HYPE applied to the modelling of hybrid biological systems

Vashti Galpin[1,4]

LFCS, School of Informatics
University of Edinburgh, Scotland

Jane Hillston[2,5,6]

CSBE and School of Informatics
University of Edinburgh, Scotland

Luca Bortolussi[3,7]

Department of Maths and Computer Science, University of Trieste
Center for Biomolecular Medicine, Area Science Park, Trieste, Italy

Abstract

HYPE is a process algebra developed to model hybrid systems – systems that show both continuous and discrete behaviour. It is novel because it allows for the modelling of individual flows which means that subcomponents can be modelled in terms of these flows and do not need to be described monolithically. Biological systems display discrete behaviour inherently, but modellers may choose to model systems in a hybrid fashion, often to deal with differences in scale. We demonstrate how HYPE can be used to model the Repressilator.

Keywords: systems biology, hybrid systems, process algebra

1 Introduction

Systems biology has provided a productive field of endeavour for theoretical computer science research, both in the area of algorithms and modelling. Stochastic process algebras such as PEPA [11,5,**?**] and the stochastic π-calculus [13,15] have

[1] Email: Vashti.Galpin@ed.ac.uk

[2] Email: Jane.Hillston@ed.ac.uk

[3] Email: luca@dmi.units.it

[4] Supported by the EPSRC SIGNAL Project, Grant EP/E031439/1

[5] The Centre for Systems Biology at Edinburgh is a Centre for Integrative Systems Biology (CISB) funded by the BBSRC and EPSRC in 2006

[6] Supported by the EPSRC under ARF EP/c543696/01

[7] Supported by PRIN "Sistemi e calcoli di ispirazione biologica e loro applicazioni" and FIRB LIBi

been applied successfully to the modelling of biological systems resulting in both stochastic models and deterministic continuous models based on ODEs, and many other process algebra approaches have been developed [3,7,14,16]. One advantage of a process algebra approach is a compositional language which allows for the construction of models of systems by the composition of models of smaller components, which avoids the need to start from a monolithic view of the system. Another advantage is the provision of various ways in which to analyse the system once it has been constructed.

Biological systems are inherently discrete and stochastic in behaviour. Due to the difficulty of modelling systems in this manner, continuous deterministic approximations are frequently used. A third approach is to model these systems as having hybrid behaviour. Hybrid behaviour contains both discrete elements and continuous elements. In the case of gene activation, a single gene is either on or off and hence it is difficult to model this as a continuous value that varies between zero and one. Meanwhile, in the same system other species may exist in such abundance as to make discrete modelling impracticable. Therefore, taking a hybrid approach allows the gene status to be modelled as a discrete component of the system and for other species to be modelled continuously.

Other aspects of a system can be modelled as discrete. For example, a system where continuous behaviour can be viewed as changing after a threshold concentration of a specific reagent has been reached, can be modelled as having two distinct modes of operation, one where the concentration is below the threshold and one where it is over the threshold. For each mode, appropriate ordinary differential equations (ODEs) describing the continuous behaviour can be defined. In the case that one reagent has a significantly lower concentration than the other reagents then the results of an ODE model may not be reliable. Taking a hybrid approach, this reagent can be modelled in a discrete fashion (in terms of molecules or in terms of *present* versus *absent*). Behaviour described by sigmoidal curves is common in biology and this can simulated by a hybrid system with modes that give piecewise constant and piecewise linear continuous behaviour [12].

HYPE is a process algebra for modelling hybrid systems. It is novel because unlike other process algebras for hybrid systems [2,6,17,19] it does not require a monolithic view of subsystems. This means that the continuous behaviour of a subsystem does not need to be understood in advance of the modelling process. HYPE achieves this by allowing the modelling of flows, which can be seen as influences on continuous variables. In systems biology, this involves identifying the different biological processes that lead to changes in the concentration of a species. For example, the following are usually considered: production, degradation, transformation and transportation. Using HYPE these can be modelled as separate subcomponents which are then composed to obtain the whole system.

In HYPE, a transition system of a model is determined by the operational semantics of the process algebra terms. In this transition system, each node is a pair or configuration consisting of a process term and a state. A state maps influence names to influence strengths and influence types (unlike the typical notion of state which maps variables names to values) and records what influences are in effect in that state. A configuration identifies a mode, and the state of that configura-

tion is used to obtain the ordinary differential equations that apply in that mode. There are three levels of semantics: the operational semantics which describe the behaviour of the system in terms of discrete events where information about continuous behaviour is recorded in the configurations, the hybrid semantics which uses the recorded behaviour to obtain the ODEs, and finally equivalence semantics which describe notions of similar behaviour. A bisimulation equivalence has been defined and it has been shown that bisimilar systems have the same ODEs [9]. It is also possible to map the transition system and ODEs to a hybrid automata. Hence, this process algebra provides a useful modelling front-end to an existing formalism with numerous tools. HYPE is also important because it could be the basis of a first-pass modelling tool where different flows could be described directly by biologists. Currently modelling is a bottleneck with respect to the amounts of data now available and a formalism that can speed up the process of modelling will be advantageous.

The usefulness of this process algebra for hybrid modelling will be shown through an example. The Repressilator is a well-known synthetic genetic regulatory network [8]. It consists of three genes *tetR*, *cI* and *lacI*. Each gene expresses a protein that affects the expression of one of the other genes. The protein from *tetR* represses the expression of *cI*, the protein from *cI* represses the expression of *lacI* and the protein from *lacI* represses the expression of *tetR*. This forms a negative feedback loop and demonstrates oscillatory behaviour. We will express this network in HYPE and show various analyses.

The next section introduces the process algebra HYPE together with its operational and hybrid semantics. After that the Repressilator is modelled in HYPE and its analysis is given. Finally, we discuss related work in hybrid biological systems modelling and we finish with a discussion of topics for future work.

2 HYPE definition

This section will present HYPE by way of a simple running example of a gene and the protein it produces to illustrate the necessary concepts. This gene has a negative feedback cycle. Once a certain amount of protein is produced, the gene switches off, the protein production stops and the protein concentration drops as a result of the stopping of production. Once it has dropped sufficiently, the gene switches back on and protein productions resumes until it reaches the threshold.

The modelling spirit of HYPE focusses on flows. In our example, we can identify two flows affecting the concentration of a protein. There is one for the production of the protein and one for the degradation of the protein. The strength and form of a flow can be modified by events. So the production activity depends on whether the gene is switched on or off. In HYPE, we model the production of Protein A by the following subcomponent.

$$P_A^{prod} \stackrel{def}{=} \underline{\text{init}} : (p_A, k_p, const).P_A^{prod} + \underline{\text{express}}_A : (p_A, k_p, const).P_A^{prod} + \underline{\text{inhibit}}_A : (p_A, 0, const).P_A^{prod}$$

This is a *summation* of prefixed actions. 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. For Protein A, the events are $\underline{\text{init}}$, the initialisation event; $\underline{\text{express}}_A$, the switching on of the expression of Gene A and $\underline{\text{on}}_1$, the switching off of the expression of Gene A.

Activities ($\alpha \in \mathcal{A}$) are *influences* on the evolution of the continuous part of the system and are used to 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* $I(\overrightarrow{X})$ which describes how that rate is to be applied to the variables involved, or the actual form of the flow [1]. In P_A^{prod}, there are two distinct activities, $(p_A, k_p, const)$ and $(p_A, 0, const)$. The first one captures the effect of the gene being switched on and protein being produced. It affects influence p_A which represents the influence of protein production on the concentration of the protein, it has strength k_p and it is associated with the function called *const*. The second gives the effect of the gene being switched off and no protein being produced: the influence name is again p_A, 0 is the strength of the protein production since none is being produced, and the form it takes is *const*. We assume that in the initial state, protein is being produced, hence the activity after $\underline{\text{init}}$ is the same as that after $\underline{\text{express}}_A$.

Notice that no actual function has been specified for *const*. The interpretation of influence types will be specified separately, so that experimentation with different functional forms of the protein flow can occur without modifying the protein subcomponent. Hence, in HYPE we separate the description of the logical structure of flows from their mathematical interpretation.

We also require a subcomponent to represent the degradation of the protein.

$$P_A^{degr}(X) \stackrel{def}{=} \underline{\text{init}} : (d_A, -k_d, linear(X)).P_A^{degr}(X)$$

The only event in this definition is the initialisation event, as degradation is an ongoing process. The influence is named d_A and its strength is $-k_d$. The influence type $linear(X)$ will be interpreted as a linear function of its formal variable X. This subcomponent must be parameterised by a variable since degradation is affected by how much of the degrading substance is available.

These two subcomponents are composed and cooperate on their only shared event $\underline{\text{init}}$. The formal variable X can be instantiated with the actual variable A to give the overall model of a protein.

$$Pr_A(A) \stackrel{def}{=} (P_A^{degr}(A) \underset{\{\underline{\text{init}}\}}{\bowtie} P_A^{prod})$$

Here $\underset{L}{\bowtie}$ represents parallel synchronisation. L is the set of events for which synchronisation must occur. Events not in L can occur independently. The way in which HYPE subcomponents are defined means that they are responsive to any event. However, we may wish to order the events in our models. The component Con_A does this by requiring that $\underline{\text{express}}_A$ and $\underline{\text{inhibit}}_A$ events alternate.

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

$$Con_A \stackrel{def}{=} \underline{\text{inhibit}}_A.\underline{\text{express}}_A.Con_A$$

In the modelling of computer systems, we call this a controller and we will use this terminology here. The system without a controller is called the *uncontrolled system* because all events are possible and no causal or temporal constraints have been imposed. Controllers only have event prefixes. 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 protein , the full system is described by

$$P \stackrel{def}{=} Pr_A \underset{M}{\bowtie} \underline{\text{init}}.Con_A \qquad \text{with} \qquad M = \{\underline{\text{init}}, \underline{\text{express}}_A, \underline{\text{inhibit}}_A\}.$$

This has defined the structure of our protein but we require some additional definitions to capture further details. We need to link each influence with a specific actual variable. This is done using the function iv and for the example, $iv(p_A) = A$ and $iv(d_A) = A$. Note that an influence can only by associated with one variable, in agreement with the interpretation of influences as flows. We also need to define the two influence types and we choose $[\![const]\!] = 1$ and $[\![linear(X)]\!] = X$. Finally, we need to capture what triggers an event, and what happens to variables immediately after an event. This is done by defining appropriate event conditions via 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. 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 an special event condition $\bot$ which means that the event can happen at some point in the future [1] . For the example, the function ec and associated event conditions are

$$\begin{aligned} ec(\underline{\text{init}}) &= (true, (A = A_0)) \\ ec(\underline{\text{express}}_A) &= ((A = 5), true) \\ ec(\underline{\text{inhibit}}_A) &= ((A = 20, true) \end{aligned}$$

The $\underline{\text{init}}$ event has an associated event condition of $true$ and so this must happen immediately. The event $\underline{\text{express}}_A$ must happen when the concentration of A drops to 5, and $\underline{\text{inhibit}}_A$ must happen as soon as the concentration reaches 20. The event $\underline{\text{init}}$ has a reset that defines the new value of A as the initial value A_0. Both $\underline{\text{express}}_A$ and $\underline{\text{inhibit}}_A$ have 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

[1] We have not done so here but probabilistic resets can be used. Tuffin *et al* [18] use a value drawn from a exponential distribution for the time until the next event.

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 2.1 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 $ConSys ::= \Sigma \underset{L}{\bowtie} \underline{\text{init}}.Con$ where $L \subseteq \mathcal{E}$. The set of controlled systems is $\mathcal{C}_{Sys}$.

A controlled system together with the appropriate sets and functions give a HYPE model.

Definition 2.2 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 types.
- $\mathcal{E}$ is a set of events of the form $\underline{a}$ and $\underline{a}_i$.
- $\mathcal{A}$ is a set of activities of the form $\alpha(\overrightarrow{X}) = (\iota, r, I(\overrightarrow{X})) \in (IN \times \mathbb{R} \times IT)$.
- $ec : \mathcal{E} \to EC$ maps events to event conditions.
- $iv : IN \to \mathcal{V}$ 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 $[\![I(\overrightarrow{X})]\!] = f(\overrightarrow{X})$ where the variables in $\overrightarrow{X}$ are from $\mathcal{X}$.
- $\mathcal{E}$, $\mathcal{A}$, IN and IT are pairwise disjoint.

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.

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

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

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

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

$$\frac{\left\langle F, \sigma \right\rangle \xrightarrow{\underline{a}} \left\langle F', \sigma' \right\rangle}{\left\langle E \underset{M}{\bowtie} F, \sigma \right\rangle \xrightarrow{\underline{a}} \left\langle E \underset{M}{\bowtie} F', \sigma' \right\rangle} (\underline{a} \notin M)$$

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

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

Table 1
Operational semantics for HYPE

2.1 Operational semantics

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

Definition 2.3 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 $\left\langle ConSys, \sigma \right\rangle$ and the set of configurations is $\mathcal{F}$.

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

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

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

When synchronisation occurs, two states must be merged and the function uses the previous state and the new states to determine which values have changed and then puts these changed values into the new state. Γ will be undefined if both the second and third argument differ from the first argument, namely if the values in the new state both differ from the old state since this represents conflicting updates. The following definition will be useful.

Definition 2.4 The *derivative set* of a controlled system P, $\mathrm{ds}(P)$ is defined as the smallest set satisfying

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

In the protein model, there are two states. Each state captures the influences that are currently active. State σ_0 reflects that the protein is being produced and degrading and σ_1 reflects that the protein is degrading only.

$$\sigma_0 = \{d_A \mapsto (-k_d, \mathit{linear}(A)),\ p_A \mapsto (k_p, \mathit{const})\}$$
$$\sigma_1 = \{d_A \mapsto (-k_d, \mathit{linear}(A)),\ p_A \mapsto (0, \mathit{const})\}$$

2.2 Hybrid semantics

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

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

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

So for each state, we have a collection of ODEs, one for each variable V. When protein is being produced, we obtain the following ODE from σ_0.

$$P_{\sigma_0} = \left\{ \frac{dA}{dt} = -k_d A + k_p \right\}$$

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, thus allowing our modelling language to be a front-end to a well-known formalism. 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 [10] for further details.

Definition 2.5 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 $\mathit{inv}(v)$ are two formulae on $\mathbf{X}$ specifying the *admissible initial conditions* and some *invariant conditions* that must be true during the continuous evolution of variables in v.
- Edges $e \in E$ of the control graph are labelled by an event $\mathit{event}(e) \in \mathcal{E}$ and by $\mathit{jump}(e)$, a predicate on $\mathbf{X}$ stating for which values of variables each transition is active, and by $\mathit{reset}(e)$, a predicate on $\mathbf{X} \cup \mathbf{X}'$ specifying the change of the variables' values after the transition has taken place. Moreover, each edge $e \in E$ can be declared urgent, by setting to true the boolean flag $\mathit{urgent}(e)$, meaning that the transition is taken at once when $\mathit{jump}(e)$ becomes true. Otherwise, the transition can be taken nondeterministically whenever $\mathit{jump}(e)$ is true.

Consider a HYPE model $(P_0, \mathcal{V}, \mathcal{X}, \mathit{IN}, \mathit{IT}, \mathcal{E}, \mathcal{A}, \mathit{ec}, \mathit{iv}, \mathit{EC}, \mathit{ID})$ 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 $\mathit{act}_{\underline{\mathrm{a}}}$ and $\mathit{res}_{\underline{\mathrm{a}}}$ the activation conditions and the resets associated with an event $\underline{\mathrm{a}} \in \mathcal{E}$, so $\mathit{ec}(\underline{\mathrm{a}}) = (\mathit{act}_{\underline{\mathrm{a}}}, \mathit{res}_{\underline{\mathrm{a}}})$.

Definition 2.6 The hybrid automaton $\mathcal{H} = (V, E, \mathbf{X}, \mathcal{E}, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$ can be obtained from the HYPE model $(P_0, \mathcal{V}, \mathcal{X}, \mathit{IN}, \mathit{IT}, \mathcal{E}, \mathcal{A}, \mathit{ec}, \mathit{iv}, \mathit{EC}, \mathit{ID})$ 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 $\langle P_1, \sigma_1 \rangle \xrightarrow{\underline{\mathrm{a}}} \langle P_2, \sigma_2 \rangle$ is a derivation for some $\underline{\mathrm{a}}$.
- $\mathbf{X} = \mathcal{V}$ is the set of variables of the HYPE system.
- $\mathcal{E}$ is the set of events $\mathcal{E}$ of P_0.
- Let $v_j = \langle P_j, \sigma_j \rangle$, then
$$\mathit{flow}(v_j)[X_i] = \sum\{r[\![I(\overrightarrow{W})]\!] \mid \mathit{iv}(\iota) = X_i \text{ and } \sigma_j(\iota) = (r, I(\overrightarrow{W}))\}$$
- $\mathit{init}(v) = \begin{cases} \mathit{res}_{\underline{\text{init}}}, \text{ if } v = \langle P, \sigma \rangle \\ \mathit{false}, \ \mathit{otherwise} \end{cases}$ with primes removed from variables [1].

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

- $inv(v) = true$.
- 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 $urgent(e) = false$.

Additionally, bisimulation has defined over HYPE models and congruence has been shown [9]. Under conditions of well-definedness, it can be shown that bisimilar HYPE models P and Q have identical ODEs [9].

3 The Repressilator

As mentioned above, the Repressilator is a synthetic genetic regulatory network [8] with a negative feedback loop and oscillatory changes in protein concentrations. It consists of three genes *tetR*, *cI* and *lacI*, and these genes code for proteins that can then affect the expression of another gene negatively. We will model the network in an abstract fashion and consider three genes, namely Gene A, Gene B and Gene C. Figure 1 illustrates the network. For Gene A, there is a mRNA transcription reaction with rate trs_A which creates $mRNA_A$ and this mRNA is then translated into the protein Pr_A with rate trl_A. Both the mRNA and the protein can degrade with rates dm_A and dp_A respectively. Genes B and C have similar reactions. The dashed lines indicate the inhibition of mRNA transcription by the relevant protein.

In modelling this network with HYPE, we will apply further abstraction as described in Figure 2. We abstract away from the details of transcription and translation and replace this by a single abstract reaction that represents the production of protein from the gene. Furthermore, we will assume that there are no differences between the different reaction rates for the different genes and use k_p and k_d as the reaction rates for production and degradation respectively. The dashed lines now represent the inhibition of the production of protein by another protein.

The HYPE model of a protein has been presented in Section 2 and is restated in a more general form in Figure 3.

The HYPE model of the Repressilator in Figure 4 consists of three proteins and their associated event-ordering components. Each protein has been instantiated with a variable representing the concentration of the protein. For clarity, each protein has a subscript indicating the protein whose production it inhibits. So the protein produced by Gene A is A_B which shows that this protein inhibits the production by Gene B of protein B_C, and similarly for the others. There is no cooperation on events apart from on the $\underline{\text{init}}$ event between proteins, and the event-ordering components for each proteins have no shared events. However, the proteins and the event-ordering components must cooperate over all events in the model.

We also need to define the other elements of the tuple for the HYPE model. The variables that measure the concentration of the three proteins make up the set $\mathcal{V}$, and the two influence types are defined to be the obvious functions: $[\![const]\!]$ is 1 and $[\![linear(X)]\!]$ is X as before. It is also necessary to associate the influence names with variables and this is done using the function iv – d_A and p_A are related to the production of the protein A_B, and likewise for the other proteins. Finally, we need to give conditions for the events. For $\underline{\text{init}}$, it must happen immediately

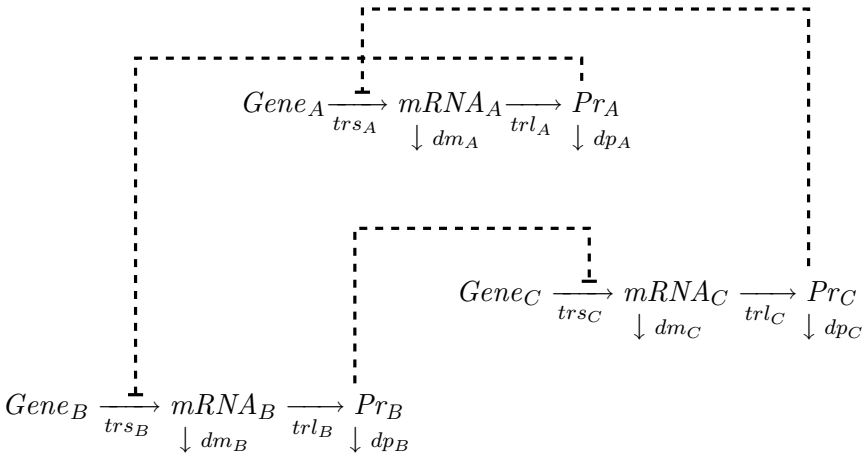

Fig. 1. The detailed Repressilator network with abstract names

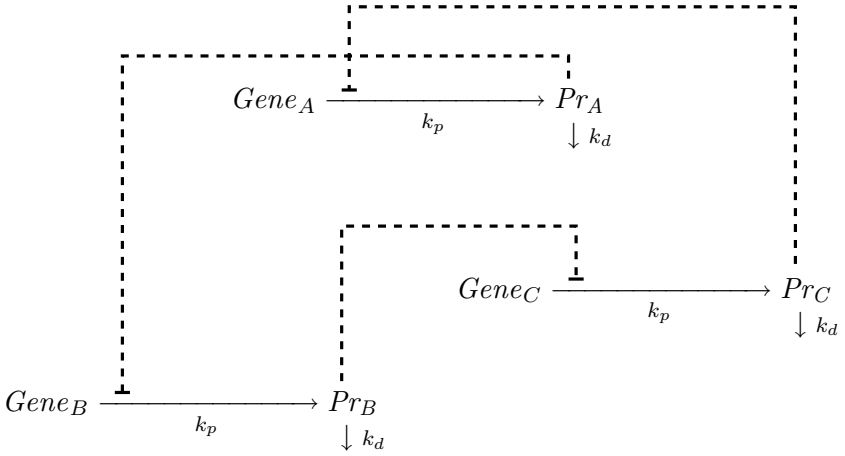

Fig. 2. The abstract Repressilator network

the system starts, hence the value *true*. The reason for having an event init is to initialise the system, hence it sets the initial values for the proteins. For protein A_B, the event inhibit_A occurs when the concentration of the protein C_A that inhibits its production reaches a certain concentration. Likewise the event express_A occurs when the concentration of the protein C_A is sufficiently low. The remaining events are defined in a similar way. No resets of variable values occur for any of these events, except init.

Once the model is defined, the operational semantics can be used to obtain the labelled transition system of the model. Figure 5 shows part of a transition derivation when init occurs at the start of the system. The symbol $*$ is used to

$$P_x^{degr}(X) \stackrel{def}{=} \underline{\text{init}} : (d_x, -k_d, linear(X)).P_x^{degr}(X)$$

$$\begin{aligned} P_x^{prod} \stackrel{def}{=} & \ \underline{\text{init}} : (p_x, k_p, const).P_x^{prod} \\ & + \underline{\text{inhibit}}_x : (p_x, 0, const).P_x^{prod} \\ & + \underline{\text{express}}_x : (p_x, k_p, const).P_x^{prod} \end{aligned}$$

$$Pr_x(X) \stackrel{def}{=} (P_x^{degr}(X) \underset{\{\underline{\text{init}}\}}{\bowtie} P_x^{prod})$$

$$Con_x \stackrel{def}{=} \underline{\text{inhibit}}_x.\underline{\text{express}}_x.Con_x$$

Fig. 3. Proteins in HYPE for $x \in \{A, B, C\}$

$$\begin{aligned} Rep \stackrel{def}{=} & (Pr_A(A_B) \underset{\underline{\text{init}}}{\bowtie} Pr_B(B_C) \underset{\underline{\text{init}}}{\bowtie} Pr_C(C_A)) \underset{L}{\bowtie} \underline{\text{init}}.(Con_A \parallel Con_B \parallel Con_C) \\ L = & \{\underline{\text{init}}, \underline{\text{inhibit}}_A, \underline{\text{inhibit}}_B, \underline{\text{inhibit}}_C, \underline{\text{express}}_A, \underline{\text{express}}_B, \underline{\text{express}}_C\} \end{aligned}$$

$$\mathcal{V} = \{A_B, B_C, C_A\}$$
$$[\![const]\!] = 1 \qquad [\![linear(X)]\!] = X$$

$$\begin{aligned} iv(d_A) &= A_B & iv(p_A) &= A_B \\ iv(d_B) &= B_C & iv(p_B) &= B_C \\ iv(d_C) &= C_A & iv(p_C) &= C_A \end{aligned}$$

$$\begin{aligned} ec(\underline{\text{init}}) &= (true, (A_B = A_0 \wedge B_C = B_0 \wedge C_A = C_0)) \\ ec(\underline{\text{inhibit}}_A) &= (C_A > p, T) & ec(\underline{\text{express}}_A) &= (C_A \leq p, T) \\ ec(\underline{\text{inhibit}}_B) &= (A_B > p, T) & ec(\underline{\text{express}}_B) &= (A_B \leq p, T) \\ ec(\underline{\text{inhibit}}_C) &= (B_C > p, T) & ec(\underline{\text{express}}_C) &= (B_C \leq p, T) \end{aligned}$$

Fig. 4. The Repressilator in HYPE

show an undefined value. This example shows how the function Γ is used to select the new value of an influence. In the case of d_A, $\tau(d_A)$ is undefined as is $\tau_2(d_A)$ hence Γ selects the value of $\tau_1(d_A)$ to use in the state that results after the transition.

There are eight configurations in the labelled transition system, each with a unique state and these states are give in Figure 6. Each gene can be *on* (producing protein) or *off* (not producing protein). The eight states represent all possible combinations of the three genes being on or off. These states then give rise to the ODEs that are given in Figure 7. In these equations, the presence of the term k_p represents the gene being on, and its absence the gene being off. Again, these sets of equations cover all possible combinations of the three genes being on or off. How do these ODEs relate to the ones proposed in the original deterministic model [8]? The

$$\frac{\dfrac{\langle P_A^{degr}(A_B),\tau\rangle \xrightarrow{\text{init}} \langle P_A^{degr}(A_B),\tau_1\rangle \quad \langle P_A^{prod},\tau\rangle \xrightarrow{\text{init}} \langle P_A^{prod},\tau_2\rangle}{\langle P_A^{degr}(A_B) \underset{\text{init}}{\bowtie} P_A^{prod},\tau\rangle \xrightarrow{\text{init}} \langle P_A^{degr}(A_B) \underset{\text{init}}{\bowtie} P_A^{prod},\tau_3\rangle}}{\langle Pr_A(A_B),\tau\rangle \xrightarrow{\text{init}} \langle Pr_A(A_B),\tau_3\rangle}$$

$$\begin{aligned}
\tau &= \{d_A \mapsto *,\ p_A \mapsto *\} \cup S \\
\tau_1 &= \tau[d_A \mapsto (-1, l(A_B))] = \{d_A \mapsto (-1, l(A_B)),\ p_A \mapsto *\} \cup S \\
\tau_2 &= \tau[p_A \mapsto (k_p, c)] = \{d_A \mapsto *,\ p_A \mapsto (k_p, c)\} \cup S \\
\tau_3 &= \Gamma(\tau, \tau_1, \tau_2) = \{d_A \mapsto (-1, l(A_B)),\ p_A \mapsto (k_p, c)\} \cup S \\
S &= \{d_B \mapsto *,\ p_B \mapsto *,\ d_C \mapsto *,\ p_C \mapsto *\}
\end{aligned}$$

Fig. 5. An example transition derivation

$$\begin{aligned}
\sigma_0 &= D \cup \{p_A \mapsto (0,c),\ p_B \mapsto (0,c),\ p_C \to (0,c)\} \\
\sigma_1 &= D \cup \{p_A \mapsto (0,c),\ p_B \mapsto (0,c),\ p_C \mapsto (k_p,c)\} \\
\sigma_2 &= D \cup \{p_A \mapsto (0,c),\ p_B \mapsto (k_p,c),\ p_C \mapsto (0,c)\} \\
\sigma_3 &= D \cup \{p_A \mapsto (0,c),\ p_B \mapsto (k_p,c),\ p_C \mapsto (k_p,c)\} \\
\sigma_4 &= D \cup \{p_A \mapsto (k_p,c),\ p_B \mapsto (0,c),\ p_C \mapsto (0,c)\} \\
\sigma_5 &= D \cup \{p_A \mapsto (k_p,c),\ p_B \mapsto (0,c),\ p_C \mapsto (k_p,c)\} \\
\sigma_6 &= D \cup \{p_A \mapsto (k_p,c),\ p_B \mapsto (k_p,c),\ p_C \mapsto (0,c)\} \\
\sigma_7 &= D \cup \{p_A \mapsto (k_p,c),\ p_B \mapsto (k_p,c),\ p_C \mapsto (k_p,c)\}
\end{aligned}$$

where $D = \{d_A \mapsto (-1, l(A_B)), d_B \mapsto (-1, l(B_C)), d_C \mapsto (-1, l(C_A))\}$.

Fig. 6. The states of the Repressilator

$$\begin{aligned}
Rep_{\sigma_0} &= \{\tfrac{dA_B}{dt} = -k_d A_B, && \tfrac{dB_C}{dt} = -k_d B_C, && \tfrac{dC_A}{dt} = -k_d C_A\} \\
Rep_{\sigma_1} &= \{\tfrac{dA_B}{dt} = -k_d A_B, && \tfrac{dB_C}{dt} = -k_d B_C, && \tfrac{dC_A}{dt} = -k_d C_A + k_p\} \\
Rep_{\sigma_2} &= \{\tfrac{dA_B}{dt} = -k_d A_B, && \tfrac{dB_C}{dt} = -k_d B_C + k_p, && \tfrac{dC_A}{dt} = -k_d C_A\} \\
Rep_{\sigma_3} &= \{\tfrac{dA_B}{dt} = -k_d A_B, && \tfrac{dB_C}{dt} = -k_d B_C + k_p, && \tfrac{dC_A}{dt} = -k_d C_A + k_p\} \\
Rep_{\sigma_4} &= \{\tfrac{dA_B}{dt} = -k_d A_B + k_p, && \tfrac{dB_C}{dt} = -k_d B_C, && \tfrac{dC_A}{dt} = -k_d C_A\} \\
Rep_{\sigma_5} &= \{\tfrac{dA_B}{dt} = -k_d A_B + k_p, && \tfrac{dB_C}{dt} = -k_d B_C, && \tfrac{dC_A}{dt} = -k_d C_A + k_p\} \\
Rep_{\sigma_6} &= \{\tfrac{dA_B}{dt} = -k_d A_B + k_p, && \tfrac{dB_C}{dt} = -k_d B_C + k_p, && \tfrac{dC_A}{dt} = -k_d C_A\} \\
Rep_{\sigma_7} &= \{\tfrac{dA_B}{dt} = -k_d A_B + k_p, && \tfrac{dB_C}{dt} = -k_d B_C + k_p, && \tfrac{dC_A}{dt} = -k_d C_A + k_p\}
\end{aligned}$$

Fig. 7. The ODEs for the Repressilator

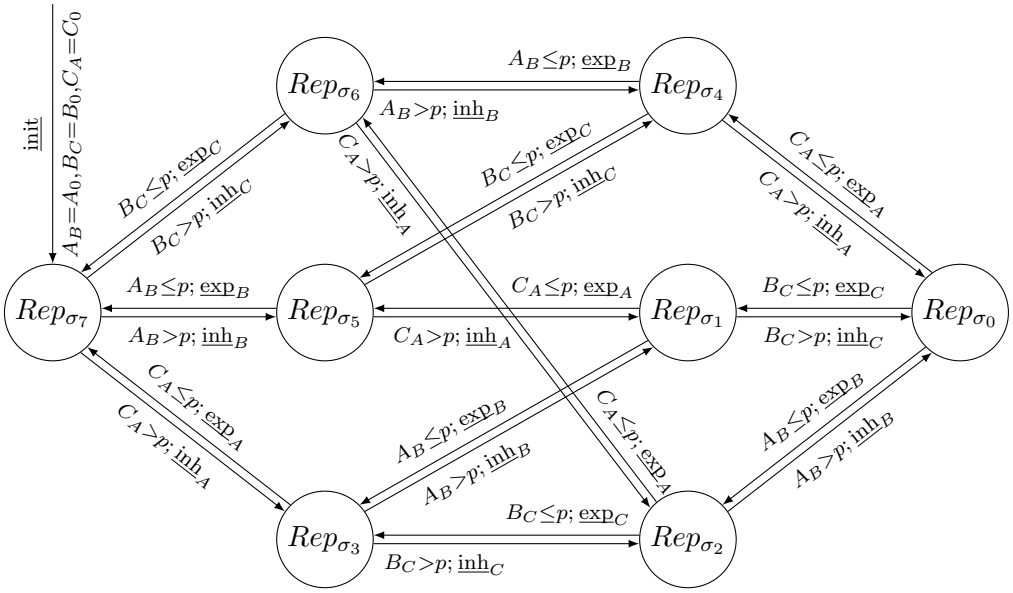

Fig. 8. The Repressilator as a hybrid automaton

original model describes explicitly both the transcription and translation phases, using two kind of variables, one for protein concentrations (A_B,B_C,C_B) and one for mRNA concentrations (M_A,M_B,M_C). Translation is assumed to be proportional to mRNA concentration, while transcription is modelled using an inhibitory Hill-like kinetics with exponent 2. The transcription of protein A, for instance, has speed $\alpha/(1+C_A^2)$. The exponent 2, in particular, assumes a cooperative effect of repressor-binding. The authors also assume a "leakiness" of translation; even in presence of a saturating amount of the repressor, there is still production of the protein with rate α_0. Introducing degradation mechanisms for proteins and mRNA, we obtain the following equations for protein A_B and mRNA M_A.

$$\frac{dM_A}{dt} = -M_A + \frac{\alpha}{1+C_A^2} + \alpha_0 \qquad \frac{dA_B}{dt} = -\beta A_B + \beta M_A$$

Our model can be seen as an approximation of this model. We have abstracted away from the intermediate step of mRNA, assuming proteins to be produced directly. In addition, our production rate equals k_p when the repressor is below the critical threshold p, and equals zero for when repressor is above p. Inspecting the transcription rate in the original model [8], and setting the leakiness rate α_0 to zero, we can observe how the term $\alpha/(1+C_A^2)$ has a sigmoid shape, approximately equal to α when the repressor is low, and to zero when the repressor is high. Therefore, by assuming an activation threshold, we are simply discretizing this sigmoidal behaviour.

As described earlier, it is possible to obtain a hybrid automata from a HYPE model by considering the operational and hybrid semantics. The hybrid automata obtained from the HYPE model of the Repressilator is given in Figure 8. Each node in the figure is labelled with the name of a set of ODEs from Figure 7. The arcs are labelled with event conditions, followed by events. The obvious abbreviations have

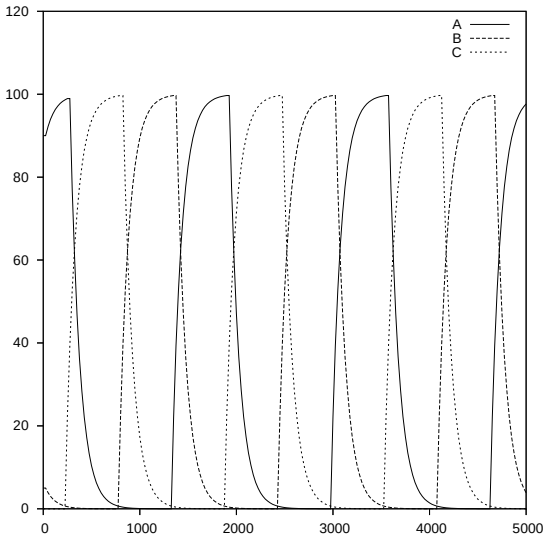

Fig. 9. Changes in protein concentration over time for $k_p = 1$, $k_d = 0.01$, $p = 1$, $A_0 = 95$, $B_0 = 5$ and $C_0 = 0$

been used for event names. The arc for the init event has no event conditions but it does have the appropriate resets. In the following, the name of the set of ODEs on a mode will be used as the name of the mode.

3.1 Analysis

Given a set of starting values, as well as parameter values for k_p, k_d and p, we can draw a graph of how the value of each protein changes over time. The graph in Figure 3.1 has $k_p = 1$, $k_d = 0.01$, $p = 1$, and initial values $A_0 = 95$, $B_0 = 5$ and $C_0 = 0$, and is similar to the graphs in [3] and [4]. A second graph is given in Figure 3.1 using the parameters $k_p = 150$, $k_d = 0.07$, $p = 100$, $A_0 = 1500$, $B_0 = 500$ and $C_0 = 0$ and is similar in amplitude and period to the graph presented in the original paper [8]. Both graphs show the oscillations we expect.

The behaviour shown is deterministic after the first two mode changes. The system starts in mode Rep_{σ_7} and immediately follows a two-edge path to mode Rep_{σ_4} representing Gene B and Gene C being off, so the concentration of proteins B_C and C_A are decreasing. Once the concentration of B_C goes below the threshold p, Gene C can switch on and the concentration of C_A starts to increase, and the system is now in mode Rep_{σ_5}. As the concentration of C_A goes over the threshold, Gene A switches off taking the system to mode Rep_{σ_1}. Continuing in this manner, it can be seem that the deterministic and repeating path in the hybrid automata is $Rep_{\sigma_4}, Rep_{\sigma_5}, Rep_{\sigma_1}, Rep_{\sigma_3}, Rep_{\sigma_2}, Rep_{\sigma_6}, Rep_{\sigma_4}$. This is demonstrated in Figure 11.

When at least one of the initial values differs from the other two, it can be shown that the model will eventually demonstrate deterministic behaviour for those given initial values and parameters. This occurs since one of the modes in the above sequence will always be reached even if the system starts in mode Rep_{σ_7} or Rep_{σ_0}.

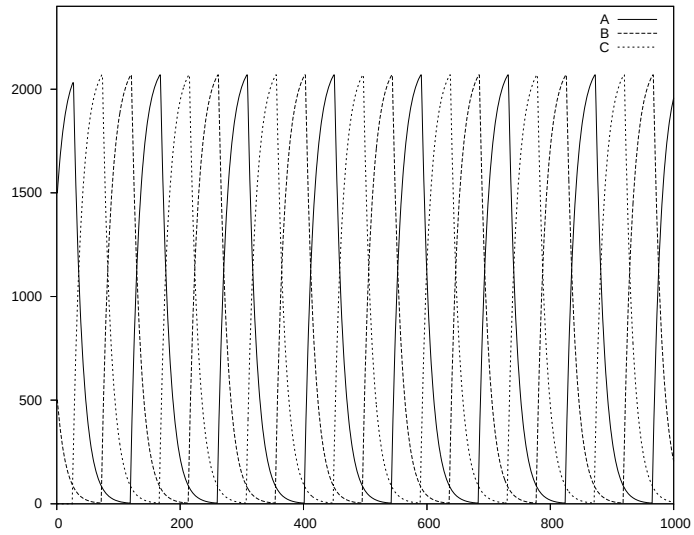

Fig. 10. Changes in protein concentration over time for $k_p = 150$, $k_d = 0.07$, $p = 100$, $A_0 = 1500$, $B_0 = 500$ and $C_0 = 0$

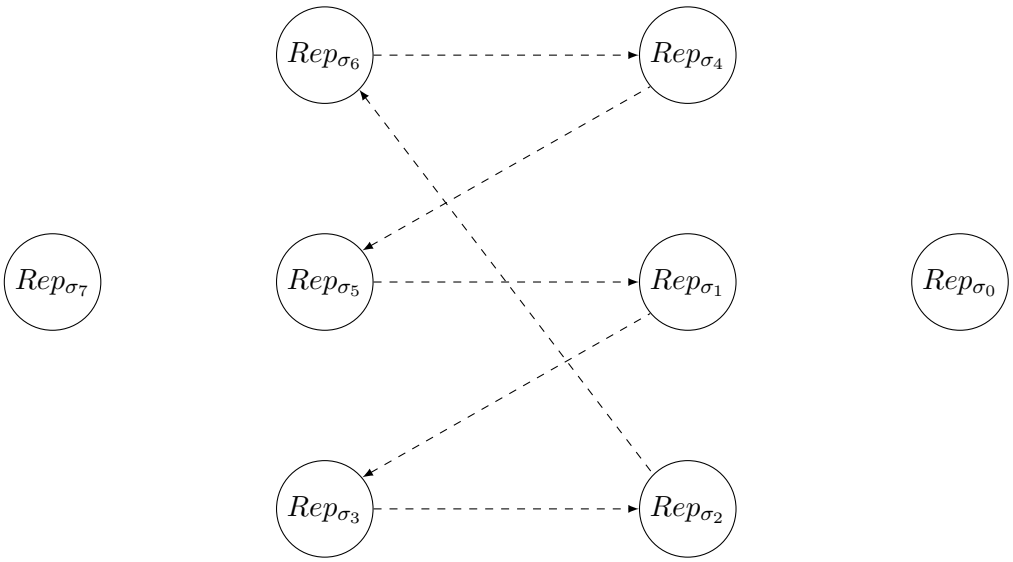

Fig. 11. The deterministic and repeating path of the Repressilator

When the initial values are identical, pathological behaviour can occur. If the systems starts in the mode Rep_{σ_7}, the protein values will increase simultaneously until they become greater than p, then nondeterministically, a path (consisting of three modes) from Rep_{σ_7} to Rep_{σ_0} will be taken with each event happening immediately. Then almost immediately, the values will drop to p and a nondeterministic path over three modes will be taken back to Rep_{σ_7} and this will repeat infinitely. Since identical quantities do not occur in nature, it would be unrealistic to use them in the model and hence the pathological case cannot occur.

4 Related work

As mentioned earlier, there are other process algebras that have been developed for hybrid systems: ACP_{hs}^{srt} [2], HyPA [6], Φ-calculus [17] and hybrid χ [19]. The main difference between modelling the Repressilator in HYPE versus any of these other process algebras is that in HYPE we do not need to explicitly embed the ODEs in the syntax of the model. So to model a single protein, say A_B, in one of these other process algebras, the model would require the equations

$$\frac{dA_B}{dt} = -k_d A_B \qquad \text{and} \qquad \frac{dA_B}{dt} = -k_d A_B + k_p$$

to both appear in the text of the model – this can be viewed as a monolithic approach. In contrast, with HYPE the idea is to identify possible influences on a variable of interest and construct a more fine-grained approach based on these influences, as can be seen by the use of the influences, p_A and d_A which both can contribute to the change in value of the variable A_B, and equations can be constructed from the model rather than added *a priori*. This gives a more natural modelling technique.

Other hybrid approaches from computer science have been applied to the modelling of biological systems but differ significantly from our approach. An example is Alur *et al* [1] who used the hybrid systems language Charon to model regulatory networks. Rather than considering some aspects of the system as having discrete behaviour and some having continuous behaviour as with HYPE, they use a continuous deterministic ODE model of the species when the concentration is sufficiently high and a discrete stochastic model and simulation when the concentration is not high enough. Ye *et al* [20] model excitable cells using cycle-linear hybrid automata which are linear within a cycle but overall show non-linear behaviour. Excitable cells tend to be either in the excited state or not whereas other aspects of the model such as voltage change continuously. The piecewise linear nature of the model allows for analytical solutions.

Finally, Bortolussi and Policriti [4] discuss a hybrid system for the Repressilator similar to ours, mainly differing in the activation conditions for the "express" and "inhibit" events. They start from a model of Repressilator written in sCCP [**?**] and obtain a hybrid automaton via a hybrid semantics defined for sCCP. Activation conditions are more complicated: they are derived automatically from the stochastic program and they use clocks to fire the event at the expected time of the corresponding stochastic events. However, the behavior of their model is basically the same as the one presented here. The Repressilator has also been modelled in a non-hybrid manner using the π-calculus and stochastic simulation [3].

5 Conclusions and further work

We have presented a process algebra for modelling the hybrid behaviour of biological systems. The process algebra has novel features that include a fine-grained approach to modelling flows and an explicit controller, and the ability to map the model to hybrid automaton in a direct manner.

Considering further work, we wish to do more modelling of both biological and computer systems. For the Repressilator, we also to build a more concrete model that demonstrates the production and use of mRNA and "leakiness" where some mRNA is produced even though the gene appears to be switched off [8]. An important step after this would be to show equivalence between the abstract and concrete models. We also wish to develop models of actual biological systems as well as synthetic systems.

References

[1] Alur, R., C. Belta, F. Ivanĉić, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin and J. Schug, *Hybrid modeling and simulation of biomolecular networks*, in: M. D. D. Benedetto and A. L. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control, 4th International Workshop (HSCC 2001),* Rome, Italy, LNCS 2034 (2001), pp. 19–32.

[2] Bergstra, J. A. and C. A. Middelburg, *Process algebra for hybrid systems*, Theoretical Computer Science **335** (2005), pp. 215–280.

[3] Blossey, R., L. Cardelli and A. Phillips, *A compositional approach to the stochastic dynamics of gene networks*, in: C. Priami, L. Cardelli and S. Emmott, editors, *Transactions on Computational Systems Biology IV*, LNCS 3939 (2006), pp. 99–122.

[4] Bortolussi, L. and A. Policriti, *Modeling biological systems in concurrent constraint programming*, Constraints **13** (2008).

[5] Bortolusssi, L. and A. Policriti, *Hybrid approximation of stochastic process algebras for systems biology*, in: *IFAC World Congress,* Seoul, South Korea, 2008.

[6] Calder, M., S. Gilmore and J. Hillston, *Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA*, in: C. Priami, A. Ingólfsdóttir, B. Mishra and H. R. Nielson, editors, *Transactions on Computational Systems Biology VII*, LNCS 4230 (2006), pp. 1–23.

[7] Ciocchetta, F. and J. Hillston, *Bio-pepa: a framework for the modelling and analysis of biological systems*, Report EDI-INF-RR-1231, School of Informatics, University of Edinburgh (2007).
URL http://www.inf.ed.ac.uk/publications/report/1231.html

[8] Cuijpers, P. J. L. and M. A. Reniers, *Hybrid process algebra*, Journal of Logic and Algebraic Programming **62** (2005), pp. 191–245.

[9] Danos, V. and C. Laneve, *Formal molecular biology*, Theoretical Computer Science **325** (2004), pp. 69–110.

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

[11] Galpin, V., L. Bortolussi and J. Hillston, *HYPE: hybrid systems modelled with flows*, Report EDI-INF-RR-1251, School of Informatics, University of Edinburgh (2008).
URL http://www.inf.ed.ac.uk/publications/report/1251.html

[12] Henzinger, T. A., *The theory of hybrid automata*, in: *LICS*, 1996, pp. 278–292.

[13] Hillston, J., "A compositional approach to performance modelling," Cambridge University Press, 1996.

[14] Lincoln, P. and A. Tiwari, *Symbolic systems biology: hybrid modeling and analysis of biological networks*, in: R. Alur and G. J. Pappas, editors, *Hybrid Systems: Computation and Control, 7th International Workshop (HSCC 2004),* Philadelphia, PA, USA, LNCS 2993 (2004), pp. 660–672.

[15] Priami, C., *Stochastic π-calculus*, The Computer Journal **38** (1995), pp. 578–589.

[16] Priami, C. and P. Quaglia, *Beta binders for biological interactions*, in: V. Danos and V. Schächter, editors, *Computational Methods in Systems Biology, International Conference (CMSB 2004),* Paris, France, LNCS 3082 (2004), pp. 20–33.

[17] Priami, C., A. Regev, E. Shapiro and W. Silverman, *Application of a stochastic name-passing calculus to representation and simulation of molecular processes*, Information Processing Letters **80** (2001), pp. 25–31.

[18] Regev, A., E. Panina, W. Silverman, L. Cardelli and E. Shapiro, *BioAmbients: an abstraction for biological compartments*, Theoretical Computer Science **325** (2004), pp. 141–167.

[19] Rounds, W. C. 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, 6th International Workshop (HSCC 2003),* Prague, Czech Republic, LNCS 2623 (2003), pp. 435–449.

[20] Tuffin, B., D. S. Chen and K. S. Trivedi, *Comparison of hybrid systems and fluid stochastic petri nets*, Discrete Event Dynamic Systems: Theory and Applications **11** (2001), pp. 77–95.

[21] van Beek, D., K. Man, M. Reniers, J. Rooda and R. Schiffelers, *Syntax and consistent equation semantics of hybrid Chi*, Journal of Logic and Algebraic Programming **68** (2006), pp. 129–210.

[22] Ye, P., E. Entcheva, R. Grosu and S. A. Smolka, *Efficient modeling of excitable cells using hybrid automata*, in: G. Plotkin, editor, *Proceedings of Computational Methods in Systems Biology (CMSB 2005),* Edinburgh, Scotland, 2005, pp. 216–227.