

Discrete Simulation of Behavioural Hybrid Process Calculus

Tomas Krilavičius* and Helen Schonenberg

Formal Methods and Tools, Univ. of Twente, 7500AE Enschede, The Netherlands,
(T.Krilavicius, M.H.Schonenberg)[@cs.utwente.nl](mailto:cs.utwente.nl),
WWW home page: <http://www.cs.utwente.nl/~krilaviciust>

Abstract Hybrid systems combine continuous-time and discrete behaviours. Simulation is one of the tools to obtain insight in dynamical systems behaviour. Simulation results provide information on performance of system and are helpful in detecting potential weaknesses and errors. Moreover, the results are handy in choosing adequate control strategies and parameters.

In our contribution we report a work in progress, a technique for simulation of Behavioural Hybrid Process Calculus, an extension of process algebra that is suitable for the modelling and analysis of hybrid systems.

1 Introduction

The growing interest in hybrid systems both in computer science and control theory has generated a new interest in models and formalisms that can be used to specify and analyse such systems. A prominent framework for hybrid systems is provided by the family of hybrid automata models (hybrid automata [1], hybrid behavioural automata [2], hybrid input/output automata [3]). More recently process algebraic models have been put forward as a vehicle for the study of hybrid systems [4,5,6,7].

Simulation is a *de facto* standard tool in both academia and industry for analysis of hybrid systems. It helps to detect potential weaknesses and errors, and provides information on performance of system. There is a number of simulation tools which provide various facilities for analysis of hybrid systems. Hybrid χ [6] provides facilities for simulation of hybrid process calculus. HYVISUAL [8] is a Java based visual modeller and simulator for hierarchical continuous-time dynamical and hybrid systems. Dymola¹, STATEFLOW/SIMULINK [9] and 20-Sim² provide industrial strength facilities for simulation of non-causal object oriented simulation language ModelicaTM [10], hierarchical formalism and bond graphs [11], respectively.

* Work is partially done in the framework of the HYCON Network of Excellence, contract number FP6-IST-511368*.

¹ See <http://www.Dynasim.se>.

² <http://www.20sim.com/>

We report a work in progress, a technique for simulation of Behavioural Hybrid Process Calculus (BHPC) [7]. BHPC is a process calculus that extends the standard repertoire of operators that combine discrete functional behaviour with features to also represent and compose continuous-time behaviour. Dynamic behaviour is represented by the evolution of variables, which are typically defined in terms of differential equations. Following [12], behaviour can be simply seen as the set of all allowed real-time evolutions, or *trajectories*, of the system variables. The operational semantics of the calculus defines the transitions for the simulator. An adapted version of the expansion law from [13] is used to solve parallel composition. As a first step towards simulation of entire BHPC, we propose a discrete simulator. It abstracts from the continuous-time behaviour and uses operational semantics rules and the expansion law to determine the next simulation step. We design it in such a way that it should be easy extendible to hybrid simulation.

2 Behaviour Hybrid Process Calculus

In this section we introduce main concepts of Behavioural Hybrid Process Calculus. See [7] for the details and proofs.

Trajectories. The continuous behaviour of hybrid systems can be seen as the set of continuous-time evolutions of system variables. We will call them *trajectories*. We assume that trajectories are defined over bounded time intervals $(0, t]$, and map to a *signal space* to define the evolution of the system. The signal space (\mathbb{W}) specifies the potentially observable continuous behaviour of the system. Components of the signal space correspond to the different aspects of the continuous behaviour, like temperature, pressure, etc. They are associated with *trajectories qualifiers* that identify them.

Hybrid Transition System. We define a hybrid transition system as a collection $HTS = \langle S, A, \rightarrow, W, \Phi, \rightarrow_c \rangle$, where S is a *state space*. The *discrete transition relation* $\rightarrow \subseteq S \times A \times S$ defines discrete changes annotated by actions ($a \in A$). The *continuous-time transition relation* $\rightarrow_c \subseteq S \times \Phi \times S$ links continuous changes to trajectories ($\varphi \in \Phi$).

Language. We introduce a language for defining hybrid processes.

$$B ::= \mathbf{0} \mid \mathbf{a} . B \mid [\varphi] . B \mid \bigoplus_{i \in I} B_i \mid B \parallel_A^H B \mid P$$

Action-prefix $\mathbf{a}.B$ defines a process that starts with action \mathbf{a} and afterwards engages in B . *Silent actions*[13] (denoted τ) are used to specify nondeterministic behaviour.

Trajectory-prefix $[\varphi].B$ models the behaviour of a process that executes a continuous trajectory φ and then continues as B .

Superposition $\bigoplus_{i \in I} B_i$ is a generalised operator on sets of behavioural expression. For action prefixes the interpretation of the operator is the same

as the ordinary choice operator Σ from classical process algebras. However, the choice among trajectories is made at the moment when the trajectories start bifurcating.

Parallel composition $B_1 \parallel_A^H B_2$ specifies the behaviour of two parallel processes. The operator explicitly attaches the sets of synchronising action names A and of synchronising trajectory qualifiers H . Synchronisation on actions has an interleaving semantics. Trajectory-prefixes can evolve in parallel only if the evolution of coinciding trajectory qualifiers is equal.

Recursion allows to define processes in terms of each other, as in the equation $B \triangleq P$, where B is the process identifier and P is a process expression that may only contain actions and signal types of B .

One of the main tools to compare systems is a *strong bisimulation*. The bisimulation for continuous dynamical systems is presented in [14]. The process algebraic version is discussed in [13]. A strong bisimulation for hybrid transition systems requires both systems to be able to execute the same trajectories and actions and to have the same branching structure. Strong bisimulation for BHPC is a congruence relation w.r.t. all operations defined above. See [7] for details.

BHPC is an assembly language for a modelling of hybrid systems. We add auxiliary constructs to increase usability of the language.

We introduce *parametrised action-prefix* $\mathbf{a}_v . B(v) \triangleq \bigoplus_{v \in V} \mathbf{a}_v . B(v)$ for convenience (as in [13, 53–58]).

We will use a *symbolic trajectory-prefix*, which extends a notion of ordinary trajectory-prefix by providing a set of continuous behaviours conforming to the certain conditions. We will define a *set of trajectory-prefixes* $[f \mid \Phi] . B(f) \triangleq \bigoplus_{\varphi \in \Phi} ([\varphi] . B(\varphi))$ where Φ is a set of trajectories and f is a *trajectory variable* for a trajectory. Furthermore, we will use $[t_1, \dots, t_m \mid \Phi \downarrow \mathcal{P}red \Downarrow \mathcal{P}red_{\text{exit}}]$ to define extended version of set of trajectory-prefixes, where t_1, \dots, t_m are trajectory qualifiers, which can be used to access corresponding parts of trajectories. Two types of restrictions on the set of trajectories are used: \downarrow states restrictions on the whole duration of trajectories and \Downarrow define the *exit conditions*, i.e., restrictions on the *end-points*.

Sometimes it is useful to check some conditions explicitly, and if they are not satisfied, to stop the progress of process. With the *guard* construct $\langle \mathcal{P}red \rangle . B$, these conditions can be given as a predicate.

Idling in BHPC is defined as $\text{idle} = [t \mid \dot{t} = 0]$, where t is a reserved variable. It does not manifest any observable behaviour, but reacts as soon, as it is invoked by another process, which communicates with the process, which follows the idling period.

Application of BHPC. Bouncing ball [7] is a simplified model of an elastic ball that is bouncing and losing a fraction of its energy with every bounce. The altitude of the ball is h , and v is a vertical speed, c is a coefficient for the lost energy. The ball moves according to the flow conditions and at the bounce time the variables are reassigned. In BHPC it can be defined in the following way:

$$\begin{aligned} \text{BB}(h_0, v_0) &\triangleq [h, v \mid \Phi(h_0, v_0) \Downarrow h = 0] . \text{BB}(0, -c * v) \\ \Phi(h_0, v_0) &= \{h, v : (0, t] \rightarrow \mathbb{R} \mid h(0) = h_0, v(0) = v_0, \dot{h} = v, \dot{v} = -g, h \geq 0\} \end{aligned}$$

Symbolic trajectory-prefix $[h, v \mid \Phi(h_0, v_0) \Downarrow h = 0]$ defines the dynamics of the ball until the bounce, and then the process continuously recursively calling itself with updated continuous variables $\text{BB}(0, -c * v)$. We extend the given specification by adding discrete actions to sense the elasticity of the bounce and increase the ball's kinetic energy, and a compensating controller (CC).

$$\begin{aligned} \text{BB}(h_0, v_0) &\triangleq [h, v \mid \Phi(h_0, v_0) \Downarrow h = 0] . \text{bounce}(c : [0, 1]). \\ &\quad [h, v \mid \Phi(0, -cv) \Downarrow v = 0] . \text{push}(v : \mathbb{R}) . \text{BB}(h, v) \\ \text{CC}(v_0) &\triangleq \text{idle} . \text{bounce}(c : [0, 1]) . \text{idle} . \text{push}((1 - c) v_0) . \text{CC}((1 - c) v_0) \\ \text{Sys}(h_0, v_0) &\triangleq \text{BB}(h_0, v_0) \parallel_{\text{push, bounce}}^v \text{CC}(v_0) \end{aligned}$$

3 Simulating BHPC

As the first step, we will focus on the simulation of discrete behaviour of BHPC. Discrete simulation can give a lot of valuable insight on the system. It helps to detect potential deadlocks. Discrete abstraction of the system can be verified and error-traces can be used in an hybrid simulator to investigate potential faults. Moreover, it comes handy in the early stages of modelling or prototyping.

To get discrete abstraction we make some choices concerning continuous-time behaviours. We discuss diverse choices for the operators individually.

- Parametrised action-prefix is left unchanged, only the parameters related with trajectories are ignored.
- We will interpret (symbolic) trajectory-prefix as a special type of action-prefix. Several options are possible. It can be seen as a silent (unobservable) action, just like the τ action described in [13]. A special action can be introduced to denote any trajectory-prefix. Then trajectory-prefixes can be treated like ordinary action-prefixes.
- Guard is treated as usually, but with all trajectories-related predicates evaluated to **true** or **false**, depending on the simulation purpose.
- With the trajectory-prefix reduced to a discrete action, the superposition operator is equal to the choice operator from ordinary process algebra.
- Without the set of synchronising trajectory qualifiers, the parallel composition operator from BHPC becomes equal to the parallel composition operator from ordinary process algebra.

Operational semantics for the language already define the transitions the simulator can take, only parallel composition requires additional reshaping. It is provided by the expansion law [7], which expresses parallel composition as a superposition of processes.

4 Future Plans

Future plans for the simulation of BHPC and the calculus include several directions.

The discrete simulator is being designed in such a way that extending it to the hybrid simulation should be doable without completely reshaping the discrete simulation part. Principally, it means adding support for continuous-time behaviour: interpretation of trajectory qualifiers, interface to solvers, etc.

Moreover, we are building the current tool not as a prototype of an industrial tool, but more as a hybrid "*sand-box*", a place to experiment with BHPC and related developments. Consequently, the architecture and implementation of the tool are being designed in such a way that it is easy to accommodate the changes in the calculus and to test the algorithms developed for hybrid systems in BHPC framework. Adaptable and well documented interfaces for ODE/DAE solvers should be provided for experimenting with different approaches for continuous-time behaviour simulation. Co-simulation also takes place in the plans. Theoretical and practical issues of the co-simulation of BHPC and Simulink³ are explored as a part of WP3 of HYCON⁴. Furthermore, the means to export a restricted subset of BHPC to Modelica™ [10] are investigated.

Examination of different simulation results visualisation techniques are of interest too. Just ordinary graphs (plots) usually used to display continuous-time simulation results do not provide sufficient information about switching behaviour. Event-traces and the message sequence charts [15] are appropriate and even beneficial when discrete behaviour should be visualised. However both techniques become inadequate when combination of continuous-time and discrete behaviour should be visualised. It calls for a combination of the aforementioned techniques or even new unconventional approaches.

One more interesting development of BHPC is *model-based testing* of hybrid systems [16]. In model-based testing we use a formal specification (a model) of the desired behaviour of the system under test (SUT). The model is used to select the input for the SUT. The model is also used to check the correctness of the output of the SUT after a certain input. One of the great advantages of model-based testing is that tools can explore the model and automatically generate and execute tests cases from the model. BHPC and the simulation tool can be extended to generate required information for a well-known on-the-fly testing tool TORX [17], and in combination with it form a hybrid systems testing framework.

5 Results

In this paper we proposed a simulation technique for Behavioural Hybrid Process Calculus. The calculus and transitions system were introduced, operators for the

³ <http://www.mathworks.com/products/simulink/>.

⁴ WP3, HYCON, <http://wp3.hycon.bci.uni-dortmund.de/>.

calculus were explained. We focussed discussion on the discrete simulation of selected operators, by abstracting from the continuous-time behaviour so that all operators from our calculus have corresponding interpretation in ordinary process algebra. The expansion law is used to resolve parallel composition. The operational semantics defines the possible transitions for the simulator.

The work in progress will have to evaluate the conceptual and practical implications of our approach. Currently we are developing techniques and tools for discrete and hybrid simulation of the calculus.

Acknowledgements. The authors thank H. Brinksma for his comments.

References

1. Alur, R., Courcoubetis, C., Henzinger, T., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H., eds.: Hybrid Systems. Volume 736 of LNCS., Springer (1993) 209–229
2. Julius, A.: On Interconnection and Equivalence of Continuous and Discrete Systems: A Behavioral Perspective. PhD thesis, SSCG, Univ. of Twente (2005)
3. Lynch, N., Segala, R., Vaandrager, F.: Hybrid I/O automata. *Information and Computation* **185** (2003) 105–157
4. Cuijpers, P.J.L., Reniers, M.A.: Hybrid process algebra. Technical report, Dept. of Math. and Comp. Science, Tech. Univ. of Eindhoven (TU/e), Eindhoven (2003)
5. Bergstra, J., Middelburg, C.: Process algebra for hybrid systems. Technical report, Dept. of Math. and Comp. Science, Tech. Univ. of Eindhoven (TU/e), Eindhoven (2003)
6. van Beek, D., Man, K., Reniers, M., Rooda, J., Schiffelers, R.: Syntax and consistent equation semantics of hybrid chi. Report CS-Report 04-37, Tech. Univ. of Eindhoven (TU/e), Eindhoven (2004)
7. Brinksma, E., Krilavičius, T.: Behavioural hybrid process calculus. Technical Report TR-CTIT-05.45, CTIT, University of Twente (2005)
8. Lee, E., Zheng, H.: Operational semantics of hybrid systems. In: Hybrid Systems: Computation and Control. LNCS (2005) 25–53
9. Hamon, G., Rushby, J.: An operational semantics for STATEFLOW. In Wermelinger, M., Margaria-Steffen, T., eds.: FASE 2004. LNCS (2004) 229–243
10. Modelica Association: Modelica - A Unified Object Oriented Language for Physical Systems Modeling: Language Specification. (2005)
11. van Amerongen, J., Breedveld, P.: Modelling of physical systems for the design and control of mechatronic systems. *Annual Reviews in Control* **27** (2003) 87–117
12. Polderman, J., Willems, J.C.: Introduction to Mathematical Systems Theory: a behavioral approach. Springer (1998)
13. Milner, R.: Communication and concurrency. Prentice-Hall, Inc. (1989)
14. van der Schaft, A.: Bisimulation of dynamical systems. In Alur, R., Pappas, G.J., eds.: HSCC. Volume 2993 of LNCS., Springer (2004) 555–569
15. ITU-T: Recommendation Z.120. Message Sequence Charts. Technical Report Z-120, Int. Tel. Union, Genève (2000)
16. Berkenkötter, K., Kirner, R.: Real-Time and Hybrid Systems Testing. In: Model-based Testing of Reactive Systems. Volume 3472 of LNCS. Springer (2005) 355–387
17. Bohnenkamp, H., Belinfante, A.: Timed testing with TORX. In: Formal Methods Europe. Volume 3582 of LNCS., Springer (2005) 173 – 188