A Machine Learning approach to Biochemical Reaction Rules Discovery

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

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 about 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 semi-automatically. 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, the possibility of performing symbolic computation on bio-molecular 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;

- numerical and boolean simulators;

- an original query language based on temporal logic CTL (Clarke et al., 1999) for boolean models and LTL with constraints for numerical models, used for expressing biological queries about 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 boolean 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).

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 using the temporal logic query language as a specification language. The learning process can be guided by the user by providing patterns for limiting the types of sought reactions, such as com-

plexation, 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.

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, 2002a,b; Koza et al., 2001) or gene interactions (Bernot et al., 2004). Our work can also be related to the whole domain of qualitative and numerical scientific discovery (Langley et al., 1987) and to the theories modified in theory revision (Todorovski and Džeroski, 2001; de Raedt, 1992). 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 describe the BIOCHAM machine learning system and illustrate its interactive use through a toy example of a negative feedback loop model refinement.

Preliminaries on BIOCHAM

BIOCHAM reaction rules primarily represent biochemical reactions between formal objects which represent chemical or biochemical compounds, ranging from ions, small molecules, to macromolecules and genes. The syntax is defined by the following simplified grammar:

molecule =	name | molecule~{name,...,name}
	| molecule- molecule
reaction =	kinetics `for` solution `=>` solution
solution =	_ | molecule | solution `+` solution

The following abbreviations are also 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~{p}` is a phosphorylation rule with catalyst `Zp`. A rich pattern language with constraints is also provided, and used to specify molecules and sets of reaction rules in a concise manner, or to restrict the rule search during the learning process.

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 molecule concentration semantics supposes that each reaction rule is given a kinetic expression (such as mass action law, Michaelis-Menten, 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 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 (Chabrier-Rivier et al., 2004a).

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 ϕ is 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 $EF(Xp)$, and is abbreviated as `reachable(Xp)` in BIOCHAM.

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

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? $EG((Xp \Rightarrow EF\ \neg Xp) \wedge (\neg Xp \Rightarrow EF\ Xp))$ (without strong fairness, this formula is in fact an approximation), 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 provides 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~{p}]` or `d([Xp])/dt < 0`).

Reachability queries are formalized with the operator F and *oscillation* queries, checking whether the derivative of the molecule concentration alternates between positive and negative n times, with $F((d[Xp]/dt > 0) \wedge F((d[Xp]/dt < 0) \wedge F((d[Xp]/dt > 0)\ldots$ (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 define the properties that the model has to satisfy.

In our approach, the biological properties of the system can be formalized in CTL as a specification. We 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 for new rules, eliminating in advance those having no biological meaning. The modifications which fulfill the specification are returned as answers and proposed to the user.

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 theory revision algorithm (Calzone et al., 2005) based on a classification of ECTL and ACTL formulae (Clarke et al., 1999):

1. Check the ACTL formulae, if a formula is false, search for rule deletions s.t. the model satisfies this formula;
2. Check the ECTL formulae, if a formula is false, search for rule additions s.t. the model satisfies this formula and the ACTL formulae (that were true);
3. Check the unclassified formulae, if a formula is false, search for rule additions or deletions s.t. the model satisfies this formula.

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. 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 an LTL 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.

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

In this section, we illustrate the learning algorithms on a toy example of model refinement for 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 considered. The three proteins involved appear in two forms, active (`Xp`, `Yp` and `Zp`) and inactive (`Xp~{p}`, ...). The known interactions are that `Xp` (resp. `Zp`, `Yp`) promotes the inactivation of `Zp` (resp. `Yp`, `Xp`). A first BIOCHAM model is written in the simplest way, using the law of mass action with some arbitrary parameter values:

```
rule1 : kax*[Xp~{p}]    for Xp~{p} => Xp.
rule2 : kix*[Xp]*[Yp]   for Xp=[Yp]=> Xp~{p}.
rule3 : kay*[Yp~{p}]    for Yp~{p} => Yp.
rule4 : kiy*[Yp]*[Zp]   for Yp=[Zp]=> Yp~{p}.
rule5 : kaz*[Zp~{p}]    for Zp~{p} => Zp.
rule6 : kiz*[Zp]*[Xp]   for Zp=[Xp]=> Zp~{p}.
parameter(kax,0.1).   parameter(kix,1.5).
parameter(kay,0.4).   parameter(kiy,1).
parameter(kaz,0.2).   parameter(kiz,1).
```

To simulate the BIOCHAM model, a set of initial conditions is provided. The CTL specification under these initial conditions expresses that `Xp` is reachable, that `Xp~{p}` and `Xp` alternate, and that `Yp` is a checkpoint for the inactivation of `Xp` (same for `Yp, Zp`):

```
present(Xp,1).  present(Yp,1).  present(Zp,1).
absent(Xp~{p}). absent(Yp~{p}). absent(Zp~{p}).
add_specs({
Ei(reachable(Xp)),     Ei(reachable(Yp)),...
Ei(reachable(Xp~{p})), Ei(reachable(Yp~{p})),...
Ai(loop(Xp,Xp~{p})),   Ai(loop(Yp,Yp~{p})),...
Ai(checkpoint(Yp,Xp~{p})),
Ai(checkpoint(Zp,Yp~{p})),
Ai(checkpoint(Xp,Zp~{p}))}).
```

In this instance, the boolean model complies with that specification, but the numerical simulation does not exhibit oscillations. After a search of the parameter space (using `trace_get` for `oscil(Yp,3)`), no parameter values are found, which suggests that the rules need to be modified. Some kind of non-linearity can be introduced to the model by imposing an intermediary step in the inactivation of one of the variables, for instance, `Yp`. The rule 4, `Yp=[Zp]=>Yp~{p}`, is thus deleted and a rule expressing the complexation with a new enzyme, `E1`, is added: `Yp+E1 =>Yp-E1`. The formation of the complex becomes necessary step to get `Yp` inactivated, by adding the following specification: `checkpoint(Yp-E1,Yp~{p})`. The command `learn_one_rule(elementary_interaction_rules)` is used to complete the model under this specification.

Only one rule, `Yp-E1=[Zp]=>Yp~{p}+E1`, is found and added to the model. The same reasoning is done for the reverse reaction. The rule `Yp~{p}=>Yp` is deleted.

The rule `Yp~{p}+E2 =>Yp~{p}-E2` and the specification `checkpoint(Yp~{p}-E2,Yp)` are added. BIOCHAM then finds the rule `Yp~{p}-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 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~{p}-E2).
```

With these values, the system still does not oscillate. The command `trace_get` then enables the modeler to search for several parameter values at the same time, for instance by varying parameters `ka2` and `kr2` with `ka1` and `kr1` fixed. Their respective domains are explored and values are searched such that Yp oscillates 3 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 making oscil(Yp,3)&F([Yp]<0.001) true:
parameter(ka2,1000000).    parameter(kr2,300).
```

With these learned parameters, the system now 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.

These issues of model refinement are ubiquitous for the modeling of complex biological systems. The symbolic learning techniques implemented in BIOCHAM automate parts of the reasoning process and help the modeler fit the model with its expected behavior, as a complement to other techniques such as bifurcation analysis.

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 biomolecular 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.

The machine learning algorithm has complexity $O(N \times T)$ where N is the number of candidates (rules or parameter values) and T is the time needed to check the specification on one model. Since encouraging results were obtained for the querying of larger size models, we investigate the use of these learning techniques for decomposing and coupling complex models.

References

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

Angelopoulos, N. and Muggleton, S. H. (2002b). Slps for probabilistic pathways: Modeling and parameter estimation. Technical Report TR 2002/12, Department of Computing, Imperial College, London, UK.

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, *Proceedings of CMSB'05*.

Chabrier, N. and Fages, F. (2003). Symbolic model cheking of biochemical networks. In Priami, C., editor, *Proceedings CMSB'03*, volume 2602 of *LNCS*, 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, *Proceedings of CMSB'04*, volume 3082 of *LNBI*, 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 CAV'02*, Copenhagen, Danmark.

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

de Raedt, L. (1992). *Interactive Theory Revision, an inductive Logic Programming Approach*. Knowledge-Based Systems. academic 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.

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.

Koza, J. R., Mydlowec, W., Lanza, G., Yu, J., and Keane, M. A. (2001). Reverse engineering of metabolic pathways from observed data using genetic programming. In *Proceedings of the 6th Pacific Symposium on Biocomputing*, pages 434–445, Hawaii, USA.

Langley, P., Simon, H. A., Bradshaw, G. L., and Zytkow, J. M. (1987). *Scientific Discovery: Computational Explorations of the Creative Processes*. MIT Press, Cambridge, MA.

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

Todorovski, L. and Džeroski, S. (2001). Theory revision in equation discovery. In Jantke, K. P. and Shinohara, A., editors, *Proceedings of the 4th International Conference on Discovery Science, DS 2001*, volume 2226 of *LNAI*, pages 389–400, Washington, DC, USA. Springer-Verlag.