Learning Transition Rules from Temporal Logic Properties

Nathalie Chabrier-Rivier, Francois Fages, Sylvain Soliman, Laurence Calzone

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Learning Transition Rules from Temporal Logic Properties

Nathalie Chabrier-Rivier — Francois Fages — Sylvain Soliman — Laurence Calzone

N° 5543

Avril 2005

Thèmes SYM et BIO

ISSN 0249-6399 ISRN INRIA/RR--5543--FR+ENG

Learning Transition Rules from Temporal Logic Properties

Nathalie Chabrier-Rivier , Francois Fages , Sylvain Soliman , Laurence Calzone

Thèmes SYM et BIO — Systèmes symboliques et Systèmes biologiques
Projets Contraintes

Rapport de recherche n° 5543 — Avril 2005 — 19 pages

Abstract: Most of the work on temporal representation issues in Machine Learning deals with the problem of learning/mining temporal patterns from a large set of temporal data. In this paper we investigate the somewhat different problem of learning the behavioral rules of a system from its observed temporal properties formalized in temporal logic. Our interest in this problem arose from Systems Biology and the development of machine learning techniques for learning biochemical reaction rules and kinetic parameters in the Biochemical Abstract Machine BIOCHAM. Our contribution is twofold. First, in the general setting of Kripke structures and concurrent transition systems, we define positive and negative CTL formulae and propose a theory revision algorithm for learning transition rules from a CTL specification. Second, in the setting of hybrid systems which add a continuous dynamics described by differential equations, we show how a similar algorithm can be built to learn parameter values from a constraint LTL specification. In the context of BIOCHAM, which is used as a running example in this paper, we report evaluation results showing the usefulness of this approach and encouraging performance figures.

Key-words: Temporal Logic, CTL, LTL, Constraints, Rule Learning, Parameter Learning, Systems Biology

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 —Télécopie : +33 1 39 63 53 30

Apprentissage de règles de transition à partir de propriétés de logique temporelle

Résumé : La plupart des travaux sur les questions de représentation temporelle en apprentissage automatique traitent du problème d'apprentissage de "motifs temporels" à partir d'un grand ensemble de données temporelles. Dans ce papier nous étudions le problème quelque peu différent d'apprendre les règles de comportement d'un système à partir de ses propriétés temporelles observées et fomalisées en logique temporelle. Notre intéret pour ce problème vient de la biologie des systèmes et du développement de techniques d'apprentissage automatique pour apprendre des règles de réactions biochimiques et des paramètres cinétiques dans la Machine Abstraite Biochimique BIOCHAM. Notre contribution est en deux temps. Premièrement, dans le cadre général des structures de Kripke et des systèmes de transition concurents, nous définissons les formules CTL positives et négatives et proposons un algorithme fondé sur la révision de théorie, pour apprendre des règles de transitions à partir de spécifications en logique temporelle CTL. Deuxièmement, dans le cadre des systèmes hybrides qui ajoutent une dynamique continue décrite par des équations différentielles, nous montrons comment un algorithme similaire peut être construit pour apprendre les valeurs des paramètres à partir de spécifications en logique temporelle LTL avec contraintes. Dans le contexte de BIOCHAM, qui est utilisé comme exemple d'application dans ce papier, nous présentons des résultats d'évaluation qui montrent l'utilité de cette approche et des temps d'exécution encourageants.

Mots-clés : Logique Temporelle, CTL, LTL, Contraintes, Apprentissage de règles, Apprentissage de paramètres, Biologie des Systèmes

1 Introduction

The representation of temporal data has been investigated for a long time in Artifical Intelligence [2, 6]. In Machine Learning, most of the work on temporal representation issues deal with the problem of learning/mining temporal patterns from a large set of temporal data [16, 21, 25, 1, 18]. In this context, the use of temporal logic has been proposed as a mean to represent the temporal structure of time series of events.

In this paper we investigate the somewhat different problem of learning the behavioral rules of a system from its observed temporal properties formalized in temporal logic. Our interest in this problem arose from Systems Biology and the development of the Biochemical Abstract Machine BIOCHAM [15] which provides an environment for modeling, querying and curating models of complex bio-molecular interaction processes. BIOCHAM is composed of a rule-based language to model bio-molecular interactions, and an original temporal logic based language to formalize the biological properties of the system [11]. Our first experimental results have been reported on a qualitative model of the mammalian cell cycle control involving about 500 variables and 2700 reaction rules [12]. In this context, we are investigating machine learning techniques for learning BIOCHAM reaction rules [9], as well as kinetic parameters, from the observed biological properties of the organism formalized in temporal logic. There has been work on the use of machine learning techniques, such as inductive logic programming [19], to infer gene functions [8], metabolic pathway descriptions [3, 4] or gene interactions [7]. However structural learning of bio-molecular interactions from temporal properties is quite new, both from the machine learning perspective and from the Systems Biology perspective.

The contribution of this paper is twofold. First, in the general setting of concurrent transition systems [22], we propose an enumerative algorithm, and a more involved theory revision algorithm [23], for learning transition rules from a behavioural specification given in Computation Tree Logic CTL [13]. This algorithm is illustrated with BIOCHAM reaction rule discovery from a partial model of reaction rules and a set of observed biological properties formalized in CTL.

Second, in the setting of hybrid systems which add a continuous dynamics described by differential equations, we show how a similar algorithm can be used to learn parameter values from a LTL specification based on first-order logic with constraints over the reals. This general method is illustrated with the learning of kinetic parameter values in BIOCHAM reaction rules from expected numerical properties, such as oscillations, thresholds, curve patterns, etc.

The performance figures reported in the context of BIOCHAM for semi-automatically curating models of bio-molecular processes are encouraging and already show the usefulness of these methods.

2 Preliminaries on Temporal Logics and BIOCHAM

2.1 Syntax: CTL and LTL

The Computation Tree Logic CTL* [13] basically extends classical logic used for describing states (we will see later examples using either propositional logic for a boolean setting or first order logic to describe arithmetic constraints over reals), with operators for reasoning on time (state transitions) and non-determinism. Several temporal operators are introduced in CTL*:

- $X\phi$ meaning ϕ is true at next step,
- $G\phi$ meaning ϕ is always true,
- $F\phi$ meaning ϕ is finally true,
- $\phi U\psi$ meaning ψ is finally true and ϕ is always true until ψ becomes true,
- and $\phi W\psi$ meaning ϕ is always true until ψ becomes true or ϕ is always true.

Note that F (resp. G) can be defined by U (resp. W). Two path quantifiers are introduced for reasoning about non-determinism:

- $A\phi$ meaning ϕ is true on all paths, and
- $E\phi$ meaning ϕ is true on some path.

In CTL, all temporal operators must be immediately preceded by a path quantifier (e.g. $AFG\phi$ is not in CTL, but $AF(EG\phi)$ is). In the Linear Time Logic, LTL, only temporal operators are used, so the reasoning is only about one single path.

Note that there is a strong duality in the operators:

- $\neg EX(\phi) = AX(\neg\phi)$
- $\neg EF(\phi) = AG(\neg\phi)$
- $\neg EG(\phi) = AF(\neg\phi)$
- $\neg E(\phi\ U\ \psi) = A(\neg\psi\ W\ \neg\phi)$

2.2 Semantics: Kripke Structures

The model theory of temporal logic is given by Kripke structures [13].

Definition 1 *Let AP be a set of atomic propositions. A* **Kripke structure** *M over AP is a four tuple $M = (S, S_0, R, L)$ where*

1. *S is a finite set of states,*

2. *$S_0 \subseteq S$ is the set of initial states, (can missing)*
3. *$R \subseteq S \times S$ is a transition relation that must be total, that is, for every state $s \in S$ there is a state $s' \in S$ such that $R(s, s')$.*
4. *$L : S \to A^{AP}$ is a function that labels each state with the set of atomic propositions true in that state.*

Definition 2 *A* **path** *in a struture M from a state s is an infinite sequence of states $\pi = s_0 s_1 s_2 ...$ such that $s_0 = s$ and $R(s_i, s_{i+1})$ holds for all $i \geq 0$.*

We note π^n the state s_n in the above sequence.

We can now define the semantics of CTL* formulae over AP w.r.t. a Kripke structure over AP as in table 1.

Conjunction and disjunction are defined as usual.

Table 1: Inductive definition of the truth relations $s \models \phi$ and $\pi \models \phi$ in a given Kripke structure.

$s \models \alpha$	iff	$\alpha \in L(s)$,
$s \models E\psi$	iff	there is a path π from s s.t. $\pi \models \psi$,
$s \models A\psi$	iff	for every path π from s, $\pi \models \psi$,
$\pi \models \phi$	iff	$\pi^0 \models \phi$,
$\pi \models X\psi$	iff	$\pi^1 \models \psi$,
$\pi \models F\psi$	iff	there exists $k \geq 0$ s.t. $\pi^k \models \psi$,
$\pi \models G\psi$	iff	for every $k \geq 0$, $\pi^k \models \psi$,
$\pi \models \psi U \psi'$	iff	there exists $k \geq 0$ such that $\pi^k \models \psi'$ and $\pi^j \models \psi$ for all $0 \leq j < k$.
$\pi \models \psi W \psi'$	iff	for all $k \geq 0$, if for every $0 \leq j < k$ $\pi^j \not\models \psi'$ then $\pi^k \models \psi$.

2.3 Generalization/Specialization of a Kripke Structure

We provide a few more definitions about Kripke structures that will be used in Sect. 4.

Definition 3 *Let $K_1 = (S_1, R_1, L_1)$ and $K_2 = (S_2, R_2, L_2)$ be two Kripke structures over AP a set of atomic propositions. such that $S_1 \subseteq S_2$, $R_1 \subseteq R_2$ and $\forall s \in S_1$ $L_2(s) = L_1(s)$. We say that K_2 is* **more general** *than K_1.*

K_2 is more general than K_1 implies that each path in K_1 is a path in K_2. Thus if we add a transition rule in a rule-based model, the new model is more general than the initial model.

Definition 4 *Let* $K_1 = (S_1, R_1, L_1)$ *and* $K_2 = (S_2, R_2, L_2)$ *be two Kripke structures over* AP *a set of atomic propositions. such that* $S_2 \subseteq S_1$, $R_2 \subseteq R_1$ *and* $\forall s \in S_2\ L_1(s) = L_2(s)$, *We say that* K_2 *is* **more specific** *than* K_1.

K_2 is **more specific** than K_1 iff K_1 is **more general** than K_2. Thus if we remove a transition rule in a rule-based model, and the relation of transition remains total, then the new model is more specific than the initial one.

3 BIOCHAM Example

The simplest way to define a Kripke structure is often to use a rule-based language to define R. In the context of Systems Biology, BIOCHAM [1] provides such a language to formalize sub-cellular biological processes as a Kripke structure, by means of biochemical reaction rules.

Here is a simple example of BIOCHAM model based on a cell cycle control model [20]:

```
present(CKI,1).        present(Y,1.5).
...
parameter(k1,0.2).     parameter(k2,0.2).
...
k1                     for _     => CycB.
k2*[CycB]              for CycB => _.
(k2u*[APC])*[CycB]     for CycB =[APC]=> _.
...
(k3*CDK*[CycB],k4*[MPF])   for CycB<=>MPF.
```

After the definition of the initial state of the system and of macros and parameter values, the rules related to the cyclin behavior are given (the full model is available in appendix A). The rules given, corresponding to biochemical reactions, show the synthesis of molecule `CycB`, its degradation, at constant rate and with catalysis of `APC`, and finally the complexation/decomplexation into `MPF`. These rules are given with kinetic expressions which correspond mostly to the mass action law in this example. The kinetic expressions are optional and can be omitted.

3.1 CTL Queries

If we abstract from kinetic expressions, as usually done in large systems, one can associate to the BIOCHAM rules a Boolean Kripke structure in which :

- the states are defined by boolean variables associated to molecules (denoting their presence or absence in the cell);

[1] `http://contraintes.inria.fr/BIOCHAM/`

- the transitions are defined by the reaction rules, taking into account the possible consumption of the reactants by the reaction. A reaction rule with n reactants among which m appear in the right hand side, is thus translated into 2^{n-m} transition rules corresponding to the possible consumption or not of the $n-m$ reactants by the reaction.

As shown in [12, 11], CTL logic provides a very powerful language to formalize the biological properties of a model. Questions of *reachability*, like, is there a pathway for synthesizing a molecule P?, can be formalized by the CTL formula $EF(P)$. Questions about *pathways*, like, can the cell reach a state s while passing by another state s_2? can be formalized by $EF(s_2 \wedge EF(s))$; or is state s_2 a necessary *checkpoint* for reaching state s? by $\neg E((\neg s_2)\ U\ s)$ which is abbreviated as `checkpoint(s2,s)`. Questions about *stability*, like, is a certain (partially described) state s of the cell a steady state? can be formalized by $s \Rightarrow EG(s)$; or, can the cell reach a given permanent state s? by $EF(EGs)$. Also questions about *oscillations*, like, can the system exhibit a cyclic behavior w.r.t. the presence of P? can be formalized by the CTL formula $EG((P \Rightarrow EF\ \neg P) \wedge (\neg P \Rightarrow EF\ P))$, which is abbreviated as `loop(P)`.

Checking the validity of CTL formulae in a given BIOCHAM model provides a form of validation of the model against biological experiments, made in wild life and mutated organisms and formalized as a set of CTL properties. For this task, BIOCHAM is interfaced to the state-of-art model checker NuSMV [10] which uses a very efficient representation of states and transitions by Binary Decision Diagrams.

Here is the result of a BIOCHAM run on the above example, asking if `CycB` is a checkpoint for the activation of `MPF`, then if `Wee1` is one:

```
biocham: check_checkpoint(CycB,MPF).
Ai(!(E(!(CycB) U MPF))) is true

biocham: check_checkpoint(Wee1,MPF).
Ai(!(E(!(Wee1) U MPF))) is false

biocham: why.
   CKI is present
1 _=>CycB.
   CycB is present
4 CycB=>MPF.
   MPF is present
   CycB is absent
Query time: 0.02 s
```

3.2 Constraint LTL Queries

If the kinetic information is available and taken into account, one can associate to the BIOCHAM model a system of Ordinary Differential Equations (ODE) which makes the

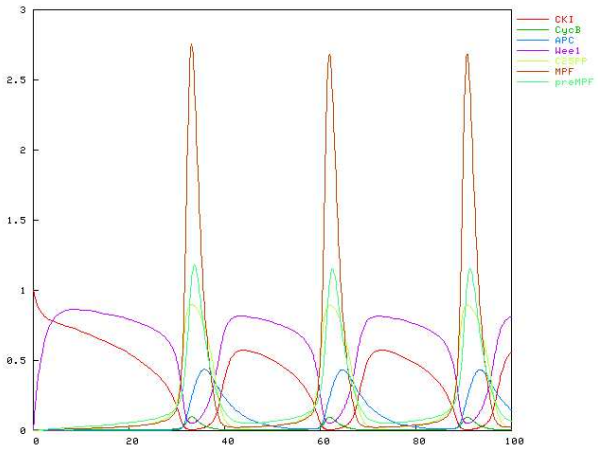

Figure 1: Simulation result for the cell cycle model of [20]

model deterministic and can be simulated with numerical methods. Figure 1 shows the result of the simulation in this example.

The simulation plots (using adaptive step size Runge-Kutta or Rosenbrock's methods) provides a time series which constitutes a linear Kripke structure amenable to LTL querying with constraints. BIOCHAM allows to check the validity of an LTL formulae with arithmetic constraints on molecule concentrations and on their derivatives, similarly to [5]. The LTL formulae with arithmetic constraints formalize properties of the curve that can be automatically checked in a given model or imposed as a specification for revising the model.

Here is the result of a BIOCHAM run, on the same example, where one asks if `MPF` oscillates (i.e. its derivative change its sign) three then four times, on a trace of length 100:

```
biocham: trace_check(oscil(MPF,3)).
oscil(MPF,3) is true.

biocham: trace_check(oscil(MPF,4)).
oscil(MPF,4) is false.
```

4 Learning Reaction Rules from CTL properties

Turning the temporal logic query language into a specification language for expressing the observed behavior of the system, opens the way to the use of machine learning techniques for completing or correcting such formal models semi-automatically.

4.1 Enumerative Algorithm

The intended behavior of a model can be described through a set of CTL properties providing a specification, with positive and negative examples, generalized by logical formulae closed by negation. A rule pattern (the bias) describing the plausible rules to add to the system can be given to guide the search of new rules, eliminating in advance rules having no biological meaning, and we want the system to come up with corrections/completions of the initial model.

After unfruitful experiments with state-of-the-art Inductive Logic Programming tools related to the complexity of temporal properties, even with the simple transitive closure program expressing only reachibility properties, we developed an ad-hoc exhaustive enumeration method: from the rule pattern, all its ground instances are generated, ordered by size and tried one by one, adding them to the model and checking the specification with the model-checker. Those rules which check all the specifications (positive examples and no negative examples) are returned as answers and proposed to the user.

This approach is somewhat limited, since it currently handles only the addition of a single rule to the model, however Sect. 4.2 shows that it already provides interesting results for a certain number of examples.

4.2 BIOCHAM Example

The example of Qu's cell cycle model and another example of the MAPK cascade were, among others, tried out in the following way: some rules were deleted, and BIOCHAM had to find them, in order to satisfy the CTL specification (constructed from the working, original model).

Table 2 summarizes the results and shows that even for models of a moderate size, it is possible to find the missing rules.

In both cases the bias provided was very permissive, and could have been refined by a modeler knowing what kind of rule/process he was looking for.

4.3 Positive and Negative CTL Formulae

Definition 5 *p is a* **positive formula** *iff:*

- *p is a propositionnal formula,*
- *$p = EX(c)$ where c is a positive formula*
- *$p = EF(c)$where c is a positive formula*
- *$p = EG(c)$where c is a positive formula*
- *$p = E(c_1\ U\ c_2)$ where c_1 and c_2 are two positive formulae,*
- *$p = E(c_1\ W\ c_2)$ where c_1 and c_2 are two positive formulae,*

Table 2: Two examples. The bias given defines all possible biological reactions ((de)complexation, synthesis, degradation or (de)phosphorylation)

deleted rule	good rules	tested rules	time
Cell cycle model by Qu et al. [20] (25 rules , 17 molecules, 46 specifications)			
synthesis of CycB deleted	19	1041	345s
inhibition of Wee1 by MPF	14	1041	655s
activation of MPF by C25	2	1041	4680s
activation of C25 by MPF	5	1042	740s
RTK-MAPK cascade by Levchenko et al. [17] (22 rules, 22 molecules, 4 spec.)			
`RAF+RAFK=>RAF-RAFK`	40	1888	1092s
`MEK+RAF~{p1}=>MEK-RAF~{p1}`	33	1888	1570s
`MEK~{p1}+RAF~{p1}=>MEK~{p1}-RAF~{p1}`	23	1888	794s
`MAPK+MEK~{p1,p2}=>MAPK-MEK~{p1,p2}`	68	1888	1382s
`MAPK~{p1}+MEK~{p1,p2}=>MAPK~{p1}-MEK~{p1,p2}`	83	1888	585s
`RAF-RAFK=>RAFK+RAF~{p1}`	35	1887	917s
same rule, other pattern tested	4	877	563s
`MEK~{p1}-RAF~{p1}=>MEK~{p1,p2}+RAF~{p1}`	29	1887	636s
`MEK-RAF~{p1}=>MEK~{p1}+RAF~{p1}`	23	1887	1189s
same rule, other pattern tested	2	877	604s
`MAPK-MEK~{p1,p2}=>MAPK~{p1}+MEK~{p1,p2}`	86	1887	1102s
`MAPK~{p1}-MEK~{p1,p2}=>MAPK~{p1,p2}+MEK~{p1,p2}`	62	1887	535s

Positive formulae are conserved by generalization of a kripke structure.

Proposition 1 (Positive formula conservation) *Let be* $K = (S, S_0, R, L)$ *and* $K' = (S', S_0, R', L)$ *be two kripke structures such that* K' *is more general as* K. *Let* ϕ *a positive formula such that* $K, s \models \phi$ *then* $K', s \models \phi$

Proof. ϕ being positive means that if $K, s \models \phi$ then, there exists a path $\pi = ss_1s_2...$ in the struture K such that $\pi \models \phi$ or $s \models \phi$ (by case on the definition of positive). K' is more general than K thus π is a path in the struture K', and $K', s \models \phi$, or s is a state of S' and $K', s \models \phi$. □

It is worth noting however that the preceding proposition is no longer true for the operators G and W if we impose fairness.

Definition 6 *p is a* **negative formula** *iff:*

- *p is a propositionnal formula,*
- $p = \neg p'$ *where* p' *is an atomic proposition,*
- $p = AX(c)$ *where c is a negative formula*
- $p = AF(c)$*where c is a negative formula*
- $p = AG(c)$*where c is a negative formula*
- $p = A(c_1)U(c_2)$ *with* c_1 *and* c_2 *two negative formulae,*
- $p = A(c_1)W(c_2)$ *where* c_1 *and* c_2 *two negative formulae,*

Proposition 2 (Duality) ϕ *is a positive formula iff* $\neg\phi$ *is a negative formula.*

Proof. By recursion of the size of the formula ϕ. For the base case, $\phi = p$ or $\phi = \neg p$ with p an atomic proposition then $\neg\phi = \neg p$ or $\neg\phi = p$, thus ϕ and $\neg\phi$ are positive and negative formulae.

Let ψ be a negative formula of size $n + 1$, different forms of ψ are:

- $\psi = AX(q)$, q is negative of size n and by induction $\neg q$ is positive, $\neg\psi = \neg AX(q) = EX(\neg q)$ which is thus a positive formula , qed.
- $\psi = AF(q)$ and $\psi = AG(q)$ are treated in the same way.
- $\psi = A(q_1)U(q_2)$ with q_1 and q_2 two negative formulae,$\neg\psi = E(\neg q_2\ W\ \neg q_1)$ is a positive formula by using induction twice.
- $\psi = A(q_1)W(q_2)$ is treated in the same way.
- the same for $\psi = q_1 \vee q_2$ and $\psi = q_1 \wedge q_2$.

□

Proposition 3 (Negative formula conservation) *Let be* $K = (S, S_0, R, L)$ *and* $K' = (S', S_0, R', L)$ *two kripke structures such that* K' *is more specific than* K*. Let* ϕ *a negative formula such that* $K, s \models \phi$ *then* $K', s \models \phi$

Proof. Let us reason by contradiction and suppose that $K', s \not\models \phi$. We have $K', s \models \neg\phi$ where, by duality, $\neg\phi$ is a positive formula. Hence by conservation, we get $K, s \models \neg\phi$, a contradiction. □

In the somewhat related context of incremental model checking introduced in [24], it is worth noticing that the concept of positive and negative formulae provides a syntactic criterion for the formulae which need not be revised.

4.4 Improved Theory Revision Algorithm

The Theory Revision framework [14, 23], of which the enumerative method is an extremely simple instance, should provide more efficient methods for structural learning:

- in order to limit the number of candidate rules according to the CTL specification, in addition to the bias pattern;
- to learn, delete or modify more than one rule.

With respect to this framework, a CTL formula can be either seen as:

- *positive*;
- *negative*;
- or unclassified, for the other formulae.

Thanks to the results of the previous section, this classification is important in order to anticipate whether one has to add or remove rules when trying to make a positive example true (or a negative example false). For instance, if $EF(a)$ is in the specification and is not true in the current model, one needs to **add** a rule (i.e. to generalize the Kripke structure) in order to make it true. If $AG(b)$ is in the specification and is not satisfied, one needs to **remove** a rule (i.e. specialize the Kripke structure).

It is worth noting that the model-checker does not only provide a yes/no answer but can, in certain cases, provide counter-examples for unsatisfied properties. In the above example, typing the command `why` in BIOCHAM after noticing that $AG(b)$ is false, will come up with a path leading to a state where b is absent. At least one rule used in this path need to be removed.

Properties about cyclicity ($AG((a \rightarrow EF(b)) \wedge (b \rightarrow EF(a)))$) remain nevertheless among the *unclassified* properties and can hardly take advantage of theory revision techniques.

A machine learning system taking into account the classification of properties to speed up the learning is currently under development.

5 Learning Parameters from Constraint LTL Properties

5.1 Enumerative Algorithm

In the same spirit as what is done for learning boolean rules from CTL properties, one can use an LTL specification with arithmetic constraints to learn parameter values of a kinetic model. Once again an enumerative method is used, and the search space is explored with a precision specified by the modeler. For each set of parameters tried, a simulation is run, and the resulting time series is used as a Linear Time Logic model on which the specifications are checked.

In a sense, the machine learning process actually replicates what most modelers do by hand, i.e. trying different values for parameters, guided by ideas about the plausible interval of values to try and the *shape* that the simulation should produce. The machine learning algorithm allows us to test parameter sets much faster once the formalization effort of that shape into an LTL specification is done.

It is worth noticing that this method applies to highly non-linear systems and proves particularly effective in this context.

5.2 Example in BIOCHAM

Starting from the same example as usual, let us "erase" the values of parameters k1 and k4 (put them to 0). The model doesn't oscillate at all any more.

Acting as a modeler we try to find values for these parameters such that the system oscillates.

The command `trace_get([k1,k4], [(0,5), (0,5)], 20, oscil(Wee1,3), 100)` searches for two parameters (`k1` and `k4`) in the interval of possible values $[0, 5]$, with only 20 different values tried for each, and such that before time 100, `Wee1` oscillates 3 times. The output is as follows:

```
biocham: trace_get([k1,k4],[(0,5),(0,5)],20,
   oscil(Wee1,3),100).
Search time: 9.76 s
Found parameters that make oscil(Wee1,3) true:
parameter(k1,0.25).
parameter(k4,0).
```

We find very fast some values, but obtain an oscillation a bit too fast and where `CycB` stays very low (see Fig. 2). This might be enough, but if we want to obtain results closer to the published ones, we can refine our query: we give a more precise interval for `k1`, since we see that low values are enough to get an oscillating system; we also ask for the precise number of oscillations that we want, and force an activation of `CycB`.

```
trace_get([k1,k4],[(0,1),(0,5)],10,
   oscil(Wee1,3)&!(oscil(Wee1,4))&F([CycB] gt
```

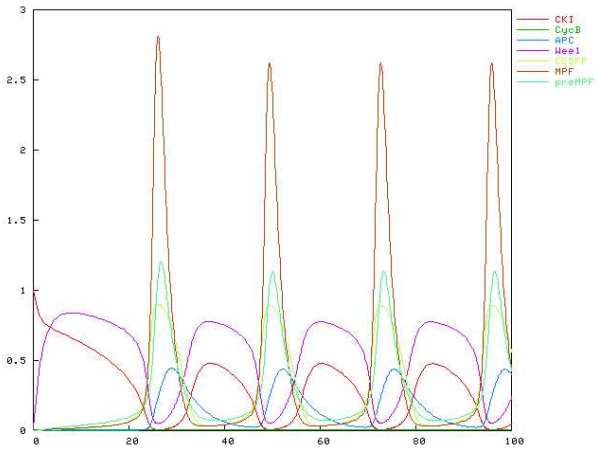

Figure 2: Simulation result after a first round of parameter learning

```
0.08),100)..
Search time: 48.03 s
Found parameters that make oscil(Wee1,3)
   &!(oscil(Wee1,4))&F([CycB] gt 0.08) true:
parameter(k1,0.2).
parameter(k4,1).
```

The resulting simulation fits almost perfectly with the original one, which shows a weak sensitivity on `k4` for this model.

It is also possible to start with very wide intervals, like $[1, 100]$, and a quite loose specification. Then once one gets an answer, the specification can be made more precise, and the intervals smaller if necessary.

6 Conclusion

With the advent of formal languages for describing on the one hand, complex systems, and on the other hand, their expected temporal properties, the design of machine learning tools becomes possible, in order to semi-automatically correct or complete a system implementation w.r.t. its behavioural specifications.

In the general settings of rule-based languages describing (non-deterministic) concurrent transition systems, and Computation Tree Logic formalizing temporal properties of the system, we have shown that a relatively simple enumerative algorithm can be used to find missing rules to satisfy a temporal specification. Furthermore, we have proposed a classification of CTL formulae into positive and negative formulae in order to design a more involved

theory revision algorithm making use of the temporal specification actively to reduce the search space.

We have also shown that the same approach can be applied to the learning of numerical parameters in hybrid systems equipped with a continuous dynamics, from a specification of their behaviour in temporal logic with numerical constraints.

These results have been illustrated with examples coming from the Biochemical Abstract Machine BIOCHAM which offers a modeling environment for Systems Biology at two levels of abstraction: the Boolean abstraction which abstracts from the quantities of molecules, and the concentration semantics which deals with kinetic laws. In this context, our first experiments are very encouraging and proved already useful in some problems of decomposition of interactions and parameter fitting. More generally, we believe that the machine learning algorithm proposed for temporal logic with numerical constraints, may also prove succesful as an original optimization numerical method for highly non-linear systems.

A BIOCHAM model after [20]

```
present(CKI,1).
absent(preMPF).       absent(MPF).
absent(C25).          absent(C25P).
absent(C25PP).        absent(Wee1).
absent(Wee1P).        absent(APC).
absent(C).            absent(CP).

macro(CDK,((c0-[preMPF]-[MPF]-[C]-[CP])/c0)).
macro(MPFT,[MPF]+[preMPF]).

%% Cyclin
k1                    for _=>CycB.
k2*[CycB]             for CycB=>_.
(k2u*[APC])*[CycB]    for CycB=[APC]=>_.

(k3*CDK*[CycB],k4*[MPF]) for CycB<=>MPF.

(k5*[preMPF])      for preMPF=>MPF.
[C25PP]*[preMPF]   for preMPF=[C25PP]=>MPF.
k6*[MPF]           for MPF=>preMPF.
[Wee1]*[MPF]       for MPF=[Wee1]=>preMPF.
k7*[MPF]           for MPF=>_.
k7u*[APC]*[MPF]    for MPF=[APC]=>_.

%% Cdc25
k8               for _=>C25.
k9*[C25]         for C25=>_.
k9*[C25P]        for C25P=>_.
```

```
k9*[C25PP]      for C25PP=>_.

bz*[C25]          for C25=>C25P.
cz*[MPF]*[C25]    for C25=[MPF]=>C25P.
az*[C25P]         for C25P=>C25.

bz*[C25P]         for C25P=>C25PP.
cz*[MPF]*[C25P]   for C25P=[MPF]=>C25PP.
az*[C25PP]        for C25PP=>C25P.

%% Wee1
k10            for _=>Wee1.
k11*[Wee1]     for Wee1=>_.
k11*[Wee1P]    for Wee1P=>_.

bw*[Wee1]         for Wee1=>Wee1P.
cw*[MPF]*[Wee1]   for Wee1=[MPF]=>Wee1P.
aw*[Wee1P]        for Wee1P=>Wee1.

%% APC
(([MPF]*[MPF])/(a*a+([MPF]*[MPF])))/tho
           for _=[MPF]=>APC.
 [APC]/tho  for APC=>_.

%% CKI
k12         for _=>CKI.
k13*[CKI]   for CKI=>_.

(k14*[CKI]*[MPF],k15*[C])  for CKI+MPF<=>C.
bi*[C]                     for C=>CP.
ci*[MPF]*[C]               for C=[MPF]=>CP.
ai*[CP]                    for CP=>C.

k16*[CP]             for CP=>MPF.
k16u*[APC]*[CP]      for CP=[APC]=>MPF.

parameter(k1,0.2).      parameter(k2,0.2).
parameter(k3,30).       parameter(k4,1).
parameter(k5,0.7).      parameter(k6,0.7).
parameter(k7,0).        parameter(k8,1).
parameter(k9,1).        parameter(k10,0.5).
parameter(k11,0.5).     parameter(k12,1).
parameter(k13,1).       parameter(k14,50).
parameter(k15,0.1).     parameter(k16,1).
parameter(k2u,2).       parameter(k7u,2).
parameter(k16u,5).      parameter(c0,200).
```

```
parameter(a,1).          parameter(tho,5).
parameter(az,1).         parameter(aw,1).
parameter(ai,1).
parameter(bz,0.1).       parameter(bw,0.1).
parameter(bi,0.1).
parameter(cz,10).        parameter(cw,10).
parameter(ci,2).
```

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. International Conference on Data Engineering*, 1995.

[2] J. Allen. Time and time again: The many ways to represent time. *International Journal of Intelligent System*, 6(4), 1991.

[3] Nicos Angelopoulos and Stephen H. Muggleton. Machine learning metabolic pathway descriptions using a probabilistic relational representation. *Electronic Transactions in Artificial Intelligence*, 7(9), 2002. also in Proceedings of Machine Intelligence 19.

[4] Nicos Angelopoulos and Stephen H. Muggleton. Slps for probabilistic pathways: Modeling and parameter estimation. Technical Report TR 2002/12, Department of Computing, Imperial College, London, UK, 2002.

[5] 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.

[6] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 16:123–191, 2000.

[7] 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.

[8] Christopher H. Bryant, Stephen H. Muggleton, Stephen G. Oliver, Douglas B. Kell, Philip G. K. Reiser, and Ross D. King. Combining inductive logic programming, active learning and robotics to discover the function of genes. *Electronic Transactions in Artificial Intelligence*, 6(12), 2001.

[9] Laurence Calzone, Nathalie Chabrier-Rivier, François Fages, Lucie Gentils, and Sylvain Soliman. Machine learning bio-molecular interactions from temporal logic properties. In Gordon Plotkin, editor, *CMSB'05: Proceedings of the third Workshop on Computational Methods in Systems Biology*, 2005.

[10] Roberto Cavada, Alessandro Cimatti, Emmanuele Olivetti, Marco Pistore, and Marco Roveri. *NuSMV 2.1 User Manual.* CMU and ITC-irst, IRST - Via Sommarive 18, 38055 Povo (Trento) - Italy, 1998–2002.

[11] 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.

[12] 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.

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

[14] Luc de Raedt. *Interactive Theory Revision, an inductive Logic Programming Approach.* Knowledge-Based Systems. academic press, 1992.

[15] 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.

[16] F. Hoppner. Discovery of temporal patterns—learning rules about the qualitative behaviour of time series. In *Proc. European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2001.

[17] Andre Levchenko, Jehoshua Bruck, and Paul W. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *PNAS*, 97(11):5818–5823, May 2000.

[18] H. Mannila, H. Toivonen, and A. Verkamo. Discovering frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[19] Stephen H. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13:245–286, 1995.

[20] Zhilin Qu, W. Robb MacLellan, and James N. Weiss. Dynamics of the cell cycle: checkpoints, sizers, and timers. *Biophysics Journal*, 85(6):3600–3611, 2003.

[21] J. Rodriguez, J. Alonso, and H. Bostrom. Learning first order logic time series classifiers: Rules and boosting. In *Proc. European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2001.

[22] Udaya A. Shankar. An introduction to assertional reasoning for concurrent systems. *ACM Computing Surveys*, 25(3):225–262, 1993.

[23] Ehud Y. Shapiro. *Algorithmic Program Debugging.* The MIT Press Classics Series, April 1983.

[24] Oleg Sokolsky and Scott A. Smolka. Incremental model checking in the modal mu-calculus. In *CAV '94: Proceedings of the 6th International Conference on Computer Aided Verification*, pages 351–363. Springer-Verlag, 1994.

[25] M. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2), 2001.

Contents

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399