Machine Learning Bio-molecular Interactions from Temporal Logic Properties

Laurence Calzone, Nathalie Chabrier-Rivier,
François Fages, Lucie Gentils, Sylvain Soliman

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

Abstract. With the advent of formal languages for modeling bio-molecular interaction systems, the design of automated reasoning tools to assist the biologist becomes possible. The biochemical abstract machine BIOCHAM software environment offers a rule-based language to model bio-molecular interactions and an original temporal logic based language to formalize the biological properties of the system. Building on these two formal languages, machine learning techniques can be used to infer new molecular interaction rules from temporal properties. In this context, the aim is to semi-automatically correct or complete models from observed biological properties of the system. Machine learning from temporal logic formulae is quite new however, both from the machine learning perspective and from the Systems Biology perspective. In this paper we present an *ad hoc* enumerative method for structural learning from temporal properties and report on the evaluation of this method on formal biological models of the literature.

1 Introduction

Recent progress in molecular biology and high-throughput data-production technologies have renewed the quest for a global understanding of molecular processes at the system level. Systems Biology is a cross-disciplinary field involving biologists, computer scientists, logicians, mathematicians, physicists, aiming at breaking the complexity walls to reason about biological systems in their own right.

The language approach to Systems Biology aims at designing formal languages for describing bio-molecular mechanisms, processes and systems at different levels of abstraction. This approach has developed rapidly since the pioneering use of the π-calculus process algebra for modeling biochemical processes in [1]. Recently, formal languages have also been proposed to model the biological properties of the system in temporal logics [2, 3]. In our work on the biochemical abstract machine BIOCHAM [4], we propose a rule-based language to model bio-molecular interactions, and an original temporal logic based language to formalize the biological properties of the system at hand. Our first experimental

results of temporal logic querying have been reported on a qualitative model of the mammalian cell cycle control developed after Kohn's map [5], involving about 500 variables and 2700 reaction rules [6].

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. There has been work on the use of machine learning techniques, such as inductive logic programming [7], to infer gene functions [8], metabolic pathway descriptions [9, 10] or gene interactions [11]. 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.

In this paper we present an *ad-hoc* machine learning algorithm to discover bio-molecular interaction rules from a partial model and constraints on the system behavior. These constraints are expressed in temporal logic with positive formulae (expected properties) and negated formulae (properties to avoid). The learning process can be guided by the user by providing interaction patterns for limiting the types of sought reactions, such as complexation, phosphorylation.

In the next section, we briefly present the BIOCHAM syntax of bio-chemical compounds, interaction rules, patterns, and temporal logic formulae for expressing biological properties. In Sect. 3, we present our *ad-hoc* enumerative learning algorithm in the general framework of theory revision learning algorithms. In Sect. 4, we evaluate this method on different examples[1] concerning the inference of interaction rules in models of the cell cycle control, the refinement of models of inhibition and activation, and the discovery of drug targets in the cell cycle. Finally we conclude on the scalability of this approach and plans for future work.

2 The Biochemical Abstract Machine BIOCHAM

The Biochemical Abstract Machine BIOCHAM [12] provides precise semantics to bio-molecular interaction maps at two abstraction levels: the quantitative level of molecular concentrations, and the qualitative level of boolean values. In this paper, we focus on the boolean abstraction level of BIOCHAM, and do not consider kinetics and BIOCHAM numerical models.

Based on these formal semantics, BIOCHAM offers:

- a compositional rule-based language for modeling biochemical systems, allowing patterns (and kinetic expressions when numerical data are available);
- a non-deterministic boolean simulator (and numerical simulator);
- a powerful query language based on temporal logic (CTL [13] for boolean models and LTL with constraints for numerical models) for expressing biological queries such as reachability, checkpoints, oscillations or stability;

[1] All data used in this paper are available on the web at `http://contraintes.inria.fr/BiochamLearning`. BIOCHAM is a free software downloadable from `http://contraintes.inria.fr/BIOCHAM`

– a machine learning system to infer interaction rules from observed temporal properties, that is the main subject of this article.

BIOCHAM manipulates formal objects which represent chemical or biochemical compounds, ranging from ions, to small molecules, macromolecules and genes. BIOCHAM objects can be used also to represent control variables and abstract biological processes. BIOCHAM reaction rules are used primarily to represent biochemical reactions. They can also be used to represent state transitions involving control variables or abstract processes, such as protein synthesis by DNA transcription without introducing RNAs in the model.

Syntax:

object = molecule | abstract
molecule = name | molecule-molecule | molecule~{name,...,name}
| gene | (molecule)
gene = #name
abstract = @name
reaction = name: reaction
| solution => solution | solution =[object]=> solution
| solution =[solution => solution]=> solution
| solution <=> solution | solution <=[object]=> solution
solution = _ | object | solution + solution | (solution)

The following abbreviations can be used for reaction rules: `A<=>B` for the two symmetrical rules, `A=[C]=>B` for the rule `A+C=>B+C` with catalyst molecule `C`, and `A=[C=>D]=>B` for the rule `A+C=>D+B`. For instance, `RAF + RAFK => RAF-RAFK` is a complexation rule. `MEK =[RAF~{p1}]=> MEK~{p1}` is a phosphorylation rule with catalyst `RAF~{p1}`. This rule is equivalent to `MEK + RAF~{p1} => MEK~{p1} + RAF~{p1}`. Following an uniqueness assumption, objects marked as "genes" with the '#' notation, or any compound built on such a molecule (such as `DMP1-#p19ARF` for instance) are not multiplied. These objects remain unique and are deterministically consumed in the form in which they appear in the left-hand side of the rule. The same goes for control variables, noted with a '@', which are deterministically consumed. We refer to [12] for the precise semantics of these reaction rules at the boolean abstraction level, which mainly reflects the capability of reasoning about all possible behaviors of the system with unknown concentration values and unknown kinetics parameters.

BIOCHAM has also a rich pattern language with constraints which is used to specify molecules and sets of reaction rules in a concise manner. Patterns introduce the special character ? and variables noted with a name beginning with a $ to denote unspecified parts of a molecule. These variables can be constrained with simple set constraints. For instance, the command `list_rules(RAF ?-? + ? => ?).` contains a pattern matching all rules reacting with any form (phosphorylated or complexed) of RAF. The command `list_rules(? =[RAFK]=> ?).` matches all rules involving the catalyst RAFK, i.e. having RAFK in their left and right-hand sides, even if they were not written with the catalyst notation. Patterns are also used to define sets of rules in a concise way and to provide a

guideline on the shape of rules to be considered during the learning process, as explained in Sect. 3 and 4.

The most original feature of BIOCHAM however, is its use of the Computation Tree Logic CTL [13] as a query language for the temporal properties of boolean models. This methodology introduced in [3, 6] is implemented in BIOCHAM with an interface to the state-of-the-art symbolic model checker NuSMV [14]. CTL basically extends propositional logic used for describing states, 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 transition, $G\phi$ meaning ϕ is always true, $F\phi$ meaning finally true, and $\phi U\psi$ meaning ϕ is always true until ψ becomes true. 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. We refer to [4, 12] for the precise semantics of CTL formulae in BIOCHAM models of bio-molecular interaction networks. We recall here that CTL is expressive enough to express a wide range of biological queries.

About reachability. Is there a pathway for synthesizing a protein P? This query is formalized by the CTL formula $EF(P)$. It can be abbreviated as `reachable(P)` in BIOCHAM.

About pathway. Can the cell reach a state s while passing by another state s_2? This is expressed in CTL by $EF(s_2 \wedge EF(s))$. Is state s_2 a necessary *checkpoint* for reaching state s? $\neg E((\neg s_2)\ U\ s)$. This formula is abbreviated as `checkpoint(s2,s)`. Can the cell reach a state s without violating certain constraints c? $E(c\ U\ s)$. Is it possible to synthesize a protein P without creating nor using protein Q? $E(\neg Q\ U\ P)$.

About stability and oscillations. Is a certain (partially described) state s of the cell a steady state? $s \Rightarrow EG(s)$; a permanent state? $s \Rightarrow AG(s)$. Can the cell reach a given permanent state s? $EF(AGs)$. Must the cell reach a given permanent state s? $AF(AGs)$. Can the system exhibit a cyclic behavior w.r.t. the presence of a product P? This query can be formalized by the CTL formula $EG((P \Rightarrow EF\ \neg P) \wedge (\neg P \Rightarrow EF\ P))$. It will be abbreviated as `loop(P)` in the following. That formula expresses that there exists a path where at all time points whenever P is present it becomes eventually absent, and whenever it is absent it becomes eventually present.

3 Machine Learning from Temporal Logic Properties

Systems biologists build models of bio-molecular interactions from experiments in wild-life and mutated organisms. These experiments tell them the properties that their model have to check, concerning the behaviour of the system under various conditions. In BIOCHAM, we have shown that most of these biological

properties can be formalized in temporal logic CTL [4, 3]. When a model does not satisfy all these properties, the machine learning system of BIOCHAM proposes rules to be added or removed, in order to improve the model.

3.1 An *Ad hoc* Enumerative Algorithm

The intended behavior of the model can thus be described through a set of CTL properties providing a specification, with positive and negative examples. 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, we developed an ad-hoc exhaustive enumeration method: from the rule pattern, we generate all its ground instances, order them by size and try them 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 shows that it already provides interesting results for a certain number of examples.

3.2 Theory Revision

The Theory Revision framework [15] , of which the above 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*, i.e. if it is false, it will remain false when removing a rule (like $EF(\phi)$ or $EG(\phi)$ where ϕ is state description formula containing no CTL operator);
- *negative*, i.e. if it is false, it will remain false when adding a rule (like $AG(\phi)$ where ϕ is a state description formula);
- or unclassified, for the other formulae.

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 in order to make it true. If $AG(b)$ is in the specification and is not satisfied, one needs to **remove** a rule to make it true. With this respect, it is important to note 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. Only rules used in this path need to be tried for removal.

It is also possible to convert *unclassified* properties into *classified* ones. In the case of $\neg E(\neg aUb)$, the translation of "a is a checkpoint for the production of b", we have for example an *unclassified* property. However, if this property is false but b is reachable (i.e. $EF(b)$), we get a *negative* property, since we know that the counter-example will be a path leading to b without going through a and that to make the checkpoint property true, one needs to remove one of the rules of this path. Properties about cyclicity remain nevertheless among the *unclassified* properties and can hardly take advantage of theory revision techniques.

3.3 Evaluation

For the purpose of evaluating the machine learning system, we start from a given BIOCHAM model, a set of temporal properties that the model satisfies is written down (in temporal logic, the formula can be negated, providing thus positive and negative examples), one reaction rule is erased and the purpose of the learning process is to let it recover that rule or discover other rules which permit to satisfy the temporal properties. In this paper, we report mainly the results of the ad-hoc enumerative algorithm, and give in Sect. 4.2 an example of a simplified use of the Theory Revision approach.

4 Examples and performances

We have collected pathway data from different sources in order to try out and evaluate pathway correction/completion based on a temporal logic specification. The first evaluations are based on well-known models, taken from the literature or imported from the Web using the standard SBML format [16], from which one interaction rule is erased. Different specifications, given in temporal logic, have been proposed for the models (more or less complete). The result is not only whether the system recovers the expected rule or not, but also if it produces other rules satisfying the property, and if the biases expressed as interaction rule patterns are expressive enough.

4.1 Rule inference in cell cycle control

The model . This model is a BIOCHAM boolean model imported from an SBML model[2] which itself was taken from the literature [17]. It provides a very basic (6 variables) model of the cell cycle control. The input of the machine learning system is in three parts:

1. A set of interaction rules:

[2] http://www-aig.jpl.nasa.gov/public/mls/cellerator/notebooks/Tyson6.html

```
_ => cyclin.
cdc2~{p} + cyclin => cdc2~{p}-cyclin~{p}.
cdc2~{p}-cyclin~{p} => cdc2-cyclin~{p}.
cdc2-cyclin~{p} => cdc2 + cyclin~{p}.
cyclin~{p} => _.
cdc2 => cdc2~{p}.
cdc2~{p} => cdc2.
```

2. An initial state. Only the kinase cdc2 is present, the other molecules are absent.
3. A set of specifications in temporal logic CTL, verified by the biological model, like the reachability of the activated complex kinase-cyclin (also called MPF), or the oscillation of the cycle's phase, plus some other simple properties.

```
reachable(cdc2-cyclin~{p}).
loop((cyclin & cyclin~{p}) & !(cdc2-cyclin~{p})).          ...
```

Finding a missing rule. For the first test we delete one rule, the activation rule of MPF (cdc2-cyclin)by dephosphorylation: `cdc2~{p}-cyclin~{p} => cdc2-cyclin~{p}` The model does not verify all CTL specifications any more. The purpose is to let the system learn few but biologically correct rules to complete the model. When asking to find a rule which is a transformation of a molecule into another, or a degradation or a synthesis, the system tests 45 rules, in 4 seconds, and returns only 3 as possible answers:

```
? learn([$Q=> $P where $P in complexes and $Q in complexes]).
_=>cdc2-cyclin~{p}
cyclin=>cdc2-cyclin~{p}
cdc2~{p}-cyclin~{p}=>cdc2-cyclin~{p}
```

From this point, the biologist can look at each rule and notice that they all produce active MPF. This is already a clue about what is missing in the model.

One can also ask more to the system, by making the rule pattern more precise to force the learning of a rule with a stronger biological meaning, like a phosphorylation. In that case, fewer rules (11) are tested and, in less than 1 second, the missing rule is found.

```
? learn([$qp=> $q where $q in complexes and $qp modif $q]).
cdc2~{p}-cyclin~{p}=>cdc2-cyclin~{p}
```

A third possibility is to add a specification like the fact that the presence of phosphorylated cdc2 is necessary to activate MPF
`checkpoint(cdc2~{p},cdc2-cyclin~{p})`. The same set of rules (45) as the first query is tested but only one rule (the missing rule) then verifies all the specifications.

```
? learn([$Q=> $P where $P in complexes and $Q in complexes]).
cdc2~{p}-cyclin~{p} => cdc2-cyclin~{p}
```

Table 1 summarizes the results of this process. To show the scalability of the method we used it on some bigger examples, results are reported in table 2.

Finding a process. During the building of a biological model, it is often a whole process that remains to be modeled, and not just a single rule. In order to simulate this problem, we delete the set of rules of the formation of the MPF complex and its activation: `delete_rules({cdc2~{p} + cyclin => cdc2~{p}-cyclin~{p}, cdc2~{p}-cyclin~{p} => cdc2-cyclin~{p}})`. The inactive form of the MPF complex is now unknown by the model, and we try to find a rule which completes the model and gives some directions for biological experiments. We first try to find a complexation or a phosphorylation but no rule is found to complete the model from 23 rules tested in 1.7 seconds.

```
? learn([$qp=>$q where $q in complexes and $qp modif $q,
         $p + $q=>$p-$q where $q in complexes and $p in complexes]).
No rule
```

The pattern can be generalized to a "synthesis", but not necessarily with a strict biological meaning. This can be interpreted like the simple presence of a molecule in the system. Once again, no rule is found to complete the model, and we can deduce that the "hole" in the model is not a only the production of a molecule.

If the pattern is further generalized to a consumption and a production. Three rules are proposed by the system (from 32 rules in 2.3 seconds), each producing active MPF and consuming the cdc2 kinase or the cyclin.

```
? learn([$R=> $P where $P in complexes and $R in complexes]).
cdc2=>cdc2-cyclin~{p}
cyclin=>cdc2-cyclin~{p}
cdc2~{p}=>cdc2-cyclin~{p}
```

The production of active MPF being apparently crucial, one queries for a rule which would produce MPF by complexation and/or phosphorylation and/or dephosphorylation. In 4 seconds, 51 rules are tested and 2 are proposed to complete the model. And if we make the presence of phosphorylated cdc2 compulsory to produce active MPF (checkpoint property), only one rule is proposed:

```
? learn([$R+$Q=>$Rp-$Qp where $Q in complexes and  $R in complexes
                        and $Rp modif $R and $Qp modif $Q]).
cdc2~{p}+cyclin=>cdc2-cyclin~{p}
```

This rule is a shortcut for the process which we had removed. See table 3 for a summary of the results and table 4 for reports of other experiments on a bigger example.

4.2 Model Refinement by Theory Revision

The enumeration method described up to now can be embedded in a slightly more complex interactive model refinement method based on Theory Revision.

A simple model. Let us consider a small hypothetical system composed of three proteins, MA, MB and MC, in which oscillations are experimentally

observed. The exact interactions among proteins are unknown and the modeler seeks for diverse possibilities. In absence of other knowledge, one can start from the simplest boolean model: `_<=>MA. _<=>MB. _<=>MC.`

Temporal properties are specified, defining both oscillations and reachability for each protein. So far, the model does not take into account protein interactions. These will be discovered, one by one, based on experimental observations formalized with CTL formulae, checking that the previous properties imposed on the system still hold (i.e. oscillation, and reachability of each protein).

The first refinement of the model comes from the observation that MC is needed for the disappearance of MB. The specification is written in temporal logic in the form of a checkpoint: `checkpoint(MC,!MB).` However, the current model does not verify this new property. In other words, there exists a path where MB can disappear even though MC has already been degraded. A rule is chosen and deleted from the path given by BIOCHAM. At this point, the new model can either verify the checkpoint specifications or they are not verified, we ask to infer a rule that would account for the interaction between MB and MC. Three rules are proposed by the system: `MB+MC=>MB~{p}+MC. MB+MC=>MC. MB+MC=>MB-MC.` One of the rules is chosen, the first one, and added to the model. The model then verifies all the properties, i.e. oscillations, reachability and `checkpoint(MC,!MB).`

The second refinement of the model comes from the fact that MA is needed for the disappearance of MC which is formalized as follows: `checkpoint(MA,!MC).` New rules are proposed to verify the new specifications of the model and added until all the observations are included.

It might happen that a rule chosen in the first step of refinement leads to a dead end when a second step of refinement is proposed and for which no rule can be found. In this case, it is possible to go back to the previous step and choose a more appropriate rule.

Scenario of refinement by simplified theory revision. The following algorithm summarizes the method:

1. If the model is satisfactory, stop, else add a new specification and go to 2.
2. Check if the model verifies all the specifications, if it does go to 1, else go to 3.
3. Ask BIOCHAM why the properties aren't verified with the command `why`. *Choose* a rule from the proposed pathway and delete it.
4. Check if the model is now valid , if it is go to 1, else go to 5.
5. Learn rules to complete the model with the command `learn`. If at least one is found, then *choose* one of them and add it to the model and go to 1, else go back to the previous choice (in 3).

4.3 Results

Assuming that the proteins can appear under two forms, active (unphosphorylated) and inactive (phosphorylated), 40 seconds and 2 backtracks (steps back from 5 to 3) were needed to find the final model:

```
_=>MA.  MA=[MB]=>_.  _=>MB.  MB=[MC]=>MB~{p}.  _=>MC.  MC=[MA]=>MC~{p}.
```

This model is an example of a three-component negative feedback loop where MA inhibits MC, MC inhibits MB and in turn, MB inhibits MA. This model is an example of a minimal system for an oscillatory behavior in numerical models assuming that an appropriate parameter set is chosen.

4.4 Scenario of drug target discovery in cell cycle control

The machine learning method described in this paper can be applied to concrete matters found in pharmaceutical research. Even though the example given here concerns a small model, the impact of such methods seems quite promising for the future in finding and testing medicines.

To illustrate the learning method, we use a modified version of a cell cycle model (Qu et al. [18]) involving 6 proteins. In this model, a kinase, Cdk, binds to a cyclin, CycB, to form a complex, `Cdk+CycB<=>Cdk-CycB`, which plays a central role in the cycle. When Cdk-CycB is active, the cell enters into mitosis and exits mitosis when Cdk-CycB is inactivated. Around this complex, many pathways and networks of proteins are organized to ensure that it is active at the right moment of the cycle. Cdk-CycB can be transformed into its inactive form by a kinase Wee1, `Cdk-CycB=[Wee1]=>Cdk-CycB~{p1}`. The reverse reaction is done by a phosphatase Cdc25 (noted C25): `Cdk-CycB~{p1}=[C25~{p1,p2}]=>Cdk-CycB`. The complex Cdk-CycB and the two proteins interact into a positive feedback loop, Cdk-CycB activating Cdc25 in two steps, `C25=[Cdk-CycB]=>C25~{p1}` and `C25~{p1}=[Cdk-CycB]=>C25~{p1,p2}`, and inactivating Wee1 :

`Wee1=[Cdk-CycB]=>Wee1~{p1}`.

Moreover, antagonist proteins interfere to keep the complex off: the inhibitors CKI that bind to Cdk-CycB to form an inactive complex, CKI-Cdk-CycB, `CKI+Cdk-CycB<=>CKI-Cdk-CycB`, and the proteases (called APC) that degrade the cyclin part of Cdk-CycB, `CycB=[APC]=>_`.

Adding a molecule to block the cycle. The cell cycle described above exhibits oscillations. In this example, the purpose is to find a protein capable of blocking the cycle. The method consists in introducing an unknown and hypothetical molecule to the model and determining what kind of interactions between this new molecule and other components of the model lead to an arrest of the cell cycle. For this purpose, a specification is added such that the introduction of this new molecule, `NewMol`, to the model stops the oscillations observed in normal conditions. BIOCHAM proposes 11 rules that will arrest the cycle out of the 81 tested in 365 seconds. Only five rules seem to have a biological meaning and the other six ones are dismissed. The remaining rules are the following:

```
Cdk-CycB+New_mol=>Cdk-CycB-New_mol.     (1)
C25=[New_mol]=>C25~{p1,p2}.             (2)
Cdk-CycB=[New_mol]=>Cdk-CycB~{p1}.      (3)
Cdk-CycB~{p1}=[New_mol]=>Cdk-CycB.      (4)
Cdk-CycB=[New_mol]=>_.                  (5)
```

All of them show direct or indirect interactions with Cdk-CycB: either the complex is kept active (the activation of MPF or Cdc25 is forced by rules 2,

4), or the complex is inhibited (rules 1, 3) or degraded (rule 5). If experiments needed to be done to block the cell cycle, this method could suggest specific properties that a protein, a pathway or a network of proteins should have to cause this arrest.

Adding a molecule to unblock the cycle. In this case, the problem is reversed. Starting from a cell arrested in one of the phases of its cycle, a new molecule is added to unblock it. First of all, the cycle needs to be blocked by deleting one important component, for example Cdc25 (a deletion of Cdc25 is lethal in most cells). Since Cdc25 is involved in a two-step process where Cdk-CycB activates Cdc25, which in turn activates Cdk-CycB more, it can be anticipated that the new molecule might have similar dynamics. Therefore, a reaction rule is added such that the new molecule is forced to interact and activate Cdk-CycB: `Cdk-CycB~{p1}=[NewMol]=>Cdk-CycB`. However, it is still unclear how NewMol has to be regulated in the system. The function "learn" proposes six rules that verify the specifications of the model and that suggest ways to unblock the cycle. Only three of them are of interest:

```
_=[CycB]=>NewMol.  _=[Cdk-CycB]=>NewMol.  Wee1~{p1}=>Wee1~{p1}+NewMol.
```

The proposed rules indicate that the new protein needs to be activated directly or indirectly by the complex Cdk-CycB. Even in the case that the rules are not totally biologically correct, they provide useful hints in terms of the necessary processes to unblock the cell cycle.

5 Conclusion

Computation Tree Logic (CTL) is a powerful formalism for expressing the biological properties of an organism, such as state reachability, checkpoints, stability and oscillations. The scalability of symbolic model checking tools to evaluate CTL formulae has been shown in large interaction networks of several hundreds of variables in [6, 3] using the BIOCHAM system [4, 12]. Here we have shown how CTL could be turned into a specification language for the observed behavior of a living system. The *ad-hoc* machine learning technique that we have presented for learning interaction rules from CTL specification, has been evaluated on small models. However we have shown that the expressivity of rule patterns makes it possible to direct the search so that the performances of the learning process scale up linearly with the number of reaction rules. The flexibility of rule patterns makes it possible also to use the learning process in an interactive way for refining models. This capability has been illustrated with a scenario of drug target discovery.

These results are thus quite encouraging for the design of automated reasoning tools to assist the biologist/modeler in Systems Biology. However, the boolean abstraction level of bio-molecular interactions considered in this paper is a rather crude abstraction, as it abstracts from the kinetics of reactions. It should thus be considered as a first step for the learning of bio-molecular interaction rules with kinetics. In particular, the BIOCHAM language supports the

specification of the kinetics of reactions as well, and compiles them into a system of ordinary differential equations. Furthermore, the kinetics of BIOCHAM reactions can be used not only for simulations but also for checking temporal properties expressed in a constraint-based extension of Linear Time Logic (as ODE systems are deterministic). We are thus currently investigating the generalization of our machine learning algorithm to BIOCHAM numerical models along the same lines.

Acknowledgments

This work has been partly supported by the European STREP project APRIL II[3]. We are especially grateful to our colleagues Stephen Muggleton and Luc de Raedt for enlightening discussions.

References

1. Regev, A., Silverman, W., Shapiro, E.Y.: Representation and simulation of biochemical processes using the pi-calculus process algebra. In: Proceedings of the sixth Pacific Symposium of Biocomputing. (2001) 459–470
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. 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
5. Kohn, K.W.: Molecular interaction map of the mammalian cell cycle control and DNA repair systems. Molecular Biology of the Cell **10** (1999) 703–2734
6. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biochemical interaction networks. Theoretical Computer Science **325** (2004) 25–44
7. Muggleton, S.H.: Inverse entailment and progol. New Generation Computing **13** (1995) 245–286
8. Bryant, C.H., Muggleton, S.H., Oliver, S.G., Kell, D.B., Reiser, P.G.K., King, R.D.: Combining inductive logic programming, active learning and robotics to discover the function of genes. Electronic Transactions in Artificial Intelligence **6** (2001)
9. Angelopoulos, N., Muggleton, S.H.: Machine learning metabolic pathway descriptions using a probabilistic relational representation. Electronic Transactions in Artificial Intelligence **7** (2002) also in Proceedings of Machine Intelligence 19.
10. Angelopoulos, N., Muggleton, S.H.: Slps for probabilistic pathways: Modeling and parameter estimation. Technical Report TR 2002/12, Department of Computing, Imperial College, London, UK (2002)

[3] http://www.aprill.org

11. 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
12. Chabrier-Rivier, N., Fages, F., Soliman, S.: The biochemical abstract machine BIOCHAM. In Danos, V., Schächter, V., eds.: CMSB'04: Proceedings of the second Workshop on Computational Methods in Systems Biology. Volume 3082 of Lecture Notes in BioInformatics., Springer-Verlag (2004) 172–191
13. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
14. Cimatti, A., Clarke, E., Enrico Giunchiglia, F.G., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: Nusmv 2: An opensource tool for symbolic model checking. In: Proceedings of the International Conference on Computer-Aided Verification, CAV'02, Copenhagen, Danmark (2002)
15. Shapiro, E.Y.: Algorithmic Program Debugging. The MIT Press Classics Series (1983)
16. Hucka, M., et al.: The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. Bioinformatics **19** (2003) 524–531
17. Tyson, J.J.: Modeling the cell division cycle: cdc2 and cyclin interactions. Proceedings of the National Academy of Sciences **88** (1991) 7328–7332
18. Qu, Z., WR, W.M., Weiss, J.: Dynamics of the cell cycle: checkpoints, sizers, and timers. Biophysics Journal **85** (2004) 3600–11
19. Levchenko, A., Bruck, J., Sternberg, P.W.: Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. PNAS **97** (2000) 5818–5823

Appendix

bias	# good rules	# rules tested	time
undefined `A => B` reaction	3	45	4s
(de)phosphorylation	1	11	<1s
add a specification: cdc2 activated is a checkpoint of MPF activation	1	45	4s

Table 1. Tyson [17] cell cycle experiment with the rule of MPF activation by dephosphorylation deleted.

deleted rule	good rules	tested rules	time
Cell cycle model by Qu et al. [18] (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	4680 s
activation of C25 by MPF	5	1042	740s
RTK-MAPK cascade by Levchenko et al. [19] (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

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

Bias	good rules	tested rules	time
complexation or phosphorylation	0	23	1.8s
synthesis	0	5	0.3s
synthesis and degradation	3	32	2.2s
very general reaction `A + B => C + D`	29	222	16.5s
Restriction to `A + B => C`	6	87	6.0s
complexation and (de)phosphorylation `A + B => Ap-Bp`	2	51	4.0s
add a specification: activation of cdc2 is a checkpoint for MPF activation	1	51	3.6s

Table 3. Tyson model with MPF activation process deleted.

deleted process	good rules	tested rules	time
degradation of (pre)MPF by APC complexation of CKI-MPF	23	1042	532s
all inhibition of MPF	0	1043	635s
other pattern `A + B<=>C`	4	5831	4497s
complexation of MPF and its activation by C25	1	1042	285s
all activation of MPF	0	1042	227s
other patterns `A + B<=>C`	0	5831	1504s
`A + B =>C + D`	0	22550	4291s

Table 4. Another example of process discovery, with Qu's cell cycle model [18]. The bias is, again, all possible biological rules.