HYPE: A Process Algebra for Compositional Flows and Emergent Behaviour

Vashti Galpin[1], Luca Bortolussi[2], 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
luca@dmi.units.it

Abstract. Several process algebras for modelling hybrid systems have appeared in the literature in recent years. These all assume that continuous variables in the system are modelled monolithically, often with the differential equations embedded explicitly in the syntax of the process algebra expression. In HYPE an alternative approach is taken which offers finer-grained modelling with each flow or influence affecting a variable modelled separately. The overall behaviour then emerges as the composition of these flows. This approach is supported by an operational semantics which distinguishes states as collections of flows and which is supported by an equivalence which satisfies the property that bisimilar HYPE models give rise to the same sets of continuous behaviours.

1 Introduction

HYPE is a novel process algebra for modelling hybrid systems. A hybrid system exhibits both discrete and continuous behaviour. It can be viewed as a system consisting of values which change continuously over time with respect to specific dynamics. Discrete events can cause discontinuous jumps in these values after which different dynamics may come into effect. These events can be triggered by conditions on the continuously changing values. The novelty of HYPE lies in how it captures the continuous dynamics of a system, and the separation of a discrete controller considered in parallel to the system under study. Unlike existing process algebras for hybrid systems, HYPE captures behaviour at a fine-grained level, composing distinct flows or influences. The dynamic behaviour then emerges, via the semantics of the language, from these compositional elements. We are inspired by the fluid flow semantics of PEPA models [15] which approximates the behaviour of large numbers of discrete components with a set of ordinary differential equations (ODEs).

Hybrid behaviour arises in a variety of systems, both engineered and natural. Consider a thermostatically controlled 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. Another example would be a genetic regulatory network, such as the Repressilator [9], in which genes can be switched on or off by interactions with their environment (more precisely, with

transcription factor proteins). The behaviour of such systems can be regarded as a collection of sets of ODEs, the discrete events shifting the dynamic behaviour from the control of one set of ODEs to another. This is the approach taken with hybrid automata [13].

Process algebras have the advantage of being compositional hence models are built out of subcomponents and we aim to fully exploit this feature in HYPE. Existing process algebras for hybrid systems include ACP_{hs}^{srt} [3], hybrid χ [22], ϕ-calculus [20] and HyPA [6]. In [16], 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 all similar in their approach in that the dynamic behaviour of each subcomponent must be fully described with the ODEs for the subcomponent given explicitly in the syntax of the process algebra, before the model can be constructed.

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. 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 the 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.

We believe that the use of flows as the basic elements of model construction has advantages such as ease and simplification of modelling. This approach assists the modeller in allowing them to identify smaller or local descriptions of the model and then to combine these descriptions to obtain the larger system. The explicit controller also helps to separate modelling concerns.

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 rise to a hybrid automaton. We consider a notion of bisimulation and our main result is that bisimilar systems have identical ODEs and finally we discuss related work.

2 HYPE Definition

This section will present HYPE by way of a running example of the temperature control system of an orbiting space vehicle. As the orbiter travels around the earth, it needs to regulate its temperature to remain within operational limits. It has insulation but needs to use a heater at low temperatures and at high temperatures it can erect a shade to reduce temperatures. The modelling spirit of HYPE focusses on flows. In our example, we identify four flows affecting the temperature. One is due to thermodynamic cooling, one is due to the heater,

one is due to the heating effect of the sun and one is due to the cooling effect of the shade. The strength and form of a flow are modified by events. We first define the heater which can be on or off.

$$Heat \stackrel{def}{=} \underline{\text{on}}{:}(h, r_h, const).Heat + \underline{\text{off}}{:}(h, 0, const).Heat + \underline{\text{init}}{:}(h, 0, const).Heat$$

This is a *summation* of prefixes. Each prefix consists of two actions. *Events* ($\underline{a} \in \mathcal{E}$) are actions which happen instantaneously and trigger discrete changes. They can be caused by a controller or happen randomly and can depend on the global state of the system, specifically values of variables. In the example, the events are on, switching on; off, switching off and init, the initialisation event.

Activities ($\alpha \in \mathcal{A}$) are *influences* on the evolution of the continuous part of the system and define flows. An activity is defined as a triple and can be parameterised by a set of variables, $\alpha(\overrightarrow{X}) = (\iota, r, I(\overrightarrow{X}))$. This triple consists of an *influence name* ι, a rate of change (or *influence strength*) r and an *influence type name* $I(\overrightarrow{X})$ which describes how that rate is to be applied to the variables involved, or the actual form of the flow[3]. In $Heat$, there are two distinct activities, $(h, r_h, const)$ and $(h, 0, const)$. The first one captures the effect of the heater being off. It affects influence h which represents the influence from the heater on the orbiter's temperature, it has strength 0 and it is associated with the function called *const*. The second gives the effect of the heater being on: the influence name is again h, r_h is the strength of the heater, and the form it takes is *const*. The interpretation of influence types will be specified separately, so that experimentation with different functional forms of the heating flow can occur without modifying the subcomponent. Hence, in HYPE we separate the description of the logical structure of flows from their mathematical interpretation. We now describe the other flows for the example.

$$Shade \stackrel{def}{=} \underline{\text{up}}{:}(d, -r_d, const).Shade + \underline{\text{down}}{:}(d, 0, const).Shade + \underline{\text{init}}{:}(d, 0, const).Shade$$

$$Sun \stackrel{def}{=} \underline{\text{light}}{:}(s, r_s, const).Sun + \underline{\text{dark}}{:}(s, 0, const).Sun + \underline{\text{init}}{:}(s, 0, const).Sun$$

$$Cool(X) \stackrel{def}{=} \underline{\text{init}}{:}(c, -1, linear(X)).Cool(X)$$

The only event in the last definition is the initialisation event, as once cooling is in effect it does not change. The influence name is c and its strength is -1. The type $linear(X)$ will be interpreted as a linear function of its formal variable X. We also need to model the change in sunlight. We do this by keeping track of time with the following component (which is kept simple for reasons of space).

$$Time \stackrel{def}{=} \underline{\text{light}}{:}(t, 1, const).Time + \underline{\text{dark}}{:}(t, 1, const).Time + \underline{\text{init}}{:}(t, 1, const).Time$$

These subcomponents can be combined and the formal variable X can be instantiated with the actual variable K to give the overall uncontrolled system.

$$Sys \stackrel{def}{=} (((Heat \underset{\{\underline{\text{init}}\}}{\bowtie} Shade) \underset{\{\underline{\text{init}}\}}{\bowtie} Sun) \underset{\{\underline{\text{init}}\}}{\bowtie} Cool(K)) \underset{\{\underline{\text{init}},\underline{\text{light}},\underline{\text{dark}}\}}{\bowtie} Time$$

Here $\underset{L}{\bowtie}$ represents parallel synchronisation. L is the set of events over which synchronisation must occur. Events not in L can occur independently. Sys is called the *uncontrolled system* because all events are possible and no causal or

[3] For convenience, we will use I for $I(\overrightarrow{X})$ when $\overrightarrow{X}$ can be inferred.

temporal constraints have been imposed yet. For instance, we need to specify that the heater can only be switched off after it has been switched on. We now give controllers/sequencers for the heater, the shade and the effect of the sun.

$$Con_h \stackrel{def}{=} \underline{\text{on}}.\underline{\text{off}}.Con_h \quad Con_d \stackrel{def}{=} \underline{\text{up}}.\underline{\text{down}}.Con_d \quad Con_s \stackrel{def}{=} \underline{\text{light}}.\underline{\text{dark}}.Con_s$$
$$Con \stackrel{def}{=} Con_h \underset{\emptyset}{\bowtie} Con_d \underset{\emptyset}{\bowtie} Con_s$$

Controllers only have event prefixes. Their behaviour is affected by the state of the system through event conditions which determine when events occur. The controlled system is constructed from synchronisation of the controller and the uncontrolled system and the controller must be prefixed by the initialisation event $\underline{\text{init}}$. For the example, the controlled system is described by

$$TempCtrl \stackrel{def}{=} Sys \underset{M}{\bowtie} \underline{\text{init}}.Con \quad \text{with} \quad M = \{\underline{\text{init}}, \underline{\text{on}}, \underline{\text{off}}, \underline{\text{up}}, \underline{\text{down}}, \underline{\text{light}}, \underline{\text{dark}}\}.$$

This has defined the structure of our system but we require additional definitions to capture further details. We need to link each influence with an actual variable. This is done using the function iv. For the example, $iv(h) = iv(s) = iv(d) = iv(c) = K$ where K is the actual variable for the temperature of the orbiter, and $iv(t) = T$, the variable for time. Note that an influence can only be associated with one variable, in agreement with the interpretation of influences as flows. This does not mean that only one variable can be affected by an event. For another variable Y that was also affected by the heater being on (power consumption, say), we could define a subcomponent with a prefix $\underline{\text{on}}{:}(p, r, I)$ and set $iv(p) = Y$.

We define the influence types as $[\![const]\!] = 1$ and $[\![linear(X)]\!] = X$. The influence types are used to describe influences are affected by variables in the system. The type *const* is used when there is no effect and $linear(X)$ is used when the value of the variable X modifies an influence. Mass action can also be defined through this mechanism.

Finally, we define what triggers an event, and how it affects variables, with the function ec. Each event condition consists of an activation condition which is a positive boolean formula containing equalities and inequalities on system variables or the symbol $\bot$, and a variable reset which 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 $V = V'$ can be left implicit. For the example, the function ec and associated event conditions are

$$ec(\underline{\text{init}}) = (true, (K' = t_0 \wedge T' = 0))$$
$$ec(\underline{\text{off}}) = (K \geq t_1, true) \qquad ec(\underline{\text{on}}) = (K \leq t_2, true)$$
$$ec(\underline{\text{up}}) = (K \geq t_3, true) \qquad ec(\underline{\text{down}}) = (K \leq t_4, true)$$
$$ec(\underline{\text{light}}) = (T = 12, true) \qquad ec(\underline{\text{dark}}) = (T = 24, T' = 0)$$

where the $t_i (1 \leq i \leq 4)$ are fixed temperature values. Most events are urgent – the event must occur as soon as its event condition is satisfied. For events that can happen randomly such as breakdowns, we introduce a special event condition $\bot$ which means that the event can happen at some point in the future[1]. In the example, the $\underline{\text{init}}$ event has an associated event condition of *true* and so this

[1] We have not done so here but probabilistic resets can be used. Tuffin *et al* [21] use a value drawn from a exponential distribution for the time until the next event.

must happen immediately and light happens when 12 hours have passed. The event init has a reset that defines the values of the variables and on has a reset of *true* meaning that no values are changed.

In the preceding informal discussion, we have introduced the main constituents of a HYPE model including the combination of flow components with a controller component, formal and actual variables, association between influences and variables, conditions that specify when events occur, and definitions for the influence type functions. To understand the dynamics of this system, we need to derive ODEs to describe how the variables change over time. To do this we present operational semantics that define the behaviour of our controlled system. Before that we present the formal definition of HYPE.

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

- Subcomponents *are defined by* $C_s(\overrightarrow{X}) = S$*, where* C_s *is the* subcomponent name *and* S *satisfies the grammar* $S' ::= \underline{a} : \alpha.C_s \mid S' + S'$ *(*$\underline{a} \in \mathcal{E}$*,* $\alpha \in \mathcal{A}$*), with the free variables of* S *in* $\overrightarrow{X}$*.*
- Components *are defined by* $C(\overrightarrow{X}) = P$*, where* C *is the* component name *and* P *satisfies the grammar* $P' ::= C_s(\overrightarrow{X}) \mid C(\overrightarrow{X}) \mid P' \underset{L}{\bowtie} P'$*, with the free variables of* P *in* $\overrightarrow{X}$ *and* $L \subseteq \mathcal{E}$*.*
- *An* uncontrolled system Σ *is defined according to the grammar* $\Sigma' ::= C_s(\overrightarrow{V}) \mid C(\overrightarrow{V}) \mid \Sigma' \underset{L}{\bowtie} \Sigma'$*, where* $L \subseteq \mathcal{E}$ *and* $\overrightarrow{V}$ *is a set of system variables, instantiating the formal variables of* C *or* C_s*.*
- 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* $Con ::= M \mid Con \underset{L}{\bowtie} Con$*.*
- *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, gives a HYPE model.

Definition 2. *A* HYPE model *is a tuple* $(ConSys, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, ec, iv, EC, ID)$ *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 type names.*
- $\mathcal{E}$ *is a set of events of the form* $\underline{a}$ *and* $\underline{a_i}$*.*
- $\mathcal{A}$ *is a set of activities of the form* $\alpha(\overrightarrow{X}) = (\iota, r, I(\overrightarrow{X})) \in (IN \times \mathbb{R} \times IT)$*.*
- $ec : \mathcal{E} \to EC$ *maps events to event conditions. Event conditions are pairs of formulas, the first with free variables in* $\mathcal{V}$ *and the second with free variables in* $\mathcal{V} \cup \mathcal{V}'$*.*
- $iv : IN \to \mathcal{V}$ *maps influence names to variable names.*
- EC *is a set of event conditions.*
- ID *is a collection of definitions consisting of a real-valued function for each influence type name* $[\![I(\overrightarrow{X})]\!] = f(\overrightarrow{X})$ *where the variables in* $\overrightarrow{X}$ *are from* $\mathcal{X}$*.*
- $\mathcal{E}$*,* $\mathcal{A}$*,* IN *and* IT *are pairwise disjoint.*

Prefix with influence:
$$\frac{}{\langle \underline{a} : (\iota, r, I).E, \sigma\rangle \xrightarrow{\underline{a}} \langle E, \sigma[\iota \mapsto (r, I)]\rangle}$$

Prefix without influence:
$$\frac{}{\langle \underline{a}.E, \sigma\rangle \xrightarrow{\underline{a}} \langle E, \sigma\rangle}$$

Choice:
$$\frac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle E + F, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle} \qquad \frac{\langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}{\langle E + F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}$$

Parallel without synchronisation:
$$\frac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E' \underset{M}{\bowtie} F, \sigma'\rangle}(\underline{a} \notin M)$$
$$\frac{\langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E \underset{M}{\bowtie} F', \sigma'\rangle}(\underline{a} \notin M)$$

Parallel with synchronisation:
$$\frac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \tau\rangle \quad \langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \tau'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E' \underset{M}{\bowtie} F', \Gamma(\sigma, \tau, \tau')\rangle}(\underline{a} \in M, \Gamma \text{ defined})$$

Constant:
$$\frac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle A, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}(A \overset{def}{=} E)$$

Fig. 1. Operational semantics for HYPE

When referring to 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. The syntax of HYPE is moderately complex because hybrid systems are complex structures displaying continuous and discrete behaviour. The example shows how it is straightforward to construct a HYPE model once flows and the controller are identified.

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 $(\mathcal{F}, \mathcal{E}, \rightarrow \subseteq \mathcal{F} \times \mathcal{E} \times \mathcal{F})$. We write $F \xrightarrow{\underline{a}} F'$ for $(F, \underline{a}, F') \in \rightarrow$. In the following, $E, F \in \mathcal{C}_{Sys}$. The rules are given in Figure 1 and are fairly standard. In Choice, Prefix without influence, Parallel without synchronisation and Constant, states are not changed by the application of the rule. For Prefix with influence, the state needs to be updated, and for Parallel with synchronisation, the two new states in the premise of the rule need to be merged using the function Γ.

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 $\Gamma : \mathcal{S} \times \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$ 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, namely if the values in the new state both differ from the old state since this represents conflicting updates.

The next two definitions will be useful in referring to the states of the model.

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* $\langle P', \sigma' \rangle \xrightarrow{\underline{a}} \langle P'', \sigma'' \rangle$ *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* $\mathrm{st}(P) = \{\sigma \mid \langle Q, \sigma \rangle \in \mathrm{ds}(P)\}$.

In the *TempCntl* model, there are eight states of interest (we have omitted the state before the $\underline{\text{init}}$ event) given in Figure 2. Each state captures the influences that are currently active. Since the influence strengths and types of c and t do not change, and each of h, d and s have two possible strengths, there are eight states. Here k abbreviates *const* and $l(X)$, $linear(X)$. For example, σ_3 reflects that the heater (h) is off (and has no effect), the shade (d) is up, the sun (s) is shining, cooling (c) is happening, and time (t) is passing.

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.

$$
\begin{aligned}
\sigma_0 &= \{h \mapsto (0,k),\ d \mapsto (0,k),\ s \mapsto (0,k),\ c \mapsto (-1,l(K)),\ t \mapsto (1,k)\}\\
\sigma_1 &= \{h \mapsto (0,k),\ d \mapsto (0,k),\ s \mapsto (r_s,k),\ c \mapsto (-1,l(K)),\ t \mapsto (1,k)\}\\
\sigma_2 &= \{h \mapsto (0,k),\ d \mapsto (-r_d,k),\ s \mapsto (0,k),\ c \mapsto (-1,l(K)),\ t \mapsto (1,k)\}\\
\sigma_3 &= \{h \mapsto (0,k),\ d \mapsto (-r_d,k),\ s \mapsto (r_s,k),\ c \mapsto (-1,l(K)),\ t \mapsto (1,k)\}\\
\sigma_4 &= \{h \mapsto (r_h,k),\ d \mapsto (0,k),\ s \mapsto (0,k),\ c \mapsto (-1,l(K)),\ t \mapsto (1,k)\}\\
\sigma_5 &= \{h \mapsto (r_h,k),\ d \mapsto (0,k),\ s \mapsto (r_s,k),\ c \mapsto (-1,l(K)),\ t \mapsto (1,k)\}\\
\sigma_6 &= \{h \mapsto (r_h,k),\ d \mapsto (-r_d,k),\ s \mapsto (0,k),\ c \mapsto (-1,l(K)),\ t \mapsto (1,k)\}\\
\sigma_7 &= \{h \mapsto (r_h,k),\ d \mapsto (-r_d,k),\ s \mapsto (r_s,k),\ c \mapsto (-1,l(K)),\ t \mapsto (1,k)\}
\end{aligned}
$$

Fig. 2. The states of the orbiter temperature control system

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.

$$CS_\sigma = \Big\{ \frac{\mathrm{d}V}{\mathrm{d}t} = \sum \big\{ r \times [\![I(\overrightarrow{W})]\!] \;\big|\; iv(\iota) = V \text{ and } \sigma(\iota) = (r, I(\overrightarrow{W})) \big\} \;\Big|\; V \in \mathcal{V} \Big\}$$

So for each state, we have a collection of ODEs, one for each variable V. We obtain the following ODE from σ_3.

$$\mathit{TempCntl}_{\sigma_3} = \Big\{ \frac{\mathrm{d}K}{\mathrm{d}t} = r_s - r_d - K,\ \frac{\mathrm{d}T}{\mathrm{d}t} = 1 \Big\}$$

Therefore, this process enables us to obtain ODEs describing how the continuous part of the system evolves, and we have different sets of ODEs to describe the different dynamics that can be in operation. We wish to combine this information with the event conditions already defined and an obvious way to do this is to translate this information into a hybrid automaton. Therefore, this well-supported formalism provides a powerful back-end for HYPE.

Hybrid automata are dynamic systems presenting both discrete and continuous evolution. They consist 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 [13] 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:*

- $\mathbf{X} = \{X_1, \ldots, X_n\}$ *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*

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* $event(e) \in \mathcal{E}$ *and by* $jump(e)$*, a formula on* $\mathbf{X}$ *stating for which values of variables each transition is active, and by* $reset(e)$*, a formula 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* $urgent(e)$*, meaning that the transition is taken at once when* $jump(e)$ *becomes true. Otherwise, the transition can be taken nondeterministically whenever* $jump(e)$ *is true.*

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

Definition 7. *The hybrid automaton*
$\mathcal{H} = (V, E, \mathbf{X}, \mathcal{E}, flow, init, inv, event, jump, reset, urgent)$ *can be obtained from the HYPE model* $(P_0, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, ec, iv, EC, ID)$ *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*
$$flow(v_j)[X_i] = \sum\{r[\![I(\overrightarrow{W})]\!] \mid iv(\iota) = X_i \text{ and } \sigma_j(\iota) = (r, I(\overrightarrow{W}))\}$$
- $init(v) = \begin{cases} res_{\underline{init}}, \text{ if } v = \langle P, \sigma \rangle \\ false, \text{ otherwise} \end{cases}$ *with primes removed from variables*[1].
- $inv(v) = true$.
- *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 3 shows the HYPE model from the example as a hybrid automaton. Note that which states are visited is determined by the values of the $t_i (0 \leq i \leq 4)$.

5 Equivalence Semantics

We define equivalent behaviour with respect to the labelled transition system given in Section 3 thereby focussing on the configuration of the system rather than evaluations of continuous variables.

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

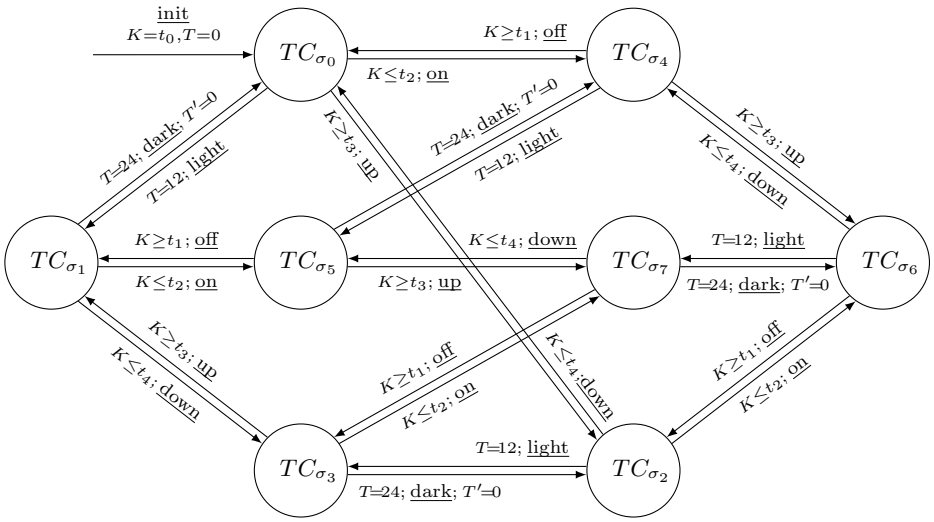

Fig. 3. Hybrid automaton of the orbiter temperature control system

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 $\langle P_1 \underset{L}{\bowtie} Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1' \underset{L}{\bowtie} Q', \Gamma(\sigma, \sigma', \sigma'')\rangle$ and $\underline{a} \in L$ then $\langle P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1', \sigma'\rangle$ and $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma''\rangle$. 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 the example, the 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* $C_s(\overrightarrow{X}) \stackrel{\text{def}}{=} S$*, the only subcomponent name that can appear in* S *is* $C_s(\overrightarrow{X})$*.*
2. *In each subcomponent* $C_s(\overrightarrow{X})$*, the event* $\underline{a}$ *can only appear once.*
3. *In each subcomponent* $C_s(\overrightarrow{X})$*, each* ι *that appears, must also appear in a prefix with* $\underline{init}$*.*
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*.*

The example is a well-defined controlled system. For such systems, we can show that the current state in a configuration does not determine which 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 $\underline{\mathrm{a}} : (\iota, r, I(\overrightarrow{X}))$. On the occurrence of $\underline{\mathrm{a}}$, the value of ι in the state will be updated exactly once and hence Γ is always defined. If we did not have Condition 4 and event and influence names appeared multiple times, Γ could be undefined which is not a desired feature of modelling in HYPE. We now consider properties of well-defined HYPE systems.

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* $\langle P, \sigma\rangle \stackrel{init}{\Longrightarrow} \langle P', \sigma'\rangle$ *and* $\langle P, \tau\rangle \stackrel{init}{\Longrightarrow} \langle P', \tau'\rangle$ *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.* □

This result allows us to tell whether two HYPE models are bisimilar by inspecting the prefixes in the model description to see if they are the same. Hence bisimulation can be checked syntactically. The next two results show that bisimilar HYPE models have the same ODEs.

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 $(P, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, ec, iv, EC, ID)$ *and* $(Q, \mathcal{V}, \mathcal{X}, IN, IT, \mathcal{E}, \mathcal{A}, ec, iv, EC, ID)$*. By Lemma 3,* $\sigma \in \mathrm{st}(P)$ *implies* $\sigma \in \mathrm{st}(Q)$*. Hence*

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

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

which are clearly the same. □

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.

6 Related Work

As mentioned in the introduction, HYPE takes a finer grained, less monolithic approach than the other process algebras for hybrid systems [3, 22, 20, 6] because it enables the modelling of individual flows. In [16] the comparison of these other process algebras is based on a train gate controller example and in each case, the train, gate and controller components have to be fully, sequentially described

and then composed in parallel. This would also be necessary for the modelling of our orbiter example. For each of these process algebras, somewhere in the syntactic description of the system, a term such as $\dot{K} = r_s - r_d - K$, as well as terms for each of the other seven ODEs for the variable K, would need to appear to describe the continuous behaviour that can occur. By comparison, a modeller using HYPE would only need to model the individual flows, and not construct the ODEs explicitly. This could allow non-experts to model hybrid systems more easily.

A classical formalism for expressing hybrid systems is hybrid automata [13]. They are usually specified by defining explicitly both the control graph and the dynamical conditions within each mode, in terms of differential equations or, more generally, differential inclusions. Two hybrid automata can be composed in parallel by synchronizing transitions on shared events in the control graph [13]. Flow conditions are combined by taking the logical conjunction of the predicates defining them. Where flows are defined by differential equations, the equation for each variable X must be defined only in one component, otherwise a logical inconsistency may arise. However, the variable may depend explicitly on variables governed by other components.

The modelling style of HYPE is quite different. Activities are identified with atomic flows acting on system variables. ODEs are then derived for an individual state by adding the different atomic flows acting on each variable. Activities can change in response to the happening of discrete events, which are controlled not by components but by an external controller and triggered by event conditions. This results in a separation of the description of the response of the system to events from the discrete control structure imposing causality on the happening of events. In contrast to hybrid automata, HYPE allows the separate description of flow conditions, event conditions, and the control graph, making easier the task of modifying the controller or the interactions with the environment. Furthermore, the fact that influence types are defined separately from the structure of the model also separates modelling concerns. In addition, compositionality of HYPE manifests on the set of activities (the state of the system) rather than on ODEs, hence we can allow different components of the system to influence the same continuous variable: the combined effect is obtained by superimposing flows, namely by addition on the right hand side of ODEs. Since a HYPE model can be expressed as a hybrid automaton, when using HYPE to model one gets the advantages of HYPE together with the formalism of hybrid automata. This is a distinct advantage over languages such as CHARON [1], SHIFT [8], and HyCharts [11]. These are all compositional formalisms describing hybrid systems which do not map so readily to hybrid automata. They differ from HYPE in that they do not have the simple syntax and structured operational semantics of a process algebra.

Another formalism which has a mechanism to combine flows is hybrid action systems [19]. This language is based on Dijkstra's guarded command language and has predicate transformer semantics. Differential actions consist of guards before ODEs, and parallel composition of these actions is defined as a linear

combination of functions with the addition of functions over any shared domains. This differs from our approach where we associate influence names with a specific variable and then sum over all influences for a given variable to obtain the ODE.

Physical systems can also be modelled by constitutive equations and bond graphs [18]. This is a modelling technique in which a system's components are described by means of equations relating main physical quantities of interest. Components are then glued together by imposing suitable conservation laws. This results in a set of differential equations with algebraic constraints, which after an algebraic manipulation, can be simplified by removing variables and constraints. There are also hybrid extensions of the bond graph method which have been represented in a hybrid process algebra [5], to deal with discontinuities in physical systems (such as a bouncing ball).

In HYPE, instead, the modelling activity concentrates around the notion of flow or influence, which is not explicitly connected with physical quantities or with conservation laws, and components are described by specifying the way they react to external events through flows modifications. This may be less natural for certain physical systems, as one has to identify the different influences acting on each variable of interest (without relying on the implicit derivation mechanism provided by conservation laws). However, HYPE's modelling style is straightforwardly applicable to a wider class of systems, at different levels of abstraction.

6.1 Bisimulations on Hybrid Systems

Other process algebras for hybrid systems use a hybrid transition system with two types of transition: one type represents discrete events and the other continuous evolution of the system [16]. By comparison, our transition system only has transitions for events. This gives a smaller transition system on which it is possible to consider simpler notions of equivalence.

Bergstra and Middelburg [3] present two bisimulations for their process algebra for hybrid systems, defined over a hybrid transition system. One bisimulation 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.

The standard notion of bisimulation of hybrid automata is defined for a transition system encoding the dynamical evolution. Both continuous transitions and discrete transitions are used. These two relations are combined (usually interleaving discrete and continuous transitions) into one relation which defines the behaviour of the system. The hybrid automata bisimulation is defined on this relation in the usual way [13, 12, 7]. Most of the research into these bisimulations focusses on finding restrictions on hybrid automata syntax implying the existence of a finite bisimulation quotient thus guaranteeing decidability of reachability and of model checking, such as [17].

The transition relation defined for HYPE is very different. It does not encode any notion of time-dynamics and hence it acts at the level of the system

description, identifying equivalent modes. In this sense, it is closer to the notion of U-bisimulation for hybrid automata which acts on the control graph [2]. If we consider a simplified form of U-bisimulation, we can show that any HYPE models that are system bisimilar, are also equated by this bisimulation. The converse does not hold since different states can lead to the same ODEs. 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.

7 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 [14] can be used to explore these systems.

As well as the orbiter temperature control example, we have successfully modelled a dual-tank system (as described by [21]), a bottling line (as described by [3]) and the abstract view of the repressilator [10] (as described by [4]).

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. Related to the idea of equivalence is axiomatisation and this will also be investigated.

Our models currently give rise to hybrid automata with invariants which are always true (see Definition 7) and we will investigate relaxing this condition. We have focussed on the embedding of HYPE models in hybrid automata; in future we wish to identify the class of hybrid automata to which HYPE models correspond as a way to understand how to solve HYPE models.

Acknowledgements Thanks to Stephen Gilmore and Pieter Cuijpers for their helpful comments. Vashti Galpin is supported by the EPSRC SIGNAL Project, Grant EP/E031439/1. Luca Bortolussi is supported by GNCS and FIRB LIBi. Jane Hillston is supported by the EPSRC under ARF EP/c543696/01.

References

1. R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional modeling and refinement for hierarchical hybrid systems. *Journal of Logic and Algebraic Programming*, 68(1-2):105–128, 2006.
2. 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 (CMSB 2003)*, pages 57–74, 2003.

3. J. A. Bergstra and C. A. Middelburg. Process algebra for hybrid systems. *Theoretical Computer Science*, 335(2-3):215–280, 2005.
4. L. Bortolusssi and A. Policriti. Hybrid approximation of stochastic process algebras for systems biology. In *IFAC World Congress,* Seoul, South Korea, July 2008.
5. P. Cuijpers, J. Broenink, and P. Mosterman. Constitutive hybrid processes: a process-algebraic semantics for hybrid bond graphs. *SIMULATION*, 8:339–358, 2008.
6. P. J. L. Cuijpers and M. A. Reniers. Hybrid process algebra. *Journal of Logic and Algebraic Programming*, 62(2):191–245, 2005.
7. J. M. Davoren and P. Tabuada. On simulations and bisimulations of general flow systems. In A. Bemporad, A. Bicchi, and G. C. Buttazzo, editors, *Hybrid Systems: Computation and Control (HSCC 2007)*, LNCS 4416, pages 145–158, 2007.
8. A. Deshpande, A. Göllü, and P. Varaiya. SHIFT: A formalism and a programming language for dynamic networks of hybrid automata. In P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems IV*, LNCS 1273, pages 113–133, 1996.
9. M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.
10. V. Galpin, J. Hillston, and L. Bortolussi. HYPE applied to the modelling of hybrid biological systems. *Electronic Notes in Theoretical Computer Science*, 218:33–51, 2008.
11. R. Grosu and T. Stauner. Modular and visual specification of hybrid systems: An introduction to HyCharts. *Formal Methods in System Design*, 21(1):5–38, 2002.
12. E. Haghverdi, P. Tabuada, and G. J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, 342(2-3):229–261, 2005.
13. T. A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, 1996.
14. 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.
15. 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.
16. 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.
17. G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, 2000.
18. H. Paynter. *Analysis and Design of Engineering Systems.* MIT Press, 1961.
19. M. Rönkkö, A. P. Ravn, and K. Sere. Hybrid action systems. *Theoretical Computer Science*, 290(1):937–973, 2003.
20. 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 (HSCC 2003)*, LNCS 2623, pages 435–449, 2003.
21. 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.
22. 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.