PROCESS ALGEBRAIC APPROACH TO HYBRID SYSTEMS

Ed Brinksma [*] **Tomas Krilavičius** [*,1]
Yaroslav S. Usenko [*]

[*] *FMT, EEMCS, University of Twente, P.O.Box 217, 7500
AE Enschede, The Netherlands,
{brinksma,t.krilavicius,usenko} at utwente.nl*

Abstract: Process algebra is a theoretical framework for the modelling and analysis of the behaviour of concurrent discrete event systems that has been developed within computer science in past quarter century. It has generated a deeper understanding of the nature of concepts such as observable behaviour in the presence of nondeterminism, system composition by interconnection of concurrent component systems, and notions of behavioural equivalence of such systems. It has contributed fundamental concepts such as bisimulation, and has been successfully used in a wide range of problems and practical applications in concurrent systems. The basic tenets of process algebra are highly compatible with the behavioural approach to dynamical systems. In the contribution an extension of classical process algebra that is suitable for the modelling and analysis of continuous and hybrid dynamical systems is presented. It provides a natural framework for the concurrent composition of such systems, and can deal with nondeterministic behaviour that may arise from the occurrence of internal switching events. Standard process algebraic techniques lead to the characterization of the observable behaviour of such systems as equivalence classes under some suitably adapted notion of bisimulation. *Copyright* © *2005 IFAC.*

Keywords: Formal specifications, Formal methods, Behavioural science, Algebraic approaches, Modelling

1. INTRODUCTION

The growing interest in hybrid systems both in computer science and control theory has generated a new interest in models and formalism that can be used to specify and analyze such systems. A prominent framework for hybrid systems is provided by the family of hybrid automata models (hybrid automata (Alur *et al.*, 1993; Henzinger, 1996), hybrid behavioural automata (Julius *et al.*, 2002), hybrid input/output automata (Lynch *et al.*, 2003)). More recently process algebraic models have been put forward as a vehicle for the study of hybrid systems (Cuijpers and Reniers, 2003; Bergstra and Middelburg, 2003).

Process algebra (Milner, 1989; Hoare, 1985; Bergstra and Klop, 1984; Bolognesi and Brinksma, 1987) is a theoretical framework for the modelling and analysis of the behaviour of concurrent discrete event systems that has been developed within computer science in past quarter century. It has generated a deeper understanding of the nature of concepts such as observable behaviour in the presence of nondeterminism, system composi-

tion by interconnection of concurrent component systems, and notions of behavioural equivalence of such systems. It has contributed fundamental concepts such as bisimulation, and has been successfully used in a wide range of problems and practical applications in concurrent systems.

We believe that the basic tenets of process algebra are highly compatible with the behavioural approach to dynamical systems. In our contribution we present an extension of classical process algebra that is suitable for the modelling and analysis of continuous and hybrid dynamical systems that can be seen as a generalization of the behavioural approach in a hybrid setting. It provides a natural framework for the concurrent composition of such systems, and can deal with nondeterministic behaviour that may arise from the occurrence of internal switching events. Standard process algebraic techniques lead to the characterization of the observable behaviour of such systems as equivalence classes under some suitably adapted notion of bisimulation, yielding a potentially interesting mathematical interpretation of the notion of hybrid behaviour. A technical advantage of our approach is that, in contrast to (Cuijpers and Reniers, 2003; Bergstra and Middelburg, 2003) strong bisimulation is a congruence relation with respect to the parallel composition of subsystems, i.e. substitution of a subsystem by a bisimilar subsystem does not affect the behaviour of the composition.

2. BEHAVIOURAL HYBRID PROCESS CALCULUS

In this section hybrid transition systems and a related notion of strong bisimulation relation are defined. Then a hybrid basic language based on the transition systems is defined. The syntax of the language is presented and its operators are explained.

2.1 Trajectories

To make this process algebraic framework work the concept of trajectory should be explained a bit. It is assumed that trajectories are defined over bounded time intervals $(0, t]$, and map to a *signal space* to define the evolution of the system. Components of the signal space correspond to different aspects of the continuous-time behaviour, like temperature, pressure, etc. They are associated with type names that identify them.

Definition 1. (Signal space). Let $\mathcal{W}$ be a set of *signal domains* (typically $\subseteq \mathbb{R}$), and $\mathcal{T}$ be a set of *signal type names*. A *signal space* $\mathbb{W}$ is a tuple $(W_1 \times \cdots \times W_n, t_1 \times \cdots \times t_n)$ with $W_i \in \mathcal{W}, t_i \in \mathcal{T}$,

where t_i denotes the signal type of W_i, and $t_i \neq t_j$ for $i \neq j$, i.e. all W_i have a different type.

Definition 2. (Trajectory). Let $\mathbb{W}$ be the signal space. Then a *trajectory* is a mapping $\varphi : (0, t] \to \mathbb{W}$, where $t \in \mathbb{R}^+$ is the duration of the trajectory, also denoted as $\mathrm{t}(\varphi)$.

The signal space $\mathbb{W}$ specifies the potentially observable continuous-time behaviour of the system. Φ will be used to denote a set of trajectories.

Definition 3. (Projection). Let $\varphi : (0, t] \to \mathbb{W}$ be a trajectory. Then a *projection* of trajectory to a type t_i of $\mathbb{W}$ is the trajectory $\pi^{t_i}(\varphi) : (0, t] \to \mathbb{W}_i$, with $\mathbb{W}_i = (W_i, t_i)$.

If the signal types of two trajectories coincide, they can be concatenated to one trajectory, which is not necessary continuous.

Definition 4. (Concatenation of trajectories). Let $\varphi : (0, t] \to \mathbb{W}$ and $\psi : (0, u] \to \mathbb{W}$ be trajectories. The *concatenation* of φ and ψ is given by the trajectory $\phi; \psi : (0, t + u] \to \mathbb{W}$ defined as:

$$\varphi \,;\, \psi(t') = \begin{cases} \varphi(t'), & 0 < t' \leqslant t \\ \psi(t' - t), & t < t' \leqslant t + u \end{cases}$$

2.2 Definition of hybrid transition system

A hybrid transition system is a labelled transition system with two types of transitions.

Definition 5. (HTS). A *hybrid transition system* is a tuple $HTS = \langle S, A, \to, W, \Phi, \to_c \rangle$, where

- S is a *state space*,
- A is a *set of (discrete) action names*,
- $\to \subseteq S \times A \times S$ is a *(discrete) transition relation*,
- $\mathbb{W}$ is a *signal space*,
- Φ is a *set of (initial) trajectories* $\varphi : (0, t] \to \mathbb{W}$ for $t \in \mathbb{R}^+$,
- $\to_c \subseteq S \times \Phi \times S$ is a *(continuous time) transition relation*,

and $s \xrightarrow{a} s'$ iff $(s, a, s') \in \to$ and $s \xrightarrow{\varphi} s'$ iff $(s, \varphi, s') \in \to_c$.

The set of discrete action names includes a *silent action*, denoted τ. It does not represent a potential communication and is not directly observable. Silent action may be used to specify a nondeterministic behaviour (as *internal actions* in (Milner, 1989, Pages 37–43)).

Density is required, i.e. for all $0 < u < t : s \xrightarrow{\varphi}_t s' \Leftrightarrow \exists s'', \varphi_1, \varphi_2 : \varphi = \varphi_1 \,;\, \varphi_2 \wedge s \xrightarrow{\varphi_1}_u s'' \wedge$

$s'' \xrightarrow{\varphi_2}_{t-u} s'$. This requirement allows us to split every trajectory into arbitrarily many parts.

Remark 6. (Notation). Greek alphabet symbols (like φ, ψ) are used to denote trajectories, which are taken on a continuous transition, and Latin alphabet (like a, b) for an action.

The label φ in $s \xrightarrow{\varphi} s'$ is a *semantic object*, viz. the set theoretic graph of the function φ.

Remark 7. The above property of density does not suffice in general, because it allows pathological transition systems, see (Jeffrey *et al.*, 1993). However, the defined process calculus cannot describe such pathological cases, so that our definition suffices.

Remark 8. Note that the trajectory transitions can be non-deterministic.

2.3 Bisimulation

One of the main tools to compare systems is strong bisimulation. The bisimulation for continuous dynamical systems is presented in (van der Schaft, 2004). The process algebraic version is nicely explained in (Milner, 1989). 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.

Definition 9. (Strong bisimulation). A binary relation $\mathcal{R} \subseteq S \times S$ on the states is a *bisimulation*, if for all $p, q \in S$, such that $p \, \mathcal{R} \, q$, holds

$$p \xrightarrow{a} p' \Longrightarrow \exists q' \text{ such that } q \xrightarrow{a} q' \text{ and } p' \, \mathcal{R} \, q'$$
$$q \xrightarrow{a} q' \Longrightarrow \exists p' \text{ such that } p \xrightarrow{a} p' \text{ and } p' \, \mathcal{R} \, q'$$
$$p \xrightarrow{\varphi} p' \Longrightarrow \exists q' \text{ such that } q \xrightarrow{\varphi} q' \text{ and } p' \, \mathcal{R} \, q'$$
$$q \xrightarrow{\varphi} q' \Longrightarrow \exists p' \text{ such that } p \xrightarrow{\varphi} p' \text{ and } p' \, \mathcal{R} \, q'.$$

The first two statements define bisimulation requirements for the discrete actions, and the last two for the continuous-time transitions.

Definition 10. (Bisimilarity). Two states p and q are *bisimilar* (denoted $p \leftrightarrows q$), if there exists a bisimulation $\mathcal{R}$, containing the pair (p, q).

2.4 Language

To define evolution and interaction of systems, a language, based on hybrid transition system (Subsection 2.2) is introduced. The syntax of
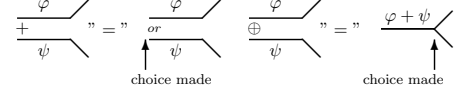

Fig. 1. Superposition

language is presented in BNF notation (Backus-Naur form).

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

The language can be easily extended with renaming, hiding operators (like in (Brinksma *et al.*, 2004)).

Syntactic functions $\mathfrak{L}(P)$ and $\mathcal{N}(P)$ will be used for collecting action and signal type names occurring in P, respectively.

2.4.1. Action-prefix $a.P$ The ordinary well-known action-prefix. Process $a.P$ describes a process which engages in the action a and then behaves as described by P.

2.4.2. Trajectory-prefix $[\varphi].P$ It denotes the hybrid behaviour that starts with a continuous trajectory denoted by φ and is followed by the behaviour specified by P.

2.4.3. Superposition $\bigoplus\{P(\overline{v}) \mid \overline{v} \in I\}$ *Superposition* is a generalized operator on sets of behaviour expressions. To generate the set arbitrary index sets I are allowed. It can be thought of as a generalization of the choice or summation operator Σ in ordinary process algebra. Indeed, if all $P(\overline{v})$ are of the form $a_{\overline{v}}.P(\overline{v})$ then the intended interpretation of

$$\bigoplus\{P(\overline{v}) \mid \overline{v} \in I\} \text{ or } \bigoplus_{\overline{v} \in I} P(\overline{v}) \quad \text{is} \quad \Sigma_{\overline{v} \in I} P(\overline{v})$$

The inactive process 0 or *STOP* can be introduced as $\bigoplus \varnothing$.

The difference between $\bigoplus$ and Σ becomes apparent in the case of trajectory-prefixes: when two trajectories are superposed the choice between them is not made at the time of superposition, but at the time when the trajectories start bifurcating. This is illustrated in Figure 1. If it is necessary to make a choice before at the beginning, it can be done using silent action. In $\tau.[\varphi] \oplus \tau.[\varphi]$ choice is made at the beginning of trajectory.

2.4.4. Parallel composition $P_1 \parallel_A^H P_2$ Parallel composition models concurrent evolution of several processes. During the evolution they may interact with each other via synchronization on discrete and continuous-time transitions. In

BHPC synchronization on identical names is assumed as the basic synchronization concept. In order to avoid context-dependent interpretations of operators, the set of action names A and the set of signal type names H that are subject to synchronization, are made explicit in the parallel operator $\|_A^H$.

This form of synchronization implies that parallel components jointly execute identical actions or trajectories with common signal evolutions that occur in their transitions and are subject to synchronization.

The basic idea of synchronizing trajectories is not much different than that of synchronizing on actions. Let $\mathcal{W}$ be a set of signal domains and

$$\varphi : (0, t] \to \mathbb{W}_\varphi, \quad \psi : (0, t] \to \mathbb{W}_\psi$$
$$\mathcal{T}_\varphi = \{t \mid \exists W \in \mathcal{W}, \pi^t(\varphi) = (W, t)\}$$
$$\mathcal{T}_\psi = \{t \mid \exists W \in \mathcal{W}, \pi^t(\psi) = (W, t)\}$$
$$\chi : (0, t] \to \mathbb{W} \text{ such that}$$
$$\forall t \in \mathcal{T}_\varphi \cap \mathcal{T}_\psi \quad \pi^t(\varphi) = \pi^t(\varphi) = \pi^t(\chi),$$
$$\forall t \in \mathcal{T}_\varphi \quad \pi^t(\varphi) = \pi^t(\chi),$$
$$\forall t \in \mathcal{T}_\psi \quad \pi^t(\psi) = \pi^t(\chi)$$

The static constraint is imposed, that $P_1 \|_A^H P_2$ is only well-formed *iff* $\mathfrak{L}(P_1) \cap \mathfrak{L}(P_2) \subseteq A$ and $\mathcal{N}(P_1) \cap \mathcal{N}(P_2) \subseteq H$.

Let P_1 and P_2 can have trajectories φ and ψ, respectively. If $\mathcal{T}_\varphi \cap \mathcal{T}_\psi \subseteq H$ and $\forall t \in \mathcal{T}_\varphi \cap \mathcal{T}_\psi$ $\pi^t(\varphi) = \pi^t(\psi)$, then χ is a synchronized trajectory of $P_1 \|_A^H P_2$ that simultaneously changes the states of P_1 and P_2.

2.4.5. Recursion The ordinary process algebraic recursion extended to work with trajectory prefix. It allows to define processes in terms of each other, like in equation $B =_{def} P$, where B is a process identifier and actions and signal types of P are only allowed actions and signal types in B.

2.5 Congruence Property

The bisimulation relation (equivalence) defined for the HTSs in Subsection 2.3 is a congruence relation w.r.t all operations defined in Subsection 2.4. It is shown in (Brinksma *et al.*, 2004) using the existing meta-theory (Aceto *et al.*, 2001) of congruence formats for the transition systems defined by means of SOS rules. It can be done because the definition of bisimulation for HTSs coincides with the bisimulation relation induced by the transition relations used in SOS rules (Brinksma *et al.*, 2004).
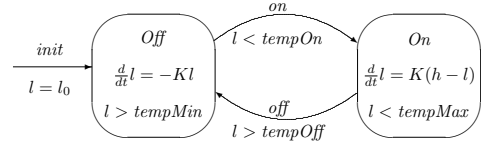

Fig. 2. A thermostat

Theorem 11. Strong bisimulation equivalence on HTSs is a congruence w.r.t the operations of BHPC defined by the in Section 2.4.

3. APPLICATIONS

3.1 Derived constructs

BHPC is an assembly language for a specification of hybrid systems. Additional language constructs should be added to make it more readable and easier to use.

3.1.1. Set of Continuous Behaviours The trajectory-prefix defines a trajectory with fixed duration. To define set of trajectories additional construct is introduced. It defines a set of trajectories, for which certain conditions hold.

$$[\varphi \mid \Phi \rhd Pred] . P(\varphi) =_{def} \bigoplus_{\substack{\varphi \in \Phi \\ Pred(\mathrm{t}(\varphi))}} ([\varphi] . P(\varphi))$$

where φ defines evolution of dynamics, and *Pred* defines *exit conditions*.

3.1.2. Infinite Continuous Behaviour Trajectory-prefix defines only finite continuous behaviour. To define an *infinite continuous behaviour* additional construct is necessary.

Definition 12. Let $\varphi : (0, t] \to \mathbb{W}$ then the *infinite continuous behaviour* is defined as follows $[\varphi] =_{def} \bigoplus_{t>0} [\varphi \upharpoonright (0, t]]_t . 0$ where $0 =_{def} \oplus \varnothing$.

3.2 Examples

To illustrate application of BHPC several examples are given.

3.2.1. Thermostat

Example 13. (Thermostat). A thermostat is one of the main introductory examples of hybrid systems. It is used in many sources (Alur *et al.*, 1994; Henzinger, 1996; Bergstra and Middelburg, 2003; Schiffelers *et al.*, 2003). The temperature of a room is controlled by a thermostat, which continuously senses the temperature and switches a heater on and off. When the heater
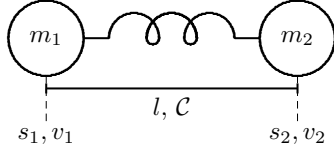
Fig. 3. Two masses and a spring

is off, the temperature decreases according to the exponential function $l(t) = l_0 e^{Kt}$, where t is time, l is the temperature in the room, l_0 is the initial temperature, and K is a constant determined by the room. When it is on, the temperature increases according to the function $l(t) = l_0 e^{-Kt} + h(1 - e^{-Kt})$, where h is a constant depending on the power of the heater. The temperature should be maintained between *tempMin* and *tempMax*. *tempOn* and *tempOff* are a maximal temperature and the minimal temperatures, when the heater can be turned on and off, respectively.

A hybrid automaton of the thermostat is shown in Figure 2. It starts with the initial temperature $l_0 \in [tempMin, tempMax]$. In locations *Off* and *On* the heater is off and on, respectively, and the temperature changes according to the flow conditions. And the same model in BHPC:

$$ThOff(l_0) =_{def}$$
$$[l \mid \Phi_{Off}(l_0) \triangleright tempOn \geqslant l \geqslant tempMin] \,.$$
$$\text{on}. ThOn(l)$$
$$ThOn(l_0) =_{def}$$
$$[l \mid \Phi_{On}(l_0) \triangleright tempOff \leqslant l \leqslant tempMax] \,.$$
$$\text{off}. ThOff(l)$$
$$\Phi_{Off}(l_0) = \{l : (0, t] \to \mathbb{R} \mid l(0) = l_0, \dot{l} = -Kl\}$$
$$\Phi_{On}(l_0) =$$
$$\{l : (0, t] \to \mathbb{R} \mid l(0) = l_0, \dot{l} = K(h - l)\}$$

3.2.2. Two Masses and a Spring

Example 14. Consider a simple system, depicted in the figure 3 consisting of the two masses and a spring. Let weights be m_1 and m_2, displacements from the reference points be s_1 and s_2 , and speeds be v_1 and v_2 correspondingly of the first mass and the second mass. The length of spring in the state of rest is l, and $\mathcal{C}$ is an elasticity of the spring. Then the system can be modelled as follows.

$$Mass(m, s_0, v_0) =_{def}$$
$$[s, v, f \mid s(0) = s_0, v(0) = v_0, \dot{s} = v, \dot{v} = mf]$$
$$Spring(l, c) =_{def}$$
$$[s_l, s_r, f_l, f_r \mid f_l = c(s_r - s_l - l), f_r = -f_r]$$
$$System =_{def} \text{new } f_l, f_r$$
$$\big(Mass(m_1, s_{01}, v_{01})[s_l/s, v_l/v, f_l/f]$$
$$\|_{s_l, f_l} Spring(l_0, \mathcal{C})$$
$$\|_{s_r, f_r} Mass(m_2, s_{02}, v_{02})[s_l/s, v_l/v, f_l/f] \big)$$

4. RELATED WORK

There are several different ways to work with hybrid systems. Overviews of different approaches are given in (Labinaz *et al.*, 1996; Antsaklis and Koutsoukos, 1998; van der Schaft and J.M.Schumacher, 2000).

In (Cuijpers and Reniers, 2003) HyPA ACP is extended with a disrupt, a flow and a reinitialization-clause to cope with hybrid behaviour. Alternative composition in HyPA is non-deterministic for both discrete and continuous actions. The passage of time influences the valuation of the model variables and can introduce choices in the system behaviour. Choice is done before action. In parallel composition flow-clauses are forced to synchronize, and can only do it if they accept the same solutions. In contrast to our calculus the strong bisimulation is not a congruence relation with respect to the parallel composition of subsystems.

χ (Schiffelers *et al.*, 2003) is a process algebra, with a two ways for a choice between the processes: a choice and an alternative composition operators. The first one is a non-deterministic for discrete actions and does not make choice for continuous actions. The second is the same for discrete actions, but uses the weak time-determinism principle, i.e. the passage of time cannot result in making a choice, if both alternatives can perform the transition with the same trajectory and the same time step. If one of the processes can perform a time transition and the other cannot, then the alternative is lost. Closed world assumption is used in modelling systems in hybrid χ.

ACP_{hs}^{srt} (Bergstra and Middelburg, 2003) is an ACP extension for hybrid systems. One of the main differences is in the choice of operations. The definition of hybrid transition system is almost the same. But instead of having a trajectory prefix operation, a *continuous signal insertion operator* $(\varphi \frown x)$ is used, in order to decorate the process x by a logical formula. This formula defines a class of trajectories the process can follow by remaining in the initial state location (state) of x. $(\sigma_{rel}^r(x))$ denotes a relative time delay of r time units, during which the continuous signal could be emitted. And in contrast with BHPC , the strong bisimulation is not a congruence relation with respect to the parallel composition of subsystems.

One of the most popular approaches to model and analyze hybrid systems is (Alur *et al.*, 1993; Henzinger, 1996) hybrid automata. In (Brinksma *et al.*, 2004) is shown, that hybrid automaton can be translated to BHPC . Inverse translation may be a lot more complex, if possible at all, because there are no corresponding constructs for the superposition and a continuous part of the parallel

composition in hybrid automata. In (Julius *et al.*, 2002; Julius, 2005) hybrid behavioral automata – a modification of hybrid automata based on use of behavioral theory is presented.

In (Lynch *et al.*, 2003) well known Input/Output Automata approach is extended to cope with hybrid systems resulting in Hybrid Input/Output Automata.

5. CONCLUSION AND FUTURE WORK

In this paper the hybrid process calculus BHPC and the underlying concept of a hybrid transition system were introduced, and their application was illustrated by a number of small examples. Together with a suitable adaptation of the classical notion of bisimulation this approach yields a mathematical interpretation of hybrid behaviour, viz. as equivalence classes of hybrid transition systems modulo bisimulation, that can been interpreted as a generalization of the behavioural approach to classical dynamic systems. In particular, this introduces a notion of nondeterminism into (hybrid) behaviour that has proved indispensable for the study of discrete concurrent systems in computer science.

Future work will have to evaluate the conceptual and practical implications of our approach. In particular, our plans include:

- detailed comparison with related models and formalisms, such as hybrid automata and other applications of process algebra to hybrid systems;
- development of analytical techniques for hybrid systems in the BHPC framework.

REFERENCES

Aceto, L., W.J. Fokkink and C. Verhoef (2001). Structural operational semantics. In: *Handbook of Process Algebra* (J. A. Bergstra, A. Ponse and S. A. Smolka, Eds.). Chap. 3, pp. 197–292. Elsevier.

Alur, R., C. Courcoubetis, T.A. Henzinger and P.-H. Ho (1993). Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In: *Hybrid Systems*. Vol. 736 of *LNCS*. Springer. pp. 209–229.

Alur, R., C. Courcoubetis, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine (1994). The algorithmic analysis of hybrid systems. In: *ICAOS 1994*. Vol. 199 of *LNCIS*. pp. 331–351. Springer.

Antsaklis, P. J. and X. D. Koutsoukos (1998). On hybrid control of complex systems: A survey. In: *Automation of Mixed Processes: Dynamic Hybrid Systems*. pp. 1–8.

Bergstra, J.A. and J.W. Klop (1984). Process algebra for synchronous communication. *Information and Computation* **60**(1/3), 109–137.

Bergstra, J. A. and C. A. Middelburg (2003). Process algebra for hybrid systems. Technical report. Dept. of Mathematics and Computer Science, Technical University Eindhoven.

Bolognesi, T. and E. Brinksma (1987). Introduction to the ISO Specification Language LOTOS. *Computer Networks* **14**, 25–59.

Brinksma, E., T. Krilavičius and Y.S. Usenko (2004). Behavioural hybrid process calculus. *http:\\www.cs.utwente.nl\˜krilaviciust*.

Cuijpers, P.J.L. and M.A. Reniers (2003). Hybrid process algebra. Technical report. Dept. of Mathematics and Computer Science, Technical University Eindhoven.

Henzinger, T.A. (1996). The Theory of Hybrid Automata. In: *LICS 1996*. pp. 278–292.

Hoare, C.A.R. (1985). *Communicating Sequential Processes*. Prentice-Hall, Inc.

Jeffrey, A.S.A., S.A. Schneider and F.W. Vaandrager (1993). A comparison of additivity axioms in timed transition systems. Report CS-R9366. CWI. Amsterdam.

Julius, A. A. (2005). On interconnection and equivalence of continuous and discrete systems: a behavioral perspective. PhD thesis. Systems Signals and Control Group, University of Twente.

Julius, A. A., S. N. Strubbe and A. J. van der Schaft (2002). Compositional modeling of hybrid systems with hybrid behavioral automata. *http:\\www.math.utwente.nl\˜julius*.

Labinaz, G., M. M. Bayoumi and K. Rudie (1996). Modeling and Control of Hybrid Systems: A Survey. In: *IFAC 13th Triennial World Congress*.

Lynch, N.A., R. Segala and F.W. Vaandrager (2003). Hybrid I/O automata. *Inf. Comput.* **185**(1), 105–157.

Milner, R. (1989). *Communication and concurrency*. Prentice-Hall, Inc.

Schiffelers, R. R. H., D. A. van Beek, K. L. Man, M. A. Reniers and J. E. Rooda (2003). Formal semantics of hybrid chi. In: *Int. Workshop on Formal Modeling and Analysis of Timed Systems*. Springer.

van der Schaft, A. (2004). Bisimulation of dynamical systems. In: *HSCC 2004*. Vol. 2993 of *LNCS*. Springer. pp. 555–569.

van der Schaft, A. J. and J.M.Schumacher (2000). *An Introduction to Hybrid Dynamical Systems*. LNCIS. Springer. London.