# Behavioural Hybrid Process Calculus

# draft

Ed Brinksma [a] Tomas Krilavičius [a,1]

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

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. 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 characterisation of the observable behaviour of such systems as equivalence classes under some suitably adapted notion of bisimulation.

*Key words:* Formal specifications, Formal methods, Behavioural science

## 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

[1] Supported by the NWO project CASH

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 [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, van Beek et al., 2004].

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 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.

We believe that the basic tenets of process algebra are highly compatible with the behavioural approach to dynamical systems [Polderman and Willems, 1998]. 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 generalisation 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 characterisation 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 [2], i.e., substitution of a subsystem by a bisimilar subsystem does not affect the behaviour of the composition.

In this chapter we propose a process algebraic calculus that extends the standard repertoire of operators that combine discrete functional behaviour with features to also represent and compose continuous-time behaviour. As mentioned above, we are inspired by the so-called behavioural approach to dynamic systems due to Polderman and Willems [1998]. In control theory, which is the relevant context in our case, the traditional presentation of dynamic

[2] In Cuijpers and Reniers [2003] the robust and stateless bisimulations are congruent.

behaviour, assumes as given, a number of continuous-time input and output variables, whose evolutions, respectively, influences and depend on the evolution of state variables. This evolution is typically defined in terms of differential equations.

Although in practice most dynamical systems are ultimately described in this format, Willems' behavioural approach starts from a more general point of view. System behaviour is characterised by a time-dependent relation between the observable or *manifest* variables of a system. Input and output become derived notions that depend on the constraints that the overall relation imposes on the individual variables. Thus behaviour can be simply seen as the set of all allowed real-time evolutions, or *trajectories*, of the system variables.

The notion of input and output as derived concepts is also well-known in process algebras with communication based on (symmetric) instantaneous synchronisation. It suggests that communication on continuous-time variables can be achieved by non-instantaneous synchronisation on (parts of) trajectories. This leads to a calculus of actions, for discrete behaviour, and trajectories, for continuous-time behaviour. As we will see, the calculus does not depend upon any particular representation of sets of allowed trajectories: it simply defines the behaviour of composed, hierarchical systems in terms of the allowed actions and trajectories of its component systems. This leads to a natural separation of concerns in which control theory is used to determine qualitative properties of dynamical behaviour (e.g., stability, controllability, etc.), and the proposed calculus describes how these propagate under complex system compositions.

Based on the above approach, this chapter introduces the concept of hybrid transition systems and defines the related notion of strong (hybrid) bisimulation that captures a natural notion of equivalent behaviour. This leads to a branching-time interpretation of hybrid behaviour, in which behaviour is not characterised by sets of trajectories and action traces, but by tree-like structures that capture also the moments in time when a choice between alternative behaviours exists.

Subsequently, a basic language for the construction of hybrid transition systems is defined. The syntax of the language is presented and its operators are explained.

## 2 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. Components of the signal space correspond to different aspects of the continuous-time behaviour,

like temperature, pressure, etc. They are associated with *trajectories qualifiers* 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* trajectories qualifiers. *A* signal space *is a tuple*

$$\mathbb{W} = (W_1 \times \cdots \times W_n, t_1 \times \cdots \times t_n)$$

*with $W_i \in \mathcal{W}, t_i \in \mathcal{T}$, where $t_i$ denotes the trajectory qualifier of $W_i$, and $t_i \neq t_j$ for $i \neq j$, i.e., all $W_i$ have different trajectory qualifiers.*

**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. Usually trajectories are defined over infinite time intervals (like in Example 6). However, hybrid systems usually evolve according to some trajectory only for a certain period of time. The restriction to interval $(0, t]$ allows to define such evolutions. In Section 6.3 we provide a tool to define infinite trajectories, thus such definition does not cause any inconvenience or loss of generality.

**Notation 2.1** *We will use $\Phi$ to denote a set of trajectories. For brevity reasons instead of writing $\varphi \restriction (0, t]$ we will write $\varphi \restriction t$, where $\varphi$ is a trajectory.*

**Definition 3 (Projection)** *Let $\varphi : (0, u] \to \mathbb{W}$ be a trajectory, such that $\mathbb{W} = (W_1 \times \cdots \times W_n, t_1 \times \cdots \times t_n)$. Then a* projection *of trajectory to a trajectory qualifier $t_i\,(i = 1, \ldots, n)$ is the trajectory*

$$\pi^{t_i}(\varphi) : (0, u] \to \mathbb{W}_i$$

*with $\mathbb{W}_i = (W_i, t_i)$ from $\mathbb{W}$.*

**Definition 4 (Trajectory qualifier)** *Let $\varphi : (0, u] \to \mathbb{W}$ be a trajectory. Then the function $\mathrm{T} : \Phi \to \mathcal{T}$, where $\Phi$ is a set of trajectories and $\mathcal{T}$ is a set of* trajectory qualifiers*, collects all trajectory qualifiers of the trajectory:*

$$\mathrm{T}(\varphi) = \{t \mid \exists W \in \mathcal{W}, \pi^t(\varphi) = (W, t)\}.$$

**Remark 5 (Extended projections)** *We will write projection for a set of trajectory qualifiers*

$$\pi^{\mathcal{T}'}(\varphi) : (0, u] \to \mathbb{W}_{\mathcal{T}'}$$

*with $\mathbb{W}_{\mathcal{T}'} = (W_1 \times \cdots \times W_m, t_1 \times \cdots \times t_m)$, $\{t_1, \ldots, t_m\} = \mathcal{T}'$ and $\forall t_i \in \mathcal{T}'\ \pi^{t_i}(\varphi) = \pi^{t_i}(\pi^{\mathcal{T}'}(\varphi))$.*

**Example 6 (Trajectories and projections)** *Let* $\mathbb{W}_{\mathrm{BB}} = (\mathbb{R}_+ \times \mathbb{R}, \mathit{Altitude} \times \mathit{Velocity})$ *be the signal space for bouncing ball example (Example 7). Then the trajectory for the bouncing ball can be defined as a mapping*

$$\varphi : (0, t] \to (\mathbb{R}_+ \times \mathbb{R}, \mathit{Altitude} \times \mathit{Velocity}) ,$$

*and, e.g., given as*

$$\frac{d}{dt}\pi^{\mathit{Altitude}}(\varphi) = \pi^{\mathit{Velocity}}(\varphi)$$
$$\frac{d}{dt}\pi^{\mathit{Velocity}}(\varphi) = -g$$

*with initial values* $\pi^{\mathit{Altitude}}(\varphi)(0) = h_0$ *and* $\pi^{\mathit{Velocity}}(\varphi)(0) = v_0$*, respectively.* If the signal types of two trajectories coincide, they can be concatenated to one trajectory, which is not necessary continuous.

**Definition 7 (Concatenation of trajectories)** *Let* $\varphi : (0, t] \to \mathbb{W}$ *and* $\psi : (0, u] \to \mathbb{W}$ *be trajectories. The* concatenation *of* $\varphi$ *and* $\psi$ *is given by the trajectory*

$$\phi; \psi : (0, t + u] \to \mathbb{W}$$

*defined by*

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

**Example 8 (Concatenation)** *Let* $\varphi$ *and* $\psi$ *be trajectories of duration* 3 *depicted by the solid and dotted lines in Figure 1, respectively. Then the concatenation* $\varphi \,;\, \psi$ *is a trajectory of duration* 6 *depicted in Figure 1 by a solid line.* For the convenience a time-shift operation is defined. It shifts a trajectory

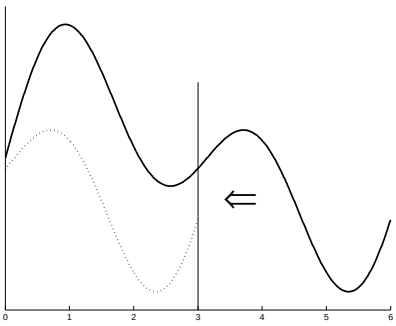


Figure 1. Time-shift

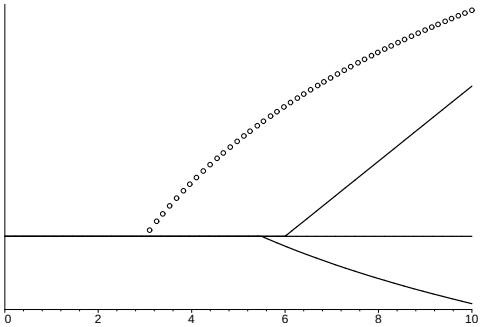


Figure 2. Set of continuations

to "the left" by some time. An example of the time shift by 3 time units is presented in Figure 1. The solid line represents the original function, and the dotted line represents the function after the time-shift.

**Definition 9 (Time-shift)** *Let $\Phi$ be a set of trajectories and $\varphi : (0, t] \to \mathbb{W}$ be a trajectory. Then a* time-shift *operator*

$$\uparrow : \Phi \times \mathbb{R}_{\geqslant 0} \to \Phi$$

*defines a shift of the trajectory by some time $t' < t$,*

$$\varphi \uparrow t' : (0, t - t'] \to \mathbb{W} \text{ such that } \forall u \in (0, t - t'] \quad \varphi \uparrow t' (u) = \varphi(t' + u).$$

If one trajectory coincides on the signal space with the initial part of the other trajectory, it is called a *prefix* of trajectory.

**Definition 10 (Prefix of trajectory)** *Let $\varphi : (0, t] \to \mathbb{W}$ and $\psi : (0, u] \to \mathbb{W}$ be a trajectories, such that $t \leqslant u$. Then $\varphi$ is a* prefix *of $\psi$ (denoted $\varphi \preceq \psi$), if $\varphi = \psi \restriction t$. Furthermore, if $\varphi \preceq \psi$ and $t < u$, then $\varphi$ is called a* strict prefix *of $\psi$ and denoted $\varphi \prec \psi$.* As a supplement to the trajectory prefix, we introduce a notion of trajectory continuation, the part of trajectory, the remainder of the taken trajectory.

**Definition 11 (Trajectory continuation)** *Let $\varphi : (0, t] \to \mathbb{W}$ and $\psi : (0, u] \to \mathbb{W}$ be a trajectories such that $\psi \preceq \varphi$. Then we define a* trajectory continuation *of $\varphi$ after taking $\psi$*

$$\varphi \backslash \psi : (0, t - u] \to \mathbb{W},$$

*such that*

$$\varphi \backslash \psi = \varphi \uparrow u.$$

Trajectory continuation defines a remainder of the taken trajectory. A generalised version of it, a *set of trajectory continuations*, singles out a subset of trajectory continuations, i.e., all remainders from the set of trajectories, which have the same initial part.

**Definition 12 (Set of trajectory continuations)** *Let $\Phi$ be a set of trajectories and $\psi$ be a trajectory or trajectory prefix of some trajectory belonging to the set. Then a* set of trajectory continuations *is a set of trajectory $\psi$ continuations*

$$\Phi \backslash \psi = \{\varphi \mid \psi ; \varphi \in \Phi\}.$$

**Example 13 (Set of trajectory continuations)** *Let us have a set of trajectories depicted in Figure 2. Let us take the trajectory prefix of duration 4, depicted by the solid line. Then the set of trajectory continuations for this prefix will include all continuations from the time point 4 depicted by the solid line, and the trajectory depicted by the circles will be excluded.*

**Definition 14 (Partial prefix)** *Let $H$ be a set of trajectory qualifiers, and*

*let* $\varphi : (0, t] \to \mathbb{W}_\varphi$, $\psi : (0, u] \to \mathbb{W}_\psi$ *be trajectories, such that* $t \leqslant u$. *Let* $\mathcal{T} = \mathrm{T}(\varphi) \cap \mathrm{T}(\psi) \subseteq H$.

- *Trajectory* $\varphi$ *is a* partial prefix *of* $\psi$ *(denoted* $\varphi \preceq^H \psi$*), if* $\pi^{\mathcal{T}}(\varphi) = \pi^{\mathcal{T}}(\psi \restriction t)$.
- *If* $\varphi \preceq^H \psi$ *and* $t < u$, *then* $\varphi$ *is called a* strict partial prefix *of* $\psi$ *and denoted* $\varphi \prec^H \psi$.
- *In case of* $t = u$ *the trajectories are equal on the coinciding trajectory qualifiers and are called* partially equal *(denoted* $\varphi =^H \psi$*).*

The notion of partial prefix loosens requirements put by prefix (Definition 10), i.e., only the projections over coinciding trajectory qualifiers are compared.

**Example 15 (Partial prefix)** *Let us have two trajectories, which define the (altitude, velocity) and (altitude, temperature) pairs, respectively. Then one of these trajectories is a partial prefix of another, if the altitude changes in the same way. It allows to define different aspects of the same object separately and then compose definitions to get a complete specification of the object.*

**Definition 16 (Synchronising prefix)** *Let* $H$ *be a set of trajectory and let* $\varphi : (0, t] \to \mathbb{W}_\varphi$ *and* $\psi : (0, u] \to \mathbb{W}_\psi$ *be trajectories. Let* $\mathcal{T} = \mathrm{T}(\varphi) \cap \mathrm{T}(\psi) \subseteq H$. *We will define a* set of common prefixes *as follows*

$$\mathrm{CP}(\varphi, \psi) = \{\varphi' \preceq^H \varphi \mid \exists \psi' \preceq^H \psi \quad \varphi' =^H \psi'\}.$$

*Furthermore, a* synchronising prefix *is the longest prefix in the set of common prefixes*

$$\varphi \downarrow \psi = \max_{\preceq^H} \mathrm{CP}(\varphi, \psi).$$

A set of common prefixes collects all coinciding prefixes of trajectories, and the synchronising prefix is the longest of them.

Based on synchronising trajectory qualifiers, two trajectories can be composed creating a new, "wider", trajectory, such that evolutions of coinciding trajectory qualifiers are merged and non-coinciding parts extend the state space.

**Definition 17 (Composition of trajectories)** *Let* $H$ *be a set of synchronising trajectory qualifiers, and let* $\varphi : (0, u] \to \mathbb{W}_\varphi$ *and* $\psi : (0, u] \to \mathbb{W}_\psi$ *be trajectories such that* $\mathrm{T}(\varphi) \cap \mathrm{T}(\psi) \subseteq H$ *and* $\pi^{\mathrm{T}(\varphi) \cap \mathrm{T}(\psi)}(\varphi) = \pi^{\mathrm{T}(\varphi) \cap \mathrm{T}(\psi)}(\psi)$. *Then a* composition of trajectories *is a relation*

$$\varphi \times_H \psi : (0, u] \to \mathbb{W},$$

*such that*

$$\begin{aligned} \mathrm{T}(\varphi \times_H \psi) &= \mathrm{T}(\varphi) \cup \mathrm{T}(\psi), \\ \pi^{\mathrm{T}(\varphi)}(\varphi \times_H \psi) &= \varphi, \\ \pi^{\mathrm{T}(\psi)}(\varphi \times_H \psi) &= \psi. \end{aligned}$$

Several different ways will be used to define sets of trajectories.

- By listing all trajectories belonging to the set: $\Phi = \{\varphi_1, \ldots, \varphi_n\}$.
- By putting restrictions on the already existing set of trajectories: $\Phi \downarrow \mathcal{P}red = \{\varphi \in \Phi \mid \mathcal{P}red(\varphi)\}$, where $\mathcal{P}red$ is a predicate.
- Because trajectories are finite, sometimes it is useful to define conditions on the *end-points* of trajectories or *exit conditions*. We will use $\Downarrow$ to denote such conditions, as restrictions on set of trajectories: $\Phi \Downarrow \mathcal{P}red_{\text{exit}} = \{\varphi : (0, u] \to \mathbb{W} \in \Phi \mid \mathcal{P}red_{\text{exit}}(\varphi(u))\}$. It is illustrated in Section 7, where, e.g., in Example 31 $h = 0$ specifically requires, that the trajectory finishes at 0 altitude.

**Notation 2.2 (Set of trajectories)** *When it is clear from the context, we will use trajectory qualifiers to access corresponding parts of trajectories, e.g., $t_i$ will mean the same as $\pi^{t_i}(\varphi)$ for $\varphi : (0, t] \to \mathbb{W}$ with $\mathbb{W} = (W_1 \times \cdots \times W_n, t_1 \times \cdots \times t_n), i = 1 \ldots n$. Furthermore, will use $t_i$ instead of $\pi^{t_i}(\varphi)(u)$ with $u \in (0, t]$ as a time, when it is clear from the context.*

*Combination of different conditions is allowed*

$$\Phi \downarrow \mathcal{P}red \Downarrow \mathcal{P}red_{\text{exit}} = \{\varphi : (0, u] \to \mathbb{W} \in \Phi \mid \mathcal{P}red(\varphi) \wedge \mathcal{P}red_{\text{exit}}(\varphi(u))\}$$

## 3 Hybrid transition system

Automata, state-transition diagrams and other similar models are often used to describe the dynamic behaviour of the systems. They consist of the states $s \in S$ (with $S$ as a set of states) and some construct, defining changes of the states. Most of the time changes of the states are defined by the *transitions*, which are given as a relation (function) over a subset of Cartesian product of the states ($S \times S$). If it is possible to change from the state $s$ to the state $s'$ in one step, then $(s, s') \in S \times S$ tells that. Often such transitions are denoted by an arrow, e.g., $(s, s') \in \to$ or $s \to s'$.

In the process algebras such transitions systems are used as a *labelled transition systems*. That is a class of transitions systems, where transitions are related with some *actions* $\mathsf{a} \in A$ (where $A$ is a set of actions), e.g., conditioned by them. Therefore, the transition relation is defined over subset of $S \times A \times S$. Then $(s, \mathsf{a}, s') \in \to$ or $s \xrightarrow{\mathsf{a}} s'$.

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

**Definition 18 (HTS)** *A* hybrid transition system *is a tuple* $HTS = \langle S, A, \rightarrow, W, \Phi, \rightarrow_c \rangle$*, where*

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

*We will write*

$$s \xrightarrow{\mathsf{a}} s' \iff (s, \mathsf{a}, s') \in \rightarrow$$
$$s \xrightarrow{\varphi} s' \iff (s, \varphi, s') \in \rightarrow_c .$$

*The set of discrete action names includes a* silent action*, denoted* $\tau$*. 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, p.37–43]).*

**Remark 19 (Density)** *We will require* density *for all trajectories*

$$s \xrightarrow{\varphi} s' \Longleftrightarrow \exists s'', \varphi_1, \varphi_2 : \ \varphi = \varphi_1 \,;\, \varphi_2 \ \wedge \ s \xrightarrow{\varphi_1} s'' \ \wedge \ s'' \xrightarrow{\varphi_2} s'.$$

This requirement allows us to split every trajectory into arbitrarily many parts.

**Notation 3.1** *We will adhere to a certain notation.*

- *Greek alphabet symbols (like* $\varphi, \psi$*) will be used to denote trajectories, which are taken on a continuous transition.*
- *Latin alphabet (like* $\mathsf{a}, \mathsf{b}$*) will be used to denote actions.*

**Remark 20 (Labels of continuous-time transitions)** *Label* $\varphi$ *in* $s \xrightarrow{\varphi} s'$ *is a* semantic object*, viz. the set theoretic graph of the function* $\varphi$*.*

**Remark 21 (Sufficiency of density)** *The above property of density does not suffice in general, because it allows pathological transition systems, see Jeffrey et al. [1993]. However, the process calculus that we define cannot describe such pathological cases, so that our definition suffices.*

**Remark 22** *Note that the trajectory transitions can be non-deterministic.*

### *3.1 Bisimulation*

One of the main tools to compare systems is a *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 23 (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*

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

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

**Definition 24 (Bisimilarity)** *States $p$ and $q$ are* bisimilar *(denoted $p \leftrightarrows q$), if there exists a bisimulation $\mathcal{R}$, containing the pair $(p, q)$.*

## 4 Language and semantics

### *4.1 Language*

To define evolution and interaction of systems, a language, based on hybrid transition system (Section 3) is introduced. The syntax of language is presented in BNF notation (Backus-Naur form).

$$
B ::= \mathbf{0} \;\Big|\; a \,.\, B \;\Big|\; [\varphi] \,.\, B \;\Big|\; \sum_{i \in I} B_i \;\Big|\; \bigoplus_{i \in I} B_i \;\Big|\; B \parallel_A^H B \;\Big|\; \mathsf{new}\ w \,.\, B \;\Big|\; B\,[\sigma] \;\Big|\; P
$$

- $\mathbf{0}$ is a *deadlock*.
- $\mathsf{a} \,.\, B$ is an *action-prefix*, where $\mathsf{a} \in A$ is a discrete action name and $B$ is a process. It denotes discrete transitions in the hybrid transition system.
- $[\varphi] \,.\, B$ is a *trajectory-prefix*, where $\varphi$ is a trajectory. It denotes continuous-time transition in the hybrid transition system.
- $\sum\limits_{i \in I} B_i$ is an *alternative composition* (choice) of processes, a generalised version of binary choice operator.

- $\bigoplus_{i \in I} B_i$ is a *superposition* of processes, a generalised version of binary superposition operator.
- $B \parallel_A^H B$ is a *parallel composition* of two processes with an *interconnection set* $H$ and a *synchronisation set* $A$, where $H \subseteq \mathcal{T}$ is a set of trajectory qualifiers for the continuous transitions and $A$ is a synchronisation set of the discrete actions for the discrete transitions.
- $\mathsf{new}\ w\ .\ B$ is a *hiding* operator, where $w$ is a set of discrete action names and trajectory qualifiers to hide.
- $B[\sigma]$ is a *renaming* operator, where $\sigma$ is a renaming function.
- $P \triangleq B$ is a *recursive equation*.

We will use syntactic functions $\mathfrak{L}(B)$ and $\mathcal{N}(B)$ for collecting action and trajectory qualifiers occurring in $B$, respectively.

**Definition 25 (Consistent signal flow)** *We will require a* consistent signal flow*, i.e., only the parallel composition is allowed to change the set of trajectory qualifiers in the process. Renaming operation can only renames them, but not change their type.*

*Let* $\mathbb{P}$ *is a* set of processes *and* $\mathcal{T}$ *is a set of trajectory qualifiers, then* $\mathcal{N}$ *is syntactical function* $\mathcal{N} : \mathbb{P} \to \mathcal{T}$*, that collects trajectory qualifiers occuring in the process. We define* $\mathcal{N}$ *recursively with given static constraints.*

- $\mathcal{N}(\mathsf{a}\ .\ B) = \mathcal{N}(B)$*;*
- $\mathcal{N}(B) = \mathrm{T}(\varphi)$*, and if* $B \neq \mathbf{0}$ *then* $\mathrm{T}(\varphi) = \mathcal{N}(B)$*;*
- $\mathcal{N}\left(\bigoplus_{i \in I} B_i\right) = \mathcal{N}(B_i)$ *for all* $i \in I$*, with static constraint* $\forall\, i, j \in I\ \mathcal{N}(B_i) = \mathcal{N}(B_i)$*;*
- $\mathcal{N}\left(B \parallel_A^H C\right) = \mathcal{N}(B) \cup \mathcal{N}(C)$*;*
- $\mathcal{N}(B\,[\sigma]) = \mathcal{N}(B) \cup \sigma_{\text{new}} \setminus \sigma_{\text{repl}}$*, where* $\sigma_{\text{new}}$ *and* $\sigma_{\text{repl}}$ *are sets of new and replaced qualifiers, and each new qualifier has the same type as the qualifier replaced by it;*
- $\mathcal{N}(P) = \mathrm{T}(P)$*, where* $T$ *a set of qualifiers associated with process name, if* $P \triangleq B$ *and* $\mathrm{T}(B) = \mathcal{N}(B)$*.*

**Notation 4.1** *When it is clear from the context we will use trajectory qualifiers to access the corresponding components of trajectories. Therefore, in the trajectory-prefix definition* $[t_i, t_j \mid \varphi]$ *trajectory qualifiers* $t_i$ *and* $t_j$ *are explicitly used to refer to* $\pi^{t_i}(\varphi)$ *and* $\pi^{t_j}(\varphi)$*, respectively. Furthermore, we will use trajectory qualifiers to denote values of trajectory at a current time (*snapshots*), e.g,* $t$ *instead of* $\pi^t(\varphi)(u)$ *with* $u$ *as a time, when it is clear from the context.*

### *4.2 Behavioural Hybrid Process Calculus operators*

In this section we explain main BHPC operators.

#### *4.2.1 Action-prefix* $\mathsf{a} \,.\, B$

The ordinary well-known action-prefix. Process $\mathsf{a}\,.\,B$ describes a process which engages in the action $\mathsf{a}$ and then behaves as described by $B$.

A special *silent action*, denoted $\tau$, is introduced. 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, p.37–43]).

$$\mathsf{a} \,.\, B \xrightarrow{\mathsf{a}} B \tag{1}$$

In Section 6.1 we define a parametrised version of action-prefix. The use of both ordinary and parameterised action-prefixes are illustrated in Section 7.

#### *4.2.2 Trajectory-prefix* $[\varphi] \,.\, B$

*Trajectory-prefix* defines the hybrid behaviour that starts with a continuous trajectory denoted by $\varphi$ and is followed by the behaviour specified by $B$.

$$[\varphi] \,.\, B \xrightarrow{\varphi} B \tag{2a}$$

$$[\varphi] \,.\, B \xrightarrow{\psi} [\varphi \backslash \psi] \,.\, B \qquad \text{for all } \psi \preceq \varphi \tag{2b}$$

In (2a) a process engages in the trajectory $\varphi$, completes it and then behaves as described by B. While in (2b) only a part of the trajectory is taken, and then the process will continue with the remainder of the trajectory $(\varphi \setminus \psi)$.

We define a symbolic version of trajectory-prefix in Section 6.2 and illustrate it in examples from Section 7.

#### *4.2.3 Concatenation*

Let $\varphi : (0, t] \to \mathbb{V}$ and $\psi : (0, u] \to \mathbb{W}$ be trajectories. If the signal spaces of both trajectories coincide ($\mathbb{V} = \mathbb{W}$), they can be concatenated. A final trajectory is not necessary continuous.

Concatenation can be illustrated by Figure 1. Let $\varphi$ and $\psi$ be trajectories depicted by the solid and dotted lines in the interval $(0, 3]$, respectively. Then the concatenation of trajectories $\varphi$ and $\phi$ is depicted by the solid line in the

interval $(0, 6]$.

$$\frac{B \xrightarrow{\psi} B'}{[\varphi] \,.\, B \xrightarrow{\varphi;\psi} B'} \tag{3}$$

*4.2.4 Superposition* $\bigoplus\{B(v) \mid v \in I\}$

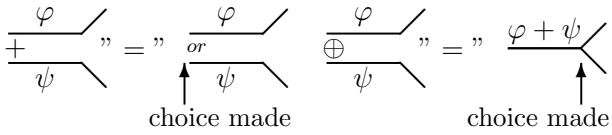


Figure 3. Superposition

*Superposition* is a generalised operator on sets of behaviour expressions. To generate the set we allow arbitrary index sets $I$. It can be thought of as a generalisation of the choice or summation operator $\sum$ in ordinary process algebra. Indeed, if all arguments are of the form $a_v \,.\, B(v)$ then the intended interpretation of

$$\bigoplus\{B(v) \mid v \in I\} \;\text{ or }\; \bigoplus_{v\in I} B(v) \quad \text{is} \quad \sum_{v\in I} B(v)$$

The inactive process $\mathbf{0}$ or $STOP$ can be introduced as $\bigoplus\varnothing$.

The difference between $\bigoplus$ and $\sum$ 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, as is illustrated in Fig. 3. If it is necessary to make a choice at the beginning, it can be done using silent action, like in, e.g. $\tau \,.\, [\varphi] \oplus \tau \,.\, [\psi]$, where choice is made before engaging in the trajectory.

$$\frac{B(w) \xrightarrow{\mathsf{a}} B'}{\bigoplus\limits_{v\in I} B(v) \xrightarrow{\mathsf{a}} B'} \qquad w \in I \tag{4a}$$

$$\frac{\{B(v) \xrightarrow{\varphi} B'(v) \mid v \in I\},\ \{B(w) \not\xrightarrow{\varphi} \mid w \in J\},\ I \neq \varnothing}{\bigoplus\limits_{v\in I\cup J} B(v) \xrightarrow{\varphi} \bigoplus\limits_{v\in I} B'(v)} \tag{4b}$$

In (4a) a superposition for action-prefixes is defined. Actually, it coincides with general choice or summation $\sum\limits_{v\in I} B(v)$ operator. Rule (4b) tells, that we can not distinguish between two signal transitions, if they can take the same trajectory, and if some processes can evolve according to a certain trajectory, and another set can not, then the choice is resolved in the favour of the processes, which can take the trajectory.

**Remark 26** *Use of rule* (4b) *in BHPC requires infinitary SOS derivation trees and transfinite induction as a proof principle.* Use of superposition is

illustred in examples from Section 7 and in extensions of basic process calculus operators in Section 6.

#### *4.2.5 Parallel composition* $B_1 \parallel_A^H B_2$

Parallel composition models concurrent evolution of several processes. During the evolution they may interact with each other via synchronisation on discrete and continuous-time transitions. In BHPC synchronisation on identical names is assumed as the basic synchronisation concept. In order to avoid context-dependent interpretations of operators, the set of action names $A$ and the set of trajectory qualifiers $H$ that are subject to synchronisation, are made explicit in the parallel operator $\parallel_A^H$.

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

The basic idea of synchronising trajectories is not much different than that of synchronising on actions. Let $B_1$ and $B_2$ are the processes which can take trajectories

$$\varphi : (0, t] \to \mathbb{W}_\varphi \quad \text{and} \quad \psi : (0, t] \to \mathbb{W}_\psi,$$

respectively. The static constraint is imposed, that $B_1 \parallel_A^H B_2$ is only well-formed iff $\mathfrak{L}(B_1) \cap \mathfrak{L}(B_2) \subseteq A$ and $\mathcal{N}(B_1) \cap \mathcal{N}(B_2) \subseteq H$. Let $\mathcal{W}$ be a set of signal domains and let

$$\mathbb{T}' = \mathrm{T}(\varphi) \cap \mathrm{T}(\psi). \tag{5}$$

If a set of coinciding trajectory quantifiers is a subset of the synchronisation set

$$\mathbb{T}' \subseteq H \tag{6a}$$

and trajectories are the same on the coinciding quantifiers

$$\pi^{\mathbb{T}'}(\varphi) = \pi^{\mathbb{T}'}(\psi), \tag{6b}$$

then the resulting trajectory is a synchronised trajectory of $B_1 \parallel_A^H B_2$ that simultaneously changes the states of $B_1$ and $B_2$, defined as

$$\chi : (0, t] \to \mathbb{W}$$

such that

$$\begin{aligned} \mathrm{T}(\chi) &= \mathrm{T}(\varphi) \cup \mathrm{T}(\psi), \\ \pi^{\mathbb{T}'}(\varphi) &= \pi^{\mathbb{T}'}(\psi) = \pi^{\mathbb{T}'}(\chi), \\ \varphi &= \pi^{\mathrm{T}(\varphi)}(\chi), \\ \psi &= \pi^{\mathrm{T}(\psi)}(\chi). \end{aligned}$$

It can be also defined composition of trajectories (Definition 17), i.e., $\chi = \varphi \times_H \psi$.

We define the following SOS rules for parallel composition

$$\frac{B_1 \xrightarrow{\mathsf{a}} B_1', B_2 \xrightarrow{\mathsf{a}} B_2'}{B_1 \parallel_A^H B_2 \xrightarrow{\mathsf{a}} B_1' \parallel_A^H B_2'} \qquad \mathsf{a} \in A \tag{7a}$$

$$\frac{B_1 \xrightarrow{\mathsf{a}} B_1'}{B_1 \parallel_A^H B_2 \xrightarrow{\mathsf{a}} B_1' \parallel_A^H B_2} \qquad \mathsf{a} \notin A \tag{7b}$$

$$B_2 \parallel_A^H B_1 \xrightarrow{\mathsf{a}} B_2 \parallel_A^H B_1'$$

$$\frac{B_1 \xrightarrow{\varphi} B_1', B_2 \xrightarrow{\psi} B_2'}{B_1 \parallel_A^H B_2 \xrightarrow{\varphi \times_H \psi} B_1' \parallel_A^H B_2'} \qquad (5) \text{ and } (6) \text{ hold} \tag{7c}$$

Rule (7b) explains interleaving semantics for the discrete behaviour, when discrete actions names do not coincide. Synchronisation on actions is defined in (7a). Rule (7c) defines the parallel composition of similar trajectories. Notice, that because of density (Remark 19) we do not have to gives rules for the trajectories with different durations.

Parallel composition is illustrated in Examples 31,33 and 35.

### *4.2.6 Hiding* new $w$ . $B$

Following the conventions of the process calculus a *hiding* is introduced as a scope restriction operator. $\mathsf{new}\ w\ .\ B$ restricts the use of the names $w$ to $B$. Hiding for discrete actions is just an ordinary hiding. It is worth emphasising, that hiding (especially in continuous case) should be used carefully, because two different trajectories can easily become observably equivalent, if only equivalent parts of the behaviour are visible. Hiding may easily influence the outcome of parallel composition and superposition.

$$\frac{B \xrightarrow{\mathsf{a}} B'}{\mathsf{new}\ w\ .\ B \xrightarrow{\tau} \mathsf{new}\ w\ .\ B'} \qquad \mathsf{a} \in w \tag{8a}$$

$$\frac{B \xrightarrow{\mathsf{a}} B'}{\mathsf{new}\ w\ .\ B \xrightarrow{\mathsf{a}} \mathsf{new}\ w\ .\ B'} \qquad \mathsf{a} \notin w \tag{8b}$$

$$\frac{B \xrightarrow{\varphi} B'}{\mathsf{new}\ w\ .\ B \xrightarrow{\pi^{\mathrm{T}(\varphi)\setminus w}(\varphi)} \mathsf{new}\ w\ .\ B'} \tag{8c}$$

The first rule states, that if an action should be hidden, it is renamed to $\tau$ (silent) action. Otherwise (the second rule) nothing changes. The third rules defines hiding for the continuous behaviour, i.e., some continuous trajectories are not visible any more.

### *4.2.7 Renaming $B\,[\sigma]$*

*Renaming operator* $B\,[\sigma]$, where $\sigma$ is a *renaming function*, is defined. Renaming of both action and signal names is allowed. The renaming function $\sigma$ changes only trajectory qualifiers, but not their type.

$$\frac{B \xrightarrow{\mathsf{a}} B'}{B[\sigma] \xrightarrow{\sigma(\mathsf{a})} B'[\sigma]} \qquad \frac{B \xrightarrow{\varphi} B'}{B[\sigma] \xrightarrow{\sigma(\varphi)} B'[\sigma]} \tag{9}$$

### *4.2.8 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 $P \triangleq B$, where $P$ is a process identifier and actions and signal types of $B$ are only allowed actions and signal types in $P$.

$$\frac{B \xrightarrow{\mathsf{a}} B'}{P \xrightarrow{\mathsf{a}} B'} \quad B \triangleq P \qquad\qquad \frac{B \xrightarrow{\varphi} B'}{P \xrightarrow{\varphi} B'} \quad P \triangleq B \tag{10}$$

## *4.3 Congruence property*

The bisimulation relation (equivalence) defined for the HTSs in Section 3.1 is a congruence relation w.r.t. all operations defined in Section 4.2. We show it using the existing meta-theory [Middelburg, 2001] of congruence formats for the transition systems defined by means of SOS rules, including the use of binding operators. 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.

**Theorem 27** *Strong bisimulation equivalence on HTSs is a congruence w.r.t. the operations of BHPC defined by the in Section 4.2.*

**PROOF.** It is not difficult to see that all SOS rules are in the *panth* format [Aceto et al., 2001, Middelburg, 2001, Mousavi, 2005]. Indeed, in each rule, all transitions in the premises end with distinct variables that do not occur in the left-hand-sides of the conclusions. The only difficulty resides in the fact that the superposition operator is potentially infinitary. This is resolved by interpreting it as a family of binding operators, with the index variable as binder and the index set as an index of the operator (name). In this interpretation, for each rule, the left-hand-sides of the conclusions have just one (binding) operator symbol occurrence, parameterised by distinct variables, which suffices.

It is also not difficult to see that a *stratification* mapping [Aceto et al., 2001, p.214] can be found for our system of SOS rules. The only rule where negative premises occur is rule (4b). By counting the number of nested occurrences of $\bigoplus$ symbol we get the stratification mapping for all of the rules except (10). To cover this recursion rule as well, we apply the same trick as in C.Verhoef [1995] for process algebra with discrete time and recursion, namely, we count each unguarded process name occurrence in a recursive specification with $\omega$.

According to Middelburg [2001] these two facts suffice to prove the statement of this theorem.

## 5 Expansion law

Expansion law (Theorem 28) expresses the parallel composition as a superposition of processes (where parallel composition of discrete actions is resolved in *interleaving* manner).

It is possible to reduce any process in BHPC to a basic form. For processes that do not involve parallel composition it is trivial. Below we show how the parallel composition can be eliminated.

**Theorem 28 (Expansion law)** *Let*

$$B = \bigoplus_{i\in I} \mathsf{b}_i \,.\, B_i \oplus \bigoplus_{j\in J} [\varphi_j] \,.\, B_j, \quad C = \bigoplus_{k\in K} \mathsf{c}_k \,.\, C_k \oplus \bigoplus_{l\in L} [\psi_l] \,.\, C_l$$

*for some process $B_i, B_j, C_k$ and $C_l$, actions $\mathsf{b}_i$ and $\mathsf{c}_k$, trajectories-prefixes $[\varphi_j]$ and $[\psi_l]$, index sets $I \cap J = K \cap L = \varnothing$. Then*

$$B \parallel_A^H C = \tag{11a}$$

$$\bigoplus_{\substack{i\in I\\ \mathsf{b}_i\notin A}} \mathsf{b}_i \,.\, \Big(B_i \parallel_A^H C\Big) \oplus \tag{11b}$$

$$\bigoplus_{\substack{k\in K\\ \mathsf{c}_k\notin A}} \mathsf{c}_k \,.\, \Big(B \parallel_A^H C_k\Big) \oplus \tag{11c}$$

$$\bigoplus_{\mathsf{a}=\mathsf{b}_i=\mathsf{c}_k\in A} \mathsf{a} \,.\, \Big(B_i \parallel_A^H C_k\Big) \oplus \tag{11d}$$

$$\bigoplus_{\varphi_j =^H \psi_l} [\varphi_j \times_H \psi_l] \,.\, \Big(B_j \parallel_A^H C_l\Big) \oplus \tag{11e}$$

$$\bigoplus_{\varphi_j \preceq^H \psi_l} [\varphi_j \times_H (\psi_l \downarrow \varphi_j)] \,.\, \Big(B_j \parallel_A^H [\psi_l \uparrow \mathrm{t}(\varphi_j)] \,.\, C_l\Big) \oplus \tag{11f}$$

$$\bigoplus_{\psi_l \preceq^H \varphi_j} [(\varphi_j \downarrow \psi_l) \times_H \psi_l] \,.\, \Big([\varphi_j \uparrow \mathrm{t}(\psi_l)] \,.\, B_j \parallel_A^H C_l\Big) \oplus \tag{11g}$$

$$\bigoplus_{\neg \begin{pmatrix} \varphi_j =^H \psi_l, \\ \varphi_j \preceq^H \psi_l, \\ \psi_l \preceq^H \varphi_j \end{pmatrix}} [(\varphi_j \downarrow \psi_l) \times_H (\psi_l \downarrow \varphi_j)] \,.\, 0 \tag{11h}$$

**PROOF.** Components (11b), (11c) and (11d) define the expansion law for the discrete components. Nice explanation and detailed proof of the expansion law for discrete actions are given in Milner [1989, p.96-97].

The expansion law for continuous behaviour is slightly more complicated. If both processes in the parallel composition take trajectories of the same duration, we get (11e). When durations of the trajectories are different, we get (11f) and (11g), respectively for the left shorter and the right shorter trajectories. Superposition (11h) defines situation, when both sides take trajectories, which lead them to the deadlock.

The proof of expansion law can be split into several steps.

It is easy to see, that

$$[\varphi] \,.\, B \parallel_A^H [\psi] \,.\, C = \begin{cases} [\varphi \times_H \psi] \,.\, \Big(B \parallel_A^H C\Big) & \varphi =^H \psi, \\ [\varphi \times_H (\psi \downarrow \varphi)] \,.\, \Big(B \parallel_A^H [\psi \uparrow \mathrm{t}(\varphi)] \,.\, C\Big) & \varphi \preceq^H \psi, \\ [(\varphi \downarrow \psi) \times_H \psi] \,.\, \Big([\varphi \uparrow \mathrm{t}(\psi)] \,.\, B \parallel_A^H C\Big) & \psi \preceq^H \varphi, \\ [(\varphi \downarrow \psi) \times_H (\psi \downarrow \varphi)] \,.\, 0 & \text{otherwise.} \end{cases} \tag{12}$$

The next step is

$$\bigoplus_{i \in I} \Big(B \parallel_A^H [\psi] \,.\, C_i\Big) = B \parallel_A^H \bigoplus_{i \in I} [\psi] \,.\, C_i \tag{13}$$

We have two cases in Equation (13)

(1) $B$ starts only with an action-prefix or a superposition of action-prefixes, then it is a deadlock on both sides of equality, because a trajectory-prefix can only be composed in parallel with other trajectory-prefix.
(2) $B$ starts with a trajectory-prefix or a superposition of trajectory-prefixes and action-prefixes. Then by the definition of superposition (Definition 4) on both sides the same trajectories are selected.

From the definition of superpositions (Definition 4) follows

$$\bigoplus_{i\in I}\bigoplus_{j\in J} B_{i,j} = \bigoplus_{i\in I, j\in J} B_{i,j} \tag{14}$$

Then we get

$$\begin{aligned}
&\bigoplus_{j\in J} [\varphi_i] \,.\, B_i \parallel^H_A \bigoplus_{l\in L} [\psi_l] \,.\, C_l = && \text{(Equation (13) and Equation (14))}\\
&\bigoplus_{j\in J,\, l\in L} \Big([\varphi_i] \,.\, B_i \parallel^H_A [\psi_l] \,.\, C_l\Big) = && \text{(Equation (12))}\\
&\bigoplus_{\substack{j\in J,\, l\in L\\ \varphi_j =^H \psi_l}} [\varphi_j \times_H \psi_l] \,.\, \Big(B_j \parallel^H_A C_l\Big) \oplus\\
&\bigoplus_{\substack{j\in J,\, l\in L\\ \varphi_j \preceq^H \psi_l}} [\varphi_j \times_H (\psi_l \downarrow \varphi_j)] \,.\, \Big(B_j \parallel^H_A [\psi_l \uparrow \mathrm{t}(\varphi_j)] \,.\, C_l\Big) \oplus\\
&\bigoplus_{\substack{j\in J,\, l\in L\\ \psi_l \preceq^H \varphi_j}} [(\varphi_j \downarrow \psi_l) \times_H \psi_l] \,.\, \Big([\varphi_j \uparrow \mathrm{t}(\psi_l)] \,.\, B_j \parallel^H_A C_l\Big) \oplus\\
&\bigoplus_{j\in J,\, l\in L} [(\varphi_j \downarrow \psi_l) \times_H (\psi_l \downarrow \varphi_j)] \,.\, 0\\
&\neg\begin{pmatrix} \varphi_j =^H \psi_l, \\ \varphi_j \preceq^H \psi_l, \\ \psi_l \preceq^H \varphi_j \end{pmatrix}
\end{aligned}$$

and in the expansion law formulation we omit $j \in J, l \in L$ in the superposition's index, because it is captured by the partial prefix and the partial equality relations (Definition 14).

## 6 Extensions of BHPC

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

We introduce parametrisation of action-prefix in Section 6.1.

In Section 6.2 we define a *symbolic trajectory-prefix*, which allows to extend the use of trajectory-prefix.

Trajectories defined in Section 2 are finite. To overcome this limitation in Section 6.3 we provide a tool to define infinite trajectories.

In Sections 6.4 and 6.5 we introduce two useful operators, $\mathsf{idle}$ and $\Delta(\mathit{delay})$, defining a trajectory-prefix without any observable behaviour and delay, respectively.

The bisimulation relation (equivalence) defined for the HTSs in Section 3.1 is

a congruence relation w.r.t. the operations defined in this section. The proof is the same, as in Theorem 4.3.

### *6.1 Parametrisation of action-prefix*

We will use parametrisation of action-prefix like in Milner [1989, p.53–58]

$$\mathsf{a}(v : V) \,.\, B(v) \triangleq \sum_{v \in V} \mathsf{a}(v) \,.\, B(v) \tag{16}$$

### *6.2 Symbolic trajectory-prefix*

We introduce a *symbolic trajectory-prefix*, which extends notion of ordinary trajectory-prefix by providing a set of continuous behaviours conforming to the certain conditions.

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

**Definition 29 (Set of trajectories)**

$$[f \mid \Phi] \,.\, B(f) \triangleq \bigoplus_{\varphi \in \Phi} ([\varphi] \,.\, B(\varphi))$$

*where* $\Phi$ *is a set of trajectories, defined in one of the ways, described in Section 2 and* $f$ *is a* placeholder (trajectory variable) *for a trajectory.*

**Notation 6.1 (Symbolic trajectory-prefix)** *We will extend notation to make use of trajectory-prefix more convenient*

$$[t_1, \ldots, t_m \mid \Phi \downarrow \mathcal{P}red \Downarrow \mathcal{P}red_{\text{exit}}]$$

*where*

- $t_1, \ldots, t_m$ *are trajectory qualifiers, which can be used to access corresponding parts of trajectories.*
- *As explained in Notation 2.2, the set of trajectories can be defined in a several different ways. We will allow such notation in symbolic trajectory-prefix definition to bring out conditions on the set of trajectories.*

*Furthermore, we will allow to define the set of trajectories directly in the definition of trajectory prefix, where commas will be used to separate conditions. We will use* $\Downarrow$ *to separate exit conditions, when it is required.*

### *6.3 Infinite continuous behaviour*

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

**Definition 30 (Infinite continuous behaviour)** *Let* $\varphi : \mathbb{R}_+ \to \mathbb{W}$ *then an* infinite continuous behaviour *can be defined as*

$$[\varphi] \triangleq \bigoplus_{t>0} [\varphi \upharpoonright (0,t]]_t \,.\, 0$$

*where* $0 \triangleq \bigoplus \varnothing$.

### *6.4 Idling*

Idling in BHPC should be treated as a continuous signal without any observable behaviour. Then it can be defined as

$$\mathsf{idle} = \left[ t \;\middle|\; \overline{0} \right].$$

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.

### *6.5 Delays*

In BHPC time and time related constructs, e.g., delays, should be treated as a continuous signals with rate 1. We define delay in a following way

$$\Delta(\mathit{delay}) \triangleq \left[ t \;\middle|\; t(0) = 0, \dot{t} = 1 \Big\Vert\!\downarrow\; t = \mathit{delay} \right], \tag{17}$$

where $t$ is a reserved variable. Such process does not manifest any observable behaviour for *delay* time units. After *delay* time units the systems progresses with the process following delay.

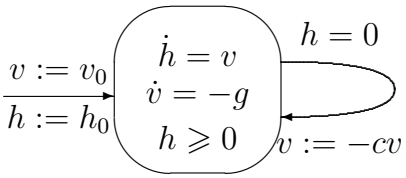


Figure 4. A bouncing ball

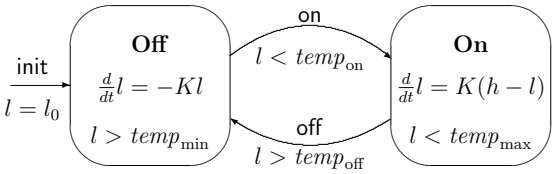


Figure 5. A thermostat

### *6.6 Guard*

Sometimes it is useful to check some conditions explicitly, and if they are not satisfied, to stop the progress of process. *Guard* is one of such constructs.

$$\frac{B \xrightarrow{\mathsf{a}} B'}{\langle \mathcal{P}red \rangle \,.\, B \xrightarrow{\mathsf{a}} B'} \quad \models \mathcal{P}red \qquad \frac{B \xrightarrow{\varphi} B'}{\langle \mathcal{P}red \rangle \,.\, B \xrightarrow{\varphi} B'} \quad \models \mathcal{P}red \tag{18}$$

The rules are very simple. They just state, that if a transition can be done, then it is done, if and only if the guard is satisfied (holds).

## 7 Application of BHPC

To illustrate the application of BHPC several examples are given.

### *7.1 Bouncing ball*

**Example 31 (Bouncing ball)** *A bouncing ball [Schutter and Heemels, 2004], [van der Schaft and J.M.Schumacher, 2000, p.37–38] is a simple example of hybrid systems. This 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:*

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

*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 continuous recursively calling itself with updated continuous variables* $\mathrm{BB}(0, -c * v)$.

*Given specification of the bouncing ball can be extended. We add discrete actions to sense the elasticity of the bounce and increase the ball's kinetic energy, and add a compensating controller.*

$$\begin{aligned}
\mathrm{BB}(h_0, v_0) &\triangleq [h, v \mid \Phi(h_0, v_0) \Downarrow h = 0] \,.\, \mathsf{bounce}(c : [0,1]). \\
&\quad\ [h, v \mid \Phi(0, -cv) \Downarrow v = 0] \,.\, \mathsf{push}(v : \mathbb{R}) \,.\, \mathrm{BB}(h, v) \\
\mathrm{Control}(v_0) &\triangleq \mathsf{idle} \,.\, \mathsf{bounce}(c : [0,1]). \\
&\quad\ \mathsf{idle} \,.\, \mathsf{push}\,((1-c)\,v_0) \,.\, \mathrm{Control}\,((1-c)\,v_0) \\
\mathrm{System}(h_0, v_0) &\triangleq \mathrm{BB}(h_0, v_0) \parallel^{v}_{\mathsf{push},\mathsf{bounce}} \mathrm{Control}(v_0)
\end{aligned}$$

### *7.2 Thermostat*

**Example 32 (Thermostat)** *A thermostat [Henzinger, 1996] is one of the main introductory examples of hybrid systems. The room temperature is controlled by a thermostat, which continuously senses the temperature and switches a heater on and off. The temperature changes are defined by the differential equations. When the heater is off, the temperature decreases according to the exponential function $l(t) = \theta e^{Kt}$, where $t$ is time, $l$ is the temperature in the room, $\theta$ is the initial temperature, and $K$ is a constant determined by the room. When the heater is on, the temperature increases according to the function $l(t) = \theta e^{-Kt} + h(1 - e^{-Kt})$, where $h$ is a constant, that depends on the power of the heater. The temperature should be maintained between $temp_{\min}$ and $temp_{\max}$. Temperatures $temp_{\mathrm{on}}$ and $temp_{\mathrm{off}}$ are the minimal and maximal thresholds, when the heater can be turned on and off, respectively. Using BHPC we get a following specification:*

$$\begin{aligned}
\mathrm{ThOff}(l_0) &\triangleq [l \mid \Phi_{\mathrm{Off}}(l_0) \Downarrow \mathit{tempOn} \geqslant l \geqslant \mathit{tempMin}] \,.\, \mathsf{on} \,.\, \mathrm{ThOn}(l) \\
\mathrm{ThOn}(l_0) &\triangleq [l \mid \Phi_{\mathrm{On}}(l_0) \Downarrow \mathit{tempOff} \leqslant l \leqslant \mathit{tempMax}] \,.\, \mathsf{off} \,.\, \mathrm{ThOff}(l) \\
\Phi_{\mathrm{Off}}(l_0) &= \{l : (0,t] \to \mathbb{R} \mid l(0) = l_0, \dot{l} = -Kl\} \\
\Phi_{\mathrm{On}}(l_0) &= \{l : (0,t] \to \mathbb{R} \mid l(0) = l_0, \dot{l} = K(h - l)\}
\end{aligned}$$

*It consists of two process.*

- *In process* ThOff *the heater is off and the symbolic trajectory-prefix defines the temperature fall. When the temperature reaches the interval $[\mathit{tempOn}, \mathit{tempMin}]$, the process can perform action* on *and switch to the process* ThOn.
- *Process* ThOn *analogously defines the period of heating.*

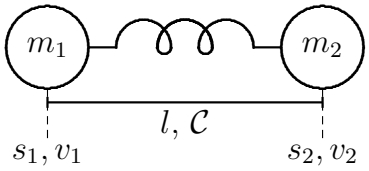


Figure 6. Two masses and a spring

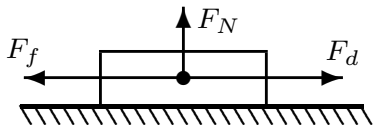


Figure 7. Dry friction

*7.3 Two masses and a spring*

**Example 33 (Two masses and a spring)** *Consider a simple system, depicted in the Figure 6, 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 $v_2$, respectively, 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.*

$$\begin{aligned}
\mathrm{Mass}(m, s_0, v_0) &\triangleq [s, v, f \mid s(0) = s_0, v(0) = v_0, \dot{s} = v, \dot{v} = mf] \\
\mathrm{Spring}(l, c) &\triangleq [s_l, s_r, f_l, f_r \mid f_l = c(s_r - s_l - l), f_r = -f_r] \\
\mathrm{System} &\triangleq \mathrm{Mass}(m_1, s_{01}, v_{01})[s_l/s, v_l/v, f_l/f] \\
&\qquad \|_{s_l, f_l} \ \mathrm{Spring}(l_0, \mathcal{C}) \ \|_{s_r, f_r} \\
&\qquad \mathrm{Mass}(m_2, s_{02}, v_{02})[s_l/s, v_l/v, f_l/f]
\end{aligned}$$

*Processes* Mass *and* Spring *define continuous behaviour of the mass and the spring, respectively. In the process* System Mass *processes are instantiated to represent the left and right masses, respectively, and composed in parallel with an instantiated* Spring *process.*

*7.4 Dry friction*

**Example 34 (Dry Friction)** *In Figure 7 a dry friction [van Beek et al., 2004] phenomenon is depicted. A driving force $F_d$ is applied to a body on a flat surface with frictional force $F_f$. When the body is moving with positive velocity $v$, the frictional force is $F_f = \mu F_n$ ($F_n = mg$). When the velocity is zero and $|F_d| < \mu_0 F_N$, the frictional force neutralises the driving force.*

*Here we provide BHPC version of it. Three processes are defined, each corresponding to a certain mode of behaviour:*

- *Process* BodyStop *defines a behaviour, when the driving force is neutralised by the friction. Corresponding dynamics are defined in $\Phi_{\mathrm{Stop}}$. Guards $\langle F_d \geqslant \mu_0 F_N \rangle$ and $\langle F_d \leqslant -\mu_0 F_N \rangle$ limit the choice, i.e., if the driving force is positive, then*

*the system switches to the process* BodyPos*, and if it is negative, then to the process* BodyNeg*.*

- *Processes* BodyPos *and* BodyNeg *define movement of the body with positive or negative velocity, respectively. Corresponding dynamics are defined in* $\Phi_{\text{Pos}}$ *and* $\Phi_{\text{Neg}}$*, respectively. If the driving force becomes to small and is neutralised by the friction, then the system switches to the process* BodyStop*.*

$$
\begin{aligned}
\text{BodyStop}(x_0, v_0, F_d^0) \triangleq\ & [x, v, F_d \mid \Phi_{\text{Stop}} \Downarrow v \neq 0, F_d \geqslant |\mu_0 F_N|]\ . \\
& \big(\langle F_d \geqslant \mu_0 F_N \rangle\ .\ \text{BodyPos}(x, v, F_d) \oplus \\
& \langle F_d \leqslant -\mu_0 F_N \rangle\ .\ \text{BodyNeg}(x, v, F_d)\big) \\
\text{BodyPos}(x_0, v_0, F_d^0) \triangleq\ & [x, v, F_d \mid \Phi_{\text{Pos}} \Downarrow v = 0, F_d \leqslant \mu F_N]\ . \\
& \langle v = 0, F_d < \mu_0 F_N \rangle\ .\ \text{BodyStop}(x, v, F_d) \\
\text{BodyNeg}(x_0, v_0, F_d^0) \triangleq\ & [t, x, v, F_d \mid \Phi_{\text{Neg}} \Downarrow v = 0, F_d \geqslant -\mu F_N]\ . \\
& \langle v = 0, F_d > -\mu_0 F_N \rangle\ .\ \text{BodyStop}(x, v, F_d) \\
\Phi_{\text{Stop}}(x_0, v_0, F_d^0) = \{ & x, v, F_d, F_n : (0, t] \to \mathbb{R} \mid \\
& x(0) = x_0, v(0) = v_0, F_d(0) = F_d^0, \\
& \dot{x} = v, F_d = \sin(t)\} \\
\Phi_{\text{Pos}}(x_0, v_0, F_d^0) = \{ & x, v, F_d, F_n : (0, t] \to \mathbb{R} \mid \\
& x(0) = x_0, v(0) = v_0, F_d(0) = F_d^0, \\
& \dot{x} = v, F_d = \sin(t), m\dot{v} = F_d - \mu F_N\} \\
\Phi_{\text{Neg}}(x_0, v_0, F_d^0) = \{ & x, v, F_d, F_n : (0, t] \to \mathbb{R} \mid \\
& x(0) = x_0, v(0) = v_0, F_d(0) = F_d^0, \\
& \dot{x} = v, F_d = \sin(t), m\dot{v} = F_d + \mu F_N\}
\end{aligned}
$$

*It is easy to see, that by bringing conditions out in the trajectory prefix we can easily increase readability and clarity of specifications in some cases.*

### *7.5 Railroad gate control*

**Example 35 (Railroad gate control)** *A railroad gate control models a train on a circular track with a controlled gate. A controller issues open and close depending on information about the train movement. A version from Henzinger [1996] is presented below.*

*The initial speed of the train is between* 40 *and* 50 *metres per second. At the distance, which is represented by a variable* $x$*, of* 1000 *meters from the gate, the train issues an* approach *event and may slow down to* 30 *meters per second. At the distance of* 100 *meters past the gate it issues an* exit *event. The circular track is between* 2 *and* 5 *kilometres long. When an* approach *event is received, the controller issues a* lower *event with a* $u$ *seconds delay, and when an* exit

*event is received, the controller issues a* raise *event within $u$ seconds. The elapsed time is represented by variable $z$. Initially the gate is open. A position of the gate, which is represented by a variable $y$, is measured in degrees, and initially is 90. When a* lower *event is received, the gate starts closing at the rate of 9 degrees per second, and when* raise *event is received, the gate starts opening at the same rate. The purpose of the model is to find $u$ - the reaction delay.*

*Specification consists of several processes. The main process* Railway *is a parallel composition of* Train, Gate *and* Controller *processes.*

$$\text{Railway} \triangleq \Big(\text{Train}(x_0) \parallel \text{Gate}(p_0)\Big) \parallel_{\{\mathsf{lower},\mathsf{raise},\mathsf{approach},\mathsf{exit}\}} \text{Controller}$$

*The* Gate *process defines behaviour of the gate. The gate can operate in four modes*

- *Processes* OpenGate *and* ClosedGate *define periods, when the gate is not moving, i.e., open or closed, respectively. Corresponding actions* lower *and* raise *can change these modes.*
- *Processes* LowerGate *and* RaiseGate *define periods, when the gate is being lowered or raised, respectively. These processes finish when the gate reaches the required position or command to change the behaviour to the opposite is issued by an external process (in this case it is* Controller*).*

$$\begin{aligned}
\text{Gate}(p_0) &\triangleq \langle p_0 = 90\rangle \,.\, \text{OpenGate}(p_0) \oplus \\
&\quad \langle p_0 = 0\rangle \,.\, \text{ClosedGate}(p_0) \\
\text{OpenGate}(p_0) &\triangleq [p \mid p(0) = p_0, \dot{p} = 0, p = 90] \,. \\
&\quad \Big(\mathsf{lower} \,.\, \text{LowerGate}(p) \oplus \mathsf{raise} \,.\, \text{OpenGate}(p)\Big) \\
\text{ClosedGate}(p_0) &\triangleq [p \mid p(0) = p_0, \dot{p} = 0, p = 0] \,. \\
&\quad \Big(\mathsf{lower} \,.\, \text{ClosedGate}(p) \oplus \mathsf{raise} \,.\, \text{RaiseGate}(p)\Big) \\
\text{LowerGate}(p_0) &\triangleq [p \mid p(0) = p_0, \dot{p} = -9, p \geqslant 0] \,. \\
&\quad \Big(\langle p = 0\rangle \,.\, \text{ClosedGate}(p) \oplus \\
&\quad \mathsf{raise} \,.\, \text{RaiseGate}(p) \oplus \\
&\quad \mathsf{lower} \,.\, \text{LowerGate}(p)\Big) \\
\text{RaiseGate}(p_0) &\triangleq [p \mid p(0) = p_0, \dot{p} = 9, p \leqslant 90] \,. \\
&\quad \Big(\langle p = 90\rangle \,.\, \text{OpenGate}(p) \oplus \\
&\quad \mathsf{raise} \,.\, \text{RaiseGate}(p) \oplus \\
&\quad \mathsf{lower} \,.\, \text{LowerGate}(p)\Big)
\end{aligned}$$

*The* Controller *process tracks the train and issues corresponding commands to the gate if necessary. It operates in three modes*

- *In* Controller *mode it just waits for the train to enter or exit the dangerous zone, then issues a corresponding command and switches to the* OpenController *or* CloseController *process, correspondingly.*
- *The* OpenController *process imitates the delay of the controller and then issues a* lower *command to the* Gate *process and returns to the* Controller.
- *The* CloseController *process imitates the delay of the controller and then issues* raise *command to the* Gate *process and returns to the* Controller. *If during delay an* approach *is issued by the train, it switches to the* OpenController.

$$
\begin{aligned}
\text{Controller} \triangleq\ & \mathsf{idle}\ .\ \big(\mathsf{exit}\ .\ \text{OpenController}(0)\ \oplus \\
& \qquad \mathsf{approach}.\text{CloseController}(0)\big) \\
\text{OpenController}(z_0) \triangleq\ & [z \mid z(0) = z_0, \dot{z} = 1, z \leqslant u]\ . \\
& \big(\langle z = u\rangle\ .\ \mathsf{raise}\ .\ \text{Controller}\ \oplus \\
& \ \mathsf{exit}\ .\ \text{OpenController}(z)\ \oplus \\
& \ \mathsf{approach}\ .\ \text{CloseController}(0)\big) \\
\text{CloseController}(z_0) \triangleq\ & [z \mid z(0) = t_0, \dot{z} = 1, z \leqslant u]\ . \\
& \big(\langle z = u\rangle\ .\ \mathsf{lower}\ .\ \text{Controller}\ \oplus \\
& \ (\mathsf{exit} \oplus \mathsf{approach})\ .\ \text{CloseController}(z)\big)
\end{aligned}
$$

*The* Train *process models a train moving on a circular track. It moves far from the railroad crossing, then it reaches the dangerous zone, issues an* approach *command, passes the crossing, and leaves the dangerous zone by issuing an* exit *command.*

$$
\begin{aligned}
\text{Train}(x_0) \triangleq\ & [x \mid x(0) = x_0, \dot{x} = [-50; -40], x \geqslant 1000 \Downarrow x = 1000]\ .\ \mathsf{approach}. \\
& [x \mid \dot{x} = [-50; -30], x \geqslant 0 \Downarrow x = 0]\ . \\
& [x \mid \dot{x} = [-50; -30], x \geqslant -100 \Downarrow x = -100]\ .\ \mathsf{exit}. \\
& \text{Train}(x_0)
\end{aligned}
$$

*Conditions* $x > 1000, x > 0$ *and* $x > -100$ *in trajectory-prefixes are not really necessary, because they are encoded in the behaviour definitions and initial conditions, but we bring them out for the clarity reasons.*

## 8 Related Work

The interest in hybrid systems has generated a set of interesting frameworks to specify and analyse such systems. In this section we compare some of them with BHPC.

HyPA Cuijpers and Reniers [2003] is an ACP version, extended with a reinitialisation clause, a disrupt and a flow to handle the hybrid phenomena. 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 synchronise, 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, however it robust and stateless bisimulations in HyPA are congruent.

Hybrid $\chi$ van Beek et al. [2004] and BHPC have some similarities, and some differences. Both of them are not extensions of any process algebras. Hybrid $\chi$ and hybrid automata Alur et al. [1993], Henzinger [1996] share the 'consistent equation semantics' van Beek et al. [2004], in contrast to BHPC. Alternative composition has some similarities to superposition in BHPC, i.e., only the time progress does not resolve choice between trajectories. But in BHPC choice is made, when the trajectories bifurcate and in $\chi$ it leads to the deadlock. Parallel composition for continuous part is the same for both process algebras. For discrete communication different paradigms were chosen: in $\chi$ communication is carried out via *directed* channels, in contrast to BHPC, where direction is not important. In both process algebras interleaving semantics are used for parallel composition of discrete actions.

$\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ 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* $(\varphi \,\mathbin{\curvearrowright}\, x)$ is used, in order to decorate the process $x$ by a logical formula. It defines a class of trajectories the process can follow by remaining in the initial state location (state) of $x$. $(\sigma^r_{\mathsf{rel}}(x))$ denotes a relative time delay of $r$ time units, during which the continuous signal could be emitted. 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 analyse hybrid systems is hybrid automata Alur et al. [1993], Henzinger [1996]. It can be 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 [2005] hybrid behavioral automata (HBA), a modification of hybrid automata, based on use of behavioral theory and interconnection is presented.

In Lynch et al. [2003] a well known Input/Output Automata approach is

extended to cope with hybrid systems resulting in Hybrid Input/Output Automata.

## 9 Conclusions

In this chapter we have introduced the hybrid process calculus BHPC and the underlying concept of a hybrid transition system, and illustrated their application 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 generalisation 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;
- Simulation of BHPC.

## References


L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, chapter 3, pages 197–292. Elsevier Science Publishers, Ltd., 2001.

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 R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *LNCS*, pages 209–229. Springer, 1993. ISBN 3-540-57318-6.

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

J.A. Bergstra and C.A. Middelburg. Process algebra for hybrid systems. Technical report, Dept. of Math. and Comp. Science, Technical University of Eindhoven (TU/e), P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands, 2003.

T. Bolognesi and H. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks*, 14:25–59, 1987.

P. J. L. Cuijpers and M. A. Reniers. Hybrid process algebra. Technical report, Dept. of Math. and Comp. Science, Technical University of Eindhoven (TU/e), P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands, 2003.

C.Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic J. of Computing*, 2(2):274–302, 1995. ISSN 1236-6064.

T.A. Henzinger. The theory of hybrid automata. In *LICS 1996*, pages 278–292. IEEE Computer Society Press, July 1996.

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

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

A.A. Julius. *On Interconnection and Equivalence of Continuous and Discrete Systems: A Behavioral Perspective*. PhD thesis, Systems Signals and Control Group, University of Twente, 2005.

A.A. Julius, S.N. Strubbe, and A.J. van der Schaft. Compositional modeling of hybrid systems with hybrid behavioral automata. *Submitted to the HSCC 2003. Available at* http:\\www.math.utwente.nl\~ julius, 2002.

N.A. Lynch, R. Segala, and F.W. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, 2003. URL `http://www.cs.kun.nl/ita/publications/papers/fvaan/HIOA.html`.

C.A. Middelburg. Variable binding operators in transition system specifications. *JLAP*, 47(1):15–45, 2001.

R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., 1989. ISBN 0-13-115007-3.

M. Mousavi. *Structuring Structural Operational Semantics*. PhD thesis, Technical University of Eindhoven (TU/e), 2005.

J.W. Polderman and J. C. Willems. *Introduction to Mathematical Systems Theory: a behavioral approach*. Springer, 1998.

B. De Schutter and W.P.M.H. Heemels. *Modeling and Control of Hybrid Systems*. DISC, September 2004. Lecture Notes of the DISC Course.

D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers. Syntax and consistent equation semantics of hybrid chi. Report CS-Report 04-37, Technical University of Eindhoven (TU/e), Eindhoven, November 2004.

A.J. van der Schaft. Bisimulation of dynamical systems. In R. Alur and G. J. Pappas, editors, *HSCC*, volume 2993 of *LNCS*, pages 555–569. Springer, 2004. ISBN 3-540-21259-0.

A.J. van der Schaft and J.M.Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *LNCIS*. Springer, London, 2000.