HYPE with stochastic events

Luca Bortolussi
Department of Maths and Computer Science, University of Trieste.
luca@dmi.units.it

Vashti Galpin Jane Hillston
Laboratory for Foundations of Computer Science, University of Edinburgh
Vashti.Galpin@ed.ac.uk, Jane.Hillston@ed.ac.uk

The process algebra HYPE was recently proposed as a fine-grained modelling approach for capturing the behaviour of hybrid systems. In the original proposal, each flow or influence affecting a variable is modelled separately and the overall behaviour of the system then emerges as the composition of these flows. The discrete behaviour of the system is captured by instantaneous actions which might be urgent, taking effect as soon as some activation condition is satisfied, or non-urgent meaning that they can tolerate some (unknown) delay before happening. In this paper we refine the notion of non-urgent actions, to make such actions governed by a probability distribution. As a consequence of this we now give HYPE a semantics in terms of Transition-Driven Stochastic Hybrid Automata, which are a subset of a general class of stochastic processes termed Piecewise Deterministic Markov Processes.

1 Introduction

Process algebras have been successfully applied to the analysis and verification of a wide variety of systems over the last thirty years. Although initially focused on semantic issues of concurrent programming, their compositional style and ability to support a number of different analysis techniques has extended their use into many application domains. In the realm of quantified analysis stochastic process algebras, in which actions are associated with a randomly distributed delay, have been used to study the dynamics of diverse systems ranging from the performance of software systems [11] to the biochemical signalling in living cells [3]. Such an analysis is inherently based on a discrete state view of the system with an underlying semantics which is generally a continuous time Markov chain (CTMC). In contrast, recently, process algebras have been used to study situations of collective dynamics in which a fluid approximation of the discrete state space is used to arrive at a semantics in terms of sets of ordinary differential equations (ODEs) [10, 16].

Hybrid behaviour arises in a variety of systems, both engineered and natural. Such systems combine elements of both the approaches outlined above as the system will undergo periods of continuous evolution, governed by ODEs, punctuated by discrete events which can alter the course of subsequent continuous evolution. 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 [17]. Another example would be a genetic regulatory network, such as the Repressilator [5, 6], 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 [9]. Given the previous success

Submitted to:
QAPL 2011

of capturing discrete and continuous scenarios with process algebras in the past it is therefore natural to consider process algebras for hybrid settings.

A number of process algebras for describing hybrid systems have appeared in recent years [12], substantially differing in the approaches taken 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. What distinguishes HYPE [7] is that it captures behaviour at a fine-grained level, composing distinct flows or influences which act on the continuous variables of the system. At a superficial level this removes the need to explicitly write ODEs in the process algebra syntax. Instead the dynamic behaviour emerges, via the semantics of the language, when these elements are composed. Moreover 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.

In the original definition of HYPE, discrete actions are termed *events* and are always considered *instantaneous* although some are subject to an activation condition which will determine when that instantaneous jump occurs. Most events are conditioned on the values of continuous variables which are evolving in the system and will be triggered when the activation condition becomes true; such events are termed *urgent*. Many systems also respond to events which are not so tightly tied to the continuous evolution of the system and may appear to occur randomly. In the original definition of HYPE such actions were given an undefined activation condition, denoted $\perp$ and termed *non-urgent*. However if we wish to carry out quantified analysis of the constructed models such events may be regarded as underspecified since we capture no information about their potential firing. Thus here we seek to refine this notion of non-urgent events, by introducing *stochastic actions*. These actions will have an activation condition which is a random variable, capturing the probability distribution of the time until the event occurs. Thus these event still occur non-deterministically and are not directly linked to the values of continuous variables, but they are now quantified and so the models admit quantitative analysis.

This small modification substantially enriches the class of underlying mathematical processes which capture the behaviour of systems modelled in HYPE. Previously we gave HYPE a semantics in terms of hybrid automata [9]. Now we give a semantics in terms of Piecewise Deterministic Markov Processes (PDMPs) [4], using the richer class of automata, Transition Driven Stochastic Hybrid Automata (TDSHA) as an intermediary. Due to space constraints, in this paper we will only show how to associate a TDSHA for a given HYPE model. Mapping TDSHAs to PDMPs can be done along the lines of [2].

TDSHA have also been used in [1] to define a hybrid semantics for PEPA, a well-known stochastic process algebra [11]. That application of TDSHA is rather different from the one presented here. In [1], we construct a hybrid system approximating the behaviour of the CTMC associated with a PEPA model by the standard semantics, using just continuous flows and stochastic events. HYPE, by contrast, is a process algebra expressly designed to model hybrid system, hence it deals with both instantaneous and stochastic events.

The rest of this paper is organised as follows. In Section 2 we briefly recall the basic notions of HYPE by means of a running example, explaining how to extend it in the stochastic setting in Section 2.1. Sections 3 and 4 are devoted to recall the definition of TDSHA and to describe how to construct a TDSHA for a given HYPE model. Finally, Sections 5 and 6 discuss related work and draw final conclusions.

2 HYPE Definition

In this section we recall the definition of non-stochastic HYPE by way of a running example. More details about the language can be found in [7, 8].

We consider an orbiter which travels around the earth and 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 reflect solar radiation and reduce temperature. Its HYPE model, is given in Table 1. The whole system is described by *TempCtrl*, and it is composed of two pieces: an uncontrolled system *Sys* and a controller *Con*, plus some additional information.

HYPE modelling is centered around the notion of *flow*, which is some sort of influence continuously modifying one variable. Both the strength and form of a flow can be changed by *events*. In our example, we identify four flows affecting the temperature, modeled by the variable K. 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.

Flows are described by the *uncontrolled system*, a composition of several sequential subcomponents, each modelling how a specific flow is changed by events. For instance, in Table 1, the subcomponent *Heat* describes the heating system, which reacts to the events turning it on and off (on and off). The tuple $(h, r_h, const)$ following event on, is called an *activity* or an *influence* and describes how the heater affects the temperature when it is working: h is the name of the influence, which provides a link to the target variable of the flow (K in our example), r_h is the strength of the influence and *const* is the influence type, identifying the functional form of the flow (which is specified separately by the interpretation $[\![const]\!] = 1$). When the heater is turned off, the influence $(h, r_h, const)$ is replaced by $(h, 0, const)$, i.e. the influence strength of the heater becomes zero. The other subcomponents affecting temperature are *Shade*, *Sun*, and $Cool(K)$, while *Time* keeps track of the flow of time. States of a HYPE model are collections of influences, one for each influence name, defining a set of ordinary differential equations describing the continuous evolution of the system. For instance, $(h, r_h, const)$ contributes to the ODE of K with the addend $r_h[\![const]\!] = r_h$.

The controller *Con*, instead, is used to impose causality on events, either due to nature (such as the alternation of day and night) or by design. For instance, Con_h expresses the fact that the heater can be turned off only if it is on. Events happen when certain conditions are met by the system. These *event conditions* are specified by a function *ec*, assigning to each event a *guard* or *activation condition* (stating when a transition can fire) and a *reset* (specifying how variables are modified by the event). For example, $ec(\underline{\text{on}}) = (K \leq k_2, true)$ states that the heater is turned on when the temperature falls below a threshold k_2 and no variable is modified and $ec(\underline{\text{dark}}) = (T = 24, T' = 0)$ states that the event dark happens after 24 hours and resets the clock T to zero. Events in HYPE are urgent, meaning that they fire as soon as their guard becomes true. HYPE has also non-urgent events, whose guard is denoted by $\perp$. They can happen after an unconstrained, non-deterministic time delay.

A full HYPE model is given by $(ConSys, \mathscr{V}, IN, IT, \mathscr{E}, \mathscr{A}, ec, iv, EC, ID)$, where *ConSys* is the controlled system, $\mathscr{V}$ is the set of continuous variables, $\mathscr{E}$ is the set of events, *EC* is the set of event conditions, $ec : \mathscr{E} \to EC$ associates event conditions to events, *IN* is a set of influence names, *IT* is a set of influence types, $\mathscr{A}$ is a set of possible influences, $iv : IN \to \mathscr{V}$ maps influence names to variable names, and *ID* associates a real-valued function with each influence type. Formally, the semantics of HYPE is defined via structured operational semantics [7, 8], which is then interpreted in terms of hybrid automata.

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

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

$$\begin{aligned}
Heat &\stackrel{def}{=} \underline{\text{on}}:(h, r_h, const).Heat + \underline{\text{off}}:(h, 0, const).Heat + \underline{\text{init}}:(h, 0, const).Heat \\
Shade &\stackrel{def}{=} \underline{\text{up}}:(d, -r_d, const).Shade + \underline{\text{down}}:(d, 0, const).Shade + \\
&\quad \underline{\text{init}}:(d, 0, const).Shade \\
Sun &\stackrel{def}{=} \underline{\text{light}}:(s, r_s, const).Sun + \underline{\text{dark}}:(s, 0, const).Sun + \underline{\text{init}}:(s, 0, const).Sun \\
Cool(K) &\stackrel{def}{=} \underline{\text{init}}:(c, -1, linear(K)).Cool(K) \\
Time &\stackrel{def}{=} \underline{\text{light}}:(t, 1, const).Time + \underline{\text{dark}}:(t, 1, const).Time + \underline{\text{init}}:(t, 1, const).Time
\end{aligned}$$

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

$$iv(t) = T \qquad iv(h) = iv(d) = iv(s) = iv(c) = K$$

$$\begin{aligned}
ec(\underline{\text{init}}) &= (true, (K' = t_0 \wedge T' = 0)) \\
ec(\underline{\text{off}}) &= (K \geq k_1, true) & ec(\underline{\text{on}}) &= (K \leq k_2, true) \\
ec(\underline{\text{up}}) &= (K \geq k_3, true) & ec(\underline{\text{down}}) &= (K \leq k_4, true) \\
ec(\underline{\text{light}}) &= (T = 12, true) & ec(\underline{\text{dark}}) &= (T = 24, T' = 0)
\end{aligned}$$

Figure 1: Orbiter model in HYPE.

2.1 HYPE with stochastic events

We now consider how the HYPE language can be enriched with stochastic transitions, namely events which are not triggered by particular values of system variables but according to a random variable, whose distribution may depend on system variables or may be independent. These transitions may be considered as a generalisation of the non-urgent transitions which were previously specified with the event condition $\perp$. In the simplest case they will correspond to an event which occurs after an exponentially distributed delay with constant fixed rate.

To illustrate the use of stochastic transitions we consider an extension of our previous orbiter example. We now suppose that as well as monitoring its own temperature in order to regulate it and maintain correct operation, the orbiter is also collecting temperature data. These data are periodically downloaded to earth. The instigation of the download comes from a control room on earth and is outside the control of the orbiter. This will be governed by an exponential distribution with a fixed, constant rate. Between downloads, data will accumulate deterministically at a constant rate. When a download is commenced its duration will depend on the amount of data which has currently accumulated and will thus be an exponential distribution with a fixed parameter which depends on a system variable. It is possible to also imagine a download rate which is dependent on the current temperature of the orbiter, which would be an exponential distribution with a variable rate.

We assume that the system variable recording the amount of data currently stored on the orbiter is

D. The value of D is governed by two influences representing the accumulation and downloading of data respectively. These are $\delta_1 = (dw, r, const)$; $\delta_0 = (dw, 0, const)$ respectively. Clearly both these correspond to a single influence name dw, with $iv(dw) = D$.

The two events that modify the status of the influence dw are $\overline{\text{request}}$ and $\overline{\text{completed}}$, and they are stochastic. We model this fact by assuming that their activation condition is a rate function, depending on the value of continuous variables, which is the parameter of the exponential distribution governing their firing time. Resets, instead, behave as for instantaneous transitions. Hence,

$$ec(\overline{\text{request}}) = (\lambda_r, true) \qquad ec(\overline{\text{completed}}) = \left(\frac{\lambda}{\mu + D}, D' = 0\right).$$

The form of the rate function for $\overline{\text{completed}}$ guarantees that its rate is λ/μ when $D = 0$ and goes monotonically to zero as D goes to infinity (i.e. expected time of the event is minimal when there is no data, and grows linearly with D). Here λ/μ represents the maximum downloading speed (which is achieved when there is no data to collect), while μ controls the amount of data required to halve the download speed.

Then, the full orbiter model is obtained by adding one more component to the uncontrolled system of previous section.

$$Dwnldr \stackrel{def}{=} \underline{\text{init}} : \delta_1 . Dwnldr + \overline{\text{request}} : \delta_0 . Dwnldr + \overline{\text{completed}} : \delta_1 . Dwnldr$$

Furthermore, the downloading events are controlled by the following controller, synchronizing them with the rest of the system:

$$Con_{dw} \stackrel{def}{=} \overline{\text{request}} . \overline{\text{completed}} . Con_{dw}$$

Note how the compositionality of HYPE allows us to extend models in a simple and natural way.

From a syntactic point of view, a stochastic HYPE model is described by a tuple $(ConSys, \mathscr{V}, IN, IT, \mathscr{E}_d, \mathscr{E}_s, \mathscr{A}, ec, iv, EC, ID)$ in a similar fashion to HYPE. The main difference with respect to non-stochastic HYPE is that events are separated into two disjoint sets, $\mathscr{E}_d$ and $\mathscr{E}_s$, the instantaneous and the stochastic events, respectively[1]. Furthermore, event conditions are different between instantaneous and stochastic events. From a semantic point of view, instead, the semantics of stochastic HYPE will be defined by associating a (Transition-Driven) Stochastic Hybrid Automaton to each HYPE model, as described in the next sections. We now give the formal definition of a stochastic HYPE model which consists of a controlled system together with the appropriate sets and functions.

Definition 1 *A* stochastic HYPE model *is a tuple* $(ConSys, \mathscr{V}, IN, IT, \mathscr{E}_d, \mathscr{E}_s, \mathscr{A}, ec, iv, EC, ID)$ *where*

- *ConSys is a controlled system as defined below.*
- $\mathscr{V}$ *is a finite set of variables.*
- *IN is a set of influence names and IT is a set of influence type names.*
- $\mathscr{E}_d$ *is the set of instantaneous events of the form* $\underline{a}$ *and* $\underline{a}_i$.
- $\mathscr{E}_s$ *is the set of stochastic events of the form* $\overline{a}$ *and* $\overline{a}_i$.
- $\mathscr{A}$ *is a set of activities of the form* $\alpha(\mathscr{W}) = (\iota, r, I(\mathscr{W})) \in (IN \times \mathbb{R} \times IT)$ *where* $\mathscr{W} \subseteq \mathscr{V}$.

[1] Events $\underline{a} \in \mathscr{E}_d$ are indicated by underlined letters, while events $\overline{a} \in \mathscr{E}_s$ are denoted by letter with a line above them. A generic event, either stochastic or instantaneous, is indicated with $a \in \mathscr{E} = \mathscr{E}_s \cup \mathscr{E}_d$.

- *$ec : \mathscr{E} \to EC$ maps events to event conditions. Event conditions are pairs of activation conditions and resets. Resets are formulae with free variables in $\mathscr{V} \cup \mathscr{V}'$. Activation conditions for instantaneous events $\mathscr{E}_d$ are formulas with free variables in $\mathscr{V}$ and the second, while for stochastic events of $\mathscr{E}_s$, they are functions $f : \mathbb{R}^{|\mathscr{V}|} \to \mathbb{R}^+$.*
- *$iv : IN \to \mathscr{V}$ 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 $[\![I(\mathscr{W})]\!] = f(\mathscr{W})$ where the variables in $\mathscr{W}$ are from $\mathscr{V}$.*
- *$\mathscr{E}$, $\mathscr{A}$, IN and IT are pairwise disjoint.*

Definition 2 *A* controlled system *is constructed as follows.*

- Subcomponents *are defined by $C_s(\mathscr{W}) = S$, where C_s is the* subcomponent name *and S satisfies the grammar $S' ::= a : \alpha.C_s \mid S' + S'$ ($a \in \mathscr{E} = \mathscr{E}_d \cup \mathscr{E}_s$, $\alpha \in \mathscr{A}$), with the free variables of S in $\mathscr{W}$.*
- Components *are defined by $C(\mathscr{W}) = P$, where C is the* component name *and P satisfies the grammar $P' ::= C_s(\mathscr{W}) \mid C(\mathscr{W}) \mid P' \underset{L}{\bowtie} P'$, with the free variables of P in $\mathscr{W}$ and $L \subseteq \mathscr{E}$.*
- *An* uncontrolled system Σ *is defined according to the grammar $\Sigma' ::= C_s(\mathscr{W}) \mid C(\mathscr{W}) \mid \Sigma' \underset{L}{\bowtie} \Sigma'$, where $L \subseteq \mathscr{E}$ and $\mathscr{W} \subseteq \mathscr{V}$.*
- Controllers *only have events: $M ::= \underline{a}.M \mid 0 \mid M + M$ with $\underline{a} \in \mathscr{E}$ and $L \subseteq \mathscr{E}$ and $Con ::= M \mid Con \underset{L}{\bowtie} Con$.*
- *A* controlled system *is $ConSys ::= \Sigma \underset{L}{\bowtie} \underline{init}.Con$ where $L \subseteq \mathscr{E}$. The set of controlled systems is $\mathscr{C}_{Sys}$.*

Remark 1 *All HYPE models that will be considered in the paper comply with the definition of well-defined HYPE models, given in [8]. Essentially, each subcomponent must be a self-looping agent of the form $S = \sum_{i=1}^{k} a_i : \alpha_i.S + \underline{init} : \alpha.S$, with each α_i of the form (i_S, r_i, I_i), where i_S is an influence name appearing only in subcomponent S. Furthermore, synchronization must involve all shared events. In the following, we will also assume that all events appearing in the uncontrolled system appear also in the controller. If an event a is not subject to any control (apart from its guard), then we always add a controller of the form $Con = a.Con$.*

3 Transition-driven Stochastic Hybrid Automata

We now present Transition-Driven Stochastic Hybrid Automata, introduced in [2], a formalization of stochastic hybrid automata putting emphasis on transitions, which can be either discrete (corresponding to instantaneous or stochastic jumps) or continuous (representing flows acting on system's variables). This formalism can be seen as an intermediate layer in defining the stochastic hybrid semantics of HYPE. In fact, TDSHA can be mapped to Piecewise Deterministic Markov Processes [4], so that their dynamics can be formally specified in terms of the latter. Due to space constraints, we will not provide a formal treatment of this construction, and refer the reader to [2] for further details. In this context, we will also consider a different notion of TDSHA-product, in which transitions can be synchronized on their labelling events.

Definition 3 *A Transition-Driven Stochastic Hybrid Automaton (TDSHA) is a tuple $\mathscr{T} = (Q, \mathbf{X}, \mathscr{TC}, \mathscr{TD}, \mathscr{TS}, \text{init}, \mathscr{E})$, where*

- Q *is a finite set of* control modes.
- $\mathbf{X} = \{X_1, \ldots, X_n\}$ *is a set of real valued* system's variables[2].
- $\mathscr{TC}$ *is the set of* continuous transitions or flows*, whose elements* τ *are triples* $(q_\tau, \mathbf{s}_\tau, f_\tau)$*, where* $q_\tau \in Q$ *is a mode,* $\mathbf{s}_\tau$ *is a vector of size* $|\mathbf{X}|$*, and* $f_\tau : \mathbb{R}^n \to \mathbb{R}$ *is a (sufficiently smooth) function.*
- $\mathscr{TD}$ *is the set of* instantaneous transitions*, whose elements* δ *are tuples of the form* $(q_1^\delta, q_2^\delta, g_\delta, r_\delta, w_\delta, e_\delta)$*. The transition goes from mode* q_1^δ *to mode* q_2^δ *and it is labeled by* $e_\delta \in \mathscr{E}$*.* $w_\delta \in \mathbb{R}^+$ *is the weight of the edge, used to solve non-determinism among two or more active transitions. The* guard g_δ *is a first-order formula with free variables from* $\mathbf{X}$*, representing the* closed set $G_\delta = \{\mathbf{x} \in \mathbb{R}^n \mid g[\mathbf{x}]\}$*, while the* reset r_δ *is a conjunction of formulae of the form* $X' = \rho(\mathbf{X})$*, for some variables of the system. Variables not appearing in* r *are not modified, so that the formula* $true$ *corresponds to the identity reset.*
- $\mathscr{TS}$ *is the set of* stochastic transitions*, whose elements* η *are tuples of the form* $\eta = (q_1^\eta, q_2^\eta, g_\eta, r_\eta, f_\eta, e_\eta)$*, where* q_1^η*,* q_2^η*,* g_η*,* e_η*, and* r_η *are as for transitions in* $\mathscr{TD}$*, while* $f_\eta : \mathbb{R}^n \to \mathbb{R}^+$ *is the rate function giving the instantaneous probability of taking transition* η*. We require transitions labeled by the same event to have* consistent *rates: if* $e_{\eta_1} = e_{\eta_2}$*, then* $f_{\eta_1} = f_{\eta_2}$*.*
- $\mathscr{E}$ *is a finite set of event names, labelling discrete transitions.* $\mathscr{E}$ *can be partitioned into* $\mathscr{E}_d \cup \mathscr{E}_s$*, such that all events labelling instantaneous transitions belong to* $\mathscr{E}_d$*, while all events labelling stochastic transitions are from* $\mathscr{E}_s$*.*
- init *is a pair* (q^{init}, inp)*, with* $q^{init} \in Q$ *and* inp *a quantifier-free first order formula with free variables in* $\mathbf{X}$*, representing a point in* $\mathbb{R}^n$*.* init *describes the initial state of the system.*

Dynamics of TDSHA. In order to formally define the dynamical evolution of TDSHA, we can map them into a well-studied model of Stochastic Hybrid Automata, namely Piecewise Deterministic Markov Processes [4]. We just sketch now some ideas about the dynamical behaviour of TDSHA.

- Within each discrete mode $q \in Q$, the system follows the solution of a set of ODE, constructed combining the effects of the continuous transitions τ acting on mode q. The function $f_\tau(\mathbf{X})$ is multiplied by the vector $\mathbf{s}_\tau$ to determine its effect on each variable and then all such functions are added together, so that the ODEs in mode q are $\dot{\mathbf{X}} = \sum_{\tau \mid q_\tau = q} \mathbf{s}_\tau \cdot f_\tau(\mathbf{X})$.
- Two kinds of discrete jumps are possible. Stochastic transitions are fired according to their rate, similarly to standard Markovian Jump Processes. Instantaneous transitions, instead, are fired as soon as their guard becomes true. In both cases, the state of the system is reset according to the specified reset policy.[3] Choice among several active stochastic or instantaneous transitions is performed probabilistically proportionally to their rate or priority.
- A trace of the system is therefore a sequence of instantaneous and random jumps interleaved by periods of continuous evolution.

Product of TDSHA. We define now a notion of product of TDSHA which, differently from the one introduced in [2], allows also the synchronization of discrete transitions on specific events. In order to do this, we must take care of resets, requiring that synchronized transitions do not reset the same variable in different ways. Hence, we say that two transitions δ_1, δ_2 (either both discrete or both stochastic)

[2] Notation: the time derivative of X_j is denoted by $\dot{X}_j$, while the value of X_j after a change of mode is indicated by X_j'

[3] Note that the formula r defines a function from $\mathbb{R}^n$ into $\mathbb{R}^n$, which will be also denoted throughout by r.

are *reset-compatible* if and only if $e_{\delta_1} \neq e_{\delta_2}$ or $r_{\delta_1} \wedge r_{\delta_2} \neq \mathit{false}$. Two TDSHA are reset-compatible if and only if all their discrete or stochastic transitions are pairwise reset-compatible. A similar notion is required for the initial conditions: Two TDSHA are *init-compatible* if and only if, given initial conditions $\mathrm{init}_1 = (q_1^{init}, \mathrm{inp}_1)$ and $\mathrm{init}_2 = (q_2^{init}, \mathrm{inp}_2)$, then $\mathrm{inp}_1 \wedge \mathrm{inp}_2 \neq \mathit{false}$.

Definition 4 *Let $\mathscr{T}_i = (Q_i, \mathbf{X_i}, \mathscr{TC}_i, \mathscr{TD}_i, \mathscr{TS}_i, \mathrm{init}_i, \mathscr{E}_i)$, $i = 1,2$ two reset-compatible and init-compatible TDSHA, and let $S \subseteq \mathscr{E}_1 \cap \mathscr{E}_2$ be the synchronization set. The S-product $\mathscr{T} = \mathscr{T}_1 \otimes_S \mathscr{T}_2 = (Q, \mathbf{X}, \mathscr{TC}, \mathscr{TD}, \mathscr{TS}, \mathrm{init}, \mathscr{E})$ is defined by*

1. $Q = Q_1 \times Q_2$;
2. $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$;
3. $\mathscr{E} = \mathscr{E}_1 \cup \mathscr{E}_2$;
4. $\mathrm{init} = (q^{init}, \mathrm{inp})$, *where* $q^{init} = (q_1^{init}, q_2^{init})$ *and* $\mathrm{inp} = \mathrm{inp}_1 \wedge \mathrm{inp}_2$.
5. *The set of continuous transitions in a mode $q = (q_1, q_2)$ contains all continuous transitions of q_1 and all those of q_2:*

$$\mathscr{TC} = \{((q_1,q_2), \mathbf{s}, f) \mid q_1 \in Q_1, q_2 \in Q_2, (q_1, \mathbf{s}, f) \in \mathscr{TC}_1 \vee (q_2, \mathbf{s}, f) \in \mathscr{TC}_2\}$$

6. *The set of instantaneous transitions $\mathscr{TD}$ is the union of non-synchronized instantaneous transitions $\mathscr{TD}_{NS}$ and of synchronized ones $\mathscr{TD}_S$, where*

$$\begin{aligned} \mathscr{TD}_{NS} \;=\; \Big\{ &((q_1,q_2),(q_1',q_2'),g,r,w,e) \mid \\ &(q_i,q_i',g,r,w,e) \in \mathscr{TD}_i \wedge q_j = q_j' \in Q_j \wedge i \neq j \wedge e \notin S \Big\}, \end{aligned}$$

and

$$\begin{aligned} \mathscr{TD}_S \;=\; \Big\{ &((q_1,q_2),(q_1',q_2'),g_1 \wedge g_2, r_1 \wedge r_2, \min\{w_1,w_2\}, e) \mid \\ &(q_1,q_1',g_1,r_1,w_1,e) \in \mathscr{TD}_1 \wedge (q_2,q_2',g_2,r_2,w_2,e) \in \mathscr{TD}_2 \wedge e \in S \Big\}. \end{aligned}$$

During synchronization, we apply a conservative policy by taking the conjunction of guards and resets, and by taking the minimum of weights.

7. *The set of stochastic transitions is defined similarly as $\mathscr{TS} = \mathscr{TS}_{NS} \cup \mathscr{TS}_S$, with*

$$\begin{aligned} \mathscr{TS}_{NS} \;=\; \Big\{ &((q_1,q_2),(q_1',q_2'),g,r,f,e) \mid \\ &(q_i,q_i',g,r,f,e) \in \mathscr{TS}_i \wedge q_j = q_j' \in Q_j \wedge i \neq j \wedge e \notin S \Big\}, \end{aligned}$$

and

$$\begin{aligned} \mathscr{TS}_S \;=\; \Big\{ &((q_1,q_2),(q_1',q_2'),g_1 \wedge g_2, r_1 \wedge r_2, f, e) \mid \\ &(q_1,q_1',g_1,r_1,f,e) \in \mathscr{TS}_1 \wedge (q_2,q_2',g_2,r_2,f,e) \in \mathscr{TS}_2 \wedge e \in S \Big\}. \end{aligned}$$

In the synchronization of stochastic transitions, we use the fact that the rate is the same for all transitions labeled by the same event, as required by the consistency condition.

4 Mapping HYPE to TDSHA

The mapping from HYPE to TDSHA works compositionally, by associating a TDSHA with each single subcomponent and with each piece of the controller, then taking their synchronized product according to the synchronization sets of the HYPE system. Guards, rates, and resets of discrete edges will be

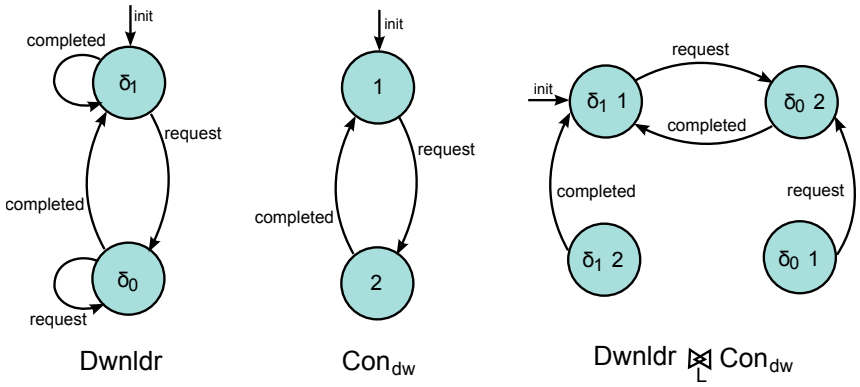

Figure 2: Schematic representation of the TDSHA $\mathscr{T}(Dwnldr)$, associated with the download module of the Orbiter, (**left**) of the TDSHA $\mathscr{T}(Con_{dw})$ associated with the download controller (**middle**) and of their TDSHA product $\mathscr{T}(Dwnldr) \otimes_L \mathscr{T}(Con_{dw})$ (**right**).

incorporated in the TDSHA of the controller, while continuous transitions will be extracted from the uncontrolled system.

Consider a HYPE model $(ConSys, \mathscr{V}, IN, IT, \mathscr{E}_d, \mathscr{E}_s, \mathscr{A}, ec, iv, EC, ID)$ with $ConSys ::= \Sigma \underset{L}{\bowtie} \underline{\text{init}}.Con$. Here Σ is the uncontrolled system and Con is the controller. In the following, we will refer to the activation condition and the reset of an event $\mathrm{a} \in \mathscr{E}$ by $act(\mathrm{a})$ and $res(\mathrm{a})$, respectively.

TDSHA of the uncontrolled system. Consider a subcomponent S, having the form $S = \sum_{i=1}^{k} \mathrm{a}_i : \alpha_i.S + \underline{\text{init}} : \alpha.S$. S is a self-looping agent which can react to events a_i modifying the state of the influence i_S, which is specific to S, see Remark 1.

First of all, we need to collect all influences and events appearing in S. The set of influences $\mathsf{is}(S)$ of a subcomponent S is defined inductively by $\mathsf{is}(\mathrm{a} : \alpha.S) = \{\alpha\}$ and $\mathsf{is}(S_1 + S_2) = \mathsf{is}(S_1) \cup \mathsf{is}(S_2)$, while the set of events $\mathsf{ev}(S)$ of S is defined by $\mathsf{ev}(\mathrm{a} : \alpha.S) = \{\mathrm{a}\}$ if $\mathrm{a} \neq \underline{\text{init}}$, $\mathsf{ev}(\mathrm{a} : \alpha.S) = \emptyset$ otherwise, and $\mathsf{ev}(S_1 + S_2) = \mathsf{ev}(S_1) \cup \mathsf{ev}(S_2)$. The set $\mathsf{is}(S)$ contains all the possible flows that can be generated by the influence with name i_S. As only one of them can be active in each state of the system, we will introduce one mode for each element of $\mathsf{is}(S)$ in the TDSHA of S. Moreover, in each such mode, the only continuous transition will be the one that can be derived from the corresponding influence. As for discrete edges, observing that the flat structure of S is such that the response to all events is always enabled, we will have an outgoing transition for each event appearing in S in each mode of the associated TDSHA. The target state of the transition will be the mode corresponding to the influence following the event. Resets and guards will be set to *true*, as event conditions will be associated with the controller. Rates of transitions derived from stochastic events $\overline{\mathrm{a}} \in \mathscr{E}_s$ will be set to $act(\overline{\mathrm{a}})$, as required by the consistency condition of TDSHA. Finally, weights will be set to 1, while the initial mode will be deduced from the init event.

Consider the subcomponent

$$Dwnldr \stackrel{def}{=} \underline{\text{init}} : \delta_1.Dwnldr + \overline{\text{request}} : \delta_0.Dwnldr + \overline{\text{completed}} : \delta_1.Dwnldr$$

describing the downloading module of the Orbiter system of Section 2. The TDSHA associated with it is visually depicted in Figure 2 (left). It has two modes, corresponding to the two different influences $\delta_1 = (dw, r, const)$ and $\delta_0 = (dw, 0, const)$, and two edges, labeled by $\overline{\text{request}}, \overline{\text{completed}}$. The initial state is the mode corresponding to δ_1.

We collect now such considerations into a formal definition.

Definition 5 *Let $S = \sum_{i=1}^{k} a_i : \alpha_i.S + \underline{init} : \alpha.S$ be a subcomponent of the HYPE model $(ConSys, \mathscr{V}, IN, IT, \mathscr{E}_d, \mathscr{E}_s, \mathscr{A}, ec, iv, EC, ID)$. The TDSHA $\mathscr{T}(S) = (Q, \mathbf{X}, \mathscr{TC}, \mathscr{TD}, \mathscr{TS}, \text{init}, \mathscr{E})$ associated with S is defined by*

1. $Q = \{q_\alpha \mid \alpha \in \mathsf{is}(S)\}$; $\mathbf{X} = \mathscr{V}$; $\mathscr{E} = \mathscr{E}_d \cup \mathscr{E}_s$;
2. $\text{init} = (q_\alpha, true)$, *where* $S = \underline{init} : \alpha.S + S'$;
3. $\mathscr{TC} = \{(q_\alpha, \mathbf{1}_{iv(i_S)}, r \cdot [\![I]\!]) \mid \alpha = (i_S, r, I) \in \mathsf{is}(S)\}$, *where $\mathbf{1}_{iv(i_S)}$ is the vector equal to 1 for the component corresponding to variable $iv(i_S)$ and zero elsewhere;*
4. $\mathscr{TD} = \{(q_{\alpha_1}, q_{\alpha_2}, 1, true, true, \underline{a}) \mid \underline{a} \in \mathsf{ev}(S) \cap \mathscr{E}_d \wedge \alpha_1 \in \mathsf{is}(S) \wedge S = \underline{a} : \alpha_2.S + S'\}$
5. $\mathscr{TS} = \{(q_{\alpha_1}, q_{\alpha_2}, true, true, act(\overline{a}), \overline{a}) \mid \overline{a} \in \mathsf{ev}(S) \cap \mathscr{E}_s \wedge \alpha_1 \in \mathsf{is}(S) \wedge S = \underline{a} : \alpha_2.S + S'\}$

Once we have the TDSHA of all subcomponents, we can build the TDSHA of the full uncontrolled system by applying the product construction of TDSHA. We capture this in the following definition.

Definition 6

1. *Let $P = P_1 \underset{L}{\bowtie} P_2$ be a component. Its TDSHA is defined recursively by $\mathscr{T}(P_1 \underset{L}{\bowtie} P_2) = \mathscr{T}(P_1) \otimes_L \mathscr{T}(P_2)$.*
2. *Let $\Sigma = \Sigma_1 \underset{L}{\bowtie} \Sigma_2$ be an uncontrolled system. Its TDSHA is defined recursively by $\mathscr{T}(\Sigma_1 \underset{L}{\bowtie} \Sigma_2) = \mathscr{T}(\Sigma_1) \otimes_L \mathscr{T}(\Sigma_2)$.*

TDSHA of the controller. Dealing with the controller is simpler, as controllers are essentially finite state automata which impose causality on the happening of events. As anticipated at the beginning of the section, event conditions will be assigned to edges of TDSHA associated with controllers. Controllers are defined by the two level syntax $M = \underline{a}.M \mid M + M$ and $Con = M \mid Con \underset{l}{\bowtie} Con$, hence sequential controllers are composed in parallel and synchronized on sets of actions. As for the uncontrolled system, we will first define the TDSHA of sequential controllers, and then combine them with the TDSHA product construction. Note that all events will be properly dealt with through this construction, as they all appear in the controller, see Remark 1.

Consider a sequential controller $M = \sum_i \mathrm{a}_i.M_i$. The derivative set of M is defined recursively by $ds(M) = \{M\} \cup \bigcup_i ds(M_i)$, where two summations coincide if they are equal up to permutation of addends.

Definition 7 *Let $(ConSys, \mathscr{V}, IN, IT, \mathscr{E}_d, \mathscr{E}_s, \mathscr{A}, ec, iv, EC, ID)$ be a HYPE model with sequential controller M. Then $\mathscr{T}(M) = (Q, \mathbf{X}, \mathscr{TC}, \mathscr{TD}, \mathscr{TS}, \text{init}, \mathscr{E})$, the TDSHA associated with M, is defined by*

1. $Q = \{q_{M'} \mid M' \in ds(M)\}$; $\mathbf{X} = \mathscr{V}$; $\mathscr{E} = \mathscr{E}_d \cup \mathscr{E}_s$;
2. $\text{init} = (q_M, res(\underline{init}))$, *where $res(\underline{init})$ is the reset associated with the $\underline{init}$ event.*
3. $\mathscr{TC} = \emptyset$;
4. $\mathscr{TD} = \{(q_{M_1}, q_{M_2}, 1, act(\underline{a}), res(\underline{a}), \underline{a}) \mid M_1 = \underline{a}.M_2,\ M_1, M_2 \in ds(M),\ \underline{a} \in \mathscr{E}_d,\ ec(\underline{a}) = (act(\underline{a}), res(\underline{a}))\}$;
5. $\mathscr{TS} = \{(q_{M_1}, q_{M_2}, true, res(\overline{a}), act(\overline{a}), \overline{a}) \mid M_1 = \overline{a}.M_2,\ M_1, M_2 \in ds(M),\ \overline{a} \in \mathscr{E}_s,\ ec(\overline{a}) = (act(\overline{a}), res(\overline{a}))\}$, *where $act(\overline{a}) : \mathbb{R}^{|\mathscr{V}|} \to \mathbb{R}^+$ is the rate of the transition;*

Definition 8 *Let $Con = Con_1 \underset{L}{\bowtie} Con_2$ be a controller. The TDSHA of Con is defined recursively as $\mathscr{T}(Con) = \mathscr{T}(Con_1) \otimes_L \mathscr{T}(Con_2)$.*

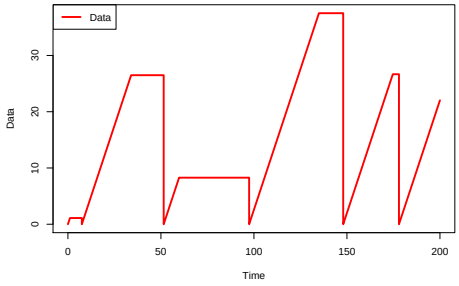

Figure 3: Sampled trajectory of the accumulated data of the extended orbiter model of Section 2.1. Data increases during accumulation phases, and remains constant during downloads. It is erased right after the download finished. Rate values, fixed just for illustrative purposes, are $r = 1.0$, $\lambda_r = 0.04$, $\lambda = 0.5$, $\mu = 10.0$.

The product construction of Definitions 6 and 8 can be carried on because the factors TDSHA are reset-compatible and init-compatible. This is trivial both for the uncontrolled system (all resets are *true*) and for the controller (resets for the same event are equal). Furthermore, stochastic transitions have consistent rates, as their rate depends only on the labelling event.

Consider the controller of the download module of the orbiter; its TDSHA is depicted in Figure 2 (middle), omitting the explicit representation of rates and resets.

TDSHA of the HYPE model. Once we have built the TDSHA of the controller and of the uncontrolled system, we simply have to take their product.

Definition 9 *Let $(ConSys, \mathscr{V}, IN, IT, \mathscr{E}_c, \mathscr{E}_s, \mathscr{A}, ec, iv, EC, ID)$ be a HYPE model, with controlled system $ConSys = \Sigma \underset{L}{\bowtie} \underline{init}.Con$. The TDSHA associated with $\mathscr{M}$ is*

$$\mathscr{T}(\mathscr{M}) = \mathscr{T}(\Sigma) \otimes_L \mathscr{T}(Con).$$

Example 1 *In Figure 2 (right) we show the product $\mathscr{T}(Dwnldr) \otimes_L \mathscr{T}(Con_{dw})$, $L = \{\overline{request}, \overline{completed}\}$, in order to give an idea of the product construction. In Figure 3, instead, we show a trajectory of the variable D, describing the amount of data collected. As we can see, periods in which the data is collected (linearly), are interleaved by downloads, in which data is not accumulated. Once the download has finished, D is set back to zero. Both the download time and the periods between two consecutive downloads are randomly distributed.*

As already evident from the previous example, the construction we have defined actually generates TDSHA with many *unreachable states*. This is a consequence of the fact that sequentiality and causality on actions is imposed just on the final step, when the controller is synchronized with the uncontrolled system. Once the TDSHA is constructed, however, it can be pruned by removing unreachable states (the TDSHA of Figure 2 (right) has indeed just two reachable states from the initial one). In order to limit combinatorial explosion, one can prune TDSHA's at each intermediate stage. A formal definition of this policy, however, would have made the mapping from HYPE to TDSHA much more complex.

Orbiter revisited. We consider now a more complex version of the orbiter, in which the download time depends also on the current temperature. The operational speed of the download can be reduced linearly down to zero if the temperature is too high or too low. In order to implement such a modification,

we simply have to modify the rate function in the event condition of event $\overline{\text{completed}}$, replacing it with a suitable function of accumulated data and temperature. A sampled trajectory is shown in Figure 4 (left), while in Figure 4 (right) we show how the firing time of $\overline{\text{completed}}$ depends on temperature.

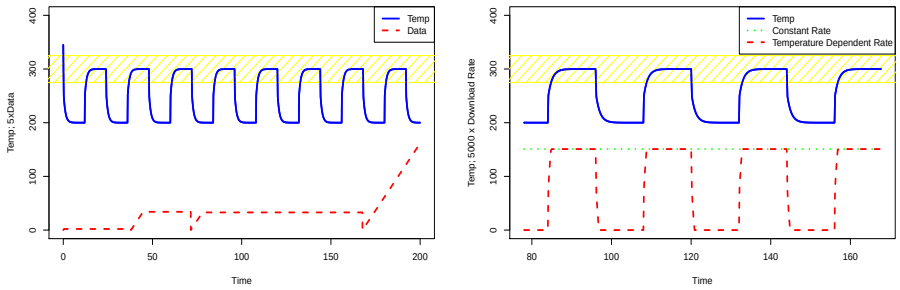

Figure 4: (**left**) Sampled trajectory of the accumulated data of the extended orbiter model of Section 2.1, with download rate depending on temperature. The rate is maximal, and equal to $\frac{\lambda}{\mu+D}$ when temperature is in the operational regime, in this example, when $275 \leq K \leq 325$. When the temperature is lower than 275 (higher than 325), the download rate linearly decreases to 0, reaching it when $K = 225$ ($K = 350$). Rate values, for the downloader are $r = 1.0$, $\lambda_r = 0.1$, $\lambda = 1.0$, $\mu = 10.0$. Rates and parameters for the temperature control mechanism are $f_h = 200$, $r_d = 100$, $r_s = 400$, $k_1 = k_2 = 250$, $k_3 = k_4 = 300$. The download time becomes longer with respect to Figure 3, as the temperature falls repeatedly below the operational regime. (**right**) Plot of the downloading rate when it is constant (green) or when it depends on temperature (red). In the latter case, the rate is periodically reduced to zero, as temperature falls below 225. The shadowed region indicates, in both plots, the interval of temperatures in which the download has maximum speed.

5 Related Work

The modelling approach of HYPE, based on the composition of individual flows, makes it different from other hybrid process algebras [12] and from hybrid automata [9]. In these other approaches the continuous dynamics is specified by embedding ODEs within the syntactic description of models, while in HYPE, ODEs emerge as a combination of active flows. A more detailed comparison between HYPE and other hybrid modelling formalisms can be found in [7, 8].

As far as stochastic hybrid systems are concerned, there has been previous work aimed at making modelling compositional. In [13], Strubbe *et al.* introduce Communicating Piecewise Deterministic Markov Processes (CPDP). This is an automata based formalism which models a system as interacting automata. Their chosen level of abstraction is somewhat lower level than ours, comparable with TDSHA. In CPDP, as in HYPE, instantaneous transitions may be triggered either by conditions of the continuous variables (boundary-hit transitions) or by the expiration of a stochastic determined delay (Markov transitions). Interaction between automata is based on *one-way* synchronisation: in each interaction one partner is active while the other is passive. In HYPE, instead, all components may be regarded as active with respect to each transition in which they participate, as activation conditions are specified uniquely in the model. Components participating in a discrete transition are determined by the construction of the HYPE model, where the synchronisation set L in $\underset{L}{\bowtie}$ specifies which actions must be shared.

The synchronization mechanics of CPDP has been extended in [14], introducing an operator which

exploits all possible interactions of active and passive actions. In [15] the authors define a notion of bisimulation for both PDMPs and CPDPs and show that if CPDPs are bisimilar then they give rise to bisimilar PDMPs. Furthermore the equivalence relation is a congruence with respect to the composition operator of CPDPs.

6 Conclusions

In this paper we extended the hybrid process algebra HYPE, allowing events to fire at (exponentially distributed) random times. Although from a syntactic point of view the modifications with respect to the original version of HYPE are minimal (non-urgent events become stochastic by replacing their activation condition $\perp$ with a functional rate), the semantics of the language is considerably enriched. The stochastic hybrid systems obtained from HYPE models fall in the class of Piecewise Deterministic Markov Processes. In the paper, we concentrated on showing how such a semantics can be defined. We used an intermediate formalism, namely Transition-Driven Stochastic Hybrid Automata, which can then be mapped to PDMPs. The way we defined the semantics in terms of TDSHA is quite different from the original definition of [7, 8], in which a hybrid automaton is extracted from the labeled transition system of a HYPE model, defined according to a suitable operational semantics. Here, instead, we directly manipulate the model at the syntactic level.

The mapping from TDSHA to PDMP is quite straightforward, except in one point: One has to check that the HYPE model is *well-behaved*, meaning that it is not possible that an infinite sequence of instantaneous transitions fires in the same time instant. Unfortunately, checking this property in general is undecidable, hence in [8] we put forward a set of decidable but stricter conditions on HYPE models, that guarantee that a model is well-behaved and that are usually satisfied in practical cases.

As the syntax of HYPE is basically unchanged, all the results of [7, 8] depending on syntactic features still hold. In particular, the notion of bisimulation of HYPE models extends untouched in this new setting. As a future investigation, we plan to compare this bisimulation relation with other bisimulations designed for PDMPs [15].

In the current version of HYPE, stochasticity has been introduced just in terms of random occurrence in the time of events. It is often useful to have stochasticity also in resets. This would allow the quantitative modelling of uncertainty in the outcome of certain actions. Such an extension can be done along the lines of the current paper, even if it requires a modification of the definition of TDSHA, allowing stochastic resets. However, the class of target stochastic processes remains that of PDMP.

Future work includes also the implementation of an efficient simulator for (stochastic) HYPE. Moreover, we will model specific case studies, to prove its effectiveness as a hybrid modelling language.

Acknowledgements This work was supported by Royal Society International Joint Project (JP090562). Vashti Galpin is supported by the EPSRC SIGNAL Project, Grant EP/E031439/1. Luca Bortolussi is supported by GNCS. Jane Hillston has been supported by EPSRC under ARF EP/c543696/01.

References

[1] L. Bortolussi, V. Galpin, J. Hillston, and M. Tribastone. Hybrid Semantics for PEPA. In *Proc. of QEST 2010*, pp. 181–190.

[2] L. Bortolussi and A. Policriti. Hybrid Semantics of Stochastic Programs with Dynamic Reconfiguration. In *Proc. of CompMod 2009*.

[3] F. Ciocchetta and J. Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410:3065-3084, 2009.

[4] M.H.A. Davis. *Markov Models and Optimization*. Chapman & Hall, 1993.

[5] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.

[6] 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.

[7] V. Galpin, J. Hillston, and L. Bortolussi. HYPE: a process algebra for compositional flows and emergent behaviour. In: *Proc. of CONCUR 2009, Lecture Notes in Computer Science*, LNCS 5710, pp. 305–320.

[8] V. Galpin, J. Hillston, and L. Bortolussi. HYPE: hybrid modelling by composition of flows. *Journal version, submitted for publication*.

[9] T. A. Henzinger. The theory of hybrid automata. In *Proc. of LICS 1996*, pages 278–292.

[10] J. Hillston. Fluid flow approximation of PEPA models. In *Proc. of QEST 2005*, pp 33–43.

[11] J. Hillston. *A Compositional Approach To Performance Modelling*. CUP, 1996.

[12] U. Khadim. A comparative study of process algebras for hybrid systems. Computer Science Report 06-23, Technische Universiteit Eindhoven, 2006. `http://alexandria.tue.nl/extra1/wskrap/publichtml/200623.pdf`.

[13] S.N. Strubbe, A.A. Julius and A.J. van der Schaft. Communicating Piecewise Deterministic Markov Processes. In *Proc. of ADHS 2003*, 349-354.

[14] S.N. Strubbe and A.J. van der Schaft. Stochastic semantics for Communicating Piecewise Deterministic Markov Processes. In *Proc. of IEEE CDC 2005*, pp. 6103–6108.

[15] S.N. Strubbe and A.J. van der Schaft. Bisimulation for Communicating Piecewise Deterministic Markov Processes. In *Proc. of HSCC 2005*, LNCS 3414.

[16] M. Tribastone, S. Gilmore and J. Hillston, Scalable differential analysis of process algebra models. *IEEE Transactions on Software Engineering*, 2010, to appear.

[17] 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.