HYPE: hybrid systems modelled with flows

Vashti Galpin[1], Luca Bortolussi[2,3], and Jane Hillston[1]

[1] Laboratory for Foundations of Computer Science, University of Edinburgh
Vashti.Galpin@ed.ac.uk, Jane.Hillston@ed.ac.uk
[2] Department of Maths and Computer Science, University of Trieste
[3] Center for Biomolecular Medicine, Area Science Park, Trieste
luca@dmi.units.it

Abstract. Hybrid systems show both continuous and discrete behaviour. We present a process algebra HYPE for hybrid systems which permits the modelling of individual flows together with a controller component. A HYPE model describes the overall behaviour of the hybrid system. Ordinary differential equations (ODEs) are generated from the states of the underlying labelled transition system and the model can be translated to a hybrid automata. We show that bisimilar HYPE models have the same ODEs.

1 Introduction

A hybrid system exhibits both discrete and continuous behaviour. It can be viewed as a system consisting of values which change continuously with respect to specific dynamics. Discrete events can cause jumps in these values after which different dynamics may come into effect. A well-known model of hybrid systems is hybrid automata [6].

Hybrid behaviour arises in switched systems. Consider a thermostatic controller of a heater. The continuous variable is air temperature, and the discrete events are the switching on and off of the heater by the thermostat in response to the air temperature. Biological systems also exhibit hybrid behaviour. The repressilator describes a genetic regulatory network [5]. It can be described abstractly as a system where genes are switched on and off and continuous behaviour changes depending on which genes are on [3].

We present a new process algebra for modelling hybrid systems. Our aim is to design a language for which it is possible to describe the basic flows in a system, to compose these in parallel together with a controller process and to obtain the overall behaviour of the system as collections of ODEs. We are influenced by the fluid flow approach in PEPA [9] which models the continuous behaviour of large numbers of continuous components by ODEs obtained from the PEPA model.

Process algebras have the advantage of being compositional hence models are built out of subcomponents. Existing process algebras for hybrid systems include ACP_{hs}^{srt} [2], hybrid χ [13], ϕ-calculus [11] and HyPA [4]. Khadim shows substantial differences in the approaches taken by these process algebras relating to syntax, semantics, discontinuous behaviour, flow-determinism, theoretical

results and availability of tools. However, they are similar in the approach taken to modelling hybrid systems. A train gate controller is considered as an example [10] and in each case, the train, gate and controller components have to be fully, sequentially described and then composed in parallel. Hence the dynamic behaviour, namely the ODEs, of these subcomponents must be understood before the model can be built.

We aim for a finer-grained approach where each subcomponent is built up from a number of flows and hence the ODEs are only obtained once the model is constructed. This approach is not particularly useful for the train gate controller but is very useful for more complex systems. By flow, we mean something that has an influence on a quantity of interest. For example, in a tank with two inlets and an outlet, both the inlets and outlet influence the tank level, hence here we would identify three separate flows. The continuous part of the system is represented by the appropriate variables and the change over time of a given variable is determined by a number of active influences which represent flows and are additive in nature. Our approach also differs in that we explicitly require a controller that consists only of events.

The existing process algebras all use a hybrid transition system with two types of transition: one type represents discrete events and the other continuous evolution of the system [10]. In comparison, our transition system only has transitions for events. The ODEs representing the continuous evolution of the system are then extracted from the transition system. This gives a smaller transition system on which it is possible to consider notions of equivalence.

The structure of the rest of the paper is as follows. In the next section we introduce our syntax for hybrid systems, explaining its components. In the following sections, we present the operational semantics and how we go from the notion of state to the ODEs which describe the system. We then discuss how this gives a hybrid automaton. Finally, we consider a notion of bisimulation and our main result is that bisimilar systems have identical ODEs.

2 Language definition

Two types of actions are distinguished in HYPE. *Events* ($\underline{a} \in \mathcal{E}$) are actions which happen instantaneously and can be viewed as signals which trigger discrete changes. They can be caused by a controller or happen randomly.

Activities ($\alpha \in \mathcal{A}$) are influences on the evolution of the continuous part of the system. An activity can be named and parameterised by a set of variables, $\alpha(\overrightarrow{X}) = (\iota, r, I(\overrightarrow{X}))$. It consists of an influence name, a rate of change (or influence strength) and an influence type which describes how that rate is to be applied to the variables involved. For convenience, we will use I for $I(\overrightarrow{X})$.

Events and activities are used to build a controlled system. The syntax is based on PEPA [8]. We assume sets of formal variables $\mathcal{X}$ and system variables $\mathcal{V}$.

Definition 1. *A* controlled system *is constructed as follows.*

- Subcomponents *are serial,* $S ::= \underline{a} : \alpha.C_s \mid S + S$ *where* $\underline{a} \in \mathcal{E}$ *and* $\alpha \in \mathcal{A}$. *The set of subcomponents is* $\mathcal{C}_s$.

– C_s *is a* subcomponent name, $C_s(\overrightarrow{X}) \stackrel{\text{def}}{=} S$ *with free variables of* S *in* X.
– Components *are composed from subcomponents,* $P ::= C(\overrightarrow{X}) \mid P \underset{L}{\bowtie} P$ *where* $L \subseteq \mathcal{E}$. *The set of components is* $\mathcal{C}_c$.
– C *is a* component name *from* $\mathcal{C}_s$ *or* $C(\overrightarrow{X}) \stackrel{\text{def}}{=} P$ *with free variables of* P *in* X.
– *An* uncontrolled system *is* $\Sigma ::= C(\overrightarrow{V}) \mid \Sigma \underset{L}{\bowtie} \Sigma$ *where* $L \subseteq \mathcal{E}$ *and* $\overrightarrow{V}$ *is a set of system variables which instantiate the formal variables. The set of uncontrolled systems is* $\mathcal{C}_{Unc}$
– Controllers *only have events:* $M ::= \underline{a}.M \mid 0 \mid M + M$ *with* $\underline{a} \in \mathcal{E}$ *and* $L \subseteq \mathcal{E}$ *and* $Con ::= M \mid Con \underset{L}{\bowtie} Con$. *The set of controllers is* $\mathcal{C}_{Con}$.
– A controlled system *is* $ConSys ::= \Sigma \underset{L}{\bowtie} \underline{init}.Con$ *where* $L \subseteq \mathcal{E}$. *The set of controlled systems is* $\mathcal{C}_{Sys}$.

A controlled system together with the appropriate sets and functions give a HYPE model. Some components will be discussed in more detail after the definition. Let $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x \geqslant 0\}$.

Definition 2. *A* HYPE model *is a tuple* $(ConSys, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, \text{ec}, \text{iv}, EC, ID)$ *where*

– $ConSys$ *is a controlled system as defined above.*
– $\mathcal{V}$ *is a finite set of variables and* $\mathcal{X}$ *is a finite set of formal variables.*
– IN *is a set of influence names and* IT *is a set of influence types.*
– $\mathcal{E}$ *is a set of events of the form* $\underline{a}$ *and* $\underline{a_i}$.
– $\mathcal{A}$ *is a set of activities of the form* $\alpha(\overrightarrow{X}) = (\iota, r, I(\overrightarrow{X})) \in (IN \times \mathbb{R}^+ \times IT)$.
– $\text{ec} : \mathcal{E} \to EC$ *maps events to event conditions.*
– $\text{iv} : IN \to \mathcal{V}$ *maps influence names to variable names.*
– EC *is a set of event conditions as defined below.*
– ID *is a collection of definitions consisting of a real-valued function for each influence type* $[\![I(\overrightarrow{X})]\!] = f(\overrightarrow{X})$ *where the variables in* $\overrightarrow{X}$ *are from* $\mathcal{X}$.
– $\mathcal{E}$, $\mathcal{A}$, IN *and* IT *are pairwise disjoint.*

The function ec associates with each event an event condition from EC. This set consists of pairs of activation conditions and variable resets. An activation condition is a positive boolean formula containing equalities and inequalities on system variables, and a variable reset is a conjunction of equality predicates on variables V and V' where V' denotes the new value that V will have after the reset, while V denotes the previous value. Resets of the form $V = V'$ are implicit.

Events are urgent – once the event conditions are satisfied, the event must occur. An event with condition *true* occurs immediately. For events that can happen randomly such as breakdowns, we introduce an event condition $\perp$ which means that the event can happen at some point in the future[1].

The function iv associates each influence name with a single variable since it can only influence one variable of the system.

In writing a HYPE model, a term for the controlled system will be used, such as P, and the tuple will be implied. In the case of two HYPE models P and Q without reference to the tuple, we will assume two implied tuples with identical elements except for the first elements.

[1] We have not done so here but probabilistic resets can be used. Tuffin *et al* [12] use a value drawn from a exponential distribution for the time until the next event.

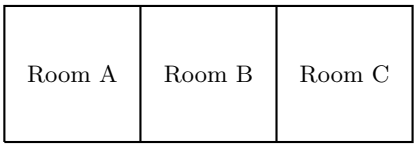

Fig. 1. Room layout of the fan system

Example 1. This is a modification of an example by Tuffin *et al* [12]. Consider three rooms in a row with A adjacent to B and B adjacent to C as in Figure 1. Multiple fan heaters can be placed in each room. Each fan has a reduced heating effect on adjacent rooms and no effect on rooms that are not adjacent. We are interested in the temperature in the central room, Room B. The HYPE model is given in Table 1. The influence of fan i in Room x affecting Room y is represented by $Fan_{i,x,y}$, $t_{i,y}$ is the influence of fan i on room y, r_i is the rate of heating of a fan and its influence is determined by $const_{\psi(x,y)}$. The $\underline{\text{init}}$ event allows for the initialisation of the temperature variable T_B. A controller component for each fan is given by Con_i. The room definition $Room_x(X)$ gives the cooling rate for the air in the room. There are two scenarios – one with a fan in Room A and Room C given by Sys_1, and one with two fans in Room C, Sys_2. The two controlled systems are MF_1 and MF_2. □

$$Fan_{i,x,y} \stackrel{def}{=} \underline{\text{on}}_i : (t_{i,y}, r_i, const_{\psi(x,y)}).Fan_{i,x,y} + \underline{\text{off}}_i : (t_{i,y}, 0, const).Fan_{i,x,y} + \underline{\text{init}} : (t_{i,y}, 0, const).Fan_{i,x,y} \qquad i \in \{1,2\} \;\; x, y \in \{A, B, C\}$$

$$Room_x(X) \stackrel{def}{=} \underline{\text{init}} : (t_{0,x}, -1, linear(X)).Room_x(X) \qquad x \in \{A, B, C\}$$

$$Sys_1 \stackrel{def}{=} (Fan_{1,A,B} \underset{\{\text{init}\}}{\bowtie} Fan_{2,C,B}) \underset{\{\text{init}\}}{\bowtie} Room_B(T_B)$$
$$Sys_2 \stackrel{def}{=} (Fan_{1,C,B} \underset{\{\text{init}\}}{\bowtie} Fan_{2,C,B}) \underset{\{\text{init}\}}{\bowtie} Room_B(T_B)$$

$$Con \stackrel{def}{=} Con_1 \underset{\emptyset}{\bowtie} Con_2 \qquad Con_i \stackrel{def}{=} \underline{\text{on}}_i.\underline{\text{off}}_i.Con_i$$
$$MF_j \stackrel{def}{=} Sys_j \underset{M}{\bowtie} \underline{\text{init}}.Con \qquad M = \{\underline{\text{init}}, \underline{\text{on}}_1, \underline{\text{off}}_1, \underline{\text{on}}_2, \underline{\text{off}}_2\}$$

$$\psi(x,y) = \begin{cases} in & \text{if } x = y \\ adj & \text{if } x \text{ and } y \text{ are adjacent} \\ far & \text{otherwise} \end{cases} \qquad \begin{aligned} \text{ec}(\underline{\text{init}}) &= (true, (T'_B = T_0)) \\ \text{ec}(\underline{\text{off}}_i) &= (\bot, true) \\ \text{ec}(\underline{\text{on}}_i) &= (\bot, true) \end{aligned}$$

$$\mathcal{V} = \{T_B\} \qquad [\![const_{in}]\!] = 1 \qquad [\![const_{adj}]\!] = 0.5 \qquad [\![const_{far}]\!] = 0$$
$$\text{iv}(t_{i,B}) = T_B \qquad [\![const]\!] = 0 \qquad [\![linear(X)]\!] = X$$

Table 1. HYPE model of the fan system

Prefix with influence:	$\dfrac{}{\langle \underline{a} : (\iota, r, I).E, \sigma\rangle \xrightarrow{\underline{a}} \langle E, \sigma[\iota \mapsto (r, I)]\rangle}$
Prefix without influence:	$\dfrac{}{\langle \underline{a}.E, \sigma\rangle \xrightarrow{\underline{a}} \langle E, \sigma\rangle}$
Choice:	$\dfrac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle E + F, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle} \qquad \dfrac{\langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}{\langle E + F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}$
Parallel without synchronisation:	$\dfrac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E' \underset{M}{\bowtie} F, \sigma'\rangle}(\underline{a} \notin M)$ $\dfrac{\langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E \underset{M}{\bowtie} F', \sigma'\rangle}(\underline{a} \notin M)$
Parallel with synchronisation:	$\dfrac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \tau\rangle \quad \langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \tau'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E' \underset{M}{\bowtie} F', \Gamma(\sigma, \tau, \tau')\rangle}(\underline{a} \in M, \Gamma \text{ defined})$
Constant:	$\dfrac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle A, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}(A \stackrel{def}{=} E)$

Table 2. Operational semantics for HYPE

3 Operational semantics

To define the operational semantics, a notion of state is required.

Definition 3. *A* state *of the system is a function* $\sigma : IN \to (\mathbb{R} \times IT)$*. The set of all states is* $\mathcal{S}$*. A* configuration *consists of a controlled system together with a state* $\langle ConSys, \sigma\rangle$ *and the set of configurations is* $\mathcal{F}$*.*

For convenience, states may be written as a set of triples of the form $(\iota, r, I(\overrightarrow{X}))$. This is the same form as an activity to reflect the fact that the state captures the activities that are currently in effect. The notion of state here is not a valuation of system variables but rather a collection of flows that occur in the system.

The operational semantics give a labelled transition system over configurations $(\mathcal{F}, \mathcal{E}, \to \subseteq \mathcal{F} \times \mathcal{E} \times \mathcal{F})$. We write $F \xrightarrow{\underline{a}} F'$ for $(F, \underline{a}, F') \in \to$. In the following, $E, F \in \mathcal{C}_{Sys}$. The rules are given in Table 2.

The updating function $\sigma[\iota \mapsto (r, I)]$ is defined by $\sigma[\iota \mapsto (r, I)](x) = (r, I)$ if $x = \iota$ and $\sigma[\iota \mapsto (r, I)](x) = \sigma(x)$ otherwise. The notation $\sigma[u]$ will also be used for an update, with $\sigma[u_1 \dots u_n]$ denoting $\sigma[u_1] \dots [u_n]$.

The partial function $\Gamma : \mathcal{S} \times \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ is defined as follows.

$$(\Gamma(\sigma, \tau, \tau'))(\iota) = \begin{cases} \tau(\iota) & \text{if } \sigma(\iota) = \tau'(\iota), \\ \tau'(\iota) & \text{if } \sigma(\iota) = \tau(\iota), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

When synchronisation occurs, two states must be merged and the function uses the previous state and the new states to determine which values have changed

and then puts these changed values into the new state. Γ will be undefined if both the second and third argument differ from the first argument.

The next two definitions will be useful.

Definition 4. *The* derivative set *of a controlled system P,* $\mathrm{ds}(P)$ *is defined as the smallest set satisfying*

- *if* $\langle P, \sigma\rangle \xrightarrow{init} \langle P', \sigma'\rangle$ *then* $\langle P', \sigma'\rangle \in \mathrm{ds}(P)$
- *if* $\langle P', \sigma'\rangle \in \mathrm{ds}(P)$ *and* $\langle P', \sigma'\rangle \xrightarrow{a} \langle P'', \sigma''\rangle$ *then* $\langle P'', \sigma''\rangle \in \mathrm{ds}(P)$.

Definition 5. *The set of* states *of the derivative set of a controlled system P is defined as* $\mathrm{st}(P) = \{\sigma \mid \langle Q, \sigma\rangle \in \mathrm{ds}(P)\}$.

Example 2. In the first model MF_1 from Example 1, there are four states.

$$\sigma_1 = \{t_{0,B} \mapsto (-1, \mathit{linear}(T_B)), t_{1,B} \mapsto (0, \mathit{const}_{adj}), t_{2,B} \mapsto (0, \mathit{const}_{adj})\}$$
$$\sigma_2 = \{t_{0,B} \mapsto (-1, \mathit{linear}(T_B)), t_{1,B} \mapsto (r_1, \mathit{const}_{adj}), t_{2,B} \mapsto (0, \mathit{const}_{adj})\}$$
$$\sigma_3 = \{t_{0,B} \mapsto (-1, \mathit{linear}(T_B)), t_{1,B} \mapsto (0, \mathit{const}_{adj}), t_{2,B} \mapsto (r_2, \mathit{const}_{adj})\}$$
$$\sigma_4 = \{t_{0,B} \mapsto (-1, \mathit{linear}(T_B)), t_{1,B} \mapsto (r_1, \mathit{const}_{adj}), t_{2,B} \mapsto (r_2, \mathit{const}_{adj})\}$$

For the second model MF_2, the same states are obtained. □

4 Hybrid semantics

We extract a set of ODEs for each state which appears in a configuration in the labelled transition system. We will label this set as CS_σ where CS is the constant used for the controlled system and σ is the state.

Given a controlled system CS, and a derivative $\langle CS', \sigma\rangle \in \mathrm{ds}(CS)$, the ODEs associated with the state σ are defined as follows.

$$CS_\sigma = \left\{ \frac{dV}{dt} = \sum \{ r[\![I(\overrightarrow{W})]\!] \mid \mathrm{iv}(\iota) = V \text{ and } \sigma(\iota) = (r, I(\overrightarrow{W})) \} \;\middle|\; V \in \mathcal{V} \right\}$$

So for each state, we have a collection of ODEs, one for each variable V.

Example 3. If both fans are on, we obtain the following ODEs for σ_4.

$$MF_{1,\sigma_4} = \left\{ \frac{dT_B}{dt} = -T_B + 0.5r_1 + 0.5r_2 \right\} = MF_{2,\sigma_4}$$ □

An obvious way of viewing the behaviour of the overall system is as a hybrid automaton and we next present how this can be done.

Hybrid automata are dynamic systems presenting both discrete and continuous evolution. They consist of a set of variables evolving continuously in time, subject to abrupt changes induced by discrete instantaneous *control* events. When discrete events happen the automaton enters its next *mode*, where the rules governing the flow of continuous variables change. See [6] for further details.

Definition 6. *A* hybrid automaton *is a tuple*
$\mathcal{H} = (V, E, \mathbf{X}, \mathcal{E}, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$*, where:*

- $\mathbf{X} = \{X_1, \dots, X_n\}$ *is a finite set of real-valued variables. The time derivative of* X_j *is* $\dot{X}_j$*, and the value of* X_j *after a change of* mode *is* X'_j*.*
- *the* control graph $G = (V, E)$ *is a finite labelled graph. Vertices* $v \in V$ *are the* (control) modes*, while edges* $e \in E$ *are called* (control) switches *and model the happening of a discrete event.*
- *Associated with each vertex* $v \in V$ *there is a set of ordinary differential equations* $\dot{\mathbf{X}} = \mathit{flow}(v)$ *referred to as the* flow conditions*. Moreover,* $\mathit{init}(v)$ *and* $\mathit{inv}(v)$ *are two formulae on* $\mathbf{X}$ *specifying the* admissible initial conditions *and some* invariant conditions *that must be true during the continuous evolution of variables in* v*.*
- *Edges* $e \in E$ *of the control graph are labelled by an event* $\mathit{event}(e) \in \mathcal{E}$ *and by* $\mathit{jump}(e)$*, a predicate on* $\mathbf{X}$ *stating for which values of variables each transition is active, and by* $\mathit{reset}(e)$*, a predicate on* $\mathbf{X} \cup \mathbf{X}'$ *specifying the change of the variables' values after the transition has taken place. Moreover, each edge* $e \in E$ *can be declared urgent, by setting to true the boolean flag* $\mathit{urgent}(e)$*, meaning that the transition is taken at once when* $\mathit{jump}(e)$ *becomes true. Otherwise, the transition can be taken nondeterministically whenever* $\mathit{jump}(e)$ *is true.*

Consider a HYPE model $(P_0, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, \mathrm{ec}, \mathrm{iv}, EC, ID)$ and suppose its initial configuration is $\langle P_0, \sigma_0 \rangle \in \mathcal{F}$. For P_0 the only possible transition is the event $\underline{\mathrm{init}}$. Let $\langle P, \sigma \rangle$ be the configuration reached after its occurrence, $\langle P_0, \sigma_0 \rangle \xrightarrow{\underline{\mathrm{init}}} \langle P, \sigma \rangle$. Moreover, we denote by $act_{\underline{\mathrm{a}}}$ and $res_{\underline{\mathrm{a}}}$ the activation conditions and the resets associated with an event $\underline{\mathrm{a}} \in \mathcal{E}$, so $ec(\underline{\mathrm{a}}) = (act_{\underline{\mathrm{a}}}, res_{\underline{\mathrm{a}}})$.

Definition 7. *The hybrid automaton*
$\mathcal{H} = (V, E, \mathbf{X}, \mathcal{E}, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$ *that can be obtained from the HYPE model* $(P_0, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, \mathrm{ec}, \mathrm{iv}, EC, ID)$ *is defined as follows.*

- *The set of modes* V *is the set of configurations reachable in 0 or more steps from* $\langle P, \sigma \rangle$*, namely* $\mathrm{ds}(P_0)$*.*
- *The edges* E *of the control graph connect two modes* (v_1, v_2) *iff* $v_1 = \langle P_1, \sigma_1 \rangle$*,* $v_2 = \langle P_2, \sigma_2 \rangle$ *and* $\langle P_1, \sigma_1 \rangle \xrightarrow{\underline{a}} \langle P_2, \sigma_2 \rangle$ *is a derivation for some* $\underline{a}$*.*
- $\mathbf{X} = \mathcal{V}$ *is the set of variables of the HYPE system.*
- $\mathcal{E}$ *is the set of events* $\mathcal{E}$ *of* P_0*.*
- *Let* $v_j = \langle P_j, \sigma_j \rangle$*, then*
  $$\mathit{flow}(v_j)[X_i] = \sum\{r[\![I(\overrightarrow{W})]\!] \mid \mathit{iv}(\iota) = X_i \text{ and } \sigma_j(\iota) = (r, I(\overrightarrow{W}))\}$$
- $\mathit{init}(v) = \begin{cases} res_{\underline{init}}, \text{ if } v = \langle P, \sigma \rangle \\ false, \text{ otherwise} \end{cases}$ *with primes removed from variables*[1]*.*
- $\mathit{inv}(v) = true$*.*

[1] $res_{\underline{\mathrm{init}}}$ is a reset so uses primed variables to refer to the new values of variables whereas $\mathit{init}(v)$ is an initialization condition and refers to variables without primes.

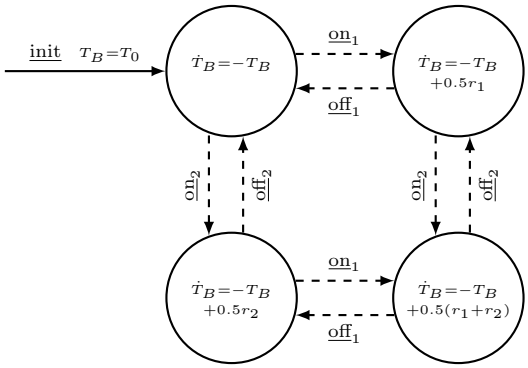

Fig. 2. Hybrid automaton of the fan system

– *Let* $e = (\langle P_1, \sigma_1\rangle, \langle P_2, \sigma_2\rangle)$ *with* $\langle P_1, \sigma_1\rangle \xrightarrow{\underline{a}} \langle P_2, \sigma_2\rangle$. *Then* $event(e) = \underline{a}$ *and* $reset(e) = res_{\underline{a}}$. *Moreover, if* $act_{\underline{a}} \neq \bot$, *then* $jump(e) = act_{\underline{a}}$ *and* $urgent(e) = true$, *otherwise* $jump(e) = true$ *and* $urgent(e) = false$.

Figure 2 shows the HYPE model from Example 1 as a hybrid automaton. The dashed edges represent non-urgent events.

5 Equivalence semantics

A reasonable definition of equivalent behaviour needs to be chosen and is defined with respect to the labelled transition system defined in Section 3.

Definition 8. *A relation* $B \subseteq \mathcal{C}_{Sys} \times \mathcal{C}_{Sys}$ *is a* system bisimulation *if for all* $(P, Q) \in B$ *whenever*

1. $\langle P, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma'\rangle$, *there exists* $\langle Q', \sigma'\rangle$ *with* $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma'\rangle$, $(P', Q') \in B$.
2. $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma'\rangle$, *there exists* $\langle P', \sigma'\rangle$ *with* $\langle P, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma'\rangle$, $(P', Q') \in B$.

P *and* Q *are* system bisimilar, $P \sim_s Q$ *if they are in a system bisimulation.*

System bisimulation is a congruence for our operators.

Theorem 1. $\sim_s$ *is a congruence for Prefix, Choice and Parallel.*

Proof sketch. Straightforward. Let $P_1 \sim_s P_2$. *Interesting cases are Prefix with influence and Parallel with synchronisation. For the former,* $\langle \underline{a} : (\iota, r, I).P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1, \sigma[\iota \mapsto (r, I)]\rangle$ *and likewise* $\langle \underline{a} : (\iota, r, I).P_2, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2, \sigma[\iota \mapsto (r, I)]\rangle$ *as required. For the latter, we need to show that* $B = \{(P_1 \underset{L}{\bowtie} Q, P_2 \underset{L}{\bowtie} Q) | P_1 \sim_s P_2\}$ *is a system bisimulation. If* $\underline{a} \in L$ *and* $\langle P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1', \sigma'\rangle$ *and* $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma''\rangle$, *then* $\langle P_1 \underset{L}{\bowtie} Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1' \underset{L}{\bowtie} Q', \Gamma(\sigma, \sigma', \sigma'')\rangle$. *Since* $P_1 \sim_s P_2$, $\langle P_2, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2', \sigma'\rangle$ *with* $P_1' \sim_s P_2'$, *and hence* $\langle P_2 \underset{L}{\bowtie} Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2' \underset{L}{\bowtie} Q', \Gamma(\sigma, \sigma', \sigma'')\rangle$ *as required.*

We are interested in the link between the ODEs obtained from bisimilar systems. Before we can consider this, some definitions and lemmas are required.

As we saw in Example 1, the $\underline{\text{init}}$ event in the controlled system allows for the initialisation of variables. Typically subcomponents are defined as a number of simple loops, reflecting events that can occur and the associated changes in the continuous part of the system. These requirements can be formalised as follows.

Definition 9.
A well-defined *controlled system has the following properties.*

1. *For each subcomponent* $\mathcal{C}_s(\overrightarrow{X}) \stackrel{\text{def}}{=} S$*, the only subcomponent name that can appear in* S *is* $\mathcal{C}_s(\overrightarrow{X})$*.*
2. *In each subcomponent* $\mathcal{C}_s(\overrightarrow{X})$*, the event* $\underline{a}$ *can only appear once.*
3. *In each subcomponent* $\mathcal{C}_s(\overrightarrow{X})$*, each* ι *that appears, must also appear in a prefix with* $\underline{\text{init}}$*.*
4. *Across all subcomponents, each pair* $\underline{a}$ *and* ι *must appear at most once together in a prefix.*
5. *In any component* $C(\overrightarrow{X})$*, any event that appears in more than one subcomponent must be synchronised on.*
6. *In the controlled system* Σ *and Con must have the same events and these must all appear in* L*.*

Example 1 is a well-defined controlled system. For well-defined controlled systems, we can show that the current state in a configuration does not determine what future events happen (only the controller can influence this) and hence the state can be discounted in certain settings. Note that Condition 4 guarantees that Γ is always defined. Consider a prefix $\underline{\text{a}} : (\iota, r, I(\overrightarrow{X}))$. On the occurrence of $\underline{\text{a}}$, the value of ι in the state will be updated exactly once and Γ is always defined since it is guaranteed that on a specific event each activity is updated at most once either in the second argument of Γ or in the third, ensuring that the other one (third or second) has the same value as the first argument. If it was the case that a particular combination of event and influence names appeared multiple times, it would be possible for Γ to be defined for the original derivation due to the old value coincidentally being the same as an update, but with the other derivation the old value is different to both updates, resulting in Γ being undefined.

Lemma 1. *In a well-defined controlled system, if* $\langle P', \sigma[u_1 \dots u_n]\rangle$ *is a derivative of* $\langle P, \sigma\rangle$*, then* $\langle P', \tau[u_1 \dots u_n]\rangle$ *is a derivative of* $\langle P, \tau\rangle$*.*

Proof sketch. It can be shown by induction on the derivation of transitions that if $\langle P', \sigma'\rangle$ *is a* n*-step derivative of* $\langle P, \sigma\rangle$*,* $\langle P', \tau'\rangle$ *is a* n*-step derivative of* $\langle P, \tau\rangle$*. Moreover,* $\sigma' = \sigma[u_1 \dots u_n]$ *and* $\tau' = \tau[u_1 \dots u_n]$ *for appropriate* $u_1, \dots, u_n$*.*

We can also consider what happens to a system after the first $\underline{\text{init}}$ event. The first transition must be an $\underline{\text{init}}$ event since *Con* is prefixed by $\underline{\text{init}}$ and all actions are synchronised. The following result shows that the starting state is irrelevant since the $\underline{\text{init}}$ action will set every value in the state.

Lemma 2. *Let P be a well-defined controlled system. If $\langle P, \sigma\rangle \xrightarrow{init} \langle P', \sigma'\rangle$ and $\langle P, \tau\rangle \xrightarrow{init} \langle P', \tau'\rangle$ then $\sigma' = \tau'$.*

Proof sketch. Since there is exactly one $\underline{init} : (\iota, r, I)$ prefix for every ι, the occurrence of an $\underline{init}$ event will update every ι value in the state and the previous value of ι is irrelevant.

The following result shows that it is the prefixes which determine the behaviour of the controlled systems because of the restrictions imposed on well-defined controlled systems.

Definition 10. *The* set of prefixes *of an uncontrolled system Sys, $pre(Sys)$ is defined structurally as follows.*

- $pre(\underline{a} : (\iota, r, I(\overrightarrow{X})).S) = \{\underline{a} : (\iota, r, I(\overrightarrow{X}))\}$
- $pre(S_1 + S_2) = pre(S_1) \cup pre(S_2)$
- $pre(P \underset{L}{\bowtie} Q) = pre(P) \cup pre(Q)$

Theorem 2. *Let $\Sigma_1 \underset{L}{\bowtie} \underline{init}.Con$ and $\Sigma_2 \underset{L}{\bowtie} \underline{init}.Con$ be two well-defined controlled systems. If $pre(\Sigma_1) = pre(\Sigma_2)$ then $\Sigma_1 \underset{L}{\bowtie} \underline{init}.Con \sim_s \Sigma_2 \underset{L}{\bowtie} \underline{init}.Con$.*

Proof sketch. We need to show that $\{(\Sigma_1, \Sigma_2)\}$ is a system bisimulation. Since the two systems are well-defined, any event can always occur and events appearing in more than one component are synchronised. Hence $\Sigma_1 \xrightarrow{\underline{a}} \Sigma_1$ and $\Sigma_2 \xrightarrow{\underline{a}} \Sigma_2$ for each event $\underline{a} \in pre(\Sigma_1)$. By congruence, we have the result.

Lemma 3. *Let P and Q be well-defined controlled systems. If $P \sim_s Q$ then their states are equal, $\mathrm{st}(P) = \mathrm{st}(Q)$.*

Proof sketch. Consider $\langle P', \sigma'\rangle$ a derivative of $\langle P, \sigma\rangle$ then $\sigma' \in \mathrm{st}(P)$. Since $P \sim_s Q$, we can find $\langle Q', \sigma'\rangle$ with $\sigma' \in \mathrm{st}(Q)$. Hence $\mathrm{st}(P) \subseteq \mathrm{st}(Q)$ and vice versa.

Theorem 3. *Let P and Q be well-defined controlled systems. If $P \sim_s Q$ then for every state $\sigma \in \mathrm{st}(P)$, $P_\sigma = Q_\sigma$.*

Proof sketch. The well-defined systems are $(P, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, \mathrm{ec}, \mathrm{iv}, EC, ID)$ and $(Q, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, \mathrm{ec}, \mathrm{iv}, EC, ID)$. Hence

$$P_\sigma = \left\{ \frac{dV}{dt} = \sum \left\{ r[\![I(\overrightarrow{W})]\!] \;\middle|\; \mathrm{iv}(\iota) = V \text{ and } \sigma(\iota) = (r, I(\overrightarrow{W})) \right\} \;\middle|\; V \in \mathcal{V} \right\} \text{ and}$$

$$Q_\sigma = \left\{ \frac{dV}{dt} = \sum \left\{ r[\![I(\overrightarrow{W})]\!] \;\middle|\; \mathrm{iv}(\iota) = V \text{ and } \sigma(\iota) = (r, I(\overrightarrow{W})) \right\} \;\middle|\; V \in \mathcal{V} \right\}$$

which are clearly the same.

Example 4. We can show by Theorems 2 and 3 that MF_1 and MF_2 give the same ODEs. Since $pre(MF_1) = pre(MF_2)$, $MF_1 \sim_s MF_2$, and therefore $MF_{1,\sigma} = MF_{2,\sigma}$ for each $\sigma \in \mathrm{st}(MF_1)$. □

The converse of Theorem 3 does not hold. It is possible for two states to be the same and hence give identical ODEs, but this does not mean that their associated derivatives are system bisimilar.

There are at least two definitions of bisimulation on hybrid automata. The usual definition is over a collection of traces representing the evolution of the variables with continuous flows interleaved by discrete jumps [6]. This obviously differs from our definition of system bisimulation. A definition in the same spirit as ours is *U-bisimulation* which acts on the control graph, and it is used for projecting away some variables and for collapsing some modes of the automaton [1]. Consider a simplified form of this U-bisimulation, where we retain all variables and collapse modes that have the same structure.

Definition 11. *Let* $\mathcal{H} = (V, E, \mathbf{X}, \mathcal{E}, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$ *be an hybrid automaton. Two modes* v_1, v_2 *are bisimilar,* $v_1 \sim_{ha} v_2$*, if and only if the following conditions hold:*

1. $\mathit{flow}(v_1) = \mathit{flow}(v_2)$*,* $\mathit{inv}(v_1) = \mathit{inv}(v_2)$*, and* $\mathit{init}(v_1) = \mathit{init}(v_2)$*;*
2. *for* $e_1 = (v_1, v_1')$ *there exists* $e_2 = (v_2, v_2')$ *with* $\mathit{event}(e_1) = \mathit{event}(e_2)$*,* $\mathit{reset}(e_1)$=$\mathit{reset}(e_2)$*,* $\mathit{jump}(e_1)$=$\mathit{jump}(e_2)$*,* $\mathit{urgent}(e_1)$=$\mathit{urgent}(e_2)$*,* $v_1' \sim_{ha} v_2'$*;*
3. *for* $e_2 = (v_2, v_2')$ *there exists* $e_1 = (v_1, v_1')$ *with* $\mathit{event}(e_1) = \mathit{event}(e_2)$*,* $\mathit{reset}(e_1)$=$\mathit{reset}(e_2)$*,* $\mathit{jump}(e_1)$=$\mathit{jump}(e_2)$*,* $\mathit{urgent}(e_1)$=$\mathit{urgent}(e_2)$*,* $v_1' \sim_{ha} v_2'$*;*

We can show that for $P_1, P_2 \in \mathcal{C}_{Sys}$ such that $P_1 \sim_s P_2$ then $\langle P_1, \sigma\rangle \sim_{ha} \langle P_2, \sigma\rangle$. The converse does not hold. Two states $\langle P, \sigma\rangle$ and $\langle Q, \tau\rangle$ of an HA can be bisimilar even if $\sigma \neq \tau$. Different states can lead to the same ODEs. Consider ι and κ mapped by iv to the same variable V. If σ contains $(\iota, 1, I)$ and $(\kappa, -1, I)$ and τ contains $(\iota, 2, I)$ and $(\kappa, -2, I)$, then both σ and τ give $\dot{V} = \mathit{flow}(\langle P, \sigma\rangle) = \mathit{flow}(\langle Q, \tau\rangle) = 0$.

Hence bisimulation for hybrid automata is coarser than that for HYPE. In a HYPE model we make explicit the source of each single flow of the system, while in a hybrid automaton flows are merged together in differential equations, and they cannot be separated out into single influences. Stated otherwise, in hybrid automata ODEs lose information about the logic of the system.

Bergstra and Middelburg [2] present two bisimulations for their process algebra for hybrid systems. One fits with their axiomatic definition, and the other gives congruence with respect to the parallel operator. Recast for our transition system and for our language these two bisimulations equate the same controlled systems and are the same as our system bisimulation.

6 Conclusions and further work

We have presented a process algebra for hybrid systems with novel features that include a fine-grained approach to modelling flows and an explicit controller. Since a HYPE model can be expressed as a hybrid automaton, tools such as the model checker HyTech [7] can be used to explore these systems.

As well as the fans example, we have successfully modelled a dual-tank system [12], a bottling line [2] and the abstract view of the repressilator [3].

Considering further work, we wish to investigate bisimulation between models which differ in more than the description of the controlled system. This would involve the use of bijections or surjections between models. We also wish to consider a more general definition of bisimulation which allows an equivalence over states yet ensures that the same ODEs are produced. Our models currently give rise to hybrid automata with invariants which are always true (see Definition 7) and we will investigate relaxing this condition.

Acknowledgements Thanks to Stephen Gilmore for his comments. Vashti Galpin is supported by the EPSRC SIGNAL Project, Grant EP/E031439/1. Luca Bortolussi is supported by PRIN "Sistemi e calcoli di ispirazione biologica e loro applicazioni" and FIRB LIBi. Jane Hillston is supported by the EPSRC under ARF EP/c543696/01.

References

1. M. Antoniotti, B. Mishra, C. Piazza, A. Policriti, and M. Simeoni. Modeling cellular behavior with hybrid automata: Bisimulation and collapsing. In C. Priami, editor, *Computational Methods in Systems Biology*, pages 57–74, 2003.
2. J. A. Bergstra and C. A. Middelburg. Process algebra for hybrid systems. *Theoretical Computer Science*, 335(2-3):215–280, 2005.
3. L. Bortolusssi and A. Policriti. Hybrid approximation of stochastic process algebras for systems biology. In To be presented at *IFAC World Congress,* Seoul, South Korea, July 2008.
4. P. J. L. Cuijpers and M. A. Reniers. Hybrid process algebra. *Journal of Logic and Algebraic Programming*, 62(2):191–245, 2005.
5. M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.
6. T. A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, 1996.
7. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1-2):110–122, 1997.
8. J. Hillston. *A compositional approach to performance modelling.* Cambridge University Press, 1996.
9. J. Hillston. Fluid flow approximation of PEPA models. In *Second International Conference on the Quantitative Evaluation of Systems (QEST 2005)*, pages 33–43. IEEE Computer Society, 2005.
10. U. Khadim. A comparative study of process algebras for hybrid systems. Computer Science Report CSR 06-23, Technische Universiteit Eindhoven, 2006. `http://alexandria.tue.nl/extra1/wskrap/publichtml/200623.pdf`.
11. W. C. Rounds and H. Song. The ϕ-calculus: A language for distributed control of reconfigurable embedded systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control 2003*, LNCS 2623, pages 435–449, 2003.
12. B. Tuffin, D. S. Chen, and K. S. Trivedi. Comparison of hybrid systems and fluid stochastic Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 11:77–95, 2001.
13. D. van Beek, K. Man, M. Reniers, J. Rooda, and R. Schiffelers. Syntax and consistent equation semantics of hybrid χ. *Journal of Logic and Algebraic Programming*, 68(1-2):129–210, 2006.