On the Hybrid Composition and Simulation of Heterogeneous Biochemical Models

Katherine Chiang[1,2], François Fages[1], Jie-Hong Jiang[2], and Sylvain Soliman[1]

[1] EPI Contraintes, Inria Paris-Rocquencourt, France
[2] Graduate Institute of Electronics Engineering, National Taiwan University, Taiwan

Abstract. Models of biochemical systems presented as a set of formal reaction rules with kinetic expressions can be interpreted with different semantics: as either deterministic Ordinary Differential Equations, stochastic continuous-time Markov Chains, Petri nets or Boolean transition systems. While the formal composition of reaction models can be syntactically defined as the (multiset) union of the reactions, the hybrid composition of models in different formalisms is a largely open issue. In this paper, we show that the combination of reaction rules with conditional events, as the ones already present in SBML, does provide the expressive power of hybrid automata and can be used in a non standard way to give meaning to the hybrid composition of heterogeneous models of biochemical processes. In particular, we show how hybrid differential-stochastic and hybrid differential-Boolean models can be compiled and simulated in this framework, through the specification of a high-level interface for composing heterogeneous models. This is illustrated by a hybrid stochastic-differential model of bacteriophage T7 infection, and by a reconstruction of the hybrid model of the mammalian cell cycle regulation of Singhania et al. as the composition of a Boolean model of cell cycle phase transitions and a differential model of cyclin activation.

1 Introduction

Systems biology aims at elucidating the high-level functions of the cell from their biochemical basis at the molecular level [24]. A lot of work has been done for collecting genomic and post-genomic data, making them available in databases [5,25], and organizing the knowledge on pathways and interaction networks into models of cell metabolism, signaling, cell cycle, apoptosis, etc. now published in model repositories (e.g. `http://biomodels.net/`). In particular, the Systems Biology Markup Language (SBML) [23] provides a common exchange format for biochemical *reaction models* and is nowadays supported by a majority of modeling tools.

According to the knowledge available on the system and to the nature of the queries that will be asked to the model, e.g. qualitative or quantitative predictions, these reaction rule-based models can be interpreted (and simulated) under different semantics as either:

- ordinary differential equations (differential semantics),
- continuous-time Markov chains (stochastic semantics),
- Petri nets (discrete semantics),

– Boolean transition systems (Boolean semantics),
– and many variants.

Some modeling tools support several of these different interpretations which can also be related by approximation [16,17,18] or abstraction [11] relationships.

In the perspective of applying engineering methods to the analysis and control of biological systems, the issue of building complex models by composition of elementary models is a central issue. While reaction rule-based models can be formally composed simply by the multiset union of reaction rules, and interpreted by one common semantics, there is also a need to compose models with different semantics. What we call a *hybrid model* is a model obtained by composition of models with heterogeneous semantics (differential, stochastic, Boolean, etc.), and *hybrid simulation* is the topic of simulating such hybrid models.

Hybrid simulation is a classical topic in physics on the one hand, e.g. for numerically solving equations describing stochastic systems using ordinary differential equations whenever possible in place of stochastic equations in order to speed-up simulations [3,31], and on the other hand, in computer science for programming and verifying hybrid systems which have both discrete and continuous dynamics [9,4,21]. Hybrid modeling is also used in systems biology for reducing the complexity of many modeling task, e.g. [29,4,13,6,26,1,33], or for speeding up stochastic simulations [32,19,22].

In this paper, we show that the combination of reaction rules with conditional events, as the one already present in SBML, does provide the expressive power of hybrid automata and can be used in a non standard way to give meaning to the hybrid composition of heterogeneous reaction models. In particular, we show how hybrid differential-stochastic and hybrid differential-Boolean models can be compiled and simulated in this formal framework of reactions plus events, through the specification of a high-level interface for composing reaction models.

This interface for composing models has been implemented as a preprocessor for Biocham [7,10]. This preprocessor transforms stochastic reaction models in events that implement Gillespie's direct method for stochastic simulation and that can be combined with the simulation of differential reaction models. Similarly, it transforms Boolean state transition models in events with extra conditions that express the links to the continuous variables and parameters of the differential reaction model.

This approach is illustrated through the hybrid stochastic-differential composition and simulation of bacteriophage T7 infection [3], and a reconstruction of the hybrid model of the mammalian cell cycle regulation of Singhania et al. [33] as the composition of a Boolean model of cell cycle phase transitions and a differential model of cyclin activation.

2 The Expressive Power of Events with Kinetic Reactions

2.1 Reactions rules with Kinetics

In the spirit of the Chemical Reaction Network Theory [12], we define our systems of study as sets of reaction rules r_i, however as in SBML [23] any function can be used as reaction rate. In the following this will be represented using Biocham syntax [10] as: v_i for $\sum_j l_{ij} \times S_j \Rightarrow \sum r_{ij} \times S_j$, where v_i is a continuous function[1] of parameters of the

[1] It would be possible to admit non-continuous functions as rates (e.g., conditional statements in v_i), and that is actually the case in many tools, however the same result can be obtained with the event mechanism described in the next section.

system and of species concentrations defining the rate of reaction i (mass action kinetics of parameter k are abbreviated as $MA(k)$), l_{ij} and r_{ij} are stoichiometric coefficients, and the S_j are the species of the model.

According to the data available on the system and to the nature of the queries that will be asked to the model, e.g. qualitative or quantitative predictions, these reaction models can be interpreted (and simulated) under different semantics: differential, stochastic, discrete or boolean. We recall here the basics of these semantics. An Ordinary Differential Equation (ODE) system can be defined from a reaction model as follows: $\frac{d[S_j]}{dt} = \sum_i (r_{ij} - l_{ij}) \times v_i$

The differential semantics corresponds to the limit of the Continuous-Time Markov Chain defined using the v_i as propensities, and realizing the solution of the Chemical Master Equation [16]. The differential semantics usually leads to numerical integration, whereas the stochastic semantics is either used for exact or approximate simulation, or for stochastic Model-checking (see for instance [27]).

The discrete semantics forgets about the rates v_i but keeps the stoichiometric information, for instance as weights in a Petri net representation [8,14].

Finally, the Boolean semantics forgets about precise stoichiometry and keeps only information about whether or not a species is active. It can be defined as an abstraction of the previous discrete semantics [11].

2.2 Semantics of Events

In this section, we present a generic notion of *events* compatible with the differential semantics of reaction models and then describe how it relates to existing concepts, most notably the events of SBML and Biocham.

An event is basically twofold, it is built by a *condition*, determining when it fires, and by an *action*, i.e., its influence on the current state (parameters, concentrations). If one wants to enforce the continuity of concentration variables, they can simply exclude them from the variables that can be modified by the *action* part of the events.

Following Biocham syntax, we will write an event as follows: *event*(*condition*, $[s_1, \ldots, s_n], [f_1, \ldots, f_n]$), where the s_i indicate the state variables that are modified by the event, the f_i are functions of the state that give the new value to s_i.

There are many possible semantics for events but the basic idea is that an event fires when its condition changes from *false* to *true*. This induces however several issues:

- what happens at the start of the simulation?
- how to find the precise time when a condition becomes true?
- what happens if some events are enabled simultaneously?

The first point is easy to settle, it is an arbitrary decision but does not have a big impact. The simplest choice is to avoid the firing of events at the initial point of the simulation and to reflect initial events by modifying accordingly the initial state.

The second point has been solved in practical tools for a long time: since numerical integration goes by steps, one detects changes in conditions only in the interval of a simulation step. One can simply go back in time until one finds—with a given precision—the first time point where a condition becomes true. Note however that if arbitrarily complex conditions appear in the events, a numerical integrator unaware of the events can hide inside a single step that a condition went from false to true and back to false again. Therefore, a cautious implementation is necessary, and often, fixed step size integration methods are recommended to use, instead of more efficient adaptive step size methods in presence of events.

The final point is again a question with multiple possible answers. Generally, the set of events that are enabled simultaneously at a given time will all be fired, whatever the actions of the events are, but what if several events modify the same variable? It is possible to assume a *synchronous* semantics, where the simultaneous events execute their actions in parallel, but then one must forbid events with conflicting actions, i.e., events that would modify in different ways the same variable at the same time point. The more common choice is an *asynchronous* semantics, that will fire all the events enabled at a given time one after the other, even if some actions invalidate the condition of other enabled events. Conflicts in actions are then solved by the ordering of events, which can be either random, i.e. non-deterministic, or given by the modeller, e.g. by the order of writing or by priorities.

The SBML choice is to keep a very flexible semantics, with asynchronous events, that may be ordered by *priorities*, and that can use either the values at the time they were enabled, or the *current* values at the time they are actually executed, after the execution of the simultaneous events with higher priority.

In Biocham, there are no priorities, the events that are enabled simultaneously are executed in the order of their writing using current values. An event with n assignments of f_i to s_i is therefore equivalent to the sequence of n events with the same condition for each assignment f_i to s_i. The semantics of events implemented in Biocham can thus be defined in SBML using the current value option and priorities corresponding to the order of writing.

2.3 Representation of Hybrid Automata by Reactions and Events

A *hybrid automaton* (HA) is a dynamical system containing both continuous and discrete components [20]. They are therefore commonly used to formalize real-life safety-critical systems and have led to various works on the verification of their different semantics and on their composition (e.g. with Hytech [21]).

Formally, a hybrid automaton is defined by a set of continuous variables, a *control graph* where edges are labelled by *jump conditions and events*, defining the discrete state changes with some labels, and vertices are labelled by *initial, invariant and flow conditions* defining the continuous change in each state. Figure 1 (left) shows the traditional thermostat example.

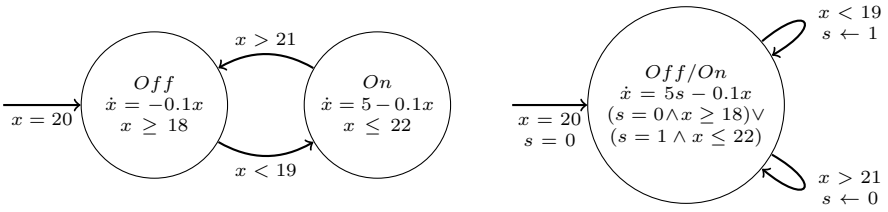

Fig. 1. The classical thermostat example encoded in a single state hybrid automaton.

Since the *jumps* describe the possible transitions with a complete description of the resulting state, there are no issues similar to what was described in the previous section to handle conflicting updates.

Note that, it is enough to restrict oneself to hybrid automata with a single state (vertex) with a big parametric system of ODEs corresponding to all the ODEs of the initial states, multiplied by a parameter that is non null only when the corresponding state is *active*. Then the jumps and event labels can remain the same, except that they go from the single state to itself, and change the state variable according to the initial state change. The invariants have as additional condition that the corresponding state must be active to be enforced. One obtains Figure 1 (right) where the ODE system has been factorized for readability.

Now to represent an hybrid automaton in the framework of reaction and events described above, one can first note that the initial and flow conditions simply define an ODE system. Such a system can be represented with reactions, for instance as a synthesis for each variable with rate corresponding to the variable's derivative in the corresponding state.

The jumps can easily be represented as events, however since they do not represent events that should fire, but, unless it violates an invariant, events that may fire, they should be accompanied by another event allowing the state not to change. This event will have as condition the fact that the current invariant is true and that a condition to leave is true. This second part is not necessary but avoids useless firings of events that do not change anything. This event will also need to be able to fire repeatedly, it will thus have a supplementary condition *can_fire* that it will itself make false, another event will always make it true again when it is false.

Note that this encoding relies on a non-deterministic asynchronous semantics for events, as discussed in Section 2.2. Here is the thermostat example as reactions and events:

```
5*s for _ => x.                                  0.1*[x] for x => _.

present(x, 20).                                  parameter(enabled1, 0).
                                                 parameter(enabled2, 0).
parameter(s, 0).                                 parameter(can_fire, 1).

event(s = 1 and [x] > 21 and can_fire = 1, [s, can_fire], [0, 0]).
event(s = 1 and [x] > 21 and [x] =< 22 and can_fire = 1,
                                                  [s, can_fire], [1, 0]).

event(s = 0 and [x] < 19 and can_fire = 1, [s, can_fire], [1, 0]).
event(s = 0 and [x] < 19 and [x] >= 18 and can_fire = 1,
                                                  [s, can_fire], [0, 0]).

event(can_fire = 0, [can_fire], [1]).
```

3 Hybrid Differential-Stochastic Semantics

Chemical reactions, originated from random collisions of particles, are discrete and stochastic in nature. Although there is no way to predict the exact state of a chemical system at a specific time point, its *statistical* behavior can be effectively calculated from known probabilistic properties. A well-mixed, non-linear chemical system can be described by a set of master equations, which in turn can be completely solved by Gillespie's stochastic simulation algorithm (SSA) [15], to be detailed in Section 3.1.

Essentially the computation cost of an SSA grows proportional to the number of reaction occurrences. Simulating a system of chemical reactions can be especially slow if one or more of the reactions have fast reaction rates (or high event occurrences) because the next reaction time will be very short due to the high probability of selecting (one of the) fast reactions for firing.

A chemical system may consist of reactions proceeding with significantly different rates. Despite the fact that all reactions are innately stochastic, those with large reactant counts and high reaction rates can be accurately approximated in terms of deterministic behavior expressed by ODEs. By incorporating both differential and stochastic semantics into one simulator, an optimal balance between simulation runtime and accuracy can be achieved. This potentially lifts the scalability of simulating large biological systems. In Section 3.2, we provide an event-based view on the SSA, that serves as basis to a hybrid differential-stochastic simulator built upon an ODE simulator with events.

3.1 Gillespie's Direct Method

A reaction model with kinetic expressions can be interpreted under the stochastic semantics as a continuous-time Markov chain (CTMC). A CTMC can be simulated with a stochastic simulation algorithm (SSA), for example, *Gillespie's direct method* [15]. Rather than solving all possible trajectories' probabilities as in the case of Master equations, the algorithm generates statistically correct trajectories.

Gillespie's direct method first calculates *when* the next reaction will occur, then decides *which* reaction should occur with the help of a random number generator. The probability that a certain reaction μ will take place in the next instant of time is given by its propensity: $\alpha_\mu = (\#\text{combinations of reactants}) \cdot k_\mu$ where k_μ is μ's rate coefficient. The algorithm repeats the following steps.

1. Calculate *how long from now* ($\triangle t$) the next reaction will occur.

$$\triangle t = \frac{-1}{\sum_j \alpha_j} \cdot \log(r_1),$$

   where r_1 is a random number within range $(0, 1)$ and the α_j are propensities at the current state.
2. Choose which reaction will occur according to the probability distribution of reactions. This is done by generating a random number r_2 within range $(0, 1)$, and letting the reaction μ_i be chosen for

$$\frac{\sum_{k=1}^{i-1} \alpha_k}{\sum_j \alpha_j} < r_2 \leqslant \frac{\sum_{k=1}^{i} \alpha_k}{\sum_j \alpha_j}.$$

3. Update the numbers of molecules to reflect the execution of reaction μ_i, and set current time to $t = t + \triangle t$.

3.2 Event Model of Stochastic Simulation

By considering every firing of a chemical reaction as one firing of an event, the *event* semantics of Section 2 enables a direct embedding of stochastic reactions into an intrinsically differential framework without additional implementation of a separate stochastic

simulation algorithm. Under this framework, **time** is the only unifying variable to keep track of current state at each instant. This event-based approach permits the simple integration of ODE and stochastic simulation as will be elaborated in Section 3.3.

Notice that, in the SSA of Section 3.1, *when* the next reaction will occur is independent of *which* reaction will occur, and also that only one reaction is chosen each time. These facts make the complete set of stochastic reaction rules be simulated correctly with a single event. Essentially the simulation can be accomplished by compiling the *when* and *which* questions Gillespie's direct method asks into an event. Specifically the event is triggered by the calculated next reaction time (`tau`); the event obtains a new random variable (`ran`) and then conditionally updates the molecular counts depending on which reaction is chosen to occur next. To accommodate all stochastic rules in one event, each update entry is composed of conditional expressions over the propensities and the random number that decides which reaction occurs.

Example 1 ([15]). Given the stochastic reaction rules $A + 2B \xrightarrow{k_1} C$ and $C \xrightarrow{k_2} 2A$ we derive their propensities by $\texttt{alpha1} = k_1 \times (\#A) \times \frac{(\#B)\times(\#B-1)}{2}$, $\texttt{alpha2} = k_2 \times (\#C)$, where "#" denotes the particle count of a species. Then the next reaction time from the current time point can be decided by $\texttt{e} = \frac{-1}{\texttt{alpha\_sum}} \cdot \log(random_1)$ for $random_1$ within $(0, 1)$ and where $\texttt{alpha\_sum} = \texttt{alpha1} + \texttt{alpha2}$. The first reaction is chosen for the next occurring reaction if $0 < (\texttt{alpha\_sum} \times random_2) \leqslant \texttt{alpha1}$, which leads to the consumption of one A and two B's and producing one C:

```
event(Time>tau, [tau, ran, #A, #B, #C],
                [Time + e, random,
                 if alpha_sum*ran =< alpha1 then #A-1 else #A+2,
                 if alpha_sum*ran =< alpha1 then #B-2 else #B,
                 if alpha_sum*ran =< alpha1 then #C+1 else #C-1]).
```

Note that the update of the particle counts of the first reaction is reflected in the three `then` entries, and that of the second reaction is reflected in the three `else` entries.

This encoding relies on the left to right ordering of the different events associated to a single trigger (see Section 2.2). This ordering is imposed to three kinds of parameters, including the random number for choosing reaction, the lower bound for particle number, and a reaction's propensity function, such that possible errors are avoided. Because these three kinds of parameters all depend on the *current* number of molecules, they are listed in front of molecular species. So their values are not changed before the *completion* of reaction firing, that is, all species' counts have been updated according to the chosen reaction.

3.3 Preprocessor for Composing Differential and Stochastic Models

The purpose of our preprocessor for composing heterogeneous biochemical models is to provide a user-friendly interface to allow users of various backgrounds to conduct hybrid simulation without knowing algorithmic details. The only work required is to decide the semantic model for each of the reactions under simulation.

In classical work on hybrid simulation [3,26], chemical reactions are divided according to their propensities and reactants' concentrations into two groups: one consisting of reactions to be simulated stochastically using SSAs, and the other consisting of reactions to be simulated deterministically using ODEs. The former is referred to as

the *stochastic reactions* and the latter *differential reactions*. While differential reactions simply advance with the pass of time, stochastic reactions fire discretely in time with frequency in accordance with their propensities. When the reactant concentrations and the propensity of a reaction are sufficiently large, ODE simulation can be faithfully applied. It avoids frequent simulation updates within a small time interval, thus accelerating simulation speed.

Hybrid species are referred to as those involved in both stochastic and differential rules. This kind of species requires special attention because they are influenced by two different mechanisms: *ODEs* that govern differential behavior by continuously changing related *concentrations*, and *events* that regulate stochastic behavior by modifying *molecule counts* discretely whenever triggered. So a hybrid species is under two kinds of modification: one targets at the evolution of macroscopic concentrations and the other targets at the changes in microscopic particle counts.

In our implementation, a fresh new variable is introduced for each hybrid species to represent its quantity (the summation of the numbers of particles from both differential and stochastic models). In all kinetic expressions, the hybrid species are expressed by the corresponding new variables. It is then a simple matter to put together the ODEs for the continuous part and the events corresponding to the encoding of the stochastic part as described in the previous section.

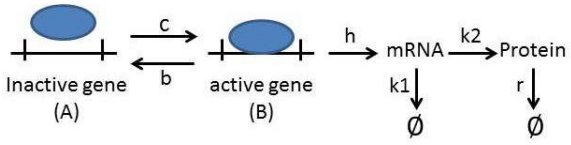

Fig. 2. Gene Regulation Network

Example 2. Let us consider the single gene regulatory model shown in Figure 2. Let the reactions for protein generation and degradation, namely, $mRNA \xrightarrow{k_2} mRNA + protein$ and $protein \xrightarrow{r} \emptyset$ be under the differential interpretation, and all other reactions, namely, $A \underset{b}{\overset{c}{\rightleftharpoons}} B$, $B \xrightarrow{h} mRNA + B$ and $mRNA \xrightarrow{k_1} \emptyset$ be under the stochastic interpretation.

```
% Differential rules
MA(k2) for mRNA => mRNA + protein.
MA(r) for protein => _.
```

```
% Stochastic rules
(MA(c), MA(b)) for A <=> B.
MA(h) for B => mRNA + B.
MA(k1) for mRNA => _.
```

Our preprocessor generates a hybrid model composed of reactions and events. Due to the stochastic nature of the reactions, there is no way to check the results point by point. Nevertheless, comparison of mean values and standard deviations shows very good agreement with purely stochastic simulations. The following table shows the CPU time improvement in this example. The number of fired events is about six times smaller and the runtime on a Macbook Pro is about four times faster.

	step size = 0.01		step size = 0.02	
method	#fired_event	CPU time (sec)	#fired_event	CPU time (sec)
stochastic	89066	63.2	83856	51.5
hybrid	14258	15.1	14183	12.9
ratio	0.16	0.24	0.17	0.25

Example 3. The reaction model of bacteriophage T7 infection described in [3] is an interesting example that can be similarly hybridized by partitioning the reactions with differential semantics for protein synthesis and with stochastic semantics for gene activation, as follows:

```
% Differential reaction rules
MA(c5) for tem => tem + struc.
MA(c6) for struc => _.
```

```
% Stochastic reaction rules
MA(c1) for gen => tem.
MA(c2) for tem => _.
MA(c3) for tem => tem + gen.
MA(c4) for gen + struc => virus.
```

In this example, `tem` and `struc` are hybrid species, while `gen` and `virus` are purely stochastic. The following table shows that the hybrid simulation improves by three orders of magnitude the simulation time over a time horizon of 100 hours with a step size of 0.01:

method	#fired_event	CPU time (sec)
stochastic	276556	218.7
hybrid	832	0.75
ratio	0.003	0.003

It is worth noting that in these examples, the user is responsible for a partition of reactions into differential and stochastic groups, that is fixed for the rest of simulation. This restriction may lead to inaccurate or inefficient simulation if the propensity and/or reactants' counts of a reaction change substantially over time and violate the underlying assumptions of differential and stochastic semantics. It is therefore desirable to dynamically adjust the reaction partition along the progress of simulation.

Interestingly, the described framework allows us to easily explore various dynamic partitioning strategies considering the crucial factors of *particle count* and *propensity value* [3]. All species become potentially hybrid and criteria are imposed such that, during the simulation run, the reactions interpreted under the differential semantics are maximized while their current particle counts and propensity values must satisfy some accuracy requirement with respect to the simulation step size.

4 Hybrid Differential-Boolean Semantics

4.1 Preprocessor for Composing Differential and Boolean Models

In this section, we consider the hybrid composition of differential reaction models with Boolean transition models. One typical use of this form of composition is for modeling the interactions between gene expression and metabolism on different time scales. Gene networks can be modeled by simple Boolean regulatory models representing the on/off states of the genes and the possible transitions from one state to another, while metabolic networks are naturally modeled by ODE systems. Hybrid models of gene expression and metabolism can thus be naturally built as hybrid Boolean-differential models, and analyzed and simulated as hybrid automata.

A Differential-Boolean composition necessitates specifying:

– the link between the continuous variables and the Boolean variables, e.g. by fixing concentration threshold values,
– the relationship between the discrete logical time of the Boolean model and the continuous real time of the ODE model, e.g. by adding delays on Boolean transitions,
– the integrity constraints between both dynamics.

There is currently no general method for these tasks. As shown in Section 2.3 however, a set of reactions and events can be interpreted as a hybrid automaton in which there is a state with a particular ODE for each combination of the trigger values, and there is a transition from one state to another state when at least one trigger changes value from false to true in the source state.

This low level mechanism provides all what is necessary to compose differential models with Boolean models, compile them in reaction rules plus events and execute them using hybrid simulations. In the following section we illustrate our composition preprocessor on a hybrid model of the cell cycle.

4.2 Hybrid Composition of Differential-Boolean Cell Cycle Models

In [33], Singhania et al. have proposed a simple hybrid model of the mammalian cell cycle regulation. This cell cycle model of low dimension has been evaluated in terms of flow cytometry measurements of cyclin proteins in asynchronous populations of human cell lines. The few kinetic constants in the model are easier to estimate from the experimental data that the numerous kinetic constants of a single large ODE model. Using this hybrid approach, modelers could thus quickly create quantitatively accurate, computational models of protein regulatory networks in cells.

In this model, Cyclin abundances are tracked by piecewise linear differential equations for cyclin synthesis and degradation. Cyclin synthesis is regulated by transcription factors whose activities are represented by discrete variables (0 or 1) and likewise for the activities of the ubiquitin-ligating enzyme complexes that govern cyclin degradation. The discrete variables change according to a predetermined sequence, with the times between transitions determined by the amount of cyclin presented as well as exponentially distributed random variables.

This model can be reconstructed in our framework as the hybrid composition of a differential reaction model of cyclin activation and degradation, with a Boolean model of cell cycle phase transitions. In our high level interface, this composition is specified by providing as input

1. the differential reaction model of cyclin activation:

```
k_sa for _ => CycA.        MA(k_da) for CycA => _.
k_sb for _ => CycB.        MA(k_db) for CycB => _.
k_se for _ => CycE.        MA(k_de) for CycE => _.
```

   with initial concentrations and symbolic kinetic expressions

```
k_sa=5+6*B_tfe+20*B_tfb.   k_da=0.2+1.2*B_cdc20a+1.2*B_cdh1.
k_sb=2.5+6*B_tfb.          k_db=0.2+1.2*B_cdc20b+0.3*B_cdh1.
k_se=0.02+2*B_tfe.         k_de=0.02+0.5*B_scf.
```

2. the Boolean transition system of the cell cycle [33]:

```
states (B_tfe,B_scf,B_tfb,B_cdc20a,B_cdc20b,B_cdh1).
(0,0,0,0,0,1) ->2 (1,0,0,0,0,1) ->3 (1,0,0,0,0,0)
->4 (1,1,0,0,0,0) ->5 (1,1,1,0,0,0) ->6 (0,1,1,0,0,0)
->7 (0,1,1,1,0,0) ->8 (0,1,1,1,1,0) ->9 (0,1,0,1,1,1)
->1 (0,0,0,0,0,1)
```

3. the specification of the interface between both models as a set conditions and actions associated to the Boolean transitions and macros:

```
delta_t=lambda*exp(random).    tau=Time-delta_t.
masst=mass*exp(0.029*(Time-start_time)).
->2 condition [Time>tau] action [lambda=2,mass=masst/2]
->3 condition [Time>tau and [CycE]*masst>=80] action [lambda=0].
->4 condition [Time>tau and [CycA]>12.5] action [lambda=0.01].
->5 condition [Time>(tau+7)] action[lambda=1].
->6 condition [Time>tau and [CycB]>21.25] action [lambda=0.5].
->7 condition [Time>tau] action [lambda=0.75].
->8 condition [Time>tau] action [lambda=1.5].
->9 condition [Time>tau] action [lambda=0.5].
->1 condition [Time>tau and [CycB]<3] action [lambda=0.025].
```

The compilation process described in Section 2.2 returns the input differential reaction model augmented with a set of events for the Boolean transitions from state 1 to 9 and back to 1, and their synchronization with the differential reaction model. In this form, the simulation over a time horizon of 100 hours takes 60 ms on a MacBook Pro.

4.3 Related Work on Boolean Regulatory Models with Delays

René Thomas's discrete modelling of gene regulatory networks (GRN) is a well known approach to study the logical dynamics of a set of interacting genes. It deals with a graph of positive and negative influences between genes and logical functions that determine the possible trajectories in the state space. Those parameters are a priori unknown, but they may generally be deduced from a large set of biologically observed behaviors in various conditions. Besides, it neglects the time delays for a gene to pass from one level of expression to another one. In [1], it is shown that one can account for time delays depending on the expression levels of genes in a GRN, while preserving powerful enough computer-aided reasoning capabilities. The characteristic of this approach is that, among possible execution trajectories in the model, one can automatically find out both viability cycles and absorption in capture basins. Model-checking techniques developed for hybrid systems are used for this purpose [2]. The authors describe a Hybrid model for the mucus production in the bacterium Pseudomonas aeruginosa and show that they are able to discriminate between various possible dynamics [1,2]. Such a model can be presented and compiled in a set of reaction rules with events as described in the previous section.

Time constraints provide another mean to refine Boolean or discrete models which are often too coarse to be useful. In [28], the authors present a new technique for over-approximating (in the sense of timed trace inclusion) continuous dynamical systems by timed automata for the purpose of efficiently checking timed (as well as untimed) properties. The essence of this technique is the partition of the state space into cubes and the allocation of a clock for each dimension. This is in contrast with other approaches

which use only one clock. This idea is close in spirit to rectangular hybrid automata in the sense of separating and bounding the dynamics of each dimension. This makes it possible to get better approximations of the behavior. The timed automata produced by these techniques can be directly composed in our preprocessor for simulation.

5 Conclusion

The combination of kinetic reaction rules with conditional events, as already present in SBML, provides the expressive power of hybrid automata for combining discrete and continuous dynamics. Although introduced in SBML for handling some discrete events, such as for instance the division of the mass by two at each cell division in cell cycle models, SBML events can be used on a large scale as a basic mechanisms allowing for the composition of heterogeneous models and implementing hybrid simulators.

We have presented a high-level interface for composing hybrid models, compiling them in reactions plus events, and running hybrid simulations. In particular we have shown that hybrid differential-stochastic reaction models can be assembled with this interface, compiled in differential reactions plus events for emulating the stochastic reactions, and executed with a *de facto* hybrid simulator with either static or dynamic strategies. This has been illustrated with the hybrid model of bacteriophage T7 infection [3].

We have also shown that hybrid Boolean-differential models can similarly be composed, compiled in reactions plus events, and simulated, through a high-level interface for specifying the input models, the conditions on the continuous variables, and the time delays of the Boolean transitions. This has been illustrated by a reconstruction of the hybrid mammalian cell cycle model of Singhania et al. [33].

This shows the expressive power of SBML events and their possible use as a low-level implementation language for representing and simulating hybrid models. This also shows the need for generating such hybrid models with a preprocessor using a high-level interface as the one prototyped here in Biocham. We are currently improving this interface to use it on more examples and on hybrid models obtained by model reduction using tropicalization methods [30].

Acknowledgements

This work has been supported by the French OSEO Biointelligence and ANR BioTempo projects, and by the European Eranet Sysbio C5Sys project. We acknowledge fruitful discussions with our partners in these projects and with Robin Philip for his early work on this topic during an internship with us.

References

1. Jamil Ahmad, Gilles Bernot, Jean-Paul Comet, Didier Lime, and Olivier Roux. Hybrid modelling and dynamical analysis of gene regulatory networks with delays. *ComplexUs*, 3:231–251, 2006.
2. Jamil Ahmad, Olivier Roux, Gilles Bernot, Jean-Paul Comet, and Adrien Richard. Analysing formal models of genetic regulatory networks with delays. *International Journal of Bioinformatics Research and Applications*, 4(3):240–262, 2008.

3. Aurélien Alfonsi, Eric Cancès, Gabriel Turinici, Barbara di Ventura, and Wilhelm Huisinga. Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems. *ESAIM: Proc.*, 14:1–13, September 2005.
4. Rajeev Alur, Calin Belta, Franjo Ivanicic, Vijay Kumar, Max Mintz, George J. Pappas, Harvey Rubin, and Jonathan Schug. Hybrid modeling and simulation of biomolecular networks. In Springer-Verlag, editor, *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01*, volume 2034 of *Lecture Notes in Computer Science*, pages 19–32, Rome, Italy, 2001.
5. Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
6. Alexander Bockmayr and Arnaud Courtois. Using hybrid concurrent constraint programming to model dynamic biological systems. In Springer-Verlag, editor, *Proceedings of ICLP'02, International Conference on Logic Programming*, volume 2401 of *Lecture Notes in Computer Science*, pages 85–99, Copenhagen, 2002.
7. Laurence Calzone, François Fages, and Sylvain Soliman. BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.
8. Claudine Chaouiya, Elisabeth Remy, and Denis Thieffry. Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms*, 6(2):165–177, June 2008.
9. Magnus Egerstedt and Bud Mishra, editors. *HSSC 08 Proceedings of the 11th International Workshop Hybrid Systems: Computation and control*, volume 4981 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008.
10. François Fages, Steven Gay, Dragana Jovanovska, Aurélien Rizk, and Sylvain Soliman. *BIOCHAM v3.4 Reference Manual*. INRIA, 2012.
11. François Fages and Sylvain Soliman. Abstract interpretation and types for systems biology. *Theoretical Computer Science*, 403(1):52–70, 2008.
12. Martin Feinberg. Mathematical aspects of mass action kinetics. In L. Lapidus and N. R. Amundson, editors, *Chemical Reactor Theory: A Review*, chapter 1, pages 1–78. Prentice-Hall, 1977.
13. Ronojoy Ghosh and Claire Tomlin. Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. In Springer-Verlag, editor, *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01*, volume 2034 of *Lecture Notes in Computer Science*, pages 232–246, Rome, Italy, 2001.
14. David Gilbert, Monika Heiner, and Sebastian Lehrack. A unifying framework for modelling and analysing biochemical pathways using petri nets. In Muffy Calder and Stephen Gilmore, editors, *CMSB'07: Proceedings of the fifth international conference on Computational Methods in Systems Biology*, volume 4695 of *Lecture Notes in Computer Science*, Edinburgh, Scotland, September 2007. Springer-Verlag.
15. Daniel T. Gillespie. General method for numerically simulating stochastic time evolution of coupled chemical-reactions. *Journal of Computational Physics*, 22:403–434, 1976.
16. Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

17. Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, July 2001.
18. Daniel T. Gillespie. Deterministic limit of stochastic chemical kinetics. *The Journal of Physical Chemistry B*, 113(6):1640–1644, 2009.
19. Andreas Hellander and Per Lotstedt. Hybrid method for the chemical master equation. *Journal of Computational Physics*, 227(1):100–122, 2007.
20. Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, pages 278–292. IEEE Computer Society Press, 1996. An extended version appeared in Verification of Digital and Hybrid Systems.
21. Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. In *CAV'97: Proceedings of the 9th International Conference on Computer Aided Verification*, pages 460–463. Springer-Verlag, 1997.
22. Thomas A. Henzinger, Linar Mikeev, Maria Mateescu, and Verena Wolf. Hybrid numerical solution of the chemical master equation. In *Proceedings of the 8th International Conference on Computational Methods in Systems Biology*, CMSB '10, pages 55–65, New York, NY, USA, 2010. ACM.
23. Michael Hucka et al. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.
24. Trey Ideker, Timothy Galitski, and Leroy Hood. A new approach to decoding life: Systems biology. *Annual Review of Genomics and Human Genetics*, 2:343–372, 2001.
25. Minoru Kanehisa and Susumu Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.
26. Thomas R. Kiehl, Robert M. Mattheyses, , and Melvin K. Simmons. Hybrid simulation of cellular behavior. *Bioinformatics*, 20(3):316–322, 2004.
27. M. Kwiatkowska, G. Norman, and D. Parker. Using probabilistic model checking in systems biology. *SIGMETRICS Performance Evaluation Review*, 35(4):14–21, 2008.
28. O. Maler and G. Batt. Approximating continuous systems by timed automata. In J. Fisher, editor, *First International Workshop on Formal Methods in Systems Biology, FMSB'08*, volume 5054 of *Lecture Notes in Bioinformatics*, pages 77–89. Springer-Verlag, 2008.
29. Hiroshi Matsuno, Atsushi Doi, Masao Nagasaki, and Satoru Miyano. Hybrid petri net representation of gene regulatory network. In *Proceedings of the 5th Pacific Symposium on Biocomputing*, pages 338–349, 2000.
30. Vincent Noël. *Modèles réduits et hybrides de réseaux de réactions biochimiques – Applications à la modélisation du cycle cellulaire*. PhD thesis, Université de Rennes 1, 2012.
31. Howard Salis and Yiannis N. Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *The Journal of Chemical Physics*, 122(5):54103, 2005.
32. Howard Salis, Vassilios Sotiropoulos, and Yiannis N. Kaznessis. Multiscale hy3s: Hybrid stochastic simulation for supercomputers. *BMC Bioinformatics*, 7(1):93, 2006.
33. Rajat Singania, R. Michael Sramkoski, James W. Jacooberger, and John J. Tyson. A hybrid model of mammalian cell cycle regulation. *PLOS Computational Biology*, 7(2), February 2011.