On Temporal Logic Constraint Solving for Analyzing Numerical Data Time Series

François Fages, Aurélien Rizk

Project-team Contraintes, INRIA Paris-Rocquencourt, BP105, 78153 Le Chesnay Cedex, France.

Abstract

Temporal logics and model-checking have proved successful to respectively express biological properties of complex biochemical systems, and automatically verify their satisfaction in both qualitative and quantitative models. In this article, we go beyond model-checking and present a constraint solving algorithm for quantifier-free first-order temporal logic formulae with constraints over the reals. This algorithm computes the domain of the real valued variables occurring in a formula that makes it true in a model. We illustrate this approach for the automatic generation of temporal logic specification from biological data time series. We provide a set of biologically relevant patterns of formulae, and apply them on numerical data time series of models of the cell cycle control and MAPK signal transduction. We show on these examples that this approach succeeds in inferring automatically semi-qualititave semi-quantitative information on concentration thresholds, amplitude of oscillations, stability properties, checkpoints and influences between species.

Key words: model-checking, temporal logic, constraint solving, data time series, systems biology

1 Introduction

Temporal logics and model-checking algorithms [11] 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 [13,8,9], discrete [3,2], stochastic [4,18] and continuous

Email addresses: Francois.Fages@inria.fr (François Fages), Aurelien.Rizk@inria.fr (Aurélien Rizk).

models [5,1,8]. This approach relies on a logical paradigm for systems biology that consists in making the following identifications [14]:

$$\begin{array}{c} \textit{biological model} = \textit{transition system} \\ \textit{biological property} = \textit{temporal logic formulae} \\ \textit{biological validation} = \textit{model-checking} \end{array}$$

Having a formal language not only for describing models, i.e. transition systems by either process calculi [24,7,23,12,21], rules [13,16,6], Petri nets [22,17], 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]. However, the formalization of the biological properties as a specification in temporal logic remains a delicate task and a bottleneck of this approach.

In this article, we investigate the use of this logical paradigm to analyze numerical data time series, and automatically infer temporal logic specifications of some pattern from them. There has been work on the inference of correlations and positive as well as negative influences between species from temporal data, especially for gene expression data [25,19]. However to our knowledge, the inference of temporal logic formulae with real valued variables from numerical data time series is new.

From a temporal logic standpoint, our work amounts to generalize model checking algorithms to constraint solving algorithms, for checking the satisfiability (instead of the validity) of temporal logic formulae in a given linear Kripke structure such as a simulation trace. To our knowledge, this generalization is also new. Previous works in this direction apply model-checking techniques to simulation traces [20,5,1] but not temporal logic constraint solvers computing the solution domain for variables.

In this article, we generalize the trace-based model-checking algorithm described in [5] and recalled in the next section, to a constraint solving algorithm for the quantifier-free fragment of LTL with numerical constraints over the reals. This first-order setting provides the ability to compute those instantiations of a formula that are true in a finite trace, by giving the complete domain of the real valued variables occurring in the formula for which it is true. A strong completeness theorem showing that the computed domain of variables describes exactly the set of solutions is given in Sec. 3, together with the time complexity of the algorithm.

Then we illustrate the relevance of this approach to the analysis of biological data time series. We provide a set of biologically relevant patterns of formulae in Sec. 4, and evaluate them on traces of cell cycle control and of signal transduction in Sec. 5. We show on these examples that this approach succeeds

in inferring automatically semi-qualititave semi-quantitative information on concentration thresholds, amplitude of oscillations, stability properties, checkpoints and influences between species.

We then conclude on the relevance of this generalization of model-checking to temporal logic constraint solving for the modelling of biological systems, the results achieved so far, and their perspectives for future work.

2 Preliminaries on Model-Checking in LTL with Constraints over the Reals

2.1 LTL with Constraints over the Reals

The *Linear Time Logic* LTL is a temporal logic [11] that extends propositional or first-order logic with modal operators for qualifying when a formula is true in a tree of timed states, called a Kripke structure. The temporal operators are X ("next", for at the next time point), F ("finally", for at some time point in the future), G ("globally", for at all time points in the future), U ("until"), and W (" weak until"). These operators enjoy some simple duality properties, $\neg X\phi = X\neg\phi$, $\neg F\phi = G\neg\phi$, $\neg G\phi = F\neg\phi$, $\neg(\psi\ U\ \phi) = (\neg\phi\ W\ \neg\psi)$, $\neg(\psi\ W\ \phi) = (\neg\psi\ U\ \neg\phi)$, and we have $F\phi = \text{true}\ U\ \phi$, $G\phi = \phi\ W$ false.

Formally, a *Kripke structure* (see for instance [11]) 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 2.1 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.

A version of LTL with constraints over the reals, called Constraint-LTL, is used in Biocham [5] to express temporal properties about molecular concentrations. A similar approach is used in the Darpa BioSpice project [1]. 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)...)))`

$s \models \alpha$	iff	α is a propositional formula true in the state s,
$s \models \psi$	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 $\pi \not\models \psi$,

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

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

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

are considered, where at each time point t_i, the trace associates the concentration values $\vec{x}_i$ to the variables, and the values of their first and second derivatives $d\vec{x}_i/dt$ and $d^2\vec{x}_i/dt^2$. This choice of derivatives is justified in section 4 as a facility for expressing positive and negative influences between entities.

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.

2.2 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 [11] can be easily adapted to Constraint-LTL as follows:

Algorithm 1 (Constraint-LTL model-checking) *[5,1]*

(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 [5].

3 Temporal Logic Constraint Solving in Quantifier-Free First-Order LTL over the Reals

3.1 Quantifier-Free First-Order LTL Formulae over the Reals

Here we consider the quantifier-free fragment of first-order LTL formula over the reals, named QFLTL($\mathbb{R}$), i.e. Constraint-LTL formula with real valued variables allowed in the constraints. More precisely, the language of QFLTL($\mathbb{R}$) formulae considered in this article is defined by the grammar given in Table 3.1.

$$
\begin{aligned}
Qfltl = & \\
& Atom \\
& \mid X(Qfltl) \\
& \mid (Qfltl)\ U\ (Qfltl) \\
& \mid (Qfltl)\ W\ (Qfltl) \\
& \mid (Qfltl) \wedge (Qfltl) \\
& \mid (Qfltl) \vee (Qfltl) \\
& \mid (Qfltl) \Rightarrow (Qfltl) \\
& \mid \neg(Qfltl) \\
Atom = & \\
& Value\ Op\ Variable \mid Value\ Op\ Value \\
Op = & \\
& < \mid > \mid \leq \mid \geq \\
Value = & \\
& float \mid [molecule] \mid d[molecule]/dt \mid d^2[molecule]/dt^2 \mid Time \\
& \mid Value + Value \mid Value - Value \mid -Value \mid Value \times Value \\
& \mid Value/Value \mid Value^{Value}
\end{aligned}
$$

Table 2
Grammar of QFLTL(ℝ) formulae.

Note that negations and implications can be eliminated, by propagating the negations down to the atomic constraints in the formula. From now on, we will assume that all QFLTL(ℝ) formulae are in negation free normal form.

3.2 QFLTL(ℝ) Constraint Solving Algorithm

Given a Kripke structure $\mathcal{K}$ with real-valued states, and a QFLTL(ℝ) formula $\phi(\vec{x})$ with n real-valued variables, the *constraint satisfaction problem*, $\exists \vec{x} \in \mathbb{R}^n$ $(\phi(\vec{x}))$, is the problem of determining the valuations $\vec{v}$ of the variables for which the formula ϕ is true. In other words, we look for the domain of validity $\mathcal{D}_\phi \subset \mathbb{R}^n$ such that $\mathcal{K} \models_{LTL} \forall \vec{v} \in \mathcal{D}_\phi\ (\phi(\vec{v}))$.

This domain of validity $\mathcal{D}_\phi$ of ϕ can be computed using an algorithm similar to the model-checking algorithm of section 2.2.

Algorithm 2 (QFLTL(ℝ) constraint solving algorithm) .

(1) label each trace point by the atomic sub-formulae of ϕ and their domain of validity as follows :

- *for an atomic formula ψ without variables label a time point t_i by $(\psi, \mathcal{D}_\psi(t_i) = \mathbb{R}^n)$ if ψ is true at time t_i, and $(\psi, \mathcal{D}_\psi(t_i) = \varnothing)$ otherwise;*

- *for an atomic formula* $[A] \geq p$ *(that is, of the form value* $\geq$ *variable) label a time point* t_i *by* $([A] \geq p, \mathcal{D}_{[A]\geq p}(t_i))$ *where* $\mathcal{D}_{[A]\geq p}(t_i)$ *is the half-space of* $\mathbb{R}^n$ *defined by* $p \leq [A](t_i)$*;*
- *proceed similarly for other atomic formulae containing variables;*

(2) label each time point t_i *by the sub-formula* $\psi_1 \vee \psi_2$ *and its domain of validity* $\mathcal{D}_{\psi_1 \vee \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cup \mathcal{D}_{\psi_2}(t_i)$*;*

(3) label each time point t_i *by the sub-formula* $\psi_1 \wedge \psi_2$ *and its domain of validity* $\mathcal{D}_{\psi_1 \wedge \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cap \mathcal{D}_{\psi_2}(t_i)$*;*

(4) label each time point t_i *by the sub-formula* $X\psi$ *and its domain of validity* $\mathcal{D}_{X\psi}(t_i) = \mathcal{D}_{\psi}(t_{i+1})$*;*

(5) label the last point of the trace t_n *by the sub-formula* $\psi_1 \; U \; \psi_2$ *and its domain of validity* $\mathcal{D}_{\psi_1 \; U \; \psi_2}(t_n) = \mathcal{D}_{\psi_2}$*. Starting from time point* t_{n-1}*, label each time point* t_i *by the sub-formula* $\psi_1 U \psi_2$ *and its domain of validity :* $\mathcal{D}_{\psi_1 \; U \; \psi_2}(t_i) = \mathcal{D}_{\psi_2}(t_i) \cup (\mathcal{D}_{\psi_1 U \psi_2}(t_{i+1}) \cap \mathcal{D}_{\psi_1}(t_i))$*;*

(6) label the last point of the trace t_n *by the sub-formula* $\psi_1 \; W \; \psi_2$ *and its domain of validity* $\mathcal{D}_{\psi_1 \; W \; \psi_2}(t_n) = \mathcal{D}_{\psi_1}$*. Starting from time point* t_{n-1}*, label each time point* t_i *by the sub-formula* $\psi_1 W \psi_2$ *and its domain of validity :*
$\mathcal{D}_{\psi_1 \; W \; \psi_2}(t_i) = (\mathcal{D}_{\psi_1}(t_i) \cap \mathcal{D}_{\psi_2}(t_i)) \cup (\mathcal{D}_{\psi_1 W \psi_2}(t_{i+1}) \cap \mathcal{D}_{\psi_1}(t_i))$*;*

(7) return the domain $\mathcal{D}_{\phi}(t_i)$ *for all time points* t_i *where it is not empty.*

In particular, given an ODE model and a temporal property ϕ with variables, to verify in a finite time horizon, one can compute a finite simulation trace, giving a linear Kripke structure to which the constaint solving algorithm can be applied to determine the domain of validity of the variables making ϕ true.

Recall that an orthotope in $\mathbb{R}^n$ is the cartesian product of n intervals over $\mathbb{R}$. The following propositions show that the solution domain computed by the algorithm is a finite union of orthotopes, and describes exactly the solution space for the chosen fragment of constraints over the reals.

Proposition 1 *The domains computed by the QFLTL(*$\mathbb{R}$*) constraint solving algorithm are finite unions of orthotopes.*

Proof. In the base case of atomic formulae, the algorithm computes orthotopes, and in the other cases, applies finite intersection and union operations on the computed domains. As a finite intersection of orthotopes is a finite union of orthotopes, the domains computed by the algorithm are always finite unions of orthotopes. □

Theorem 2 (Strong completeness) *The constraint solving algorithm is correct and complete: a valuation* $\vec{v}$ *makes a QFLTL(*$\mathbb{R}$*) formula* ϕ *true at time* t_i*,* $T, t_i \models_{LTL} (\phi(\vec{v}))$*, if and only if* $\vec{v}$ *is in the computed domain of* ϕ *at* t_i*,* $\vec{v} \in \mathcal{D}_{\phi}(t_i)$*.*

Proof. Let us prove inductively on the QFLTL($\mathbb{R}$) formula structure that for any time t, any QFLTL formula ϕ and any instantiation $\vec{v}$ of the variables, if $\phi(\vec{v}, t_i)$ is true then $\vec{v} \in \mathcal{D}_\phi(t_i)$ and if $\vec{v} \in \mathcal{D}_\phi(t_i)$ then $\phi(\vec{v}, t_i)$ is true :

- The atomic QFLTL formulae considered here are of the form $Value\ Op\ Variable$ or $Value\ Op\ Value$ where $Value$ is an evaluable arithmetic expression and Op an inequality operator. For all these atomic formulae the algorithm returns the exact validity domain. For instance, formula $([A] \leq p)(t_i)$ is true if and only if p is greater or equal to $[A](t_i)$ and the validity domain returned is the half-space defined by $p \geq [A](t_i)$;
- $\phi_1 \wedge \phi_2$. By algorithm construction $\mathcal{D}_{\phi_1 \wedge \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)$ hence : $\vec{v} \in \mathcal{D}_{\phi_1 \wedge \phi_2}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_{\phi_1}(t_i)$ *and* $\vec{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\vec{v}, t_i) \ \wedge \ \phi_2(\vec{v}, t_i) \Leftrightarrow (\phi_1 \wedge \phi_2)(\vec{v}, t_i)$;
- $\phi_1 \vee \phi_2$. By algorithm construction $\mathcal{D}_{\phi_1 \vee \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cup \mathcal{D}_{\phi_2}(t_i)$ hence : $\vec{v} \in \mathcal{D}_{\phi_1 \vee \phi_2}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_{\phi_1}(t_i)$ *or* $\vec{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\vec{v}, t_i) \ \vee \ \phi_2(\vec{v}, t_i) \Leftrightarrow (\phi_1 \vee \phi_2)(\vec{v}, t_i)$;
- $X(\phi)$. By algorithm construction $\mathcal{D}_{X(\phi)}(t_i) = \mathcal{D}_\phi(t_{i+1})$ hence : $\vec{v} \in \mathcal{D}_{X(\phi)}(t_i) \Leftrightarrow \vec{v} \in \mathcal{D}_\phi(t_{i+1}) \Leftrightarrow X(\phi)(\vec{v}, t_i)$;
- $\phi_1\ U\ \phi_2$. By algorithm construction : $\mathcal{D}_{\phi_1 U \phi_2}(t_i) = \mathcal{D}_{\phi_2}(t_i) \cup (\mathcal{D}_{\phi_1 U \phi_2}(t_{i+1}) \cap \mathcal{D}_{\phi_1}(t_i))$ hence : $\vec{v} \in \mathcal{D}_{(\phi_1 U \phi_2)}(t_i) \Leftrightarrow \phi_2 \ \vee \ (\phi_1 \wedge X(\phi_1(U\phi_2))(\vec{v}, t_i)$. Or formula $(\phi_1 U \phi_2)$ can be rewritten as $(\phi_1 U \phi_2) = \phi_2 \ \vee \ (\phi_1 \wedge X(\phi_1(U\phi_2))$;
- $\phi_1\ W\ \phi_2$. By algorithm construction : $\mathcal{D}_{\phi_1\ W\ \phi_2}(t_i) = (\mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)) \cup (\mathcal{D}_{\phi_1\ W\ \phi_2}(t_{i+1}) \cap \mathcal{D}_{\phi_1}(t_i))$ hence : $\vec{v} \in \mathcal{D}_{(\phi_1 U \phi_2)}(t_i) \Leftrightarrow (\phi_1 \ \wedge \phi_2) \ \vee \ (\phi_2 \wedge X(\phi_1 W \phi_2))(\vec{v}, t_i)$. Or formula $(\phi_1\ W\ \phi_2)$ can be rewritten as $(\phi_1\ W\ \phi_2) = (\phi_1 \wedge \phi_2) \vee \ (\phi_1 \wedge X(\phi_1\ W\ \phi_2))$.

□

The size of a QFLTL formula is the number of symbols in the formula. Let us define the *size* of a finite union of orthotopes $\mathcal{D}$, as the least integer k such that $\mathcal{D} = \bigcup_{i=1}^{k} \mathcal{R}_i$ where the R_i's are orthotopes.

Theorem 3 (complexity of the solution domain) *The validity domain of a QFLTL formula of size f containing v variables on a trace of length n is a union of orthotopes of size less than $(nf)^{2v}$.*

Proof. Let us consider the number of possible bounds appearing in the domain of validity $\mathcal{D}_\phi$ of a formula ϕ for a single variable x.

First, we examine the number of possible bounds generated by the atomic formulae. Each occurrence of variable x in ϕ is in a constraint of the form $Value(t_i)\ Op\ Variable$. Such a constraint can eventually be evaluated on each time point of the trace thus creating at most n different bounds for x. The maximum number of bounds for variable x is then n times the number of

occurrences of x in ϕ which is less than $n \times f$. Note that this size complexity is reached for instance in the formula $F([A] = u \vee [A]+1 = u \vee \cdots \vee [A]+f = u)$.

Now, by rewriting temporal operators U and W as :
$\phi_1\ U\ \phi_2(t_i) = \bigvee_{j \geq i}(\phi_2(t_j) \wedge \bigwedge_{i \leq k < j} \phi_1(t_k))$ and
$\phi_1\ W\ \phi_2(t_i) = \bigwedge_{j \geq i}(\phi_1(t_j) \vee \bigvee_{i \leq k < j} \phi_2(t_k))$ we remark that all QFLTL formulae can be rewritten with $\vee$ and $\wedge$ without changing their set of solutions. If $\mathcal{B}_v(\phi)$ is the set of possible bounds for variable x in ϕ, and if ϕ_1 and ϕ_2 are subformulae of ϕ, we have $\mathcal{B}_v(\phi_1 \vee \phi_2) \subset \mathcal{B}_v(\phi)$ and $\mathcal{B}_v(\phi_1 \wedge \phi_2) \subset \mathcal{B}_v(\phi)$. It is thus clear that intersections and unions of orthotopes do not generate new bounds.

As an orthotope is a cartesian product of intervals, it is defined by two bounds for each variable. With less than $n \times f$ bounds per variable, one can thus form less than $(nf)^{2v}$ orthotopes. Therefore, the solution domain computed by the algorithm is a union of orthotopes (Prop. 1) of size less than $(nf)^{2v}$. □

Corollary 4 *The time complexity of algorithm 2 is in* $\mathcal{O}((nf)^{2v})$.

Proof. Simply remark that in the worst case for the analysis of the domain size, each operation of the algorithm increases the size of the domain, and that the time complexity of each operation is bounded by the size of the domains. Hence the time complexity of the algorithm is in $\mathcal{O}((nf)^{2v)})$. □

As for the tightness of these bounds, note that the following formula
$F([A_1] = X_1 \vee [A_1] + 1 = X_1 \vee ... \vee [A_1] + f = X_1)\ \wedge\ ...$
$\wedge\ F([A_v] = X_v \vee [A_v] + 1 = X_v \vee ... \vee [A_v] + f = X_v)$
has a solution domain of size $(nf)^v$ on a trace of n values for the $[A_i]$'s such that the values $[A_i] + k$ are all different for $1 \leq i \leq v$ and $0 \leq k \leq f$.

4 Biologically Relevant Patterns of QFLTL(ℝ) Formulae

Temporal logic is sufficiently expressive to formalize a wide range of biological properties known from experiments under various conditions [8,5]. The constraint solving algorithm given for QFLTL(ℝ) formulae makes it possible to analyze concentration traces and obtain semi-quantitative information formalized as QFLTL(ℝ) formulae. In particular, a quantitative counterpart of the purely qualitative properties in propositional CTL studied in [8] can be expressed as follows, where variables are written using lowercase letters:

Reachability : `F([A]>=p)`, for expressing what threshold p does species A attain in the trace;
Checkpoints : `not (([A]<p1)U([B]>p2))`, for expressing on which thresholds

$p1$ and $p2$ is it false that $[A]$ is lower than $p1$ until $[B]$ is above $p2$, i.e. for which $p1$ and $p2$ `[A] >= p1` is compulsory for having `[B]>p2`.

Stability : `G([A]=<p1 & [A]>=p2)`, for formalizing the range of values taken by $[A]$; This range can be looked for in some context given by a condition like in `G(Time>10 -> ([A]<p1 & [A]>p2))`.

Oscillation : `F((d([A])/dt>0 & [A]>v1) & (F((d([A])/dt<0 & [A]<v2))))`, for the amplitude $(v1-v2)$ attained in at least one oscillation. An oscillation is defined as the change of sign of the derivative. This formula can be extended for more oscillations and is abbreviated by `oscil(M,K,p)`. It states that `M` must have amplitude `p` in at least `K` oscillations. By applying the algorithm for each value of `K`, beginning with 1, we can find the number of oscillations in the trace and minimal amplitude `p` attained by `K` oscillations for any `K`.

Influence : `G(d[A]/dt>p1 -> d2[B]/dt2>=0)`, for expressing above which threshold does the derivative of A have an influence on B. The influence is positive if a high value of `d[A]/dt` entails a positive second derivative of `[B]`. It is worth noticing that, as multiple species might influence B, this formula only indicates a correlation between the value of the derivative of A and the second derivative of B and gives no proof of direct influence.

5 Application to the Inference of Temporal Properties from Biological Time Series

5.1 Cell Cycle Data

In this section we present the application of the constraint solving algorithm 2.2 to the budding yeast cell cycle data. For the purpose of evaluation of the method, we do not use experimental data but simulation data obtained from the model of [10]. The application of the method to experimental data is discussed in section 5.3. Concentration traces are obtained by simulating the cell cycle control model in Biocham. Then, we try to recover relevant properties of the model by automatically analyzing the traces.

The reaction rules of the model are the following:

```
(1) _=>Cyclin.
(2) Cyclin+Cdc2~{p1}    => Cdc2-Cyclin~{p1,p2}
(3) Cdc2-Cyclin~{p1,p2} => Cdc2-Cyclin~{p1}
(4) Cdc2-Cyclin~{p1,p2} =[Cdc2-Cyclin~{p1}]=> Cdc2-Cyclin~{p1}
(5) Cdc2-Cyclin~{p1}    => Cyclin~{p1}+Cdc2
(6) Cyclin~{p1} =>_
(7) Cdc2        => Cdc2~{p1}
(8) Cdc2~{p1}   => Cdc2
```

Notations `~{p1}` and `~{p1,p2}` denote phosphorylated forms of a molecule. Figure 1 displays the obtained simulation traces for four species of this model.

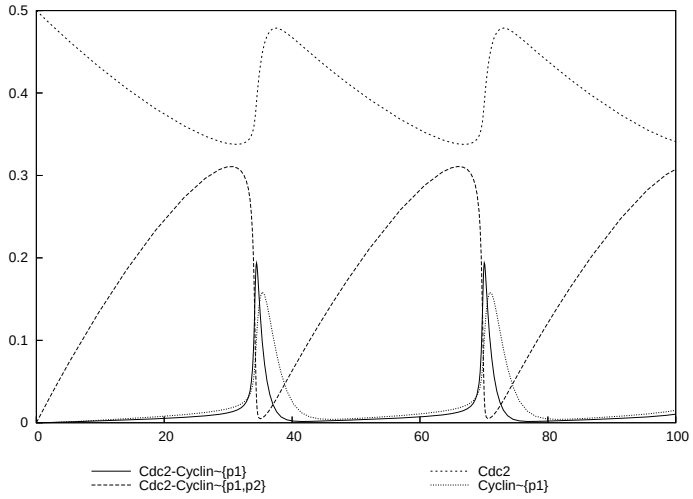

Fig. 1. Budding yeast cell cycle simulation trace over 100 time units made of 94 time points.

Such traces are remarkably informative, however to automate reasoning on them, we propose to rely on QFLTL($\mathbb{R}$) queries. For instance, a reachability query provides the maximum concentration attained by an entity:

```
biocham: trace_analyze(F([Cdc2-Cyclin~{p1}]>=v)).
[[v=<0.194]]
```

The result returned is a list of domains represented by lists of constraints on the variables, here a single domain is returned with a single constraint on v. In formulae like `F([Cdc2-Cyclin~{p1}]>=v)` where the variable only appears in inequalities of the form $Value \geq Variable$ or $Value > Variable$, the most relevant point of the domain is the highest value of `v` in the domain, i.e. its boundary. Its value is here `0.194`, the maximum concentration of `Cdc2-Cyclin~{p1}` in the trace. Table 3 gives the maximum reachable values for the four species displayed in Figure 1.

For stability, let us find the range of values taken by `[Cdc2]` :

```
biocham: trace_analyze(G([Cdc2]=<v1 & [Cdc2]>=v2)).
[[v1>=0.500, v2=<0.338]]
```

The domain is defined by the conjunction of the two constraints `v1 >= 0.500` and `v2 =< 0.338`. These values are the maximum and minimum values attained by `[Cdc2]`. The results for the other species are given in Table 3.

Species	Reachability	Stability	Amplitude of at least n oscillations	
			$n = 1$	$n = 2$
`Cdc2`	0.500	(0.338,0.500)	0.141	0.138
`Cdc2-Cyclin~{p1,p2}`	0.311	(0.000,0.311)	0.306	0.306
`Cdc2-Cyclin~{p1}`	0.194	(0.000,0.194)	0.192	0.192
`Cyclin~{p1}`	0.159	(0.000,0.159)	0.155	0.154

Table 3
Results for reachability (maximum value), stability (bottom and top values) and amplitude of at least n oscillations.

An oscillation query may compute several interval domains:

```
biocham: trace_analyze(oscil(Cdc2,1)).
[[v2>=0.338, v1=<0.479], [v2>=0.341, v1=<0.479]]
```

The result is the union of two boxes. In such domains, the most relevant point is not obvious. Here we look for the maximum amplitude $v1 - v2$. The maximum is obtained in the domain with $v1 - v2 = 0.479 - 0.338 = 0.141$. This result states that at least one oscillation of `Cdc2` has an amplitude greater or equal to 0.141. The number of oscillations is then incremented until obtaining an empty validity domain. It is obtained for `Cdc2` with the query `oscil(Cdc2,3)`, stating that there are only two oscillations of `Cdc2` in the trace.

The results for the other species are given in Table 3. Obtaining the amplitude of the oscillations is useful to distinguish between mixed amplitudes oscillations in the trace. For instance, in noisy data the amplitude can be used to count the number of oscillations regardless of small noise induced oscillations.

Whether `Cdc2-Cyclin~{p1,p2}` acts as a checkpoint for `Cdc2-Cyclin~{p1}` can be investigated with the following formula:

```
not([Cdc2-Cyclin~{p1,p2}]<v1 U [Cdc2-Cyclin~{p1}]>v2)
```

The resulting domain is a union of ten boxes. Interpreting it requires examining each box to find interesting points of the domain. Checkpoint queries are thus more delicate and perhaps not well suited for automatic analysis. In the example, the values $v1 = 0.311$ and $v2 = 0.014$ are in the domain, stating that `Cdc2-Cyclin~{p1,p2}` is not always less than 0.311 until `Cdc2-Cyclin~{p1}` exceeds 0.014. In other words `Cdc2-Cyclin~{p1,p2}` goes beyond 0.311 before `Cdc2-Cyclin~{p1}` exceeds 0.014 pointing out that `Cdc2-Cyclin~{p1,p2}` is indeed a checkpoint.

Now, the influence of a molecule A on a molecule B is looked for with formula

Species	Cdc2	Cdc2-Cyclin~{p1,p2}
Cdc2	0.00	0.11
Cdc2~{p1}	0.01	0.12
Cyclin	0.00	**0.34**
Cdc2-Cyclin~{p1,p2}	0.00	0.02
Cdc2-Cyclin~{p1}	**0.90**	0.00
Cyclin~{p1}	0.50	0.09

Table 4
Positive influence scores of all species on Cdc2 and Cdc2-Cyclin~{p1,p2}. Molecules appearing in rows (resp. columns) act as molecule A (resp. B) in formulae $G(d[A]/dt > v1 \Rightarrow d^2[B]/dt^2 > 0)$ and $G(d[A]/dt < v2 \Rightarrow d^2[B]/dt^2 < 0)$ used to compute these scores.

G(d[A]/dt>p1 -> d2[B]/dt2>0). The idea behind this formula is that if a species B appears only in a reaction rule of the form $A \rightarrow B$ with a mass action law kinetic, the following QFLTL($\mathbb{R}$) formulae are true : $G(d[A]/dt > 0 \Rightarrow d^2[B]/dt^2 > 0)$ and $G(d[A]/dt < 0 \Rightarrow d^2[B]/dt^2 < 0)$.

In a typical system each species concentration is the result of the combined effect of several other species. QFLTL($\mathbb{R}$) formula search determines above which threshold the above formulae are true, i.e. validity domains of variables v1 and v2 in formulae $G(d[A]/dt > v1 \Rightarrow d^2[B]/dt^2 > 0)$ and $G(d[A]/dt < v2 \Rightarrow d^2[B]/dt^2 < 0)$.

By comparing these domains to the range of values of $d[A]/dt$, a score $s \in [0, 1]$ is obtained indicating the influence of the derivative of [A] over the second derivative of [B]. More precisely, if the domain of validity is $v1 \geq 0$ it means that the formula is true for any positive value of $d[A]/dt$ resulting in a score 1. If the domain of validity is $v1 >= \frac{\max(d[A]/dt)}{2}$ it means that the formula is true for half of the positive values of $d[A]/dt$ resulting in a score 0.5. Table 4 gives influences scores computed by this method for species Cdc2 and Cdc2-Cyclin~{p1,p2}.

According to the reaction rules, the only species having a positive influence on [Cdc2] is [Cdc2-Cyclin~{p1}] (reaction (5)). The influence scores returned correctly reflect this. The score obtained by Cyclin~{p1} is due to its closeness with [Cdc2-Cyclin~{p1,p2}] as it can be seen in the trace. These two species both have a concentration rise coinciding with [Cdc2] own concentration rise. Nevertheless, influence scores defined above enable to distinguish [Cdc2-Cyclin~{p1}] over [Cyclin~{p1}] as molecule having a positive influence on Cdc2.

According to the reaction rules, the two species having a positive influence on

`Cdc2-Cyclin~{p1,p2}` are `[Cyclin]` and`[Cdc2~{p1}]` (reaction (2)). Notice that as more species influence `Cdc2-Cyclin~{p1,p2}` than `Cdc2`, it is harder to find correlations between single species and `Cdc2-Cyclin~{p1,p2}`. Therefore overall influence scores are smaller in this case. In spite of this, the two species having the highest scores are the correct ones.

5.2 MAPK Signal Transduction Data

The MAPK signal transduction data model is used in the same way as the cell cycle model to evaluate the analysis method. Reaction rules used to simulate concentration traces, displayed in Figure 2 are given below. All reactions rules have mass action law kinetics.

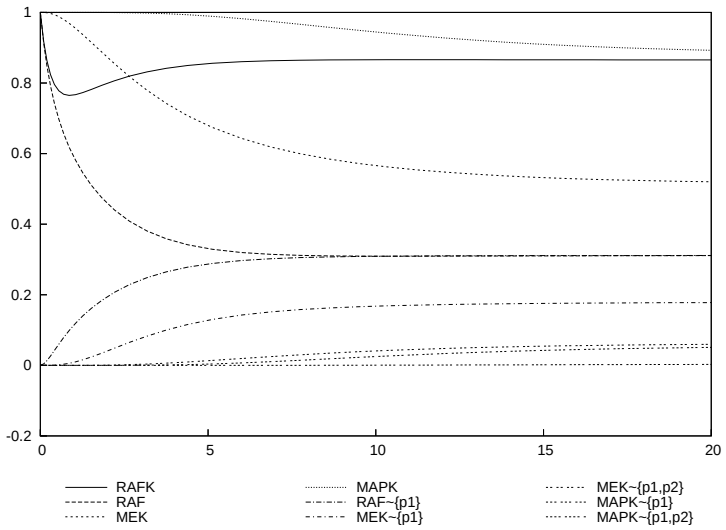

Fig. 2. MAPK model simulation trace over 20 time units made of 50 time points.

```
(1) RAF + RAFK             <=> RAF-RAFK
(2) RAF~{p1} + RAFPH       <=> RAF~{p1}-RAFPH
(3) MEK~$P + RAF~{p1}      <=> MEK~$P-RAF~{p1}
   where p2 not in $P
(4) MEKPH + MEK~{p1}~$P    <=> MEK~{p1}~$P-MEKPH
(5) MAPK~$P + MEK~{p1,p2} <=> MAPK~$P-MEK~{p1,p2}
   where p2 not in $P
(6) MAPKPH + MAPK~{p1}~$P <=> MAPK~{p1}~$P-MAPKPH
(7) RAF-RAFK                => RAFK + RAF~{p1}
(8) RAF~{p1}-RAFPH          => RAF + RAFPH
(9) MEK~{p1}-RAF~{p1}       => MEK~{p1,p2} + RAF~{p1}
(10) MEK-RAF~{p1}           => MEK~{p1} + RAF~{p1}
```

Species	Reachability	Stability	Amplitude of at least n oscillations $n = 1$
RAFK	1	(0.765,1.000)	0.001
RAF	1	(0.309,1.000)	-
MEK	1	(0,519,1.000)	-
MAPK}	1	(0.891,1.000)	-
RAF~{p1}}	0.311	(0.000,0.311)	-
MEK~{p1}	0.178	(0.000,0.178)	-
MEK~{p1,p2}	0.060	(0.000,0.060)	-
MAPK~{p1}	0.052	(0.000,0.052)	-
MAPK~{p1,p2}	0.003	(0.000,0.003)	-

Table 5
Results for Reachability (maximum value) and Stability (bottom and top values)

```
(11) MEK~{p1}-MEKPH          => MEK + MEKPH
(12) MEK~{p1,p2}-MEKPH       => MEK~{p1} + MEKPH
(13) MAPK-MEK~{p1,p2}        => MAPK~{p1} + MEK~{p1,p2}
(14) MAPK~{p1}-MEK~{p1,p2} => MAPK~{p1,p2} + MEK~{p1,p2}
(15) MAPK~{p1}-MAPKPH        => MAPK + MAPKPH
(16) MAPK~{p1,p2}-MAPKPH     => MAPK~{p1} + MAPKPH
```

Reachability, stability and oscillations queries results are given in Table 5. There are no oscillations of the species except a very small one for RAFK.

This model is made of a cascade of phosphorylation reactions. According to the reaction rules, RAFK acts as a kinase on RAF (reactions 1 and 7), RAF acts as a kinase on MEK (reactions 3, 9 and 10) and MEK acts as a kinase on MAPK (reactions 5,13 and 14).

We looked for positive influence of any species an all phosphorylated forms of RAF, MEK and MAPK. The highest score for RAF~{p1} is 0.96 and is attained by species [RAF-RAFK] while [RAFK] 's score is 0. This is consistent with the way phosphorylation reactions are written in the model, that is a complexation reaction and then a decomplexation-phosphorylation rule. Highest influence score for the phosphorylated form of MEK is correctly obtained for [MEK-RAF~{p1}], while in the case of [MAPK~{p1}] the correct complex [MAPK-MEK~{p1,p2}] only gets the second highest score, and the situation is even more confused for [MAPK~{p1,p2}].

Notice that lots of other species have relatively high influence score, which is no surprising given the similar shape of all curves in the trace. Nevertheless retaining only species having the highest scores as having positive influence,

gives an overall good indication of the direct influences between species.

Species	RAF~{p1}	MEK~{p1}	MEK~{p1,p2}	MAPK~{p1}	MAPK~{p1,p2}
[RAFK]	0.00	0.11	0.46	**0.77**	**0.50**
[RAF]	0.00	0.00	0.00	0.20	0.02
[MEK]	0.26	0.00	0.00	0.00	0.00
[MAPK]	0.50	0.47	0.11	0.00	0.00
[MAPKPH]	0.50	0.49	0.20	0.01	0.00
[MEKPH]	0.48	0.06	0.00	0.00	0.00
[RAFPH]	0.14	0.00	0.00	0.00	0.00
[RAF-RAFK]	**0.96**	0.50	0.50	0.00	**0.50**
[RAFPH-RAF~{p1}]	0.00	0.22	0.47	0.42	**0.50**
[MEK-RAF~{p1}]	0.50	**0.79**	**0.66**	0.42	**0.50**
[MEK~{p1}-RAF~{p1}]	0.00	0.00	0.27	0.41	0.48
[MEKPH-MEK~{p1}]	0.00	0.00	0.34	0.45	0.49
[MEKPH-MEK~{p1,p2}]	0.00	0.00	0.00	0.60	0.34
[MAPK-MEK~{p1,p2}]	0.00	0.00	0.00	0.62	0.37
[MAPK~{p1}-MEK~{p1,p2}]	0.00	0.00	0.00	0.00	0.09
[MAPKPH-MAPK~{p1}]	0.00	0.00	0.00	0.00	0.19
[MAPKPH-MAPK~{p1,p2}]	0.00	0.00	0.00	0.00	0.00

Table 6
Positive influence scores of all species on phosphorylated forms of `RAF, MEK and MAPK.`

5.3 Experimental Data

Experimental data for measuring the evolution over time of gene expression levels or of protein concentrations, typically involve between 6 and 50 time points taken at regular intervals. Furthermore, experimental data are noisy, and it is not one trace but several ones that have to be analyzed in order to extract their significant features. The strategy here is thus to analyze the traces separately and retain the intersection set of their properties, or the most frequent ones only.

In order to evaluate the constraint solving algorithm on similar experimental-like concentration traces, we extracted eleven equally spaced time points from the cell cycle simulation trace. An example trace is displayed in Figure 3.

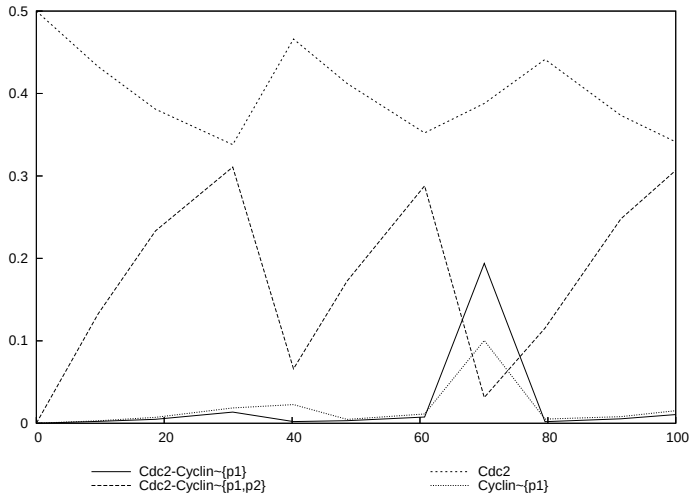

Fig. 3. Curve of concentrations every 10 units of time extracted from the cell cycle simulation trace.

Species	Reachability	Stability	Amplitude of at least n oscillations	
			$n = 1$	$n = 2$
`Cdc2`	0.500	(0.341,0.500)	0.125	0.089
`Cdc2-Cyclin~{p1,p2}`	0.311	(0.000,0.311)	0.279	0.222
`Cdc2-Cyclin~{p1}`	0.194	(0.000,0.194)	0.192	0.012
`Cyclin~{p1}`	0.100	(0.000,0.100)	0.095	0.018

Table 7
Results for reachability, stability and oscillation queries in experimental-like data.

Species	`Cdc2`	`Cdc2-Cyclin~{p1,p2}`
`Cdc2`	**0.59**	0.00
`Cdc2~{p1}`	**0.59**	0.00
`Cyclin`	0.00	**0.73**
`Cdc2-Cyclin~{p1,p2}`	0.00	0.59
`Cdc2-Cyclin~{p1}`	0.49	0.00
`Cyclin~{p1}`	0.48	0.00

Table 8
Positive influence scores of all species on `Cdc2 and Cdc2-Cyclin~{p1,p2}`.

We applied on this trace the same queries than on the original simulated one, results are given in Tables 7 and 8. Oscillations properties are still obtained but with smaller amplitudes, because the peaks are missed in the sampling.

For instance, `Cdc2-Cyclin~{p1}` has one oscillation of size 0.192 but two oscillations of size only greater than 0.012. This is a limit inherent to a low number of time points as the first peak of `Cdc2-Cyclin~{p1}` almost disappeared in this trace. Having a small number of time points also tends to give high self positive influence scores but considering only highest scores except self influence still correctly determines the influence between species.

6 Conclusion

Considering the difficulty to specify in temporal logic the biological properties of a system known from experiments, we have proposed an algorithm for computing the temporal logic formulae with constraints that are true in a given numerical data time series. To this end, the propositional Constraint-LTL model-checking algorithm described in [5] has been generalized to a constraint solving algorithm in the quantifier free fragment of first-order LTL with numerical constraints over the reals. A strong completeness theorem stating that the orthotopes of real valued variables computed for a formula in this fragment describe exactly the solution space, has been shown, together with the time complexity in $\mathcal{O}((nf)^{2v})$ where on n is the number of time points in the series, f is the size of the formula and v is the number of variables.

For the purpose of evaluating the method, we worked with data time series generated from models by simulation, and considered one experimental-like time series extracted from the simulation trace with a few time points taken at regular intervals of time. Currently, we are applying this method to the analysis of experimental temporal data of FSH signaling proteins for designing a model of FSH signal transduction together with its temporal specification, and proceed similarly with cell cycle and circadian cycle data for cancer chronotherapies in the framework of the EU project Tempo [1]. It should be clear however that the presented method may be relevant to the modelling of dynamical systems in other domains.

One obvious generalization of this work would be to consider larger fragments of constraints over the reals, trading the strong completeness theorem for a weak completeness theorem stating that the computed domains over-approximate the solution set, instead of being equal. Another generalization under investigation is the abandonment of the restriction to linear Kripke structures.

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

Acknowledgements

This research was partially supported by the EU FP7 STREP project Tempo.

References

[1] Marco Antoniotti, Alberto Policriti, Nadia Ugel, and Bud Mishra. Model building and model checking for biochemical processes. *Cell Biochemistry and Biophysics*, 38:271–286, 2003.

[2] Grégory Batt, Damien Bergamini, Hidde de Jong, Hubert Garavel, and Radu Mateescu. Model checking genetic regulatory networks using gna and cadp. In *Proceedings of the 11th International SPIN Workshop on Model Checking of Software SPIN'2004*, Barcelona, Spain, April 2004.

[3] Gilles Bernot, Jean-Paul Comet, Adrien Richard, and J. Guespin. A fruitful application of formal methods to biological regulatory networks: Extending thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, 2004.

[4] Muffy Calder, Vladislav Vyshemirsky, David Gilbert, and Richard Orton. Analysis of signalling pathways using the continuous time markow chains. In Gordon Plotkin, editor, *Transactions on Computational Systems Biology VI*, volume 4220 of *Lecture Notes in BioInformatics*, pages 44–67. Springer-Verlag, November 2006. CMSB'05 Special Issue.

[5] Laurence Calzone, Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. Machine learning biochemical networks from temporal logic properties. In Gordon Plotkin, editor, *Transactions on Computational Systems Biology VI*, volume 4220 of *Lecture Notes in BioInformatics*, pages 68–94. Springer-Verlag, November 2006. CMSB'05 Special Issue.

[6] Laurence Calzone, François Fages, and Sylvain Soliman. BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. *BioInformatics*, 22(14):1805–1807, 2006.

[7] Luca Cardelli. Brane calculi - interactions of biological membranes. In Vincent Danos and Vincent Schächter, editors, *CMSB'04: Proceedings of the second international workshop on Computational Methods in Systems Biology*, volume 3082 of *Lecture Notes in BioInformatics*, pages 257–280. Springer-Verlag, 2004.

[8] Nathalie Chabrier and François Fages. Symbolic model cheking of biochemical networks. In Corrado Priami, editor, *CMSB'03: Proceedings of the first workshop on Computational Methods in Systems Biology*, volume 2602 of *Lecture Notes in Computer Science*, pages 149–162, Rovereto, Italy, March 2003. Springer-Verlag.

[9] Nathalie Chabrier-Rivier, Marc Chiaverini, Vincent Danos, François Fages, and Vincent Schächter. Modeling and querying biochemical interaction networks. *Theoretical Computer Science*, 325(1):25–44, September 2004.

[10] Katherine C. Chen, Attila Csikász-Nagy, Bela Györffy, John Val, Bela Novàk, and John J. Tyson. Kinetic analysis of a molecular model of the budding yeast cell cycle. *Molecular Biology of the Cell*, 11:396–391, 2000.

[11] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[12] Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[13] Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer, and M. Kemal Sönmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the seventh Pacific Symposium on Biocomputing*, pages 400–412, January 2002.

[14] François Fages. Temporal logic constraints in the biochemical abstract machine biocham (invited talk). In Springer-Verlag, editor, *Proceedings of Logic Based Program Synthesis and Transformation, LOPSTR'05*, number 3901 in Lecture Notes in Computer Science, London, UK, September 2005.

[15] François Fages. From syntax to semantics in systems biology - towards automated reasoning tools. *Transactions on Computational Systems Biology IV*, 3939:68–70, December 2006.

[16] François Fages, Sylvain Soliman, and Nathalie Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2):64–73, October 2004.

[17] David Gilbert, Monika Heiner, and Sebastian Lehrack. A unifying framework for modelling and analysing biochemical pathways using petri nets. In *CMSB'07: Proceedings of the fifth international conference on Computational Methods in Systems Biology*, volume 4695 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.

[18] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In *Proc. Computational Methods in Systems Biology (CMSB'06)*, volume 4210 of *Lecture Notes in Computer Science*, pages 32–47. Springer-Verlag, 2006.

[19] Iftach Nachman, Aviv Regev, and Nir Friedman. Inferring quantitative models of regulatory networks from expression data. In *ISMB/ECCB (Supplement of Bioinformatics)*, pages 248–256, 2004.

[20] Dejan Nickovic and Oded Maler. Amt: a property-based monitoring tool for analog systems. In *Proceedings of 5th International Conference on Formal Modelling and Analysis of Times Systems, FORMATS'07*, Lecture Notes in Computer Science, Salzburg, Austria, 2007. Springer-Verlag.

[21] Andrew Phillips and Luca Cardelli. A correct abstract machine for the stochastic pi-calculus. *Transactions on Computational Systems Biology*, to appear. Special issue of BioConcur 2004.

[22] V. N. Reddy, M. L. Mavrovouniotis, and M. N. Liebman. Petri net representations in metabolic pathways. In L. Hunter, D. B. Searls, and J. W. Shavlik, editors, *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 328–336. AAAI Press, 1993.

[23] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro. Bioambients: An abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, September 2004.

[24] Aviv Regev, William Silverman, and Ehud Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Proceedings of the sixth Pacific Symposium of Biocomputing*, pages 459–470, 2001.

[25] Rui Xu, Xiao Hu, and Donald C. Wunsch II. Inference of genetic regulatory networks from time series gene expression data. In *JCNN*, volume 2, pages 1215–1220, 2004.