A MACHINE LEARNING APPROACH TO BIOCHEMICAL REACTION RULES DISCOVERY

Laurence Calzone, Nathalie Chabrier-Rivier, François Fages and Sylvain Soliman
INRIA Rocquencourt - Projet CONTRAINTES
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY CEDEX - FRANCE

Abstract

Beyond numerical simulation, the possibility of performing symbolic computation on biomolecular interaction networks opens the way to the design of new automated reasoning tools for biologists/modelers. The Biochemical Abstract machine BIOCHAM provides a precise semantics to biomolecular interaction maps as concurrent transition systems. Based on this formal semantics, BIOCHAM offers a compositional rule-based language for modeling biochemical systems, and an original query language based on temporal logic for expressing biological queries, such as reachability, checkpoints, oscillations or stability.

Turning the temporal logic query language into a specification language for expressing the observed behavior of the system (in wild-life and mutated organisms) makes it possible to use machine learning techniques for completing or correcting biological models in BIOCHAM. 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 report on the machine learning system of BIOCHAM which allows to discover, on the one hand, interaction rules from a partial model with constraints on the system behavior expressed in temporal logic, and on the other hand, kinetic parameter values from a temporal logic specification with constraints on numerical concentrations.

Keywords

Pathways, Machine Learning, Temporal Logic

Introduction

The mass production of post-genomic data, such as ARN expression, protein production and protein-protein interaction, raises the need for a strong effort on the formal representation of biological systems. Knowledge on gene interactions and pathways is currently gathered in databases such as KEGG, BioCyc, etc. in the form of annotated diagrams. Tools such as BioSpice, Copasi, GON, E-cell, etc. have been developed for making simulations based on these databases when numerical data are present.

Beyond numerical simulation however, the possibility of performing symbolic computation on biomolecular interaction networks opens the way to the design of a new kind of automated reasoning tools for biologists/modelers. Our project with the Biochemical Abstract Machine (Fages et al., 2004) (`http://contraintes.inria.r/BIOCHAM/`), started in 2002, is one attempt in this direction. BIOCHAM provides a precise semantics to qualitative biomolecular interaction maps as concurrent transition systems (Chabrier-Rivier et al., 2004b). Based on this 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 a numerical simulator;

- an original query language based on temporal logic CTL (Clarke et al., 1999) for boolean models and LTL with constraints for numerical models for expressing biological queries such as reachability, checkpoints, oscillations or stability (Chabrier and Fages, 2003; Eker et al., 2002);

- a machine learning system to infer interaction rules and kinetic parameters from observed temporal properties.

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 (Kohn, 1999) involving about 500 variables and 2700 reaction rules (Chabrier-Rivier et al., 2004a).

Now, turning the temporal logic query language into a specification language for expressing the observed behavior of the system (in wild-life and mutated organisms) makes it possible to use machine learning techniques for completing or correcting BIOCHAM models.

There has been work on the use of machine learning techniques, such as inductive logic programming (Muggleton, 1995) or genetic programming, to infer gene functions (Bryant et al., 2001), metabolic pathway descriptions (Angelopoulos and Muggleton, 2002; **?**; **?**) or gene interactions (Bernot et al., 2004). Our work can also be related to the whole domain of qualitative and numerical scientific discovery (**?**) and to the theories modified in theory revision (**??**). 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 machine learning system in BIOCHAM allows to discover interaction rules from a partial model and constraints on the system behavior (Calzone et al., 2005). These constraints are expressed in temporal logic with positive formulae (expected properties) and negated formulae (properties to forbid). The learning process can be guided by the user by providing patterns for limiting the types of sought reactions, such as complexation, phosphorylation, etc. The machine learning system supports similarly the learning of kinetic parameter values from a specification in temporal logic with constraints on numerical quantities.

In this paper, we describe the BIOCHAM machine learning system from temporal logic formulae. We present an example of rule and parameter learning and show the interactive use of the learning system to refine the specification and the patterns until a biologically satisfactory solution is obtained.

Preliminaries on BIOCHAM

BIOCHAM manipulates formal objects which represent not only chemical or biochemical compounds, ranging from ions, to small molecules, macromolecules and genes but also control variables and abstract biological processes. BIOCHAM reaction rules primarily represent biochemical reactions but can also be used to represent state transitions involving control variables or abstract processes.

Syntax

molecule =	name | molecule~{name,...,name}
	| molecule-molecule | (molecule)
reaction =	kinetics for solution => solution
solution =	_ | molecule | 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`. For instance, `Yp + E1 => Yp-E1` is a complexation rule. `Yp =[Zp]=> Yp~{i}` is a phosphorylation rule with catalyst `Zp`. A default kinetic expression is provided for rules where no such expression is given.

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 also provide a guideline on the shape of rules to be considered during the learning process, as explained in a later section.

Semantics

The semantics of BIOCHAM is defined at two levels of abstraction: the molecule concentration semantics and the boolean semantics which only deals with the presence or absence of molecules. The boolean semantics reflects the capability of drawing inferences about all possible behaviors of the system with unknown concentration values and unknown kinetic parameters. In the boolean semantics, the reaction rules are interpreted as a concurrent (asynchronous) transition system. A rule like `A+B=>C+D` defines four possible transitions taking into account the possible consumption or not of the reactants `A` and `B`. In the next state, molecules `C` and `D` are present, while molecules `A` and `B` can be non-deterministically consumed or present.

The molecule concentration semantics supposes that each reaction rule is given a kinetic expression (such as mass action law, Michaelis-Mentens, Hill kinetics, etc.). In that case the rules can be compiled in a system of (highly non-linear) ordinary differential equations. Given a set of initial concentrations for each molecule, the evolution of the system becomes fully deterministic.

The most original feature of BIOCHAM is the use of temporal logic (Clarke et al., 1999) as a query language for the biological properties of the models. The Computation Tree Logic CTL is used for the boolean semantics as it is non-deterministic. This logic 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. In CTL, a temporal operator has to be immediately preceded by a path quantifier. As shown in Chabrier and Fages (2003) CTL is expressive enough to express a wide range of biological queries:

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

About pathway. Is state Yp a necessary *checkpoint* for reaching state $Xp \sim \{i\}$? $!(E((!(Yp)UXp \sim \{i\})$. This formula is abbreviated as `checkpoint(Yp,Xp~{i})`.

About stability and oscillations. Is a certain (partially described) state s of the cell a steady state? $s \Rightarrow EG(s)$. Can the system exhibit a cyclic behavior w.r.t. the presence of a product Xp? This query can be formalized by the CTL formula $EG((Xp \Rightarrow EF\ \neg Xp) \wedge (\neg Xp \Rightarrow EF\ Xp))$. It will be abbreviated as `loop(Xp,!Xp)`.

The CTL query language for boolean models is implemented in BIOCHAM with an interface to the state-of-the-art symbolic model checker NuSMV of Cimatti et al. (2002).

Linear Time Logic LTL with arithmetic constraints is used for the molecule concentration semantics, in a way similar to Antoniotti et al. (2003). LTL is a temporal logic without path quantifier and is suitable to reason about deterministic systems, such as kinetic models. The same biological properties as above can be expressed in LTL except that:

- only one path is considered at a time. Practically, it is a time series describing the values of the different concentrations of each compound (and their derivatives) that will provide a model for an LTL query.

- the basic formulae on which LTL queries are built are made with arithmetic constraints about the concentrations or their derivatives (like `[Yp] > [Yp~{i}]` or `d([Xp])/dt < 0`).

Thus, *reachability* queries are formalized as $F([Yp] > [Yp \sim \{i\}])$ and *oscillation* queries can be formalized as checking whether the derivative of the molecule concentration alternates between positive and negative a certain number of times n (e.g. $F((d[Xp]/dt > 0) \wedge F((d[Xp]/dt < 0) \wedge F((d[Xp]/dt > 0) \ldots$, abbreviated as `oscil(Xp,n)`).

Machine Learning Interaction Rules from CTL Formulae

Systems biologists build models of bio-molecular interactions from experiments in wild-life and mutated organisms. These experiments inform on the properties that the model has to check in order to reproduce the behavior of the system under various conditions.

In our approach, the biological properties of the system can be formalized in CTL. The intended behavior of the model can thus be described through a set of CTL properties providing a specification, with positive and negative examples. This leads us to develop machine learning techniques to automatically propose rules to be added to, or removed from the model in order to fulfill the specification. A rule pattern (the bias) describing the plausible rules to add to the system is given to guide the search of new rules, eliminating in advance rules having no biological meaning.

After unfruitful experiments with state-of-the-art Inductive Logic Programming tools related to the complexity of temporal properties computation, we developed an ad-hoc exhaustive enumeration method: all the ground instances of the rule pattern are generated, and tried sequentially by adding them to the model and checking the CTL 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 algorithm is somewhat limited, since it handles only the addition of a single rule to the model. However the following sections show that it already provides interesting results. Moreover, the use of theory revision (Shapiro, 1983) techniques allow a more efficient search and the search for more than one rule, as explained in Calzone et al. (2005).

Machine Learning Kinetic Parameters from Constraint LTL Formulae

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 parameters 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 specification is checked.

For instance, the command `trace_get([ka1,kr1], [(400,4000),(100,1000)], 20, oscil(Xp,4), 40)` searches for two parameters (`ka1` and `kr1`) in the respective intervals of possible values $[400, 4000]$ and $[100, 1000]$, with only 20 different values tried for each, and such that before time 40, `Xp` oscillates 4 times. It is also possible to start with very wide intervals and a quite loose specification, like $[1, 10000]$ for the parameters above, and `oscil(Xp,2)`. Then, once one gets an answer, the specification can be made more precise, and the intervals smaller if necessary.

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.

Negative Feedback Example

The learning methods are illustrated in this section on a toy model of a negative feedback loop. Both methods for learning rules and kinetic parameters are coupled to get a kinetic model that fits the experimental behavior of the system: with the appropriate kinetics and parameters, this kind of models is expected to oscillate.

A simple network composed of three components is chosen. The three proteins involved appear in two forms, active (`Xp`, `Yp` and `Zp`) and inactive (`Xp~{i}`, ...). The known interactions are that `Xp` (resp. `Zp`, `Yp`) promotes the inactivation of `Zp` (resp. `Yp`, `Xp`). An initial BIOCHAM model is written in the simplest way, using the law of mass action with some arbitrary parameter values:

```
rule1 : kax*[Xp~{i}]    for Xp~{i} => Xp.
rule2 : kix*[Xp]*[Yp]   for Xp=[Yp]=> Xp~{i}.
rule3 : kay*[Yp~{i}]    for Yp~{i} => Yp.
rule4 : kiy*[Yp]*[Zp]   for Yp=[Zp]=> Yp~{i}.
```

```
rule5 : kaz*[Zp~{i}]    for Zp~{i} => Zp.
rule6 : kiz*[Zp]*[Xp]   for Zp=[Xp]=> Zp~{i}.

parameter(kax,0.1).   parameter(kix,1.5).
parameter(kay,0.4).   parameter(kiy,1).
parameter(kaz,0.2).   parameter(kiz,1).

% Initial conditions
present(Xp,1).  present(Yp,1).  present(Zp,1).
absent(Xp~{i}). absent(Yp~{i}). absent(Zp~{i}).

% CTL specifications
add_specs({
Ei(reachable(Xp)),     Ei(reachable(Yp)),...
Ei(reachable(Xp~{i})), Ei(reachable(Yp~{i})),...
Ai(loop(Xp,Xp~{i})),   Ai(loop(Yp,Yp~{i})),...
Ai(checkpoint(Yp,Xp~{i})),
Ai(checkpoint(Zp,Yp~{i})),
Ai(checkpoint(Xp,Zp~{i}))}).
```

To simulate the BIOCHAM model, a set of initial conditions and a list of CTL specifications that account for experimental results are provided with the model. The model is considered to be "correct" when all the specifications are satisfied. In this toy example, the CTL specifications indicate, for instance, that `Xp` is reachable, that `Xp~{i}` and `Xp` alternate, and that `Yp` is a checkpoint for the inactivation of `Xp`.

With these specifications, the boolean model complies with the expected behavior, but the kinetic model does not show the wanted oscillations. After a search of the parameter space, no parameter values are found that exhibit oscillations. These results might suggest that the rules need to be modified.

We choose to introduce some kind of non-linearity to the model by adding an intermediary step in the inactivation of one of the variables, for instance, `Yp`.

The rule we intend to replace (rule 4) is deleted: `delete_rules(Yp=[Zp]=>Yp~{i})`. The model is no longer correct as two of the specifications are now false (`reachable(Yp~{i})` and `loop(Yp,Yp~{i})`).

Then, a rule involving a new enzyme, `E1`, is added to the model: `Yp` associates with `E1` to form a complex `Yp+E1 =>Yp-E1`. We force the formation of the complex to be a necessary step to get Yp inactivated, which is translated into the following specification: `checkpoint(Yp-E1,Yp~{i})`.

BIOCHAM is asked to find a rule that could satisfy the model's specifications with the command `learn_one_rule(elementary_interaction_rules)`. Out of 1066 rules tested, only one rule is proposed by BIOCHAM: `Yp-E1=[Zp]=>Yp~{i}+E1`. This rule is added to the current list of rules and the specifications are now verified by the boolean model.

The same reasoning is done for the reverse reaction. The rule `Yp~{i}=>Yp` is deleted and the rule `Yp~{i}+E2 =>Yp~{i}-E2` added. BIOCHAM proposes `Yp~{i}-E2 =>Yp` to satisfy the specification `checkpoint(Yp~{i}-E2,Yp)`.

The next step is to test parameter values to get an oscillatory behavior. The form of the two-step reaction vaguely resembles Michaelis Menten kinetics. As a first approximation, the modeler may choose parameter values for `ka1`, `ka2`, `kr1` and `kr2` accordingly.

```
parameter(ka1,5e6).     parameter(kr1,1000).
parameter(ka2,5e6).     parameter(kr2,1000).
present(E1,0.001).      absent(Yp-E1).
present(E2,0.001).      absent(Yp~{i}-E2).
```

With these values, the system does not oscillate. The function `trace_get` enables the modeler to search for several parameter values at the same time. We vary two parameters `ka2` and `kr2` with `ka1` and `kr1` fixed. Their respective domains are explored and values are searched such that Yp oscillates three times on an interval of 40 units of time and that Yp concentration gets close to 0 at some point.

```
trace_get([ka2,kr2],[(1000000,10000000),(0,1000)],
10,(oscil(Yp,3)&F([Yp]<0.001)),40).
Search time: 8.92 s
Found parameters that make oscil(Yp,3)&F([Yp]<0.001)
true:
parameter(ka2,1000000).    parameter(kr2,300).
```

For these parameters, the system oscillates according to the specification. Note that the user can further refine the LTL specification to get a different or more accurate shape of the curves.

The use of both learning methods helped the modeler find the best rules and parameters that exhibit the appropriate behavior. The modeler, though, remains active in the process of finding rules and parameters.

Conclusion

With the advent of formal languages for describing systems of bio-molecular interactions as well as their biological properties, machine learning techniques can be used to curate models and integrate semi-automatically new data coming from biological experiments.

We have shown that in the Biochemical Abstract Machine BIOCHAM, the rule-based language for modeling bio-molecular interactions, and the temporal logics used for formalizing biological properties of the system, can be combined in a machine learning process for discovering new reaction rules and estimating kinetic parameters.

Obviously, the experiments reported in this paper were chosen for sake of simplicity, however the machine learning algorithm scales up linearly with the number of candidates (rules or parameters) and we plan to use it on a large scale for the development of models for cancer therapies.

References

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

Antoniotti, M., Policriti, A., Ugel, N., and Mishra, B. (2003). Model building and model checking for biochemical processes. *Cell Biochemistry and Biophysics*, 38:271–286.

Bernot, G., Comet, J.-P., Richard, A., and Guespin, J. (2004). 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.

Bryant, C. H., Muggleton, S. H., Oliver, S. G., Kell, D. B., Reiser, P. G. K., and King, R. D. (2001). Combining inductive logic programming, active learning and robotics to discover the function of genes. *Electronic Transactions in Artificial Intelligence*, 6(12).

Calzone, L., Chabrier-Rivier, N., Fages, F., Gentils, L., and Soliman, S. (2005). Machine learning biomolecular interactions from temporal logic properties. In Plotkin, G., editor, *CMSB'05: Proceedings of the third Workshop on Computational Methods in Systems Biology*.

Chabrier, N. and Fages, F. (2003). Symbolic model cheking of biochemical networks. In Priami, C., 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. Springer-Verlag.

Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., and Schächter, V. (2004a). Modeling and querying biochemical interaction networks. *Theoretical Computer Science*, 325(1):25–44.

Chabrier-Rivier, N., Fages, F., and Soliman, S. (2004b). The biochemical abstract machine BIOCHAM. In Danos, V. and Schächter, V., editors, *CMSB'04: Proceedings of the second Workshop on Computational Methods in Systems Biology*, volume 3082 of *Lecture Notes in BioInformatics*, pages 172–191. Springer-Verlag.

Cimatti, A., Clarke, E., Enrico Giunchiglia, F. G., Pistore, M., Roveri, M., Sebastiani, R., and Tacchella, A. (2002). Nusmv 2: An opensource tool for symbolic model checking. In *Proceedings of the International Conference on Computer-Aided Verification, CAV'02*, Copenhagen, Danmark.

Clarke, E. M., Grumberg, O., and Peled, D. A. (1999). *Model Checking*. MIT Press.

Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., and Sönmez, M. K. (2002). Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the seventh Pacific Symposium on Biocomputing*, pages 400–412.

Fages, F., Soliman, S., and Chabrier-Rivier, N. (2004). Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2):64–73.

Goldbeter, A. (1991). A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *PNAS*, 88(20):9107–9111.

Kohn, K. W. (1999). Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Molecular Biology of the Cell*, 10(8):703–2734.

Muggleton, S. H. (1995). Inverse entailment and progol. *New Generation Computing*, 13:245–286.

Shapiro, E. Y. (1983). *Algorithmic Program Debugging*. The MIT Press Classics Series.