Formal Cell Biology in Biocham

François Fages and Sylvain Soliman

Projet Contraintes, INRIA Rocquencourt,
BP105, 78153 Le Chesnay Cedex, France
http://contraintes.inria.fr

Abstract. Biologists use diagrams to represent interactions between molecular species, and on the computer, diagrammatic notations are also employed in interactive maps. These diagrams are fundamentally of two types: reaction graphs and activation/inhibition graphs. In this tutorial, we study these graphs with formal methods originating from programming theory. We consider systems of biochemical reactions with kinetic expressions, as written in the Systems Biology Markup Language (SBML), and interpreted in the Biochemical Abstract Machine (Biocham) at different levels of abstraction, by either an asynchronous boolean transition system, a continuous time Markov chain, or a system of Ordinary Differential Equations over molecular concentrations. We show that under general conditions satisfied in practice, the activation/inhibition graph is independent of the precise kinetic expressions, and is computable in linear time in the number of reactions. Then we consider the formalization of the biological properties of systems, as observed in experiments, in temporal logics. We show that these logics are expressive enough to capture semi-qualitative semi-quantitative properties of the boolean and differential semantics of reaction models, and that model-checking techniques can be used to validate a model w.r.t. its temporal specification, complete it, and search for kinetic parameter values. We illustrate this modelling method with examples on the MAPK signalling cascade, and on Kohn's map of the mammalian cell cycle.

1 Introduction

Biologists use diagrams to represent interactions between molecular species, and on the computer, diagrammatic notations like the ones introduced in Kohn's map [1] are also employed in interactive maps like, for instance, MIM[1]. This type of notation encompasses two types of information : interactions (binding, complexation, protein modification, etc.) and regulations (of an interaction or of a transcription).

The Systems Biology Markup Language (SBML) [2] uses a syntax of reaction rules with kinetic expressions to define reaction models in a precise way, and more and more models are described in such a formalism, like in the `biomodels.net` repository. This type of language is well suited to describe interactions (and

[1] `http://discover.nci.nih.gov/mim/`

M. Bernardo, P. Degano, and G. Zavattaro (Eds.): SFM 2008, LNCS 5016, pp. 54–80, 2008.

in a limited manner their regulations through the notion of *modifiers*) but not directly molecule to molecule activations and inhibitions.

On the other hand, formal influence graphs for activation and inhibition have been introduced in the setting of gene regulatory networks [3] as an abstraction of complex reaction networks. These graphs completely abstract from the precise interactions, especially at post-transcriptional level, and retain only the activation and inhibition effects between genes. In these influence graphs, the existence of a positive circuit (resp. a negative circuit) has been shown to be a necessary condition for multistationarity (resp. oscillations) in different settings [4,5,6,7,8]. There are several tools providing different kinds of analyses for either reaction models or influence graphs. The only formal relationship relating the two seems to be the extraction of the second one from the Jacobian matrix derived from the first one, when equipped with precise kinetic expressions and parameter values.

In this tutorial, we first provide a syntax for denoting objects in the cell, such as molecular compounds and compartments, and for denoting their interaction and transport. We use the rule-based syntax of the biochemical abstract machine Biocham [9,10] which is similar to (and compatible with) the Systems Biology Markup Language (SBML) [2] nowadays supported by a majority of modeling tools [11,12]. Then we present the different semantics of Biocham models which correspond to different abstraction levels: namely the differential, stochastic, discrete and boolean semantics [13,14,15].

Then in section 3 we study the formal relationship between reaction models and activation/inhibition influence graphs. We show that under the general condition of strongly increasing monotonicity of the kinetic expressions, and in absence of both activation and inhibition effects from one molecule to the same target, the influence graph inferred from the stoichiometric coefficients of the reactions, called the syntactical influence graph, is equal to the influence graph defined by the signs of the coefficients of the Jacobian matrix of the differential semantics, called the differential influence graph. Under these conditions, satisfied by mass action law, Michaelis-Menten and Hill kinetics, the influence graph is thus independent of the kinetic expressions for the reactions, and is computable in linear time in the number of reactions. We show that this remarkable property applies to the transcription of Kohn's map of the mammalian cell cycle control [1] into an SBML model of approx. 800 reactions [16]. On this example, the syntactical influence graph is computed in less than a second, and our equivalence theorem shows that this influence graph remains unchanged for any standard kinetics and any parameter values. The same property of independence from the kinetic expressions holds for the influence graph inferred from the MAPK signalling model of Levchenko et al. [17]. This influence graph exhibits positive as well as negative feedbacks that are hidden in the purely directional cascade of the reaction graph [18] and were the subject of a misinterpretation in [19].

In section 4 we show how temporal logics, as introduced for circuit and program verification, can be used for formalizing the biological properties of a system, and automatically check their satisfaction in a given model by model-checking techniques. Furthermore, by turning the temporal language into a specification

language, we show how a temporal specification formalizing the biological data can be used to search for kinetic parameter values. We illustrate how these techniques may be useful to the modeler with the same example as above.

Finally we conclude on these achievements in Biocham and on their perspectives for future work.

2 Reaction Models

2.1 Syntax

Following SBML [2] and Biocham [9,10] conventions, a model of a biochemical system is formally a set of reaction rules of the form e `for` S `=>` S' where S is a set of molecules given with their stoichiometric coefficient, called a *solution*, S' is the transformed solution, and e is a kinetic expression involving the concentrations of molecules. The reaction rules represent biomolecular interactions between chemical or biochemical compounds, ranging from small molecules to proteins and genes.

The syntax of the formal objects involved and their reactions, is given by the following (simplified) grammar:

object = *molecule* | *molecule* `::` *location*
molecule = *name* | *molecule*`-`*molecule* |*molecule*`~{`*name*,...,*name*`}`
reaction = *solution* `=>` *solution* | *kinetics* `for` *solution* `=>` *solution*
solution = `_` | *object* | *number*`*`*object* | *solution* `+` *solution*

The basic object is a molecular compound. Thanks to the `::` operator, it can be given a precise location, which is simply a name denoting a (fixed) compartment, such as the nucleus, the cytoplasm, the membrane, etc. The binding operator `-` is used to represent complexations and other forms of intermolecular bindings. The alteration operator `~` makes it possible to attach to a compound a set of modifications, such as the set of phosphorylated sites of a protein. For instance, `A~{p}` denotes a phosphorylated form of the compound `A`, and `A~{p}-B` denotes its complexation with `B`.

Reaction rules transform one formal solution into another one. The following abbreviations are used: `A =[C]=> B` for the catalyzed reaction `A+C => C+B`, and `A <=> B` for the reversible reaction equivalent to the two symmetrical reactions `A => B` and `B => A`. The constant `_` represents the empty solution. It is used for instance in protein *degradation rules*, like `A => _`, and in *synthesis rules*, like `_ =[G]=> A` for the synthesis of `A` by the (activated gene) catalyst `G`. The other main rule schemas are *(de)complexation rules*, like `A + B => A-B` for the complexation of `A` and `B`, *(de)phosphorylation rules*, like `A =[B]=> A~{p}` for the phosphorylation of `A` catalyzed by the kinase `B`, and *transport rules*, like `A::nucleus => A::cytoplasm` for the transport of `A` from the nucleus to the cytoplasm.

Reactions can be given kinetic expressions. For instance,
`k*[A]*[B] for A=[B]=>A~{p}` specifies a mass action law kinetics with parameter k for the reaction. Classical kinetic expressions are the mass action law kinetics

$$k * \prod_{i=1}^{n} {x_i}^{l_i}$$

for a reaction with n reactants x_i, where l_i is the stoichiometric coefficient of x_i as a reactant, Michaelis-Menten kinetics

$$V_m * x_s/(K_m + x_s)$$

for an enzymatic reaction of the form $x_s = [x_e] => x_p$, where[2] $V_m = k * (x_e + x_e * x_s/K_m)$, and Hill's kinetics

$$V_m * {x_s}^n/({K_m}^n + {x_s}^n)$$

of which Michaelis-Menten kinetics is a special case with $n = 1$. Kinetic expressions can be written either explicitly, allowing any kinetics, or using shortcuts like `MA(k)` for a Mass Action law with parameter k, or `MM(Vm,Km)` for a Michaelian kinetics.

Example 1. The Mitogen-Activated Protein Kinase (MAPK) cascades are a well-known example of signal transduction, since they appear in many receptor-mediated signal transduction schemes. They are actively considered in pharmaceutical research, for their applications to cancer therapies. The MAPK/ERK pathway is indeed hyperactivated in 30% of all human cancer tumours [20].

The structure of a MAPK cascade is a sequence of activations of three kinases in the cytosol. The last kinase, MAPK, when activated, has an effect on different substrates in the cytosol but also on gene transcription in the nucleus.

Since this cascade has been studied a lot, mathematical models of it appear in most model repositories, like for instance that of Cellerator [21] or the SBML repository page [2], both coming from [17]. This cascade was also the first example treated by Regev, Silverman and Shapiro [22] in the pi-calculus process algebra which was an initial source of inspiration for our own work.

Our first running example in this paper is the MAPK model without scaffold of Levchenko et al. [17], transcribed in Biocham as follows:

```
declare MEK~parts_of({p1,p2}).
declare MAPK~parts_of({p1,p2}).
parameter(k1, 1).
parameter(k2, 0.4).
(MA(k1), MA(k2)) for RAF + RAFK <=> RAF-RAFK.
parameter(k3, 0.5).
parameter(k4, 0.5).
(MA(k3),MA(k4)) for RAF~{p1} + RAFPH <=> RAF~{p1}-RAFPH.
parameter(k5, 3.3).
parameter(k6, 0.42).
(MA(k5),MA(k6)) for MEK~$P + RAF~{p1} <=> MEK~$P-RAF~{p1}
```

[2] $x_e * x_s/K_m$ is the concentration of the enzyme-substrate complex, supposed constant in the Michaelian approximation and $x_e + x_e * x_s/K_m$ is thus the total amount of enzyme.

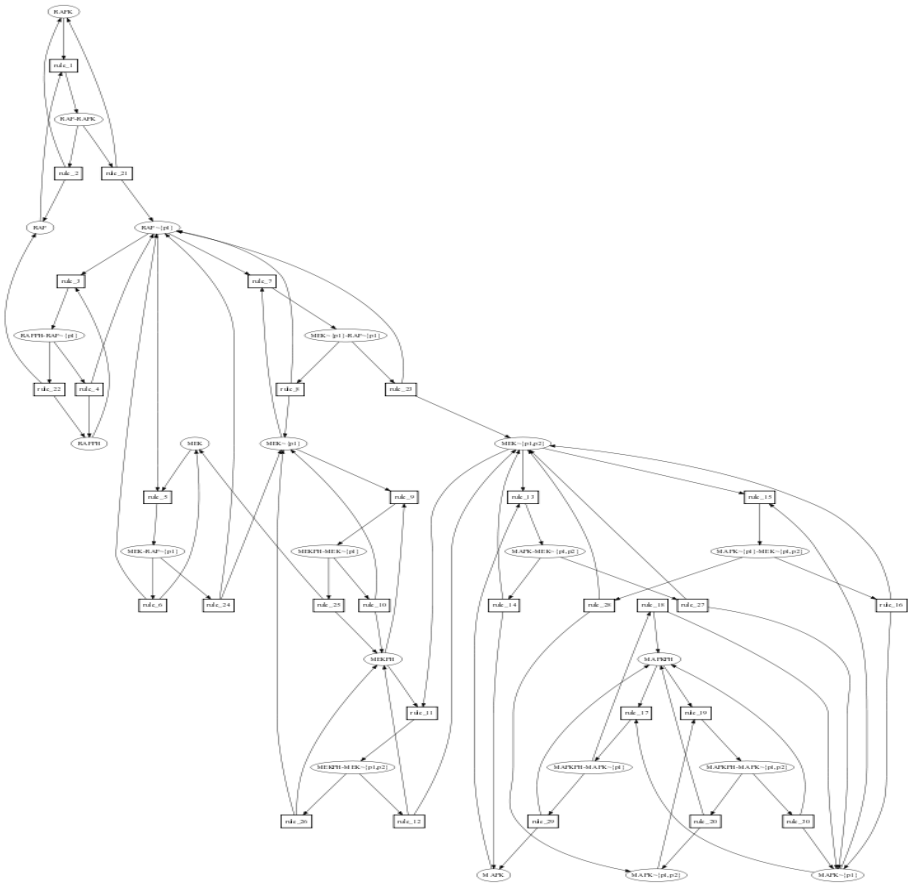

Fig. 1. Reaction (hyper)graph of the MAPK model of[17]

```
                    where p2 not in $P.
parameter(k7, 10).
parameter(k8, 0.8).
(MA(k7),MA(k8)) for MEKPH + MEK~{p1}~$P <=> MEK~{p1}~$P-MEKPH.
parameter(k9, 20).
parameter(k10, 0.7).
(MA(k9),MA(k10)) for MAPK~$P + MEK~{p1,p2} <=> MAPK~$P-MEK~{p1,p2}
                     where p2 not in $P.
parameter(k11, 5).
parameter(k12, 0.4).
(MA(k11),MA(k12)) for MAPKPH + MAPK~{p1}~$P <=> MAPK~{p1}~$P-MAPKPH.
parameter(k13, 0.1).
MA(k13) for RAF-RAFK => RAFK + RAF~{p1}.
parameter(k14, 0.1).
MA(k14) for RAF~{p1}-RAFPH => RAF + RAFPH.
```

```
parameter(k15, 0.1).
parameter(k16, 0.1).
MA(k15) for MEK~{p1}-RAF~{p1} => MEK~{p1,p2} + RAF~{p1}.
MA(k16) for MEK-RAF~{p1} => MEK~{p1} + RAF~{p1}.
parameter(k17, 0.1).
parameter(k18, 0.1).
MA(k17) for MEK~{p1}-MEKPH => MEK + MEKPH.
MA(k18) for MEK~{p1,p2}-MEKPH => MEK~{p1} + MEKPH.
parameter(k19, 0.1).
parameter(k20, 0.1).
MA(k19) for MAPK-MEK~{p1,p2} => MAPK~{p1} + MEK~{p1,p2}.
MA(k20) for MAPK~{p1}-MEK~{p1,p2} => MAPK~{p1,p2} + MEK~{p1,p2}.
parameter(k21, 0.1).
parameter(k22, 0.1).
MA(k21) for MAPK~{p1}-MAPKPH => MAPK + MAPKPH.
MA(k22) for MAPK~{p1,p2}-MAPKPH => MAPK~{p1} + MAPKPH.
present(MAPK,0.3).
present(MAPKPH,0.3).
present(MEK,0.2).
present(MEKPH,0.2).
present(RAF,0.4).
present(RAFK,0.1).
present(RAFPH,0.3).
```

For sake of simplicity, the pattern variables noted `$P` in the rules have not been described in the syntax. They represent variables bounded by the `declare` statement and that can be constrained in the `where` statement to represent rule schemas, i.e. sets of rules defined by a pattern. The last statements define the initial conditions, i.e. the concentrations of the initially present molecules, the others being set to 0. Figure 1 depicts the reaction (hyper)graph of this model, represented by a bipartite graph where molecules are in circles and reactions in boxes. □

Example 2. Our second running example in this paper will be the map of Kohn [1] for the mammalian cell cycle control. It has been transcribed in Biocham [16] to serve as a large benchmarking example of approx. 500 species and 800 rules.□

2.2 Differential Semantics

A set of reaction rules $\{e_i$ `for` S_i `=>` $S'_i\}_{i=1,...,n}$ over molecular concentration variables $\{x_1, ..., x_m\}$, can be interpreted under different semantics. The traditional *differential semantics* interpret the rules by the following system of Ordinary Differential Equations (ODE):

$$dx_k/dt = \sum_{i=1}^{n} r_i(x_k) * e_i - \sum_{j=1}^{n} l_j(x_k) * e_j$$

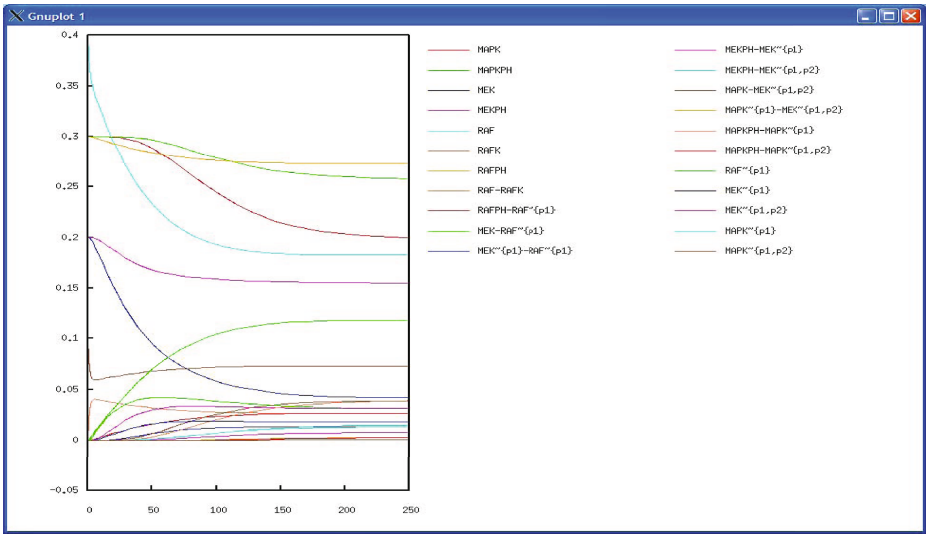

Fig. 2. Simulation result of the ODEs associated to the MAPK cascade

where $r_i(x_k)$ (resp. l_i) is the stoichiometric coefficient of x_k in the right (resp. left) member of rule i.

Example 3. Models based on ordinary differential equations (ODE) like the MAPK cascade of example 1 allow us to reproduce simulation results like the one pictured out in Figure 2, where the concentration of the visualized compounds is represented on the vertical axis, and time on the horizontal axis. It is possible to see from such simulations how the cascade evolves in time. It is possible to change input quantities to check for a significant change in the output of the cascade. Similarly, the sensitivity of the system to the values of the parameters can be checked by running different simulations with different values of the parameters, and this process can of course be automated. □

2.3 Stochastic Semantics

The most realistic interpretation of biochemical reaction models is provided by the *stochastic semantics*. In that semantics, a reaction model is interpreted as a (continuous time) Markov chain, and the kinetic expressions as transition rates. This interpretation is correct w.r.t. the Master Chemical Equation if we suppose that the reactions happen in a well stirred environment (i.e. "instantaneous" diffusion) with constant pressure, temperature and volume [23].

For a given volume V_k of the location where the compound x_k resides, a concentration C_k for x_k is translated into a molecule number $N_k = \lfloor C_k \times V_k \times N_A \rfloor$, where N_A is Avogadro's number. A state in the stochastic semantics will be a vector of integers indicating the numbers of molecules for each species.

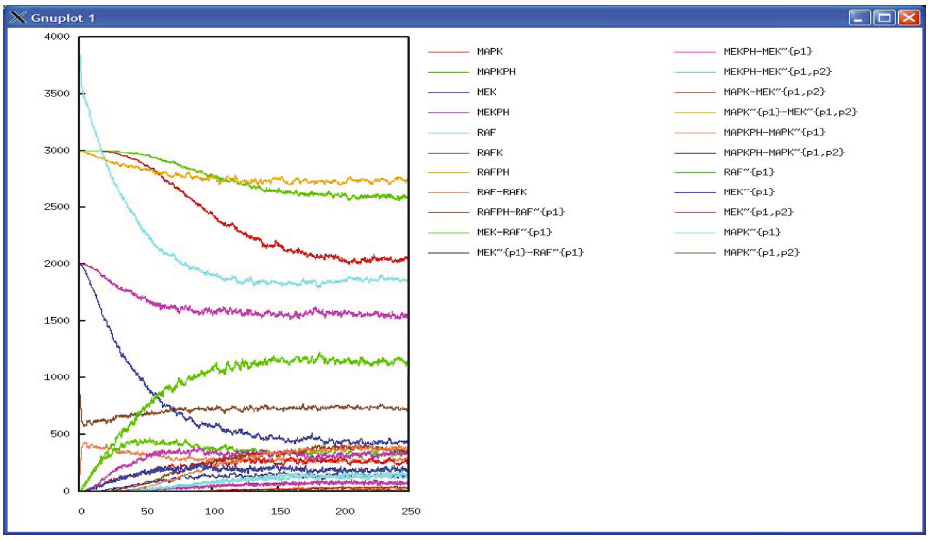

Fig. 3. Stochastic simulation of the MAPK cascade

Formally, given a fixed finite set $\mathcal{M}$ of molecule names, let a *discrete state* be a vector of positive integers of dimension $|\mathcal{M}|$. The universe $\mathcal{S}$ of *stochastic transitions* is the set of triplets (S, S', τ) where S and S' are discrete states and $\tau \in \mathbb{R}^+$ is a weight. The *stochastic transition semantics domain* is the powerset $\mathcal{D}_\mathcal{S} = \mathcal{P}(\mathcal{S})$.

Note first that discrete states have the same mathematical structure as solutions in reaction rules, and can both be represented by $|\mathcal{M}|$-dimensional vectors of positive integers. Note also that in a stochastic transition model $s \in \mathcal{D}_\mathcal{S}$, there can be more than one transition from one state to another one, labelled with different real numbers. We define the *weight* in s of a transition from state S_i to S_j as the sum of the weights $\tau_{ij} = \sum_{(S_i,S_j,\tau)\in s} \tau$.

Now, an element $s \in \mathcal{D}_\mathcal{S}$ of the domain precisely defines a Markov chain where the *probability* p_{ij} of having a transition from state S_i to state S_j is obtained by normalizing the transition weights into $p_{ij} = \frac{\tau_{ij}}{\sum_k \tau_{ik}}$. Then the transition time can be computed as usual. Stochastic simulation techniques like Gillespie's algorithm [24] compute realizations of the processes described by models in the stochastic domain, where random variables range over the probability and the time of transition.

In order to relate the stochastic semantics domain to the syntactical domain of reaction rules, let us consider a reaction rule model $\{e_i \texttt{ for } l_i\texttt{=>}r_i\}_{i\in I}$, and denote by $S \rightarrow_i S'$ the fact that *rule i fires in state S* resulting in state S', i.e. if $S \geq l_i$ (pointwise) and $S' = S - l_i + r_i$. In a given state S, the numbers of molecules are fixed integer values and the kinetic expression e_i evaluates into a (positive) real valued *reaction rate*, noted $e_i(S)$. We denote by $\alpha_{\mathcal{RS}} : \mathcal{D}_\mathcal{R} \rightarrow \mathcal{D}_\mathcal{S}$ the function that associates to a reaction model $\{e_i \texttt{ for } l_i\texttt{=>}r_i\}_{i\in I}$ the *stochastic transition model* $\{(S, S', e_i(S)) \in \mathcal{S} \mid i \in I,\ S \rightarrow_i S'\}$.

Example 4. In the example 1 of the MAPK cascade, a stochastic simulation of the model is depicted in Figure 3. As expected in this example, the realizations of this stochastic process are simply noisy versions of the differential simulation. However, in models with for instance, very few molecules of some kind, qualitatively different behaviors may appear in the stochastic simulation, and thus justify the recourse to that semantics in such cases. A classical example is the model of the lambda phage virus [25] in which a small number of molecules, promotion factors of two genes, can generate an explosive multiplication (lysis) after a more or less long period of passive wait (lysogeny). □

2.4 Asynchronous Discrete Semantics

The discrete semantics of reaction models can be defined as the trivial abstraction $\alpha_{\mathcal{SD}} : \mathcal{D}_{\mathcal{S}} \rightarrow \mathcal{D}_{\mathcal{D}}$ from the domain of stochastic transition models to the domain of discrete transition systems, that simply forgets the transition probabilities. The states, represented by integer numbers of molecules, and the transition without the weights are thus the same as in the stochastic semantics. The discrete semantics is asynchronous and non-deterministic but not probabilistic. It is worth noticing that the discrete semantics corresponds to the classical Petri net semantics of reaction models [26,27,28,29]. As a consequence, classical Petri net analysis tools can be used for the analysis of reaction models at this abstraction level. For instance, the elementary mode analysis of metabolic networks [30] has been shown in [31] to be equivalent to the classical analysis of Petri nets by T-invariants. These analyses apply to the discrete semantics of reaction models in all generality.

2.5 Asynchronous Boolean Semantics

The boolean semantics is purely qualitative, and provides somehow the most abstract semantics of reaction models. The rules are interpreted by a (non-deterministic) asynchronous transition system over boolean states representing the absence or presence of molecules. It can be applied to large models for which the kinetic data may be not available such as example 2.

Let a *boolean state* be a vector of booleans of dimension $|\mathcal{M}|$ indicating the presence of each molecule in the state. The universe $\mathcal{B}$ of *boolean transitions* is the set of pairs of boolean states which defines the domain $\mathcal{D}_{\mathcal{B}} = \mathcal{P}(\mathcal{B})$ of boolean transition models as its powerset.

The boolean semantics of a reaction model can be defined from its discrete transition semantics by the *zero/non-zero abstraction* from the integers to the booleans, and its pointwise extension from discrete states to boolean states, which provides the abstraction function $\alpha_{\mathcal{DB}} : \mathcal{D}_{\mathcal{D}} \rightarrow \mathcal{D}_{\mathcal{B}}$ from discrete models to boolean models.

In Biocham however, the boolean semantics of reaction models is computed directly from the syntax of rules, by associating to each reaction rule a set of boolean transition rules that take into account the possible complete consumption or not of the reactants by the reaction [32]. For instance, a reaction rule like `A+B=>C+D` is interpreted by four boolean transition rules :

1. $A \wedge B \longrightarrow A \wedge B \wedge C \wedge D$
2. $A \wedge B \longrightarrow \neg A \wedge B \wedge C \wedge D$
3. $A \wedge B \longrightarrow A \wedge \neg B \wedge C \wedge D$
4. $A \wedge B \longrightarrow \neg A \wedge \neg B \wedge C \wedge D$

Given a reaction model R, let us denote by S_{BB} the set of boolean transitions obtained by applying these boolean transition rules to each state. The following theorem shows that the Biocham boolean semantics of reaction models *over-approximates* the boolean semantics obtained from the quantitative semantics. The non-existence of a behaviour in the Biocham boolean semantics thus entails its non-existence in the quantitative semantics of the rules whatever the kinetic expressions are.

Theorem 1 ([13]). *For any reaction model R, $\alpha_{\mathcal{DB}}(\alpha_{\mathcal{SD}}(\alpha_{\mathcal{RS}}(R))) \subseteq S_{BB}$.*

It is worth noticing that this property does not hold for the boolean semantics of reaction models that always assume either incomplete consumption, or complete consumption, like in Pathway Logic [33] or in boolean Petri nets [29]. In these formalisms, the correctness of the boolean interpretation w.r.t. a quantitative interpretation is thus left to the modeler who is in charge of explicitly adding reaction rules for the different cases of consumption of the reactants.

Example 5. Figure 4 depicts one boolean simulation of the MAPK model of example 1. In this figure, the horizontal axis represents a logical time axis, where one reaction rule is fired at each time step. Just like the stochastic semantics, there are many possible boolean simulations, but unlike the stochastic semantics, they all have the same probability of realisation.

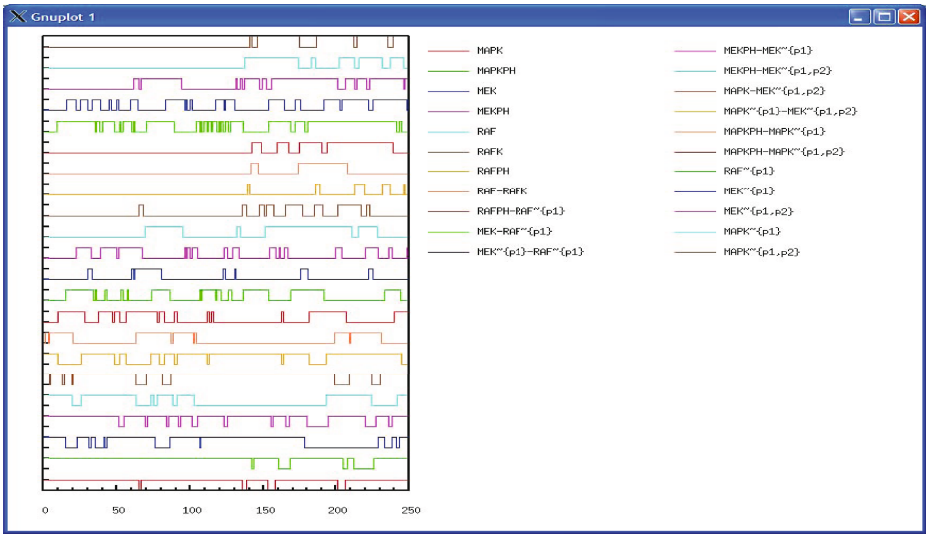

Fig. 4. Boolean simulation of the MAPK cascade

One single boolean simulation is thus not very informative, and we shall rely in section 4 on model-checking techniques to query the set of all possible boolean simulations.

In particular if one behavior is not possible in the boolean semantics, theorem 1 tells us that it is not possible to obtain such a behavior in the stochastic semantics either, whatever the kinetic expressions are. □

2.6 Hierarchy of Semantics

In [13], the different semantics of Biocham, as well as the syntactical model of reaction rules, are formally related by Galois connections in the framework of abstract interpretation [34,35,36], with the noticeable exception of the differential semantics which does not belong to this hierarchy. These results go beyond the scope of this tutorial, however they establish the formal abstraction relationships between the syntactical, stochastic, discrete and boolean interpretations of reaction rule sets ordered by inclusion. As a consequence, these abstractions can be composed and their commutation with further abstractions (such as for instance the influence graph derived from the reaction model) can be analyzed. On the other hand, the differential semantics is not compatible with the rule set inclusion ordering as the addition of kinetic terms may make them disappear in the differential equations [13].

3 Influence Graphs of Activation and Inhibition

Influence graphs for activation and inhibition have been introduced for the analysis of gene expression in the setting of gene regulatory networks [3]. Such influence graphs are in fact an abstraction of complex reaction networks, and can be applied as such to protein interaction networks. However the distinction between the influence graph and the reaction (hyper)graph is crucial to the application of Thomas's conditions of multistationarity and oscillations [3,6] to protein interaction network, and there has been some confusion between the two kinds of graphs [19].

Here we consider two definitions of the influence graph associated to a reaction model, and show their equivalence under general assumptions.

3.1 Definition from the Jacobian Matrix

In the differential semantics of a reaction rule model $M = \{e_i \texttt{ for } l_i\texttt{=>}r_i \mid i \in I\}$ we have $\dot{x_k} = dx_k/dt = \sum_{i=1}^{n}(r_i(x_k) - l_i(x_k)) * e_i$. The Jacobian matrix J is formed of the partial derivatives $J_{ij} = \partial \dot{x_i}/\partial x_j$.

Definition 1. *The* differential influence graph *associated to a reaction model is the graph having for vertices the molecular species, and for edge-set the following two kinds of edges:*

$\{A$ *activates* $B \mid \partial \dot{x_B}/\partial x_A > 0$ *in some point of the space*$\}$
$\cup\{A$ *inhibits* $B \mid \partial \dot{x_B}/\partial x_A < 0$ *in some point of the space*$\}$

3.2 Definition from the Stoichiometric Coefficients

Definition 2. *The* syntactical influence graph *associated to a reaction model M is the graph having for vertices the molecular species, and for edges the following set:*

$$\begin{aligned}
&\{A \textit{ inhibits } B \quad \mid \exists (e_i \texttt{ for } l_i \texttt{ => } r_i) \in M, \\
&\qquad\qquad\qquad\quad l_i(A) > 0 \textit{ and } r_i(B) - l_i(B) < 0\} \\
\cup&\{A \textit{ activates } B \mid \exists (e_i \texttt{ for } l_i \texttt{ => } r_i) \in M, \\
&\qquad\qquad\qquad\quad l_i(A) > 0 \textit{ and } r_i(B) - l_i(B) > 0\}
\end{aligned}$$

In particular, we have the following influences for elementary reactions of complexation, modification, synthesis and degradation:

$\alpha(\{\texttt{A} + \texttt{B} \texttt{ => } \texttt{C}\}) = \{$ A inhibits B, A inhibits A, B inhibits A, B inhibits B, A activates C, B activates C$\}$
$\alpha(\{\texttt{A} = [\texttt{C}] \texttt{ => } \texttt{B}\}) = \{$C inhibits A, A inhibits A, A activates B, C activates B$\}$
$\alpha(\{\texttt{A} = [\texttt{B}] \texttt{ => } \_\}) = \{$ B inhibits A, A inhibits A$\}$
$\alpha(\{\_ = [\texttt{B}] \texttt{ => } \texttt{A}\}) = \{$ B activates A$\}$

The inhibition loops on the reactants are justified by the negative sign in the Jacobian matrix of the differential semantics of such reactions. Unlike the differential influence graph, this graph is clearly trivial to compute by browsing the syntax of the rules:

Proposition 1. *The syntactical influence graph of a reaction model of n rules is computable in $O(n)$ time.*

Example 6. Let us consider the MAPK signalling model of [17]. Figure 1 depicts the reaction graph as a bipartite graph with round boxes for molecules and rectangular boxes for rules. Figure 5 depicts the syntactical influence graph, where activation (resp. inhibition) is materialized by plain (resp. dashed) arrows.

This computed graph reveals the negative feedbacks that are somewhat hidden in a purely directional signalling cascade of reactions. Furthermore, as no molecule is at the same time an activator and an inhibitor of a same molecule, this graph is largely independent of the kinetics of the reactions, as shown by Theorem 3 of next section. It is indeed identical to the differential influence graph for any standard kinetic expressions with any (non zero) kinetic parameter values.

These negative feedbacks, a necessary condition for oscillations [3,7,8], have been formally analyzed in [18] and interpreted as enzyme sequestration in complexes. Furthermore, oscillations in the MAPK cascade model have been shown in [37].

The influence graph also exhibits positive circuits. These are a necessary condition for multistationarity [3,6] that has been observed in the MAPK model, and experimentally in Xenopus oocytes [19]. Note that the absence of circuit in the (directional) reaction graph of MAPK was misinterpreted as a counterexample to Thomas' rule in [19] because of a confusion between both kinds of graphs. □

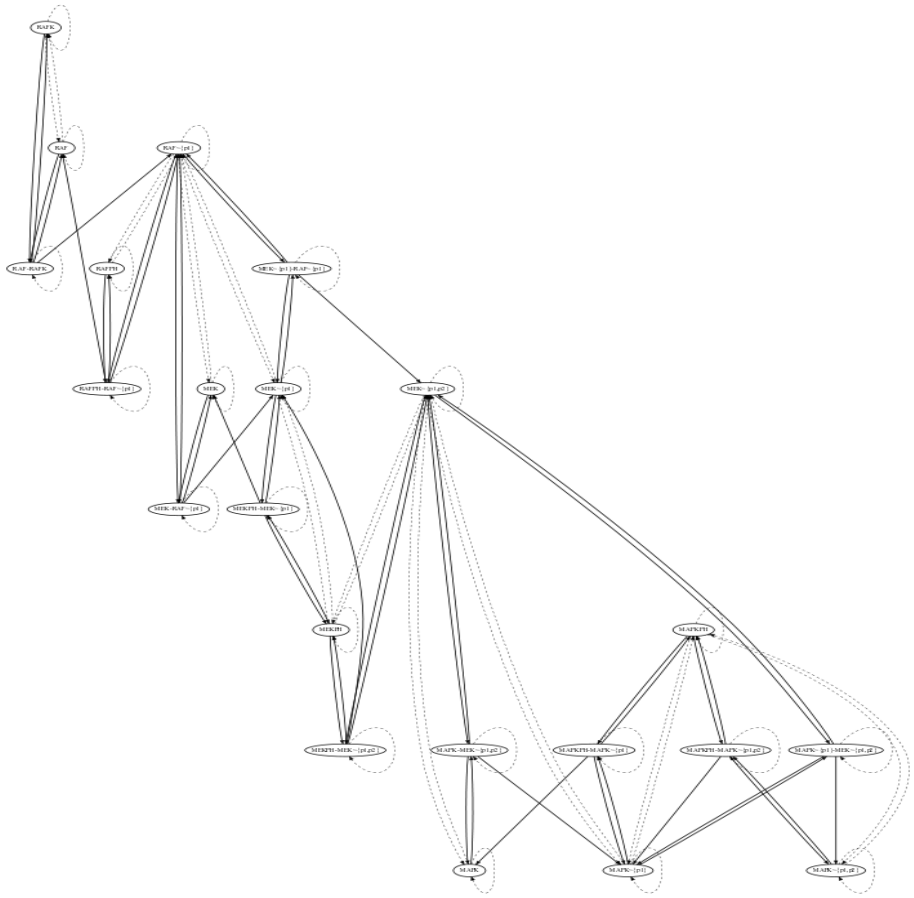

Fig. 5. Influence graph inferred from the MAPK reaction model

Example 7. On the map of Kohn, Example 2, the computation of activation and inhibition influences takes less than one second CPU time (on a PC 1,7GHz) for the complete model, showing the efficiency of the syntactical inference algorithm. The influence graph is composed of 1231 activation edges and 1089 inhibition edges.

Furthermore in this large example no molecule is both an activator and an inhibitor of the same target molecule. Theorem 3 thus entails that the computed influence graph is equal to the differential graph that would be obtained in any kinetic model of Kohn's map for any standard kinetic expressions and for any (non zero) parameter values.

Since there is a lot of kinetic data missing for such a big model, the possibility to nevertheless obtain the exact influence graph without having to estimate parameters or even to choose precise kinetic expressions is quite remarkable, and justifies the use of purely qualitative models for the analysis of feedback circuits. □

3.3 Over-approximation Theorem

Comparing the differential influence graph and the syntactical influence graph requires that the information in the kinetic expressions and in the reactions be compatible. This motivates the following definition where the first property forbids the presence of purely kinetic inhibitors not represented in the reaction, and the second property enforces that the variables appearing in the kinetic expressions do appear as reactants or enzymes in the reaction.

Definition 3. *In a reaction rule e* `for` *l=>r, we say that a kinetic expression e is* increasing *iff for all molecules x_k we have*

1. *$\partial e/\partial x_k \geq 0$ in all points of the space,*
2. *$l(x_k) > 0$ whenever $\partial e/\partial x_k > 0$ in some point of the space.*

A reaction model has an increasing kinetics *iff all its reaction rules have an increasing kinetics.*

One can easily check that:

Proposition 2. *Mass action law kinetics for any reaction, as well as Michaelis Menten and Hill kinetics for enzymatic reactions, are increasing.*

On the other hand, negative Hill kinetics of the form $k_1/(k_2+y^n)$ are not increasing. They represent an inhibition by a molecule y not belonging to the reactants, and thus not reflected in the syntax of the reaction.

Theorem 2. *For any reaction model with an increasing kinetics, the differential influence graph is a subgraph of the syntactical influence graph.*

Proof. If (A activates B) belongs to the differential influence graph then $\partial \dot{B}/\partial A > 0$. Hence there exists a term in the differential equation for B, of the form $(r_i(B) - l_i(B)) * e_i$ with $\partial e_i/\partial A$ of the same sign as $r_i(B) - l_i(B)$.

Let us suppose that $r_i(B) - l_i(B) > 0$ then $\partial e_i/\partial A > 0$ and since e_i is increasing we get that $l_i(A) > 0$ and thus that (A activates B) in the syntactical graph. If on the contrary $r_i(B) - l_i(B) < 0$ then $\partial e_i/\partial A < 0$, which is not possible for an increasing kinetics.

If (A inhibits B) is in the differential graph then $\partial \dot{B}/\partial A < 0$. Hence there exists a term in the differential semantics, of the form $(r_i(B) - l_i(B)) * e_i$ with $\partial e_i/\partial A$ of sign opposite to that of $r_i(B) - l_i(B)$.

Let us suppose that $r_i(B) - l_i(B) > 0$ then $\partial e_i/\partial A < 0$, which is not possible for an increasing kinetics. If on the contrary $r_i(B) - l_i(B) < 0$ then $\partial e_i/\partial A > 0$ and since e_i is increasing we get that $l_i(A) > 0$ and thus that (A activates B) is in the syntactical influence graph.

Corollary 1. *For any reaction model with an increasing kinetics, the differential influence graph restricted to the phase space w.r.t. some initial conditions, is a subgraph of the syntactical influence graph.*

Proof. Restricting the points of the phase space to those points that are accessible from some initial states, restricts the number of edges in the differential influence graphs which thus remains a subgraph of the syntactical influence graph.

It is worth noticing that even in the simple case of mass action law kinetics, the differential influence graph may be different from the syntactical influence graph. For instance let x be the following model :

$k_1 * A$ for $A => B$

$k_2 * A$ for $\_ = [A] => A$

In the syntactical influence graph, *A activates B, A activates A* and *A inhibits A*, however $\dot{A} = (k_2 - k_1) * A$, hence $\partial\dot{A}/\partial A$ can be made always positive or always negative or always null, resulting in the absence of respectively, *A inhibits A, A activates A* or both, in the differential influence graph.

3.4 Equivalence Theorem

Definition 4. *In a reaction rule e* `for` *l=>r, a kinetic expression e is* strongly increasing *iff for all molecules x_k we have*

1. *$\partial e/\partial x_k \geq 0$ in all points of the space,*
2. *$l(x_k) > 0$ iff there exists a point in the space s.t. $\partial e/\partial x_k > 0$*

A reaction model has a strongly increasing kinetics *iff all its reaction rules have a strongly increasing kinetics.*

Note that *strongly increasing* implies *increasing.*

Proposition 3. *Mass action law kinetics for any reaction, as well as Michaelis Menten and Hill kinetics for enzymatic reactions, are strongly increasing.*

Proof. For the case of Mass action law, the kinetics are of the form:

$$e_i = k_i * \prod_{l=1}^{m} x_l^{l_i(x_l)}$$

with $k_i > 0$ and $l_i(x_l) \geq 0$. We thus have $\partial e_i/\partial x_k = 0$ if $l_i(x_k) = 0$ and $\partial e_i/\partial x_k = k_i * l_i(x_k) * x_k^{l_i(x_k)-1} \prod_{l \neq k} x_l^{l_i(x_l)}$ otherwise, which clearly satisfies (1) and (2).

In the case of Hill kinetics (of which Michaelis Menten is a subcase), we have:

$$e_i = \frac{V_m * x_s^n}{K_m^n + x_s^n}$$

for the reaction $x_s + x_e => x_p + x_e$ and where $V_m = k_2 * x_e^{tot} = k_2 * (x_e + k_1 * x_e * x_s/(k_{-1} + k_2))$ from the steady state approximation. It is obvious that $\partial e_i/\partial x_k = 0$ for all x_k other than x_s and x_e since they do not appear in e_i and one can easily check that with all the constants n, k_1, k_{-1}, k_2 strictly positive, both $\partial e_i/\partial x_e$ and $\partial e_i/\partial x_s$ are greater than 0 at some point in the space.

Lemma 1. *Let M be a reaction model with a strongly increasing kinetics,*

Of (A activates B) is an edge in the syntactical influence graph, and not (A inhibits B), then (A activates B) belongs to the differential influence graph.

If (A inhibits B) is an edge in the syntactical influence graph, and not (A activates B), then (A inhibits B) belongs to the differential influence graph.

Proof. Since $\partial \dot{B}/\partial A = \sum_{i=1}^{n}(r_i(B) - l_i(B)) * \partial e_i/\partial A$ and all e_i are increasing we get that $\partial \dot{B}/\partial A = \sum_{\{i \leq n | l_i(A) > 0\}}(r_i(B) - l_i(B)) * \partial e_i/\partial A$.

Now if (A activates B) is in the syntactical influence graph, but not (A inhibits B), then all rules such that $l_i(A) > 0$ verify $r_i(B) - l_i(B) \geq 0$ and there is at least one rule for which the inequality is strict. We thus get that $\partial \dot{B}/\partial A$ is a sum of positive numbers, amongst which one is such that $r_i(B) - l_i(B) > 0$ and $l_i(A) > 0$ which, since M is strongly increasing, implies that there exists a point in the space for which $\partial e_i/\partial A > 0$. Hence $\partial \dot{B}/\partial A > 0$ at that point, and (A activates B) is thus in the differential influence graph.

For inhibition the same reasoning applies with the opposite sign for the $r_i(B) - l_i(B)$ and thus for the finale partial derivative.

This lemma establishes the following equivalence result:

Theorem 3. *In a reaction model with a strongly increasing kinetics and where no molecule is at the same time an activator and an inhibitor of the same target molecule, the differential and syntactical influence graphs coincide.*

This theorem shows that for standard kinetic expressions, the syntactical influences coincide with the differential influences based on the signs of the coefficients in the Jacobian matrix, when no molecule is at the same time an activator and an inhibitor of the same molecule. The theorem thus provides a linear time algorithm for computing the differential influences in these cases, simply by computing the syntactical influences. It shows also that the differential influence graph is independent of the kinetic expressions.

Corollary 2. *The differential influence graph of a reaction model of n rules with a strongly increasing kinetics is computable in time $O(n)$ if no molecule is at the same time an activator and an inhibitor.*

Corollary 3. *The differential influence graph of a reaction model is independent of the kinetic expressions as long as they are strongly increasing, if no molecule is at the same time an activator and an inhibitor.*

4 Biological Properties Formalized in Temporal Logic

Temporal logics and model-checking algorithms [38] have proved useful to respectively express biological properties of complex biochemical systems and automatically verify their satisfaction in both qualitative and quantitative models, i.e. in boolean [33,32,16], discrete [39,40], stochastic [41,42] and continuous models [14,43,32]. This approach relies on a logical paradigm for systems biology that consists in making the following identifications [44]:

biological model = transition system
biological property = temporal logic formulae
biological validation = model-checking

Having a formal language not only for describing models, i.e. transition systems by either process calculi [22,45,46,47,48], rules [33,10,9], Petri nets [26,29], etc..., but also for formalizing the biological properties of the system known from biological experiments under various conditions, opens a whole avenue of research for designing automated reasoning tools inspired from circuit and program verification to help the modeler [15].

The temporal logics CTL (*Computation Tree Logic*), LTL (*Linear Time Logic*) and PLTL (*Probabilistic LTL*) with numerical constraints are used in the three semantics of reaction models, respectively, in the boolean semantics, the differential semantics and the stochastic semantics.

4.1 Temporal Logics CTL*, CTL, LTL and PLTL

The *Computation Tree Logic* CTL* [38] is an extension of classical logic that allows reasoning about an infinite tree of state transitions. It uses operators about branches (non-deterministic choices) and time (state transitions). Two path quantifiers A and E are thus introduced to handle non-determinism: $A\phi$ meaning that ϕ is true on all branches, and $E\phi$ that it is true on at least one branch. The time operators are F, G, X, U and W; $X\phi$ meaning ϕ is true at the next transition, $G\phi$ that ϕ is always true, $F\phi$ that ϕ is eventually true, $\phi\ U\ \psi$ meaning ϕ is always true until ψ becomes true, and $\phi\ W\ \psi$ meaning ϕ is always true until ψ might become true. In this logic, $F\phi$ is equivalent to $true\ U\ \phi$, $\phi\ W\ \psi$ to $(\phi\ U\ \psi)|G\phi$. We have the following duality properties: $!(E(\phi)) = A(!\phi)$, $!(F(\phi)) = G(!\phi)$, $!(\phi\ U\ \psi) = (!\psi\ W\ !\phi)$ where ! denotes negation.

Formally, a *Kripke structure* (see for instance [38]) is a couple $K = (S, R)$ where S is a set of states in which atomic formulas can be evaluated, and $R \subseteq S \times S$ is the transition relation between states, supposed to be total (i.e. $\forall s \in S, \exists s' \in S$ s.t. $(s, s') \in R$). A path in K, starting from state s_0 is an infinite sequence of states $\pi = s_0, s_1, \cdots$ such that $(s_i, s_{i+1}) \in R$ for all $i \geq 0$. We denote by π^k the path $s_k, s_{k+1}, \cdots$. Table 4.1 recalls the inductive definition of the truth value of an LTL formula in a state s or on a path π, in a given Kripke structure K.

The computation Tree Logic CTL is the fragment of CTL* where each temporal operator must be preceded by a path operator, and each path operator has to be immediately followed by a temporal operator.

The Linear Time Logic LTL is the fragment of CTL* without path quantifiers, and where a formula is true in a Kripke structure if it is true on all paths.

The Probabilistic Computation Tree Logic PCTL quantifies the different paths by replacing the E and A modalities of CTL by probabilities.

4.2 Qualitative Biological Properties in CTL

As shown in [32], CTL is sufficiently expressive for formalizing qualitative biological properties, such as :

Table 1. Inductive definition of the truth value of a propositional CTL* formula in a state s or a path π, in a given Kripke structure K

$s \models \alpha$	iff α is a propositional formula true in state s,
$s \models E\psi$	iff there exists a path π starting from s s.t. $\pi \models \psi$,
$s \models A\psi$	iff for all paths π starting from s, $\pi \models \psi$,
$\pi \models \phi$	iff $s \models \phi$ where s is the first state of π,
$\pi \models X\psi$	iff $\pi^1 \models \psi$,
$\pi \models \psi\ U\ \psi'$	iff there exists $k \geq 0$ s.t. $\pi^k \models \psi'$ and $\pi^j \models \psi$ for all $0 \leq j < k$.
$\pi \models \psi\ W\ \psi'$	iff either for all $k \geq 0$, $\pi^k \models \psi$. or there exists $k \geq 0$ s.t. $\pi^k \models \psi \& \psi'$ and for all $0 \leq j < k$, $\pi^j \models \psi$.
$\pi \models !\psi$	iff $\pi \not\models \psi$,
$\pi \models \psi\ \&\ \psi'$	iff $\pi \models \psi$ and $\pi \models \psi'$,
$\pi \models \psi \mid \psi'$	iff $\pi \models \psi$ or $\pi \models \psi'$,
$\pi \models \psi \Rightarrow \psi'$	iff $\pi \models \psi'$ or $\pi \not\models \psi$,

- reachability where `reachable(P)` stands for $EF(P)$;
- steady states where `steady(P)` stands for $EG(P)$;
- stable states where `stable(P)` stands for $AG(P)$;
- checkpoints where `checkpoint(Q,P)` stands for $!E(!Q\ U\ P)$;
- oscillations where `oscil(P)` stands for $AG(EF\ !P \wedge EF\ P)$.
- and `loop(P,Q)` stands for $AG((P \Rightarrow EF\ Q) \wedge (Q \Rightarrow EF\ P))$.

Without strong fairness assumption, it is worth noting that the last two abbreviations are actually necessary but not sufficient conditions for oscillations. The correct formula for oscillations is indeed a CTL* formula that cannot be expressed in CTL: $EG(F\ !P \wedge F\ P)$.

In Biocham, these abbreviations can be used inside CTL formulae. For instance, the formula `reachable(steady(P))` expresses that the steady state denoted by formula P is reachable, or the formula `AG(!P -> checkpoint(Q,P))` expresses that Q is a checkpoint for P not only in the initial state but in all reachable states.

Such boolean CTL specification can also be used to complete or revise a model with machine learning algorithms. In [14], a model revision algorithm is described with the ability to not only add rules to, but also remove rules from a model in order to satisfy a CTL specification.

Example 8. In our running example of the MAPK cascade, one can use Biocham to enumerate, for instance, all simple reachability, stability, checkpoints and oscillation properties that are true in the model. This generates 112 CTL properties that can be taken as a specification. Then the model can be automatically reduced by deleting rules that do not change the satisfaction of the specification. In this model, 20 rules are left and 10 rules are deleted, essentially reverse reaction rules that do not change the specification of the cascade at this boolean abstraction level.

```
biocham: reduce_model.
1: deleting RAF-RAFK=>RAF+RAFK
2: deleting RAFPH-RAF~{p1}=>RAFPH+RAF~{p1}
3: deleting MEK-RAF~{p1}=>MEK+RAF~{p1}
4: deleting MEKPH-MEK~{p1}=>MEKPH+MEK~{p1}
5: deleting MAPK-MEK~{p1,p2}=>MAPK+MEK~{p1,p2}
6: deleting MAPKPH-MAPK~{p1}=>MAPKPH+MAPK~{p1}
7: deleting MEK~{p1}-RAF~{p1}=>MEK~{p1}+RAF~{p1}
8: deleting MEKPH-MEK~{p1,p2}=>MEKPH+MEK~{p1,p2}
9: deleting MAPK~{p1}-MEK~{p1,p2}=>MAPK~{p1}+MEK~{p1,p2}
10: deleting MAPKPH-MAPK~{p1,p2}=>MAPKPH+MAPK~{p1,p2}
```

Furthermore, temporal specifications can be used to correct a model automatically, with the model revision algorithm described in [14] for adding and removing rules in order to satisfy the temporal formulas. For instance, in the original model, if we delete one useful rule, the rule, or another model revision, can be automatically found to satisfy the specification:

```
biocham: delete_rules(RAF~{p1}+RAFPH=>RAF~{p1}-RAFPH).
RAFPH+RAF~{p1}=>RAFPH-RAF~{p1}
biocham: check_all.
The specification is not satisfied.
This formula is the first not verified: Ai(oscil(RAF))
biocham: learn_one_addition(elementary_interaction_rules).
Rules tested: 2027
Possible rules to add: 1
RAFPH+RAF~{p1}=>RAFPH-RAF~{p1}
```

In this example, the deleted rule is recovered from the temporal specification and no other choice is possible for the given pattern of elementary rules to search for. Note that this pattern generates 2027 rules to check, and that this number can be drastically reduced by integrating in the pattern type information such as protein function kinase or phosphatase to restrict the search [49,13]. □

Example 9. In example 2 of Kohn's map with 800 reaction rules over 500 molecular compounds, simple CTL properties have been model-checked in Biocham in a few seconds [16] using the symbolic model-checker NuSMV [50]. This shows the efficiency of model-checking techniques for querying all possible behaviors of a reaction model under the boolean semantics. Omissions in Kohn's map, such as for instance the absence of synthesis for cyclin B, can be immediately detected by the absence of possibility to get oscillations for cyclin B, unlike cyclin A for instance.

4.3 Quantitative Biological Properties Formalized in LTL with Constraints over the Reals

LTL with Constraints Over the Reals. A version of LTL with constraints over the reals, called Constraint-LTL, is used in Biocham [14] to express temporal properties about molecular concentrations. A similar approach is used in

the DARPA BioSpice project [43]. Constraint-LTL considers first-order atomic formulae with equality, inequality and arithmetic operators ranging over real values of concentrations and of their derivatives. For instance `F([A]>10)` expresses that the concentration of `A` eventually gets above the threshold value 10. `G([A]+[B]<[C])` expresses that the concentration of C is always greater than the sum of the concentrations of A and B. Oscillation properties, abbreviated as `oscil(M,K)`, are defined as a change of sign of the derivative of M at least K times: `F((d[M]/dt > 0) & F((d[M]/dt < 0) & F((d[M]/dt > 0)...)))` The abbreviated formula `oscil(M,K,V)` adds the constraint that the maximum concentration of M must be above the threshold V in at least K oscillations.

In this context, the Kripke structures in which the LTL formula are interpreted are linear Kripke structures which represent either an experimental data time series or a simulation trace, both completed with loops on terminal states. For instance, in a model described by a system of ordinary differential equations (ODE), and under the hypothesis that the initial state is completely defined, numerical integration methods (such as Runge-Kutta or Rosenbrock method for stiff systems) provide a discrete simulation trace. This trace constitutes a linear Kripke structure in which Constraint-LTL formulae can be interpreted. Since constraints refer not only to concentrations, but also to their derivatives, traces of the form

$$(< t_0, \boldsymbol{x}_0, d\boldsymbol{x}_0/dt, d^2\boldsymbol{x}_0/dt^2 >, < t_1, \boldsymbol{x}_1, d\boldsymbol{x}_1/dt, d^2\boldsymbol{x}_1/dt^2 >, ...)$$

are considered, where at each time point t_i, the trace associates the concentration values $\boldsymbol{x}_i$ to the variables, and the values of their first and second derivatives $d\boldsymbol{x}_i/dt$ and $d^2\boldsymbol{x}_i/dt^2$.

It is worth noting that in adaptive step size integration methods of ODE systems, the step size $t_{i+1} - t_i$ is not constant and is determined through an estimation of the error made by the discretization.

Constraint-LTL Model-Checking Algorithm. Let us assume a finite linear Kripke structure, i.e. a finite chain of states containing a loop on the last state. For these structures, the standard model-checking algorithms [38] can be easily adapted to Constraint-LTL as follows:

Algorithm 1 (Constraint-LTL model-checking). *[14,43]*

1. *label each edge with the atomic sub-formulae of ϕ that are true at this point;*
2. *add sub-formulae of the form $X\phi$ to the immediate predecessors of points labeled with ϕ;*
3. *add sub-formulae of the form $\phi_1\ U\ \phi_2$ to the points preceding a point labeled with ϕ_2 as long as ϕ_1 holds;*
4. *add sub-formulae of the form $\phi_1\ W\ \phi_2$ to the last state if it is labeled by ϕ_1, and to the predecessors of the points labeled by $\phi_1\ W\ \phi_2$ as long as ϕ_1 holds and add sub-formulae of the form $\phi_1\ W\ \phi_2$ to the points preceding a point labeled with $\phi_1 \wedge \phi_2$ as long as ϕ_1 holds;*
5. *return the edges labeled by ϕ.*

In particular, given an ODE model and a temporal property ϕ to verify within a finite time horizon, the computation of a finite simulation trace by numerical integration provides a linear Kripke structure where the terminal state is completed with a loop. Note that the notion of *next state* (operator X) refers to the state of the following time point in a discretized trace, and thus does not necessarily imply real time neighborhood. The rationale of this algorithm is that the numerical trace contains enough relevant points, and in particular those where the derivatives change abruptly, to correctly evaluate temporal logic formulae. This has been very well verified in practice with various examples of published mathematical models [14].

In [51], the model checking algorithm for constraint LTL is generalized to a constraint LTL solving algorithm with the capability to compute domains of real valued variables (such as thresholds) for which a constraint LTL formula is true. This is used for the analysis of numerical data time series in temporal logic and the automatic generation of a temporal specification of some pattern from biological experiment data time series.

Search of Kinetic Parameter Values from Constraint-LTL Properties. One can use constraint LTL model-checking to design a *generate and test* algorithm for finding parameter values such that a given LTL specification is satisfied.

A set of parameters, together with intervals of possible values and a precision parameter, are input to an enumeration algorithm. All value combinations are then scanned with a step size corresponding to the given precision, until the specification is satisfied.

Example 10. In the example 1 of the MAPK model, this parameter search algorithm can be used, for instance, to increase the overshoot for the complexation `RAF-RAFK` observed in the simulation of figure 2 as follows:

```
biocham: add_ltl(F([RAF-RAFK]>0.05)).
biocham: check_ltl.
F([RAF-RAFK]>0.05) is false.
biocham: search_parameters([k1], [(0,10)], 40, 20).
First values found that make F([RAF-RAFK]>0.05) true:
parameter(k1,1.75).
Search time: 2.96 s
```

The resulting simulation with the new value found for the complexation parameter `k1=1.75` is depicted in figure 6.

This search procedure actually replicates and automates part of what the modeler currently does by hand: trying different parameter values, between bounds that are thought reasonable, or computed by other methods such as bifurcation diagrams, in order to obtain behaviors in accordance with the experimental knowledge. Biocham provides a way to explore much faster this parameter space, once the effort for formalizing the expected behavior in LTL is done. The main novel feature of this method is its capability to express and combine in LTL both

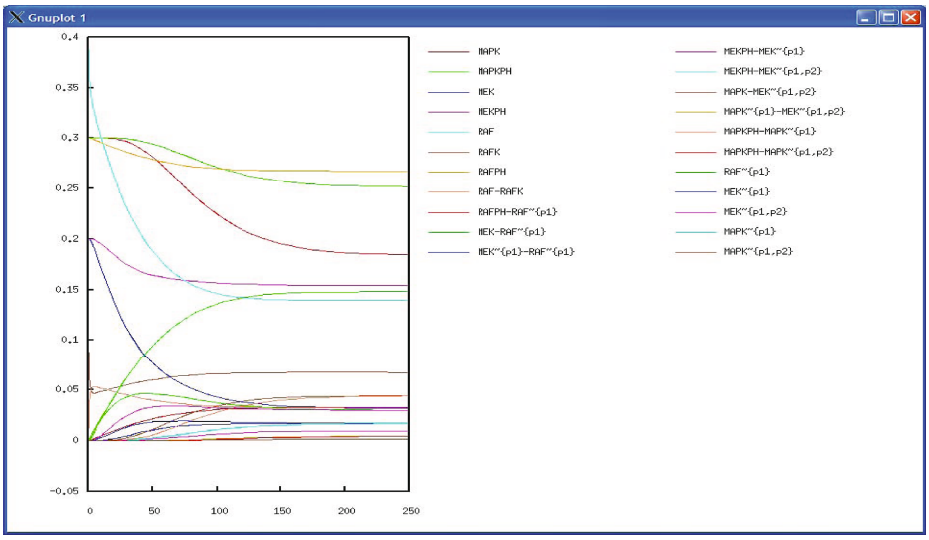

Fig. 6. Simulation result of the MAP cascade with new parameter value inferred for increasing the overshoot on `RAF-RAFK`

qualitative and quantitative constraints on the expected behavior of the model. In [52] it is used for exploring the conditions of entrainment in period of the cell cycle by the circadian cycle in a coupled model of these cycles.

The computational complexity of the parameter values scanning grows linearly in the number of combinations of parameter values to try, that is in $O(d^n)$ where n is the number of parameter values to find and d the number of values to try for each parameter. The difficulty to use other search algorithms better than generate-and-test (such as local search or simulated annealing for instance) comes from the criterion of satisfaction of LTL formulae which is naturally boolean and for which it is not obvious to define a multi-valued measure of satisfaction.

4.4 Probabilistic Model-Checking

For the stochastic semantics, it is natural to consider the PCTL logic [53] which basically replaces the path operators of CTL, E and A, by the operator $P_{\bowtie p}$. This operator represents a constraint $\bowtie_p$ on the probability that the formula under $P_{\bowtie p}$ is true. For instance, $A(\psi\ U\ \psi')$ becomes $P_{\geq 1}(\psi\ U\ \psi')$, i.e. the probability that $\psi\ U\ \psi'$ is realized is 1. The atomic formulae considered here are first-order formulae with arithmetic constraints, ranging on integers representing numbers of molecules.

However, he existing probabilistic model-checking tools, like that of PRISM [54], do not handle well highly non-deterministic examples, nor those where variables have a large domain as it is the case in BIOCHAM models' stochastic

semantics.. This led us to actually consider the PLTL fragment of PCTL formulae in which the $P_{\bowtie_p}$ operator can only appear once as head of the formula, and to use a Monte-Carlo method as done in the APMC system [55]. To evaluate the probability of realization of the underlying LTL formula, BIOCHAM samples a certain number of stochastic simulations using standard algorithms like that of Gillespie [24] or of Gibson [56]. The outer probability is then estimated by counting. It is worth noting that this method provides a real estimate of realization of the LTL formula, whereas PCTL expresses the boolean satisfaction of a probability constraint ($\bowtie_p$) over the formula.

In principle, the Monte-Carlo algorithm can thus be used for model-checking and kinetic parameter learning along the same lines as in the differential semantics and constraint-LTL. However, both the stochastic simulation process and the model-checking process are computationally more expensive than in the differential semantics by several orders of magnitude. For this reason, such search algorithms are currently not practical with the stochastic semantics.

5 Conclusion

Systems biology can benefit from formal methods originating from programming theory in many ways. By formalizing the different semantics of a reaction model, we have shown that, to a large extent, the influence graph of a reaction model is independent of the kinetic parameters and kinetic expressions, and that it can be computed in linear time simply from the syntax of the reactions. This happens for strongly increasing kinetics such as classical mass action law, Michaelis-Menten and Hill kinetics, when no molecule is at the same time an activator and an inhibitor of a same target molecule. The inference of the syntactical influence graph from a reaction model has been implemented in Biocham, and applied to various models. On a transcription of Kohn's map into approx. 800 reaction rules, this implementation shows that no molecule is at the same time an activator and an inhibitor of a same molecule, and therefore, our equivalence theorem states that the differential influence graph would be the same for any standard kinetics with any parameter values. On the MAPK signalling cascade that does not contain any feedback reaction, the implementation does reveal both positive and negative feedback circuits in the influence graph, which has been a source of confusion for the correct application of Thomas' rules. Furthermore, in this example again, no molecule is at the same time an activator and an inhibitor of another molecule, showing the independence of the influence graph from the kinetics.

By formalizing the biological properties observed in experiments, in temporal logic, we have illustrated the expressivity of these logics in this context, and we have shown that classical as well as new model-checking techniques can be applied for validating reaction models w.r.t. temporal specifications. The beauty of this approach is that it deals not only with the boolean semantics but also with the differential semantics (and stochastic semantics) of reaction models. Furthermore, such semi-qualitative semi-quantitative temporal specifications can be

also used for searching parameter values, in a complementary fashion to classical mathematical methods such as bifurcation diagrams. The improvement of this method by the definition of a measure of satisfaction of a temporal formula with constraints, and a gradient descent analog, are currently under investigation.

Acknowledgements. The material presented in this tutorial has been developed with colleagues since 2001. We are especially grateful to Laurence Calzone, Nathalie Chabrier-Rivier, Aurélien Rizk, and all those who have contributed to the development of Biocham at one stage or another. The first author thanks his successive students of the University of Paris for their reactions to his lectures on this topic. This work benefited from support of the ARC CPBIO (02-04) and ARC MOCA (05-07) `http://contraintes.inria.fr/cpbio` and `moca`, INRA Agrobi Insight (07-09) `http://contraintes.inria.fr/~heitzler/INSIGHT/Home.html`, European Union FP6 Strep projects APrIL2 (04-07) `http://www.aprill.org/`, TEMPO (06-09) `http://www.chrono-tempo.org`, and network of excellence REWERSE (04-08) `http://www.rewerse.net`.

References

1. Kohn, K.W.: Molecular interaction map of the mammalian cell cycle control and DNA repair systems. Molecular Biology of the Cell 10, 2703–2734 (1999)
2. Hucka, M., et al.: The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. Bioinformatics 19, 524–531 (2003)
3. Thomas, R., Gathoye, A.M., Lambert, L.: A complex control circuit: regulation of immunity in temperate bacteriophages. European Journal of Biochemistry 71, 211–227 (1976)
4. Kaufman, M., Soulé, C., Thomas, R.: A new necessary condition on interaction graphs for multistationarity. Journal of Theoretical Biology 248, 675–685 (2007)
5. Soulé, C.: Mathematical approaches to differentiation and gene regulation. C.R. Biologies 329, 13–20 (2006)
6. Soulé, C.: Graphic requirements for multistationarity. ComplexUs 1, 123–133 (2003)
7. Snoussi, E.: Necessary conditions for multistationarity and stable periodicity. J. Biol. Syst. 6, 3–9 (1998)
8. Gouzé, J.L.: Positive and negative circuits in dynamical systems. J. Biol. Syst. 6, 11–15 (1998)
9. Calzone, L., Fages, F., Soliman, S.: BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. BioInformatics 22, 1805–1807 (2006)
10. Fages, F., Soliman, S., Chabrier-Rivier, N.: Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. Journal of Biological Physics and Chemistry 4, 64–73 (2004)
11. Hlavacek, W.S., Faeder, J.R., Blinov, M.L., Posner, R.G., Hucka, M., Fontana, W.: Rules for modeling signal-transduction systems. Science STKE 344, 6 (2006)
12. Soliman, S., Fages, F.: CMBSlib: a library for comparing formalisms and models of biological systems. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 231–235. Springer, Heidelberg (2005)

13. Fages, F., Soliman, S.: Abstract interpretation and types for systems biology. In: Theoretical Computer Science (to appear, 2008)
14. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 68–94. Springer, Heidelberg (2006)
15. Fages, F.: From syntax to semantics in systems biology - towards automated reasoning tools. Transactions on Computational Systems Biology IV 3939, 68–70 (2006)
16. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biochemical interaction networks. Theoretical Computer Science 325, 25–44 (2004)
17. Levchenko, A., Bruck, J., Sternberg, P.W.: Scaffold proteins biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. PNAS 97, 5818–5823 (2000)
18. Ventura, A.C., Sepulchre, J.A., Merajver, S.D.: A hidden feedback in signaling cascades is revealed. In: PLoS Computational Biology (to appear, 2008)
19. Markevich, N.I., Hoek, J.B., Kholodenko, B.N.: Signaling switches and bistability arising from multisite phosphorylation in protein kinase cascades. Journal of Cell Biology 164, 353–359 (2005)
20. Kolch, W., Kotwaliwale, A., Vass, K., Janosch, P.: The role of raf kinases in malignant transformation. In: Expert Reviews in Molecular Medicine, vol. 25, Cambridge University Press, Cambridge (2002)
21. Shapiro, B.E., Levchenko, A., Meyerowitz, E.M., Wold, B.J., Mjolsness, E.D.: Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. Bioinformatics 19, 677–678 (2003)
22. Regev, A., Silverman, W., Shapiro, E.Y.: Representation and simulation of biochemical processes using the pi-calculus process algebra. In: Proceedings of the sixth Pacific Symposium of Biocomputing, pp. 459–470 (2001)
23. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry 81, 2340–2361 (1977)
24. Gillespie, D.T.: General method for numerically simulating stochastic time evolution of coupled chemical-reactions. Journal of Computational Physics 22, 403–434 (1976)
25. Gibson, M.A., Bruck, J.: A probabilistic model of a prokaryotic gene and its regulation. In: Bolouri, H., Bower, J. (eds.) Computational Methods in Molecular Biology: From Genotype to Phenotype, MIT press, Cambridge (2000)
26. Reddy, V.N., Mavrovouniotis, M.L., Liebman, M.N.: Petri net representations in metabolic pathways. In: Hunter, L., Searls, D.B., Shavlik, J.W. (eds.) Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB, pp. 328–336. AAAI Press, Menlo Park (1993)
27. Sackmann, A., Heiner, M., Koch, I.: Application of petri net based analysis techniques to signal transduction pathways. BMC Bioinformatics 7 (2006)
28. Chaouiya, C.: Petri net modelling of biological networks. Briefings in Bioinformatics (2007)
29. Gilbert, D., Heiner, M., Lehrack, S.: A unifying framework for modelling and analysing biochemical pathways using petri nets. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, Springer, Heidelberg (2007)
30. Schuster, S., Pfeiffer, T., Moldenhauer, F., et al.: Exploring the pathway structure of metabolism: decomposition into subnetworks and application to mycoplasma pneumoniae. Bioinformatics 18, 51–61 (2002)

31. Zevedei-Oancea, I., Schuster, S.: Topological analysis of metabolic networks based on petri net theory. Silico Biology 3 (2003)
32. Chabrier, N., Fages, F.: Symbolic model cheking of biochemical networks. In: Priami, C. (ed.) CMSB 2003. LNCS, vol. 2602, pp. 149–162. Springer, Heidelberg (2003)
33. Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., Sönmez, M.K.: Pathway logic: Symbolic analysis of biological signaling. In: Proceedings of the seventh Pacific Symposium on Biocomputing, pp. 400–412 (2002)
34. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL 1977: Proceedings of the 6th ACM Symposium on Principles of Programming Languages, Los Angeles, pp. 238–252. ACM Press, New York (1977)
35. Cousot, P.: Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. Theoretical Computer Science 277, 47–103 (2002)
36. Cousot, P.: Types as abstract interpretation. In: POP 1997: Proceedings of the 24th ACM Symposium on Principles of Programming Languages, pp. 316–331. ACM Press, New York (1997)
37. Qiao, L., Nachbar, R.B., Kevrekidis, I.G., Shvartsman, S.Y.: Bistability and oscillations in the huang-ferrell model of mapk signaling. PLoS Computational Biology 3, 1819–1826 (2007)
38. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)
39. Bernot, G., Comet, J.P., Richard, A., Guespin, J.: A fruitful application of formal methods to biological regulatory networks: Extending thomas' asynchronous logical approach with temporal logic. Journal of Theoretical Biology 229, 339–347 (2004)
40. Batt, G., Bergamini, D., de Jong, H., Garavel, H., Mateescu, R.: Model checking genetic regulatory networks using gna and cadp. In: Graf, S., Mounier, L. (eds.) SPIN 2004. LNCS, vol. 2989, Springer, Heidelberg (2004)
41. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signalling pathways using the continuous time markow chains. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 44–67. Springer, Heidelberg (2006)
42. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. In: Priami, C. (ed.) CMSB 2006. LNCS (LNBI), vol. 4210, pp. 32–47. Springer, Heidelberg (2006)
43. Antoniotti, M., Policriti, A., Ugel, N., Mishra, B.: Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics 38, 271–286 (2003)
44. Fages, F.: Temporal logic constraints in the biochemical abstract machine biocham (invited talk). In: Hill, P.M. (ed.) LOPSTR 2005. LNCS, vol. 3901, Springer, Heidelberg (2006)
45. Cardelli, L.: Brane calculi - interactions of biological membranes. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 257–280. Springer, Heidelberg (2005)
46. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. Theoretical Computer Science 325, 141–167 (2004)
47. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science 325, 69–110 (2004)
48. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. Transactions on Computational Systems Biology Special issue of BioConcur (to appear, 2004)

49. Fages, F., Soliman, S.: Type inference in systems biology. In: Priami, C. (ed.) CMSB 2006. LNCS (LNBI), vol. 4210, Springer, Heidelberg (2006)
50. Cimatti, A., Clarke, E., Enrico Giunchiglia, F.G., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: Nusmv 2: An opensource tool for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, Springer, Heidelberg (2002)
51. Fages, F., Rizk, A.: On the analysis of numerical data time series in temporal logic. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 48–63. Springer, Heidelberg (2007)
52. Fages, F., Soliman, S.: Model revision from temporal logic properties in systems biology. In: Probabilistic Inductive Logic Programming. LNCS, vol. 4911, pp. 287–304. Springer, Heidelberg (2008)
53. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 512–535 (1994)
54. Kwiatkowska, M.Z., Norman, G., Parker, D.: Prism 2.0: A tool for probabilistic model checking. In: st International Conference on Quantitative Evaluation of Systems (QEST 2004), pp. 322–323. IEEE Computer Society, Los Alamitos (2004)
55. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 73–84. Springer, Heidelberg (2004)
56. Gibson, M.A., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. Journal of Physical Chemistry 104, 1876–1889 (2000)