On the Analysis of Numerical Data Time Series in Temporal Logic

François Fages, Aurélien Rizk

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

Abstract. Temporal logics and model-checking techniques 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 paper, we propose a finite time horizon model-checking algorithm for the existential fragment of LTL with numerical constraints over the reals, with the ability to compute the range of values of the real variables occurring in a formula that makes it true in a model. We illustrate this approach for the analysis of biological data time series, provide a set of biologically relevant patterns of formulas, and evaluate them on models of the cell cycle control and MAPK signal transduction.

1 Introduction

Temporal logics and model-checking techniques [1] 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 [2–4], discrete [5, 6], stochastic [7, 8] and continuous models [9, 10, 3]. This approach relies on a logical paradigm for systems biology that consists in making the following identifications [11]:

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

Having a formal language not only for describing models, i.e. transition systems by either process calculi [12–16], rules [2, 17, 18], Petri nets [19], 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 [20]. However, the formalization of the biological properties as a specification in temporal logic remains a delicate task and a bottleneck of the method.

In this paper, we investigate the use of this logical paradigm to analyze numerical data, and infer temporal logic specifications from experimental data time

series. There has been work on the inference of correlations and positive and negative influences between entities from numerical data time series, especially for gene expression temporal data [21, 22]. However to our knowledge, the inference of temporal logic formulae with real valued variables from numerical data time series is new.

In this paper, we generalize the finite time horizon model-checking algorithm described in [9] and recalled in the next section, to the existential 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 model, by giving the range of values of the real valued variables occurring in the formula for which it is true. The completeness of the algorithm is shown for the considered fragment of constraint-LTL in Sec. 3.

We illustrate the relevance of this approach to the analysis of biological data time series, by providing a set of biologically relevant patterns of formulas in Sec. 4, and by evaluating them on models of cell cycle control and of signal transduction in Sec. 5. We then conclude on the results achieved so far, their generality, and their use in on-going work.

2 Preliminaries on Model-Checking in Constraint-LTL over the Reals

2.1 Constraint-LTL over the Reals

The *Linear Time Logic* LTL is a temporal logic [1] 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), and U ("until"). These operators enjoy some simple duality properties, $\neg X\phi = X\neg\phi$, $\neg F\phi = G\neg\phi$, $\neg G\phi = F\neg\phi$, $\neg(\psi U\phi) = G\neg\phi \vee (\neg\phi U(\neg\phi \wedge \neg\psi))$, and $F\phi = \text{true}\ U\ \phi$.

A first-order version of LTL with constraints over the reals, called constraint-LTL, is used in Biocham [9] to express temporal properties about molecular concentrations. A similar approach is used in the Darpa BioSpice project [10]. Constraint-LTL considers first-order atomic formulae with equality, inequality and arithmetic operators ranging over real values of concentrations and of their derivatives. For instance `F([A]>10)` expresses that the concentration of `A` eventually gets above the threshold value 10. `G([A]+[B]<[C])` expresses that the concentration of C is always greater than the sum of the concentrations of A and B. Oscillation properties, abbreviated as `oscil(M,K)`, are defined as a change of sign of the derivative of M at least K times:
`F((d[M]/dt > 0) & F((d[M]/dt < 0) & F((d[M]/dt > 0)...)))` The abbreviated formula `oscil(M,K,V)` adds the constraint that the maximum concentration of M must be above the threshold V in at least K oscillations.

2.2 Model-Checking Algorithm

In an ODE model, 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, we consider traces of the form

$$(< t_0, x_0, dx_0/dt, d^2x_0/dt^2 >, < t_1, x_1, dx_1/dt, d^2x_1/dt^2 >, ...)$$

where at each time point, t_i, the trace associates the concentration values x_i of the variables, and the values of their first and second derivatives dx_i/dt and d^2x_i/dt^2. This choice of derivatives is justified in section 4 as a facility for expressing positive or negative influences between entities. It is worth noting that in adaptive step size integration methods, the step size $t_{i+1} - t_i$ is not constant and is determined through an estimation of the error made by the discretization.

Algorithm 1 (trace-based constraint-LTL model-checking) *[9, 10] Given an ODE model and a temporal property ϕ to verify within a finite time horizon,*

1. *compute a finite simulation trace;*
2. *label each trace point by the atomic sub-formulae of ϕ that are true at this point;*
3. *add sub-formulae of the form $F\phi$ (resp. $X\phi$) to the predecessors (resp. immediate predecessor) of a point labeled with ϕ;*
4. *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;*
5. *add sub-formulae of the form $G\phi$ to the last state if it is labeled by ϕ, and to the predecessors of the points labeled by $G\phi$ as long as ϕ holds.*
6. *return the time points labeled by ϕ.*

Note that being limited to finite traces, and since $G\phi = !F(!\phi)$, we chose to label the last state by $G\phi$ if it satisfies ϕ, just like if the trace was completed to the infinite by a loop on its last state. Note also 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 [9].

3 Formula Instantiation in ∃-Constraint-LTL over the Reals

3.1 The Existential Fragment ∃-Constraint-LTL

Here we consider the existential fragment of constraint-LTL, where real valued variables are allowed in the constraints, with an implicit existential quantifica-

tion. More precisely, the language of ∃-constraint-LTL formulae considered in this paper is defined by the following grammar :

$$
\begin{aligned}
Constraint-ltl = \\
&Atom \mid F(Constraint-ltl) \\
&\mid G(Constraint-ltl) \mid X(Constraint-ltl) \\
&\mid (Constraint-ltl)U(Constraint-ltl) \\
&\mid (Constraint-ltl)\ and\ (Constraint-ltl) \\
&\mid (Constraint-ltl)\ or\ (Constraint-ltl) \\
&\mid (Constraint-ltl) \Rightarrow (Constraint-ltl) \\
&\mid not\ (Constraint-ltl) \\
Atom = \\
&Value < Variable \mid Value \leq Variable \\
&\mid Value > Variable \mid Value \geq Variable \\
&\mid Value < Value \mid Value \leq Value \\
&\mid Value > Value \mid Value \geq Value \\
Value = \\
&float \mid [molecule] \mid d[molecule]/dt \mid d^2[molecule]/dt^2 \mid Time \\
&\mid Value + Value \mid Value - Value \mid -Value \mid Value \times Value \\
&\mid Value/Value \mid Value^{Value}
\end{aligned}
$$

By an obvious transformation, 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 ∃-constraint-LTL formulae are in negation free normal form.

3.2 Formula Instantiation Algorithm

Given a trace T representing a linear Kripke structure, and an ∃-constraint-LTL formula ϕ with n variables, the *formula instantiation problem*, $\exists \boldsymbol{v} \in \mathbb{R}^n\ (\phi(\boldsymbol{v}))$, is the problem of determining the valuation $\boldsymbol{v}$ of the variables for which the formula ϕ is true in T. In other words, we look for the domain of validity $\mathcal{D}_\phi \subset \mathbb{R}^n$ such that $T \models_{LTL} \forall \boldsymbol{v} \in \mathcal{D}_\phi\ (\phi(\boldsymbol{v}))$.

The 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 (trace-based ∃-constraint-LTL formula instantiation) *Given an ODE model and a temporal property ϕ with variables, to verify in a finite time horizon,*

1. *compute a finite simulation trace;*

2. *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;*
3. *starting from the end of the trace, label each time point t_i by the sub-formula $F\psi$ and its domain of validity $\mathcal{D}_{F\psi}(t_i) = \mathcal{D}_{F\psi}(t_{i+1}) \cup \mathcal{D}_\psi(t_i)$;*
4. *starting from the end of the trace, label each time point t_i by the sub-formula $G\psi$ and its domain of validity $\mathcal{D}_{G\psi}(t_i) = \mathcal{D}_{G\psi}(t_{i+1}) \cap \mathcal{D}_\psi(t_i)$;*
5. *starting from the end of the trace, 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 each time point t_i by the sub-formula $X\psi$ and its domain of validity $\mathcal{D}_{X\psi}(t_i) = \mathcal{D}_\psi(t_{i+1})$;*
7. *label each time point t_i by the sub-formula ψ_1 or ψ_2 and its domain of validity $\mathcal{D}_{\psi_1 \text{ or } \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cup \mathcal{D}_{\psi_2}(t_i)$;*
8. *label each time point t_i by the sub-formula ψ_1 and ψ_2 and its domain of validity $\mathcal{D}_{\psi_1 \text{ and } \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cap \mathcal{D}_{\psi_2}(t_i)$;*
9. *return the domain $\mathcal{D}_\phi(t_i)$ for all time points t_i where it is not empty.*

This algorithm enjoys a strong completeness theorem for the chosen fragment of constraints over the reals.

Theorem 1. *The instantiation algorithm is correct and complete: a valuation $\boldsymbol{v}$ makes ϕ true at time t_i, $T, t_i \models_{LTL} (\phi(\boldsymbol{v}))$, if and only if $\boldsymbol{v}$ is in the computed domain of ϕ at t_i, $\boldsymbol{v} \in \mathcal{D}_\phi(t_i)$.*

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

- Atomic constraint-LTL formulae considered are of the form $Value\ \mathcal{R}\ Variable$ or $Value\ \mathcal{R}\ Value$ where $Value$ is an evaluable arithmetic expression and $\mathcal{R}$ 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 $p \geq [A](t_i)$;
- ϕ_1 *and* ϕ_2 . By algorithm construction $\mathcal{D}_{\phi_1 \text{ and } \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)$ hence : $\boldsymbol{v} \in \mathcal{D}_{\phi_1 \text{ and } \phi_2}(t_i) \Leftrightarrow \boldsymbol{v} \in \mathcal{D}_{\phi_1}(t_i)$ *and* $\boldsymbol{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\boldsymbol{v}, t_i)$ *and* $\phi_2(\boldsymbol{v}, t_i) \Leftrightarrow (\phi_1$ *and* $\phi_2)(\boldsymbol{v}, t_i)$;
- ϕ_1 *or* ϕ_2 . By algorithm construction $\mathcal{D}_{\phi_1 \text{ or } \phi_2}(t_i) = \mathcal{D}_{\phi_1}(t_i) \cup \mathcal{D}_{\phi_2}(t_i)$ hence : $\boldsymbol{v} \in \mathcal{D}_{\phi_1 \text{ or } \phi_2}(t_i) \Leftrightarrow \boldsymbol{v} \in \mathcal{D}_{\phi_1}(t_i)$ *or* $\boldsymbol{v} \in \mathcal{D}_{\phi_2}(t_i) \Leftrightarrow \phi_1(\boldsymbol{v}, t_i)$ *or* $\phi_2(\boldsymbol{v}, t_i) \Leftrightarrow (\phi_1$ *or* $\phi_2)(\boldsymbol{v}, t_i)$;

- $F(\phi)$. By algorithm construction $\mathcal{D}_{F(\phi)}(t_i) = \bigcup_{j \geq i}(\mathcal{D}_\phi(t_j))$ hence : $\boldsymbol{v} \in \mathcal{D}_{F(\phi)}(t_i) \Leftrightarrow \exists j \geq i, \boldsymbol{v} \in \mathcal{D}_\phi(t_j) \Leftrightarrow F(\phi)(\boldsymbol{v}, t_i)$;
- $G(\phi)$. By algorithm construction $\mathcal{D}_{F(\phi)}(t_i) = \bigcap_{j \geq i}(\mathcal{D}_\phi(t_j))$ hence : $\boldsymbol{v} \in \mathcal{D}_{G(\phi)}(t_i) \Leftrightarrow \forall j \geq i, \boldsymbol{v} \in \mathcal{D}_\phi(t_j) \Leftrightarrow G(\phi)(\boldsymbol{v}, t_i)$;
- $X(\phi)$. By algorithm construction $\mathcal{D}_{X(\phi)}(t_i) = \mathcal{D}_\phi(t_{i+1})$ hence : $\boldsymbol{v} \in \mathcal{D}_{X(\phi)}(t_i) \Leftrightarrow \boldsymbol{v} \in \mathcal{D}_\phi(t_{i+1}) \Leftrightarrow X(\phi)(\boldsymbol{v}, t_i)$;
- $\phi_1 U \phi_2$. $\phi_1 U \phi_2(t_i)$ is true if and only if $\phi_2(t_i)$ is true or $\phi_1(t_i)$is true and $(\phi_1 U \phi_2)(t_{i+1})$ is true. Moreover by construction $\mathcal{D}_{(\phi_1 U \phi_2)}(t_i) = \mathcal{D}_{\phi_2}(t_i) \cup (\mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i))$ hence : $\boldsymbol{v} \in \mathcal{D}_{(\phi_1 U \phi_2)}(t_i) \Leftrightarrow \boldsymbol{v} \in \mathcal{D}_{\phi_2}(t_i) \cup (\mathcal{D}_{\phi_1}(t_i) \cap \mathcal{D}_{\phi_2}(t_i)) \Leftrightarrow (\phi_1 U \phi_2)(\boldsymbol{v}, t_i)$.

Now, let us define a box of $\mathbb{R}^n$ as a finite intersection of half-spaces and the size $S(\mathcal{D})$ of a domain as the minimum number of boxes the domain is made of. Note that for a one dimension domain $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ (resp. $\mathcal{D} = \mathcal{D}_1 \cap \mathcal{D}_2$), the maximum size is $S(\mathcal{D}_1) + S(\mathcal{D}_2)$ (resp. $\max(S(\mathcal{D}_1), S(\mathcal{D}_2))$).

Theorem 2. *In the worst case, the size of the validity domain of a LTL formula of size k on a trace of length n is n^{k^2}.*

Proof. Let us prove inductively that the size of the projection on one variable of the validity domain (i.e., the validity domain of a single variable) of a LTL formula of size k, is at most n^k. The size of the validity domain of an atomic formula is at most 1. The maximum size of a one dimension domain of a formula of size k is:

$$\begin{aligned}
\max(S(\mathcal{D}_{\phi_1}(t_i)), S(\mathcal{D}_{\phi_2}(t_i))) &\text{ for } \mathcal{D}_{\phi_1 \text{ and } \phi_2}(t_i) \\
S(\mathcal{D}_{\phi_1}(t_i)) + S(\mathcal{D}_{\phi_2}(t_i)) &\text{ for } \mathcal{D}_{\phi_1 \text{ or } \phi_2}(t_i) \\
\sum_{j \geq i} S(\mathcal{D}_\phi(t_j)) &\text{ for } \mathcal{D}_{F(\phi)(t_i)} \\
\max_{j \geq i} S(\mathcal{D}_\phi(t_j)) &\text{ for } \mathcal{D}_{G(\phi)(t_i)} \\
S(\mathcal{D}_\phi(t_{i+1})) &\text{ for } D_{X(\phi)}(t_i) \\
\max(S(\mathcal{D}_{\phi_1 U \phi_2}(t_{i+1})), S(\mathcal{D}_{\phi_1}(t_i))) + S(\mathcal{D}_{\phi_2}(t_i)) &\text{ for } D_{\phi_1 U \phi_2}(t_i)
\end{aligned}$$

In all these cases except operators F an U, the size of the domain is less than the sum of the domains' size of the subformulae at one time point, which entails a size smaller than $2n^{k-1}$. The U and F operators make a sum on all time points which entails a size of at most $n \times n^{k-1} = n^k$. Each projection's size of the validity domain is thus at most n^k. The size of the validity domain of a formula containing v variables is at most $(n^k)^v \leq (n^k)^k = n^{k^2}$.

The instantiation algorithm thus computes for each subformulae and each time point a validity domain of size at most n^{k^2}.

4 Biologically Relevant Patterns of ∃-constraint-LTL Formulae

Temporal logic is sufficiently expressive to formalize a wide range of biological properties known from experiments under various conditions. The formula instantiation algorithm in ∃-constraint-LTL makes it possible to analyze concentration traces and obtain semi-quantitative information. In particular, a quantitative counterpart of the purely qualitative properties in propositional CTL studied in [3] can be expressed as follows, where variables are written using lowercase letters:

Reachability : `F([A]>=p)`, what threshold p species A attain in the trace ?

Checkpoints : `not (([A]<p1)U([B]>p2))`, for 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 a checkpoint of `[B]>p2`?

Stability : `G([A]=<p1 & [A]>=p2)`, what is 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))))`, what amplitude $(v1 - v2)$ is 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)`, 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 instantiation algorithm 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 [23]. 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.

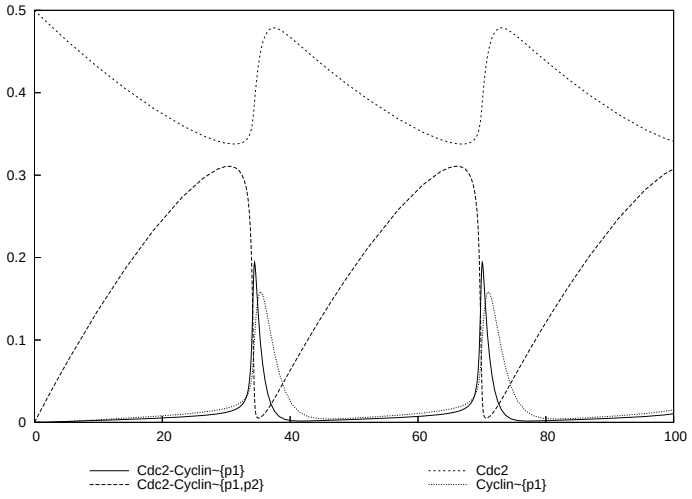

Fig. 1. Budding yeast cell cycle simulation trace over 100 time units made of 94 time points.

Such traces are remarkably informative, however to automate reasoning on them, we propose to rely on constraint-LTL 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 1 gives the maximum reachable values for the four species displayed in Figure 1.

Species	Reachability	Stability	Amplitude of at least n oscillations	
			$n = 1$	$n = 2$
`Cdc2`	0.500	(0.338,0.479)	0.141	0.138
`Cdc2-Cyclin~{p1,p2}`	0.311	(0.005,0.310)	0.306	0.306
`Cdc2-Cyclin~{p1}`	0.194	(0.002,0.194)	0.192	0.192
`Cyclin~{p1}`	0.159	(0.004,0.158)	0.155	0.154

Table 1. Results for reachability (maximum value), stability (bottom and top values in the last third part of the trace) and amplitude of at least n oscillations.

For stability, let us find the range of values taken by `[Cdc2]` in the last third part of the trace:

```
biocham: trace_analyze(G(Time>66 -> ([Cdc2]=<v1 & [Cdc2]>=v2))).
[[v1>=0.479, v2=<0.338]]
```

The domain is defined by the conjunction of the two constraints `v1 >= 0.479` and `v2 =< 0.338`. These values are the maximum and minimum values attained by `[Cdc2]` in the last third part of the trace. The results for the other species are given in Table 1.

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 1. 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 `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 constraint-LTL 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. $\exists$-constraint-LTL 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 2 gives influences scores computed by this method for species `Cdc2` and `Cdc2-Cyclin~{p1,p2}`.

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 2. 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.

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.

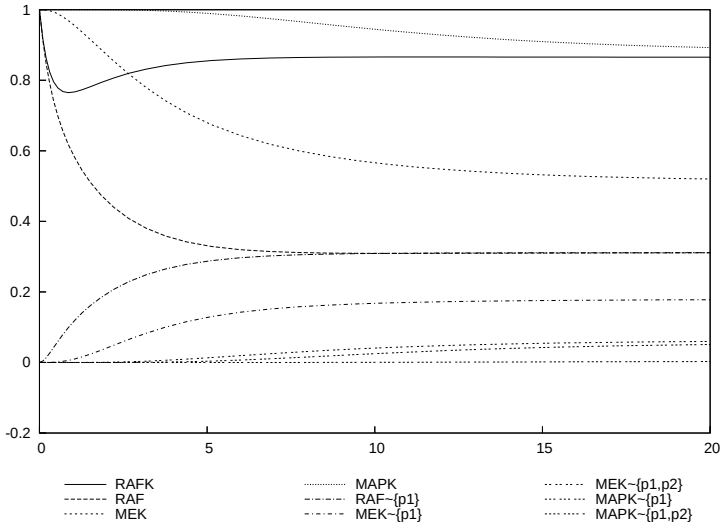

Fig. 2. MAPK model simulation trace over 20 time units made of 50 time points.

```
(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}
(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 3. Stability queries return very small ranges of values in the last third part of the trace for most species indicating stable behaviors. For instance, `RAF` is inside the $[0.310, 0.311]$ interval in the last third part of the trace. There are no oscillations of the species except a very small one for `RAFK`.

Species	Reachability	Stability	Amplitude of at least n oscillations $n = 1$
RAFK	1	(0.865,0.866)	0.001
RAF	1	(0.310,0.311)	-
MEK	1	(0,519,0.537)	-
MAPK}	1	(0.891,0.916)	-
RAF~{p1}}	0.311	(0.311,0.311)	-
MEK~{p1}	0.178	(0.174,0.178)	-
MEK~{p1,p2}	0.060	(0.052,0.060)	-
MAPK~{p1}	0.052	(0.039,0.052)	-
MAPK~{p1,)2}	0.003	(0.001,0.003)	-

Table 3. Results for Reachability (maximum value) and Stability (bottom and top values in the last third part of the trace)

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 4. 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 instantiation algorithm on similar experimental-like concentration traces, we extracted eleven equally spaced time points from the cell cycle simulation trace. The obtained trace is displayed in Figure 3.

We applied on this trace the same queries than on the original simulated one, results are given in Tables 5 and 6. 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

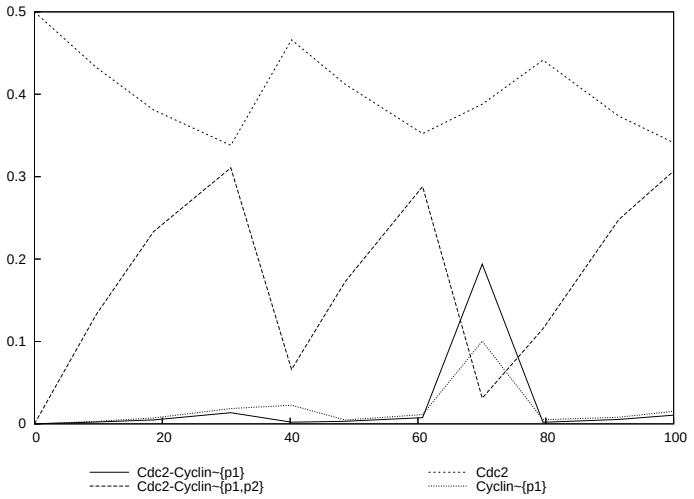

Fig. 3. Curve of concentrations every 10 units of time extracted from the cell cycle simulation trace.

Species	Reachability	Stability	Amplitude of at least n oscillations	
			$n = 1$	$n = 2$
`Cdc2`	0.500	(0.341,0.441)	0.125	0.089
`Cdc2-Cyclin~{p1,p2}`	0.311	(0.031,0.308)	0.279	0.222
`Cdc2-Cyclin~{p1}`	0.194	(0.002,0.194)	0.192	0.012
`Cyclin~{p1}`	0.100	(0.005,0.100)	0.095	0.018

Table 5. Results for reachability, stability and oscillation queries in experimental-like data.

influence scores but considering only highest scores except self influence still correctly determines the influence between species.

6 Conclusion

Considering the bottleneck of specifying in temporal logic with numerical constraints the biological properties of a system known from experiments, we have proposed an algorithm for inferring constraint-LTL formulae from numerical data time series. To this end, the finite time horizon model-checking algorithm described in [9] has been generalized to an instantiation algorithm in the existential fragment of LTL with numerical constraints over the reals. A strong completeness theorem stating that the ranges of real valued variables computed for a formula describe exactly the solution space, has been shown, together with

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

a complexity bound in n^{k^2} on the representation of the domain, where n is the number of time points and k the size of the formula.

For the purpose of evaluating the method, we worked with time series generated from models by simulation, and considered one experimental-like time series extracted from the simulation trace in few time points taken at regular intervals of time. In the near future, we plan to apply the 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].

References

1. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
2. 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. (2002) 400–412
3. Chabrier, N., Fages, F.: Symbolic model cheking of biochemical networks. In Priami, C., ed.: CMSB'03: Proceedings of the first workshop on Computational Methods in Systems Biology. Volume 2602 of Lecture Notes in Computer Science., Rovereto, Italy, Springer-Verlag (2003) 149–162
4. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biochemical interaction networks. Theoretical Computer Science **325** (2004) 25–44
5. 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** (2004) 339–347
6. Batt, G., Bergamini, D., de Jong, H., Garavel, H., Mateescu, R.: 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 (2004)
7. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signalling pathways using the continuous time markow chains. In Plotkin, G., ed.: Transactions on Computational Systems Biology VI. Volume 4220 of Lecture Notes in BioInformatics. Springer-Verlag (2006) 44–67 CMSB'05 Special Issue.

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

8. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. In: Proc. Computational Methods in Systems Biology (CMSB'06). Volume 4210 of Lecture Notes in Computer Science., Springer-Verlag (2006) 32–47
9. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In Plotkin, G., ed.: Transactions on Computational Systems Biology VI. Volume 4220 of Lecture Notes in BioInformatics. Springer-Verlag (2006) 68–94 CMSB'05 Special Issue.
10. Antoniotti, M., Policriti, A., Ugel, N., Mishra, B.: Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics **38** (2003) 271–286
11. Fages, F.: Temporal logic constraints in the biochemical abstract machine biocham (invited talk). In Springer-Verlag, ed.: Proceedings of Logic Based Program Synthesis and Transformation, LOPSTR'05. Number 3901 in Lecture Notes in Computer Science, London, UK (2005)
12. 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. (2001) 459–470
13. Cardelli, L.: Brane calculi - interactions of biological membranes. In Danos, V., Schächter, V., eds.: CMSB'04: Proceedings of the second international workshop on Computational Methods in Systems Biology. Volume 3082 of Lecture Notes in BioInformatics., Springer-Verlag (2004) 257–280
14. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. Theoretical Computer Science **325** (2004) 141–167
15. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science **325** (2004) 69–110
16. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. Transactions on Computational Systems Biology (to appear) Special issue of BioConcur 2004.
17. 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** (2004) 64–73
18. Calzone, L., Fages, F., Soliman, S.: BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. BioInformatics **22** (2006) 1805–1807
19. Gilbert, D., Heiner, M., Lehrack, S.: 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. Lecture Notes in Computer Science, Springer-Verlag (2007)
20. Fages, F.: From syntax to semantics in systems biology - towards automated reasoning tools. Transactions on Computational Systems Biology IV **3939** (2006) 68–70
21. Xu, R., Hu, X., II, D.C.W.: Inference of genetic regulatory networks from time series gene expression data. In: JCNN. Volume 2. (2004) 1215–1220
22. Nachman, I., Regev, A., Friedman, N.: Inferring quantitative models of regulatory networks from expression data. In: ISMB/ECCB (Supplement of Bioinformatics). (2004) 248–256
23. Chen, K.C., Csikász-Nagy, A., Györffy, B., Val, J., Novàk, B., Tyson, J.J.: Kinetic analysis of a molecular model of the budding yeast cell cycle. Molecular Biology of the Cell **11** (2000) 396–391