Model Revision from Temporal Logic Properties in Computational Systems Biology

François Fages and Sylvain Soliman

INRIA Rocquencourt, France
Francois.Fages@inria.fr, Sylvain.Soliman@inria.fr

Abstract. Systems biologists build models of bio-molecular processes from knowledge acquired both at the gene and protein levels, and at the phenotype level through experiments done in wild-life and mutated organisms. In this chapter, we present qualitative and quantitative logic learning tools, and illustrate how they can be useful to the modeler. We focus on biochemical reaction models written in the Systems Biology Markup Language SBML, and interpreted in the Biochemical Abstract Machine BIOCHAM. We first present a model revision algorithm for inferring reaction rules from biological properties expressed in temporal logic. Then we discuss the representations of kinetic models with ordinary differential equations (ODEs) and with stochastic logic programs (SLPs), and describe a parameter search algorithm for finding parameter values satisfying quantitative temporal properties. These methods are illustrated by a simple model of the cell cycle control, and by an application to the modelling of the conditions of synchronization in period of the cell cycle by the circadian cycle.

1 Introduction

One promise of computational systems biology is to model biochemical processes at a sufficiently large scale so that complex system behaviors can be predicted under various conditions. The biochemical reaction systems involved in these processes may contain many cycles and exhibit complex multistationarity and oscillating behaviors. While usually neglected in metabolic networks, these characteristics are preponderant in models of signal transduction and cell control. They thus provide a challenge to representation and inference methods, and the issue of representing complex biochemical systems and their behavior at different levels of abstraction is a central one in systems biology.

The pioneering use in [1] of the π-calculus process algebra for modeling cell signalling pathways, has been the source of inspiration of numerous works in the line of process calculi [2,3,4] and their stochastic extensions [5]. Recently, the question of formalizing the biological properties of the system has also been raised, and formal languages have been proposed for this task, most notably using temporal logics in either boolean [6,7], discrete [8,9,10] or continuous models [11,12].

L. De Raedt et al. (Eds.): Probabilistic ILP 2007, LNAI 4911, pp. 287–304, 2008.

The biochemical abstract machine BIOCHAM[1] [13,14] has been designed as a simplification of the process calculi approach using a logic programming setting and a language of reaction rules compatible with the Systems Biology Markup Language SBML [15] (`http://www.sbml.org/`). This opens up the whole domain of mathematical biology, through repositories like BioModels.net (`http://www.biomodels.net`), CMBSlib (`http://contraintes.inria.fr/CMBSlib/`), PWS (`http://jjj.biochem.sun.ac.za/`), etc. This rule-based language is used in BIOCHAM for modeling biochemical networks at three abstraction levels:

- The boolean semantics, where one reasons on the presence/absence of molecules,
- The differential semantics, where one reasons on molecular concentrations,
- The stochastic semantics, where one reasons on molecule numbers and reaction probabilities.

A second language is used to formalize the biological properties known from experiments in temporal logic (the *Computation Tree Logic* CTL, *Linear Time Logic* LTL or Probabilistic LTL with constraints, according to the qualitative, quantitative or stochastic nature of the properties). Such a formalization is a first step toward the use of logic learning tools to help the modeler in his tasks [16]. When a model does not satisfy all the expected properties, the purpose of the machine learning system of BIOCHAM is to propose rules or kinetic parameter values in order to curate the model w.r.t. a given specification [12]. This novel approach to biological modeling has been applied to a data set of models about the cell cycle control in different organisms, and signal transduction network (see `http://contraintes.inria.fr/APrIL2/`).

There has been work on the use of machine learning techniques, such as inductive logic programming (ILP, see Chapter 1 or [17]), to infer gene functions [18], metabolic pathway descriptions [19,20] or gene interactions [8]. However learning biochemical reactions from temporal properties is quite new, both from the machine learning perspective and from the systems biology perspective. A precursor system of this type was the system KARDIO used in drug target discovery [21]. The novelty in our approach is the use of the temporal logic setting to express semi-qualitative semi-quantitative properties of the behaviour of the system to be captured by the model.

In the following, we present successively:

- The boolean semantics of reaction models in Datalog, the representation of biological properties in temporal logic CTL, the application of ILP and model revision from temporal properties,
- The representation of kinetic models, of quantitative properties in temporal logic LTL with constraints, and a parameter search algorithm,
- The evaluation on an application: the modelling of the synchronization in period of the cell cycle by the circadian cycle.

[1] BIOCHAM is available for download at `http://contraintes.inria.fr/BIOCHAM`

2 Reaction Rule Learning from Temporal Properties

2.1 Biochemical Reaction Models in Datalog

From a syntactical point of view, SBML and BIOCHAM models basically consists in a set of reaction rules between molecules, protein complexes and modified proteins such as by phosphorylation. Each reaction rule for synthesis, degradation, complexation, phosphorylation, etc. can be given with a kinetic expression.

Example 1. Here is for instance a simple model of the cell cycle control after Tyson (1991). Each rule is given here with an arithmetic expression (its rate) followed by the keyword `for` and then a list of reactants separated by `+` on the left side of the reaction arrow `=>` and a list of products on the right side. The notation `_` represents the empty list.

```
k1                          for _=>Cyclin.
k2*[Cyclin]                 for Cyclin=>_.
k3*[Cyclin]*[Cdc2~{p1}]     for Cyclin+Cdc2~{p1}=> Cdc2~{p1}-Cyclin~{p1}.
k4p*[Cdc2~{p1}-Cyclin~{p1}] for Cdc2~{p1}-Cyclin~{p1}=> Cdc2-Cyclin~{p1}.
k4*([Cdc2-Cyclin~{p1}])^2*[Cdc2~{p1}-Cyclin~{p1}]
         for Cdc2~{p1}-Cyclin~{p1}=[Cdc2-Cyclin~{p1}]=> Cdc2-Cyclin~{p1}.
k5*[Cdc2-Cyclin~{p1}]       for Cdc2-Cyclin~{p1}=> Cdc2~{p1}-Cyclin~{p1}.
k6*[Cdc2-Cyclin~{p1}]       for Cdc2-Cyclin~{p1}=> Cyclin~{p1}+Cdc2.
k7*[Cyclin~{p1}]            for Cyclin~{p1}=>_.
k8*[Cdc2]                   for Cdc2=> Cdc2~{p1}.
k9*[Cdc2~{p1}]              for Cdc2~{p1}=> Cdc2.
```

The first rule represents the synthesis of a cyclin with a constant rate $k1$. The second rule represents the degradation of the cyclin with a reaction rate proportional to the cyclin concentration. The third rule represents the phosphorylation of the cyclin when it gets complexed with the kinase `Cdc2~{p1}`. The fourth rule is an autocatalyzed dephosphorylation of the complex, etc. For a more complete account of BIOCHAM syntax see for instance [12].

From a semantical point of view, reaction rules can be interpreted under different semantics corresponding to different abstraction levels. The most abstract semantics of BIOCHAM rules is the boolean semantics that associates to each molecule a boolean variable representing its presence or absence in the system, and ignores the kinetic expressions. Reaction rules are then interpreted as an *asynchronous transition system*[2] over states defined by the vector of boolean variables. A rule such as `A+B=>C+D` defines four possible transitions corresponding to the complete or incomplete consumption of the reactants `A` and `B`. Such a rule can only be applied when both `A` and `B` are present in the current state. In the next state, `C` and `D` are then present, while `A` and `B` can either be present (partial consumption) or absent (complete consumption).

[2] In this context asynchronous refers to the fact that only one transition is fired at a time, even if several are possible. This choice is justified by the fundamental biochemical phenomena of competition and masking between reaction rules.

The boolean semantics can be straightforwardly represented in Datalog. We use Prolog here for convenience. A state is represented by a Prolog term `state(mol1,...,molN)` where the molecule variable `mol` is 0 if absent, 1 if present, and a variable `_` if it can take any value. Transitions are represented by facts `transition(predecessor_state, successor_state)` with variables linking successor and predecessor values.

Example 2. The boolean semantics of the previous cell cycle model can be represented in Prolog as follows:

```
dimension(6).
names('Cyclin','Cdc2~{p1}','Cdc2-Cyclin~{p1,p2}',
      'Cdc2-Cyclin~{p1}','Cdc2','Cyclin~{p1}').
transition(state(_,A,B,C,D,E),state(1,A,B,C,D,E)).
transition(state(1,A,B,C,D,E),state(_,A,B,C,D,E)).
transition(state(1,1,_,A,B,C),state(_,_,1,A,B,C)).
transition(state(A,B,1,_,C,D),state(A,B,_,1,C,D)).
transition(state(A,B,1,1,C,D),state(A,B,_,1,C,D)).
transition(state(A,B,_,1,C,D),state(A,B,1,_,C,D)).
transition(state(A,B,C,1,_,_),state(A,B,C,_,1,1)).
transition(state(A,B,C,D,E,1),state(A,B,C,D,E,_)).
transition(state(A,_,B,C,1,D),state(A,1,B,C,_,D)).
transition(state(A,1,B,C,_,D),state(A,_,B,C,1,D)).
```

Formally, the boolean semantics of a reaction model is a *Kripke structure* (see for instance [22]) $K = (S, R)$ where S is the set of states defined by the vector of boolean variables, 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 $s_k, s_{k+1}, \cdots$.

2.2 Biological Properties in Temporal Logic CTL

In the boolean semantics of reaction models, the biological properties of interest are *reachability properties*, i.e. whether a particular protein can be produced from an initial state; *checkpoints*, i.e. whether a particular protein or state is compulsory to reach another state; *stability*, i.e. whether the system can (or will) always verify some property; etc.

Such properties can be expressed in the *Computation Tree Logic* CTL* [22] that is an extension of propositional logic for reasoning about an infinite tree of state transitions. CTL* uses operators about branches (non-deterministic choices) and time (state transitions). Two path quantifiers A and E are 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 either always true or until and when ψ becomes true. Table 1 recalls the truth value of a formula in a given Kripke structure.

Table 1. Inductive definition of the truth value of a 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 the 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, $\pi \models \psi$,
$s \models !\psi$	iff $s \not\models \psi$,
$s \models \psi \ \& \ \psi'$	iff $s \models \psi$ and $s \models \psi'$,
$s \models \psi \mid \psi'$	iff $s \models \psi$ or $s \models \psi'$,
$s \models \psi \Rightarrow \psi'$	iff $s \models \psi'$ or $s \not\models \psi$,
$\pi \models \phi$	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 $\pi \not\models \psi$,

In this logic, $F\phi$ is equivalent to $true\ U\ \phi$, $G\phi$ to $\phi\ W\ false$, and the following duality properties hold: $!(E\phi) = A(!\phi)$, $!(X\phi) = X(!\phi)$, $!(F\phi) = G(!\phi)$, $!(\phi\ U\ \psi) = !\psi\ W\ !\phi$ and $!(\phi\ W\ \psi) = !\psi\ U\ !\phi$, where ! denotes negation. The following abbreviation are used in BIOCHAM:

- `reachable(P)` stands for $EF(P)$;
- `steady(P)` stands for $EG(P)$;
- `stable(P)` stands for $AG(P)$;
- `checkpoint(Q,P)` stands for $!E(!Q\ U\ P)$;
- `oscillates(P)` stands for $EG((F\ !P) \wedge (F\ P))$.

These temporal properties can be checked in the Prolog representation of reaction rules, by using a symbolic model-checker written in Prolog. The BIOCHAM model checker in Prolog proceeds by computing both backward and forward frontiers of states, starting from the initial states (resp. the goal states) leading to a goal state (resp. an initial state). These sets of states are represented by Prolog facts with variables. Their cardinalities are reduced by subsumption checks in this representation. In its simplest form, the forward reachability analysis proceeds by computing the transitive closure of the transition relation, starting from the initial state, up to the reaching of a state in the query. The simplest case in such a model checker is thus a standard transitive closure algorithm in Prolog.

For performance reasons in large reaction models however, the symbolic model checker NuSMV [23] based on ordered binary decision diagram (OBDD) is preferred and is used by default in BIOCHAM, through an interface. NuSMV is restricted to the fragment CTL of CTL* in which each time operator must be immediately preceded by a path quantifier. This restriction causes a difficulty

for the oscillation properties only, since they cannot be expressed in CTL. In CTL, oscillation properties are thus approximated by the necessary but not sufficient formula $EG((EF\ !P) \wedge (EF\ P))$. We refer to [7,24] for the expressivity and scalability of this approach in reaction models containing several hundreds of variables and rules.

2.3 Model Revision from Temporal Properties

Having the model and the properties defined by a Prolog program, ILP techniques can in principle be used for learning reaction rules from temporal properties, i.e. structure learning of the underlying logic program (see Chapter 1). Here the positive and negative examples are uniformly given as a list of temporal properties to satisfy (expressed in a language closed by negation), instead of by positive and negative *facts*. Because of the relative complexity of the model checker in Prolog, this approach is currently limited to reachability properties. For learning from more general temporal properties, the NuSMV model checker is used in BIOCHAM as a black box, within an enumeration algorithm of all possible rule instances of some given rule pattern.

Furthermore, in the general framework of model revision, one wants to discover deletions as well as additions of reaction rules (of some pattern given as a bias) in order to satisfy a set of CTL formulas given as positive and negative examples. CTL properties can be classified into ECTL and ACTL formulas (i.e. formulas containing only E or A path quantifiers respectively) in order to anticipate whether reaction rules need be added or deleted. Indeed if an ECTL (resp. ACTL) formula is false in a Kripke structure, it remains false in a Kripke structure with less (resp. more) transitions. We refer to [12] for the details of the model revision algorithm implemented in BIOCHAM along these lines.

We show here our results on the model of example 1. For the structure learning phase, some CTL formulae are entered as a specification, expressing here reachability, oscillation and checkpoint properties:

```
add_specs({
       reachable(Cdc2~{p1}),
       reachable(Cdc2),
       reachable(Cyclin),
       reachable(Cyclin~{p1}),
       reachable(Cdc2-Cyclin~{p1}),
       reachable(Cdc2~{p1}-Cyclin~{p1})}).

add_specs({
       oscil(Cdc2,
       oscil(Cdc2~{p1})),
       oscil(Cdc2~{p1}-Cyclin~{p1}),
       oscil(Cdc2-Cyclin~{p1}),
       oscil(Cyclin),
           checkpoint(Cdc2~{p1}-Cyclin~{p1}, Cdc2-Cyclin~{p1})}).
```

These properties are satisfied by the model and can be automatically checked by the model-checker. The simplest example to illustrate the structural learning

method is to delete one rule in the model and let the learning system revise the model in order to satisfy the specification.

```
biocham: delete_rules(Cyclin+Cdc2~{p1}=>Cdc2~{p1}-Cyclin~{p1}).
Cyclin+Cdc2~{p1}=>Cdc2~{p1}-Cyclin~{p1}

biocham: check_all.
The specification is not satisfied.
This formula is the first not verified: Ai(oscil(Cdc2~{p1}-Cyclin~{p1}))

biocham: revise_model(more_elementary_interaction_rules).
Success
Modifications found:
  Deletion(s):
  Addition(s):
Cyclin+Cdc2~{p1}=[Cdc2]=>Cdc2~{p1}-Cyclin~{p1}.
```

The first solution found is correct, even though it does not correspond to the deleted rule. In fact, there are four solutions consisting in adding one rule, the third one corresponds to the original model:

```
biocham: learn_one_addition(elementary_interaction_rules).
(1) Cyclin+Cdc2~{p1}=[Cdc2]=>Cdc2~{p1}-Cyclin~{p1}
(2) Cyclin+Cdc2~{p1}=[Cyclin]=>Cdc2~{p1}-Cyclin~{p1}
(3) Cyclin+Cdc2~{p1}=>Cdc2~{p1}-Cyclin~{p1}
(4) Cyclin+Cdc2~{p1}=[Cdc2~{p1}]=>Cdc2~{p1}-Cyclin~{p1}
```

It is worth noting that in these algorithms, the use of types [25] specifying the protein functions for instance, has the effect of reducing the number of possibilities and improving the performances in terms of both adequacy of results and computation time.

3 Parameter Search from Quantitative Temporal Properties

For relatively small networks of less than a hundred of proteins, kinetic models have been proved successful to perform quantitative analyses and predictions. Since the models of most datasets are in SBML, it is quite natural to handle the kinetic expressions provided in those models, especially for relating them to quantitative biological properties. In this section, we recal the two most usual semantics for those expressions, the differential semantics and the stochastic semantics, and relate them to PILP representations. We then show that the Linear Time Logic LTL with numerical constraints provides the expressive power necessary to represent both qualitative and quantitative properties of biological systems. Similarly to what is done in the boolean case, a model-checker is then used as basis for a learning process allowing here to find parameter values fitting a given LTL specification of the biological properties that the model is supposed to reproduce. This is shown on example 1 and is developed in an application in the next section.

3.1 Continuous Semantics with ODE's

The concentration semantics of BIOCHAM associates to each molecule a real number representing its concentration. Reaction rules are in fact interpreted with their kinetic expressions by a set of nonlinear ordinary differential equations (ODE)[3]. Formally, to a set of BIOCHAM reaction rules $E = \{e_i$ `for` S_i `=>` $S_i'\}_{i=1,...,n}$ with variables $\{x_1, ..., x_m\}$, one associates the system of ODEs:

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

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

Given an initial state, i.e. initial concentrations for each of the objects, the evolution of the system is deterministic, and numerical integration algorithms compute a time series describing the temporal evolution of the system variables. The integration methods actually implemented in BIOCHAM are the adaptive step-size Runge-Kutta method and the Rosenbrock implicit method for stiff systems, which both produce simulation traces with variable time steps and are implemented in Prolog.

3.2 Stochastic Semantics with SLPs

The stochastic semantics is the most realistic semantics but also the most difficult to compute. This semantics associates to each BIOCHAM object an integer representing the number of molecules in the system. Rules are interpreted as a continuous time Markov chain where transition probabilities are defined by the kinetic expressions of reaction rules.

Stochastic simulation techniques [26] compute realizations of the process. The results are generally noisy versions of those obtained with the concentration semantics. 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 [27] 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).

In the stochastic semantics, for a given volume V of the location where a compound is situated, its concentration C is translated into a number of molecules $N = C \times V \times K$, where K is Avogadro's number. The kinetic expression e_i for the reaction i is converted into a transition rate τ_i by replacing all concentrations by the corresponding number of molecules multiplied by volume. After normalization on all possible transitions, this gives the transition probability $p_i = \frac{\tau_i}{\sum_{j=1}^{n} \tau_j}$.

[3] The kinetic expressions in BIOCHAM can actually contain conditional expressions, in which case the reaction rules are interpreted by a deterministic hybrid automaton.

This semantics is close to SLPs. Two points however render unusable the classical learning techniques, and suggest an extension of the SLP framework:

- Kinetic expressions, and thus the corresponding transition probabilities τ_i, can contain variables representing the molecular concentrations (resp. number) of the reactants in each rule. A faithful translation of those models into SLP would thus involve dynamic probabilities according to variables values, like in the stochastic semantics of BIOCHAM by continuous time Markov chains [12]. On the other hand, SLPs as defined in Section 3 of Chapter 2, are restricted to constant probabilities on each rule.
- In stochastic simulation and Gillespie algorithms [26], the time is a random variable over reals, which cannot be mixed with SLPs in the current version of the formalism.

3.3 Biological Properties in LTL with Numerical Constraints

The *Linear Time Logic*, LTL is the fragment of CTL* that uses only temporal operators. A first-order version of LTL is used to express temporal properties about the molecular concentrations in the simulation trace. A similar approach is used in the DARPA BioSpice project [11]. The choice of LTL is motivated by the fact that the concentration semantics given by ODEs is deterministic, and there is thus no point in considering path quantifiers. The version of LTL with arithmetic constraints we use, 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.

For practical purposes, some limited forms of quantified first-order LTL formulae are also allowed. As an example of this, constraints on the periods of oscillations can be expressed with a formula such as `period(A,75)`, defined as $\exists t\ \exists v\ F(Time = t\ \&\ [A] = v\ \&\ d([A])/dt > 0\ \&\ X(d([A])/dt < 0)\ \&\ F(Time = t + 75\ \&\ [A] = v\ \&\ d([A])/dt > 0\ \&\ X(d([A])/dt < 0)))$ where $Time$ is the time variable. This very formula is used extensively in the example of the next section.

Note that the notion of *next state* (operator X) refers to the state of the following time point computed by the (variable step-size) simulation, and thus does not necessarily imply real-time neighborhood. Nevertheless, for computing local maxima as in the formula above for instance, the numerical integration methods do compute the relevant time points with a very good accuracy.

3.4 Parameter Search from Temporal Properties

We implemented a dedicated LTL model checker for biochemical properties over simulation traces and proceeded, as in the boolean case, to use it for a learning method. Actually, it is mostly a search method automatically evaluating the fitness of a given parameter set w.r.t. an LTL specification.

The same method could theoretically be used to sample for a probability of satisfaction of an LTL specification for the stochastic semantics, however experimental trials proved to be too computationally expensive.

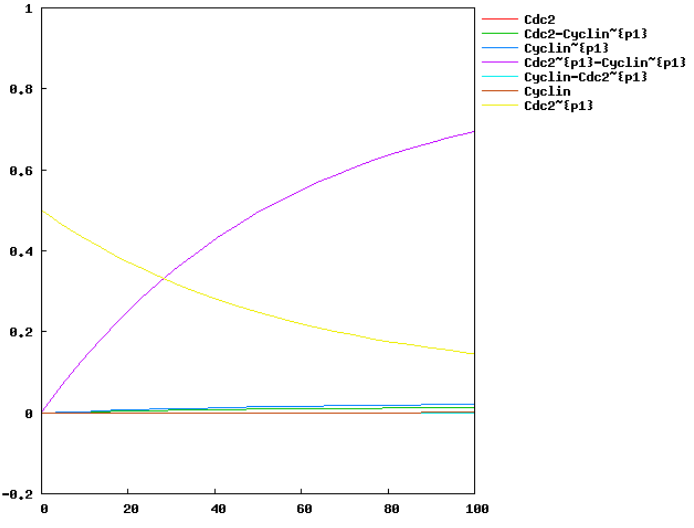

Fig. 1. Broken cell cycle model with parameter $k4 = 0$

The parameter learning method can be illustrated by changing the value of some parameter like $k4$ for instance. Figure 1 shows that the model is not oscillating as it should when $k4$ is set to zero.

The property of oscillation can be added as a temporal logic constraint to the parameter value search system as follows:

```
learn_parameters([k4],[(0,200)],20,oscil(Cdc2-Cyclin~{p1},3),100).
First values found that make oscil(Cdc2-Cyclin~{p1},3) true:
parameter(k4,200).
```

The value 200 found for $k4$ is close to the original value (180) and satisfies the experimental results formalized in LTL, as depicted in Figure 2.

Note that beacuase of the highly non-linear nature of the kinetics used in most biological models of the literature it is not possible to rely on usual tools of control theory for this kind of parameter estimation. The other available techniques are mostly local optimization based (simulated annealing and derivatives) but require to optimize with respect to a precise quantitative objective function, whereas the presented technique allows to mix qualitative and quantitative data.

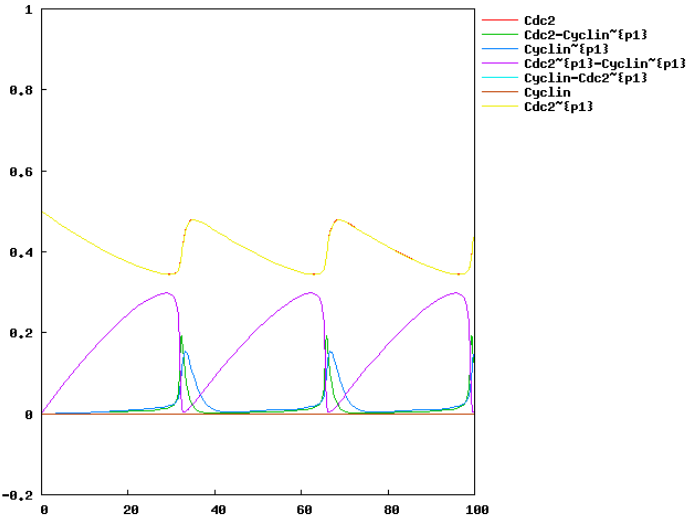

Fig. 2. Curated cell cycle model showing oscillations with the inferred parameter value $k4 = 200$

4 Application to Modelling the Synchronization in Period of the Cell Cycle by the Circadian Cycle

Cancer treatments based on the administration of medicines at different times of the day have been shown to be more efficient against malign cells and less damaging towards healthy ones. These results might be related to the recent discovery of links between the circadian clock (controlled by the light/dark cycle of a day) and the cell cycle. However, if many models have been developed to describe both of these cycles, to our knowledge none has described a real interaction between them.

In the perspective of the European Union project TEMPO[4] on temporal genomics for patient tailored chronotherapies, we developed a coupled model at the molecular level and studied the conditions of synchronization in period of these cycles, by using the parameter learning features of the modeling environment BIOCHAM. More specifically, the learning of parameter values from temporal properties with numerical constraints has been used to search how and where in the parameter space of our model the two cycles get synchronized. The technical report [28] describes the conditions of synchronization (i.e. synchronization by forcing the period of the target oscillator to be the same as that of the forcing oscillator) of the cell cycle by the circadian cycle via a common protein kinase WEE1 (see Figure 3).

[4] `http://www.chrono-tempo.org`

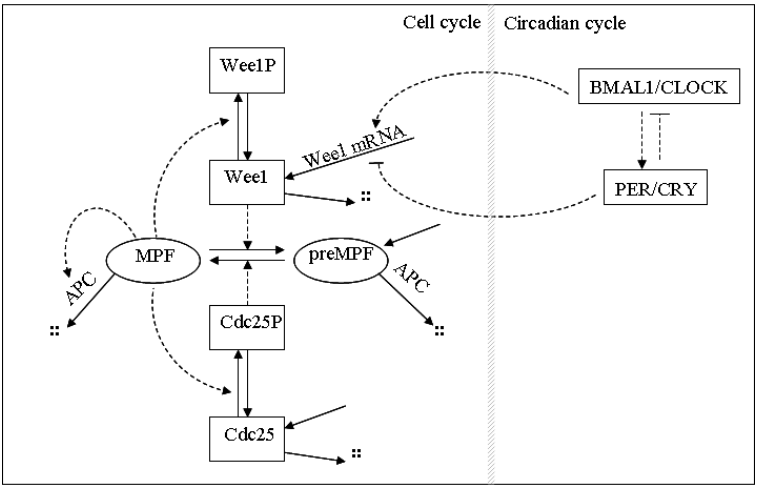

Fig. 3. Linking the circadian and the cell cycles via WEE1

The proteins chosen to illustrate the cell cycle are MPF, preMPF, the degradation factor of the cyclins, APC, the WEE1 kinase and the CDC25 phosphatase (Figure 3). Early in the cycle, MPF is kept inactive because the cyclin is not synthesized and WEE1 is present. As the cyclin is slowly synthesized, MPF activates and reaches a threshold that both inactivates WEE1 and activates CDC25 which maintains MPF in its active state. The cell enters mitosis. With a short delay, APC is activated and degrades the cyclin component of MPF. The cell exits mitosis and repeats its cycle. The model is composed of two positive feedback loops (CDC25 activates MPF which in turn activates CDC25, and WEE1 inactivates MPF which in turn inactivates WEE1) and a negative feedback loop (MPF activates APC through an intermediary enzyme X and APC degrades the cyclin component of the complex MPF). See Figure 4.

The two models describing the cell and circadian cycles are linked through the transcription of WEE1. In the model of the cell cycle alone, *wee1* mRNA was a parameter equal to 1. In the coupled model, the production of Wee1m is a function of the nuclear form of the complex BMAL1/CLOCK (BN) and the unphosphorylated nuclear form of the complex PER/CRY (PCN).

To find values for which synchronization occurs, the parameter space for each parameter has been explored using the BIOCHAM learning features from temporal properties. The values of three parameters appeared to be more significant than others: *ksweem*, *kswee* and *kimpf*. The two parameters *ksweem* and *kswee* both control the level of the WEE1 protein and show such similarities that in the following discussion, we will only report on *kswee*. The parameter values are varied in a given interval and reveal domains of synchronization reported in Figure 5. The parameters are plotted as a function of the period of three proteins that account for the behavior of the two cycles, BN (BMAL1/CLOCK nuclear) for the circadian cycle, MPF for the cell cycle, and their link, WEE1. For low

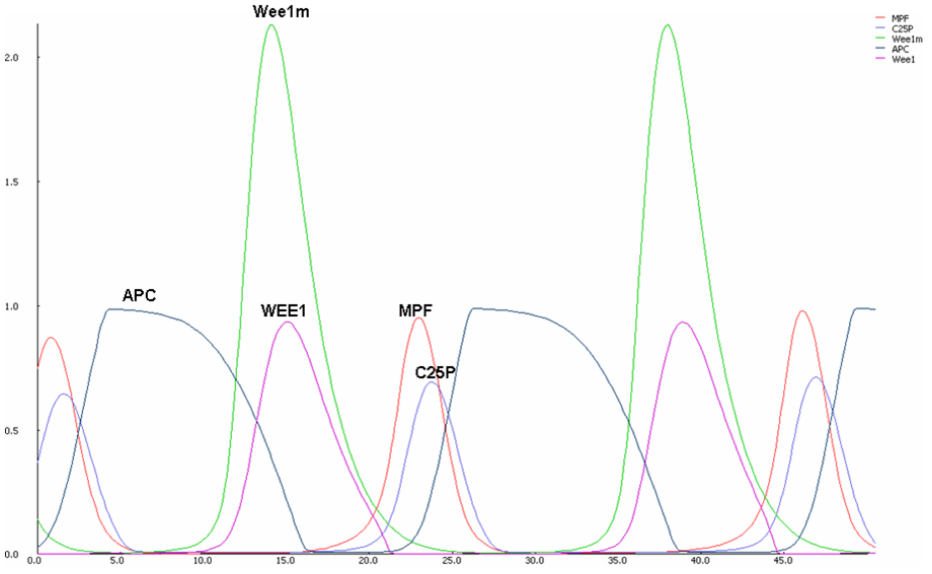

Fig. 4. Temporal simulation of a generic cell cycle

values of the parameters (region 1), MPF and BN have independent periods of oscillations of 22.4h and 23.85h respectively and no sustained oscillations are observed for WEE1. The reason for the perturbation in WEE1 oscillations in this region is that WEE1 receives simultaneously two influences: from the circadian cycle that controls the transcription of the protein mediated by the circadian transcription factors BMAL1/CLOCK and PER/CRY; and from the cell cycle that controls the activity of the protein via phosphorylation by MPF. WEE1 is produced but as soon as MPF activates, it is inactivated because WEE1 has no or little effect on MPF activation and MPF inhibits WEE1 protein. The two influences operate on WEE1 at different times as they both have different periods, perturbing WEE1 period.

For intermediate values of the parameters (region 2), WEE1 starts to play a more significant role in the cell cycle by inhibiting MPF activity, and as a result, disturbing MPF oscillations. It is only when the parameters reach a high value (either *kimpf*=1.2 or *kswee*=0.4) that the oscillations of MPF become stable again but with a period similar to that of the circadian cycle (region 3) revealing the synchronization of the cell cycle through WEE1 activity (through *kimpf*) or protein level (through *kswee*).

However, the study of *kimpf*, the parameter controlling the activity of WEE1 on MPF inactivation, shows that the synchronization does not solely depend on the value of the parameter but more particularly on the ratio *kimpf*/*kampf* since both CDC25 and WEE1 are involved in the positive feedback loops that activate MPF and therefore responsible for the G2-M transition. To investigate this dual effect, the limit of synchronization is measured as the two parameters *kimpf* and

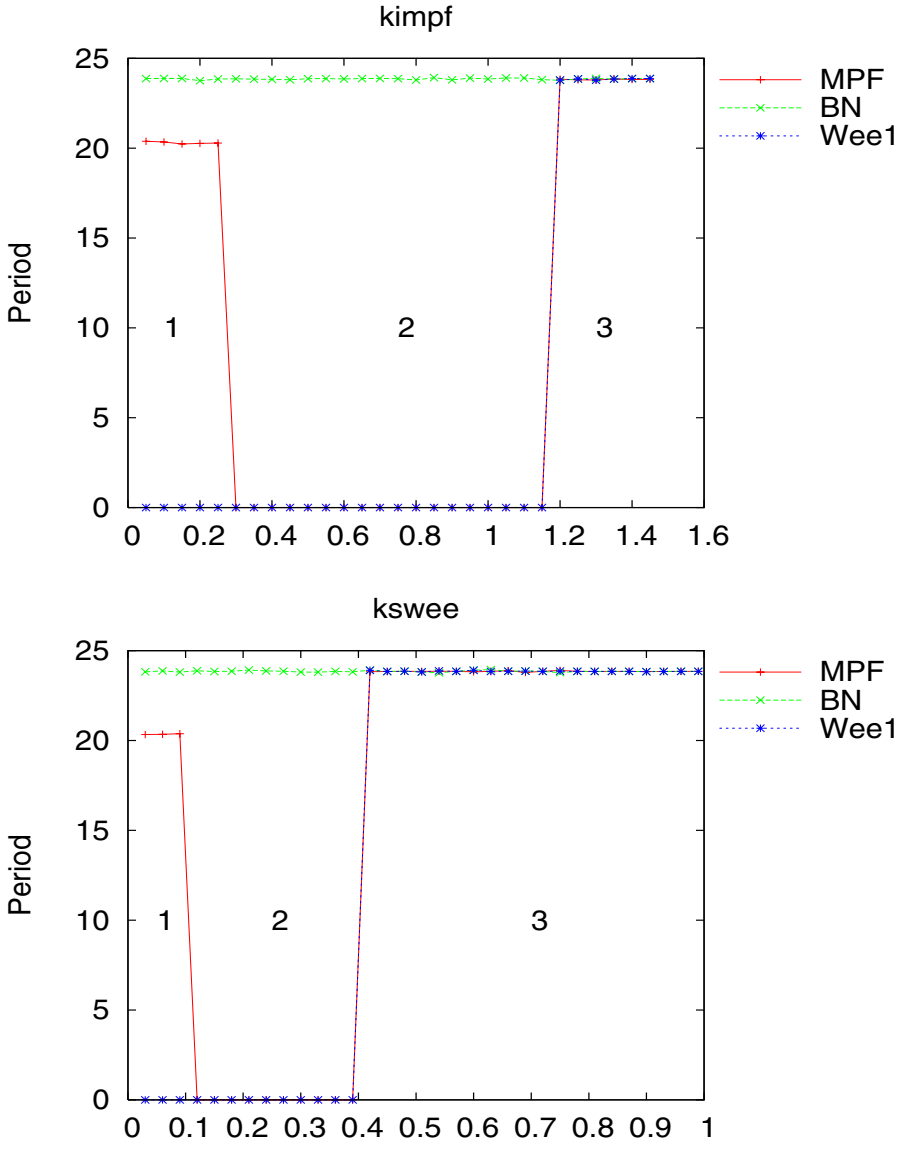

Fig. 5. Plot of the period as a function of the parameter *kimpf* from 0.01 to 1.6 and *kswee* from 0.01 to 1. The system shows synchronization for values superior to 1.2 for *kimpf* and 0.4 for *kswee*. For our purposes, constant periods are defined as follows: the last 11 peaks of the simulation over 500 time units show no more than 4% difference in their maxima and the length of the periods. MPF starts with an autonomous period of 22.4h and BN a period of 23.85h. As *kimpf* and *kswee* increase, MPF oscillations (accounting for cell cycle) lose stability and are entrained, along with WEE1, for higher values of the parameter with a period of 23.85h.

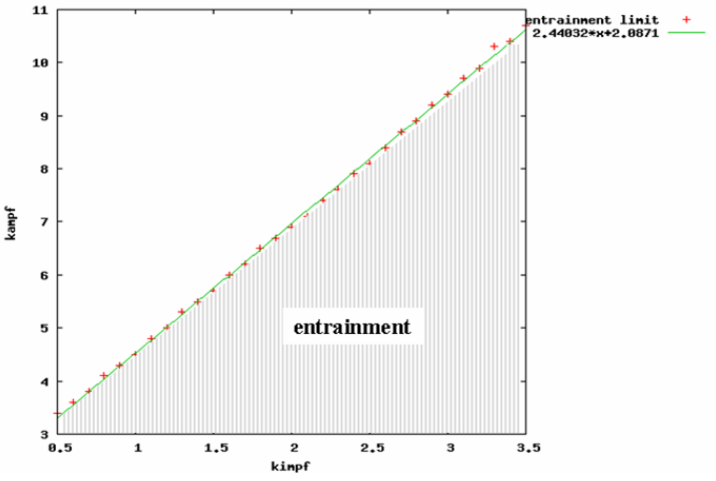

Fig. 6. BIOCHAM-generated plot of the synchronization in period of the cell cycle by the circadian cycle for different values of *kimpf* (action of WEE1 on MPF) and *kampf* (action of CDC25 on MPF). The limit of synchronization computed by BIOCHAM (red crosses) is interpolated by the linear function $kampf = 2.44832 \cdot kimpf + 2.0071$ (solid line).

kampf are varied simultaneously. A linear function of the form: $kampf = 2.44832 \cdot kimpf + 2.0071$ is obtained, the region below the line being the synchronization region (Figure 6).

These studies have been carried out thanks to the machine learning features of BIOCHAM to express the condition of synchronization in period by a first-order LTL formula with time constraints, and to explore the parameter values satisfying that formula. These methods are complementary to and have no counterpart in traditional tools for the study of dynamical systems such as the computation of bifurcation diagrams.

5 Discussion and Conclusions

Temporal logic is a powerful formalism for expressing the biological properties of a living system, such as state reachability, checkpoints, stability, oscillations, etc. This can be done both qualitatively and quantitatively, by considering respectively propositional and first-order temporal logic formulae with numerical constraints.

In the propositional case, we have given a Datalog representation of general biochemical reaction models allowing the use of ILP techniques for discovering reaction rules from given CTL properties. Because of the relative complexity of the CTL model checker however, this approach was limited to reachability properties. For general CTL properties, our approach was to use the symbolic OBDD model checker NuSMV as a black box within a model revision algorithm

that searches for rule additions and deletions in order to satisfy a CTL specification. The first results are encouraging but also show some limitations concerning the boolean abstraction that simply forgets the kinetic expressions. Less crude abstractions are however possible and are currently under investigation. Furthermore, when restricting to pure reachability properties between completely defined states, ILP methods have been shown more efficient. There is thus a perspective for combining the best of both methods in this general setting.

Kinetic models of biochemical systems have been considered too with their two most usual interpretations, by ODEs, and by continuous time Markov chains. The second interpretation has been related to PILP representations with a generalized notion of SLPs involving dynamic probabilities according to variable values. However the stochastic interpretation of kinetic expressions is computationally too expensive, while the interpretation by differential equations does scale up to real-size quantitative models. This is the reason why the continuous semantics of BIOCHAM rules based on non-linear ordinary differential equations, instead of the stochastic semantics based on continuous-time Markov chains is used in these applications. We have shown that the inference of parameter values from a temporal logic specification is flexible enough to be easy to use, and provides accurate results with reasonable execution times. These functionalities are completely new and complement the other tools the modeler can use to estimate the range of parameter values. This has been illustrated by an original application to the modelling of the synchronization in period of the cell cycle by the circadian cycle.

Acknowledgements. The authors would like to thank Stephen Muggleton and Luc de Raedt for extensive discussions on the topics of this paper, and Nathalie Chabrier-Rivier and Laurence Calzone for their contributions. This work has been partly supported by the EC Sixth Framework Project Application of Probabilistic Inductive Logic Programming II (APrIL II) (Grant Ref: FP-508861).

References

1. 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)
2. 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)
3. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. Theoretical Computer Science 325, 141–167 (2004)
4. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science 325, 69–110 (2004)
5. 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)

6. 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)
7. 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)
8. 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)
9. 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)
10. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signalling pathways using the prism model checker. In: Plotkin, G. (ed.) CMSB 2005: Proceedings of the third international conference on Computational Methods in Systems Biology (2005)
11. Antoniotti, M., Policriti, A., Ugel, N., Mishra, B.: Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics 38, 271–286 (2003)
12. 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) (CMSB 2005 Special Issue)
13. 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)
14. Calzone, L., Fages, F., Soliman, S.: BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. BioInformatics 22, 1805–1807 (2006)
15. 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)
16. Fages, F.: From syntax to semantics in systems biology - towards automated reasoning tools. Transactions on Computational Systems Biology IV 3939, 68–70 (2006)
17. Muggleton, S.H.: Inverse entailment and progol. New Generation Computing 13, 245–286 (1995)
18. Bryant, C.H., Muggleton, S.H., Oliver, S.G., Kell, D.B., Reiser, P.G.K., King, R.D.: Combining inductive logic programming, active learning and robotics to discover the function of genes. Electronic Transactions in Artificial Intelligence, 6 (2001)
19. Angelopoulos, N., Muggleton, S.H.: Machine learning metabolic pathway descriptions using a probabilistic relational representation. Electronic Transactions in Artificial Intelligence 7 (2002) (also in Proceedings of Machine Intelligence 19)
20. Angelopoulos, N., Muggleton, S.H.: Slps for probabilistic pathways: Modeling and parameter estimation. Technical Report TR 2002/12, Department of Computing, Imperial College, London, UK (2002)
21. Bratko, I., Mozetic, I., Lavrac, N.: KARDIO: A study in Deep and Qualitative Knowledge for Expert Systems. MIT Press, Cambridge (1989)
22. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)

23. 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)
24. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biochemical interaction networks. Theoretical Computer Science 325, 25–44 (2004)
25. Fages, F., Soliman, S.: Type inference in systems biology. In: Priami, C. (ed.) CMSB 2006. LNCS (LNBI), vol. 4210, Springer, Heidelberg (2006)
26. Gillespie, D.T.: General method for numerically simulating stochastic time evolution of coupled chemical-reactions. Journal of Computational Physics 22, 403–434 (1976)
27. 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)
28. Calzone, L., Soliman, S.: Coupling the cell cycle and the circadian cycle. Research Report 5835, INRIA (2006)