture Notes in Control and Information Sciences*. Springer.