Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM

François Fages, Sylvain Soliman* and Nathalie Chabrier-Rivier

Projet Contraintes, INRIA Rocquencourt, BP105, 78153 Le Chesnay CEDEX, France

Recent progress in high-throughput data-production technologies pushes research toward systems biology, focusing on the global interaction between the components of biomolecular processes. In this article we present a formal modelling environment for network biology, called the Biochemical Abstract Machine (BIOCHAM). Biocham delivers precise semantics to biomolecular interaction maps. Based on this formal semantics, the Biocham system offers automated reasoning tools for querying the temporal properties of the system under all its possible behaviours. We present the main features of Biocham and report on our modelling experience with this language.

Keywords: computational biology, model checking, protein interaction, systems biology

1. INTRODUCTION

Recent progress in high-throughput data-production technologies pushes research toward systems biology, focusing on the global interaction between the components of biomolecular processes [16]. In network biology, the complexity of the systems at hand (metabolic networks, extracellular and intracellular networks, networks of gene regulation) clearly shows the necessity of software tools for reasoning globally about biological systems. Several formalisms have been proposed in recent years for modelling biochemical processes either qualitatively [25, 24, 12] or quantitatively [22, 15, 1, 14, 3]. State-of-the-art tools integrate a graphical user interface and a simulator, yet few formal tools are available for reasoning about these processes and proving their properties. Our focus in Biocham has been on the design of a biochemical rule language and a query language of the model in temporal logic, which are intended to be used by biologists.

Biocham is a language and a programming environment for modelling biochemical systems, making simulations, and checking temporal properties. Biocham is composed of :

(i) A rule-based language for modelling biochemical systems;

(ii) A simple simulator;

(iii) A powerful query language based on Computation Tree Logic (CTL);

(iv) An interface to the NuSMV [10] model checker for automatically evaluating CTL queries.

The use of Computation Tree Logic (CTL) [11] for querying the temporal properties of the system provides an alternative technique to numerical models based on differential equations, in particular when numerical data are missing. The model-checking tools associated with CTL automate reasoning on all the possible behaviours of the system modelled in a purely qualitative way. The semantics of Biocham ensures that the set of possible behaviours of the model over-approximates the set of all behaviours of the system corresponding to different kinetic parameters.

Biocham shares several similarities with the Pathway Logic system [12] implemented in Maude. Both systems rely on an algebraic syntax and are rule-based languages. One difference is the use in Biocham of CTL logic which allows us to express a wider variety of biological queries, and the use of a state-of-the-art symbolic model checker for handling the complexity of large highly non-deterministic models.

The first experimental results of this approach for querying models of biochemical networks in temporal logic have been reported in [4, 5], on a qualitative model of the mammalian cell cycle control [9, 18] and in [5] on a quantitative model of gene expression [3]. In this paper we describe the Biocham system, which provides a modelling environment supporting this methodology.

2. A SIMPLE EXAMPLE

The mitogen-activated protein kinase (MAPK) cascades are a well-known example of signal transduction, since they appear in many receptor-mediated signal transduction schemes. They are actively considered in pharmaceutical research for their applications to cancer therapies. The MAPK/ERK pathway is indeed hyperactivated in 30% of all human cancer tumours [19].

The structure of a MAPK cascade is a sequence of activations of three kinases in the cytosol. The last kinase, MAPK, when activated, has an effect on

*Corresponding author. website: http://contraintes.inria.fr

Journal of Biological Physics and Chemistry **4** (2004) 64–73
Received 30 April 2004

different substrates in the cytosol but also on gene transcription in the nucleus.

Since this cascade has been studied a lot, mathematical models of it appear in most model repositories, like for instance that of Cellerator [27] or the SBML repository page [13], both coming from [20]. This cascade was also the first example treated by Regev, Silverman and Shapiro [25] in the pi-calculus process algebra, which was an initial source of inspiration for our own work.

Models based on ordinary differential equations (ODE) allow us to reproduce simulation results like the one pictured in Figure 1, where the concentration of the visualized compounds is represented on the vertical axis and time on the horizontal axis. In Figure 2, the concentration axis has been simply split for each compound and rescaled to its maximum value.

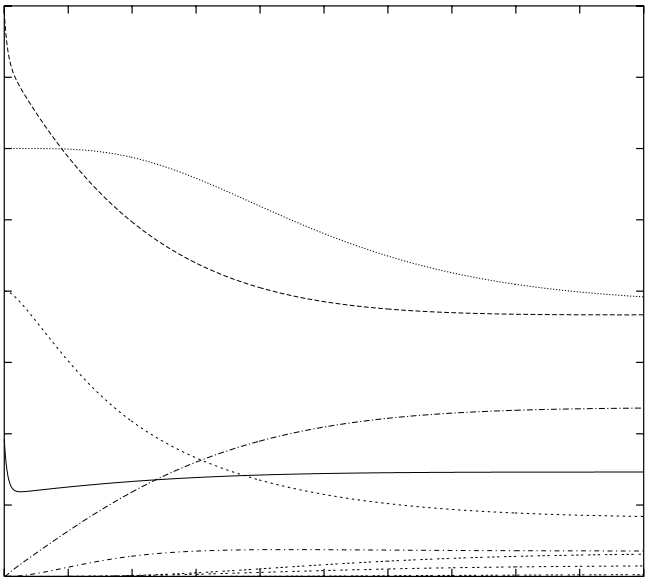

Figure 1. Simulation result of an ODE model of the MAPK cascade. ——— RAFK, -------- RAF, - - - - - - MEK, ---------- MAPK, -·-·-·-·- RAF~{p1}, - · - · - · - · - MEK~{p1}, -- -- -- MEK~{p1,p2}, --- --- MAPK~{p1}, ---- ---- MAPK~{p1,p2}.

It is possible to see from such simulations how the cascade evolves in time. It is possible to change input quantities to check for a significant change in the outcome of the simulation. Similarly, the sensitivity of the system to the values of the parameters can be checked by running different simulations with different values of the parameters.

Our aim in Biocham is to introduce complementary techniques to automate reasoning on all possible behaviours of the system modelled in a purely qualitative way. Taking the above model, one sees that it is built quite directly from the enzymatic reactions and Michaelis-Menten kinetics.

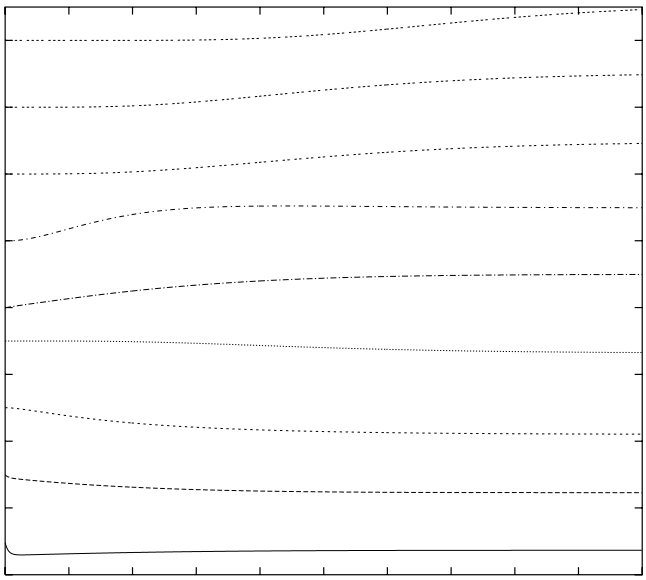

Figure 2. The same simulation as figure 1 (side by side rescaled view). ——— RAFK, -------- RAF, - - - - - - MEK, ---------- MAPK, -·-·-·-·- RAF~{p1}, - · - · - · - · - MEK~{p1}, -- -- -- MEK~{p1,p2}, --- --- MAPK~{p1}, ---- ---- MAPK~{p1,p2}.

Abstracting the kinetics part, one gets a system of biochemical reactions that can be interpreted as a non-deterministic transition system over Boolean variables denoting the presence or absence of the compounds in the signalling cascade. The source code of this example is given in the Appendix in Biocham syntax (explained in the next section) and in graphical form in Figure 4. The semantics of Biocham (explained in sections 3.2 and 3.3) ensures that the set of the possible behaviours of the Boolean model over-approximates the set of all behaviours of the system for all kinetic parameter values.

Biocham uses Computation Tree Logic (CTL) [11] as a query language for querying the temporal properties of the system under all possible conditions, given a partial description of the initial state. A biological query like for example "Is the activation of the second kinase of the cascade (MEK) compulsory for the cascade?" asks whether the phosphorylated form of MEK, noted in Biocham `MEK~{p1}`, is necessary for the production of the activated MAPK, noted `MAPK~{p1,p2}`, which is the output of the cascade. This query asks whether `MEK~{p1}` is a checkpoint. In Biocham, one expresses this query by the CTL formula

```
biocham: !(E(!(MEK~{p1}) U MAPK~{p1,p2}))
true
```

This formula expresses the non (!) existence (E) of a path on which `MEK~{p1}` is absent (!) until (U) `MAPK~{p1,p2}` becomes present, that is to say that `MEK~{p1}` is a checkpoint. This formula is checked automatically by the system.

The same query about a complex with a phosphatase, such as the complex `MEK~{p1}-MEKPH`, is false. These complexes are thus not checkpoints. The `why` command computes a counterexample in the form of a pathway which validates the negation of the query:

```
biocham:!(E(!(MEK~{p1}-MEKPH) U MAPK~{p1,p2}))
false
biocham: why
Step 1  Initial state
Step 2  rule 1    RAF-RAFK                present
Step 3  rule 21   RAF~{p1}                present
Step 4  rule 5    MEK-RAF~{p1}            present
Step 5  rule 24   MEK~{p1}                present
Step 6  rule 7    MEK~{p1}-RAF~{p1}       present
Step 7  rule 23   MEK~{p1,p2}             present
Step 8  rule 13   MAPK-MEK~{p1,p2}        present
Step 9  rule 27   MAPK~{p1}               present
Step 10 rule 15   MAPK~{p1}-MEK~{p1,p2}   present
Step 11 rule 28   MAPK~{p1,p2}            present
```

This means that the complexes with a phosphatase (xxxPH) are intermediate products that do not strictly participate in the signal transduction. They are here to regulate the cascade, but they are not mandatory for the signal transduction in this model. A similar trace is obtained when asking a simple accessibility query like `EF(MAPK~{p1,p2})`, that is the existence (E) of a path on which at some time point (F) MAPK is fully phosphorylated.

It is worth noting that imposing the absence of an intermediate product is generally difficult in an ODE-based simulation tool without touching the model. Complex CTL queries thus have no natural counterpart in a numerical model and complement the information that can be deduced from interaction maps.

Querying a Biocham model in CTL temporal logic provides a means to analyse exhaustively all possible behaviours of the system from the first principles of enzymatic reactions, in particular when numerical data are not available. The simulation of Biocham models is also possible. Since a Biocham model is highly non-deterministic, simulations are randomized, which means that at each time step, one of the possible reactions is chosen randomly. Figure 3 depicts one random simulation of the MAPK cascade, which is only one possible behaviour of the system at the Boolean abstraction level. One can notice that this trace is not a Boolean abstraction (by thresholds) of the numerical simulation. On the other hand, the semantics of Biocham ensures that the numerical simulation can be abstracted in a feasible Boolean trace.

Biocham has been designed in the framework of the ARC CPBIO on "Process Calculi and Biology of Molecular Networks" [2] which aims at pushing forward a declarative and compositional approach to

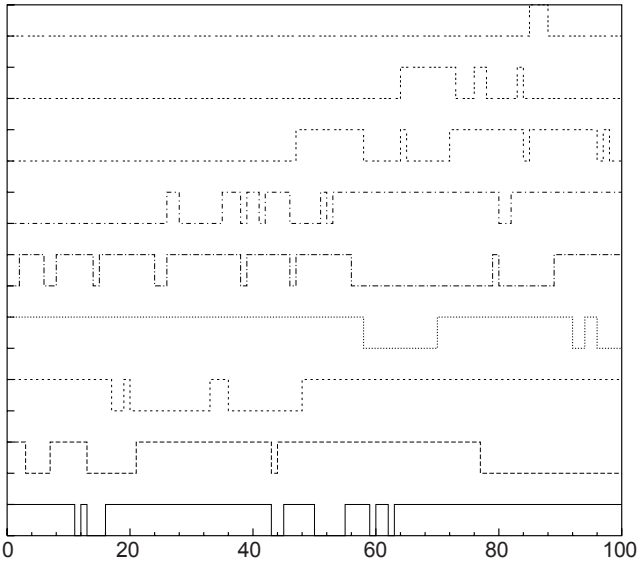

Figure 3. Random simulation of the Biocham model of the MAPK cascade. ——— RAFK, -------- RAF, - - - - - - MEK, ---------- MAPK, -·-·-·-·- RAF~{p1}, - · - · - · - · - MEK~{p1}, -- -- -- MEK~{p1,p2}, --- --- MAPK~{p1}, ----- ----- MAPK~{p1,p2}.

modelling languages in systems biology. The largest example treated so far with Biocham is a model of mammalian cell cycle control [4] developed after Kohn's map [18], involving 500 proteins and genes and 147 rule patterns which expand into 2733 rule instances. The computational results reported in [4] show the feasibility of this approach on such large examples, as the CTL queries can be evaluated in a few seconds using state-of-the-art symbolic model-checking tools.

3. MODELLING BIOCHEMICAL PROCESSES IN BIOCHAM

3.1 A simple algebra of biochemical compounds

Biocham manipulates formal objects that 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, which are not described here for the sake of simplicity. We refer to [7] for more technical details.

Syntax:

```
molecule = name|molecule-molecule|molecule~
           {name,...,name}|gene|(molecule)
    gene = #name
```

In the simplest and the most flexible syntactical form, a molecule is simply given a name. Multimolecular complexes are denoted with the linking operator −. This binary operator is assumed to be associative and commutative, hence the order of the elements in a complex does not matter. Note that the same hypothesis is made in pathway logic [12] and other systems [21]. In

the cases where one would like to distinguish between different orders of association, one can denote the different complexes with specific names. A third syntactical form serves to write modified forms of molecules, like attaching the set of phosphorylated sites with the operator ~. Several sets can be attached. The order of the elements is irrelevant.

***Example*:** `cdk1,cdk1-cycB` and `cdk1~{tyr15,thr161}-cycB` are valid Biocham notations for, respectively, the cyclin dependent kinase1, the complex `cdk1` with cyclin B, and the phosphorylated form at sites tyrosine 15 and threonine 161 of cdk1 in the complex cdk1-cycB. `(cdk1-cycB)~{tyr15, thr161}` is another notation for the same phosphorylated form of the complex without making precise the constituent which is phosphorylated.

The fourth syntactical form is used to denote genes or gene promoters, with a name beginning with #. These objects are assumed to be *unique*, which has a consequence on the way reactions involving such objects are interpreted by Biocham, as explained in the next section.

Example: `DMP1-#p19ARF` can be used to denote the binding of protein DMP1 on the promotor of the gene producing protein `p19ARF` noted `#p19ARF`.

3.2 Reaction rules

Biocham reaction rules are used primarily to represent biochemical reactions. They can be used also to represent state transitions involving control variables or abstract processes, or to represent the main effects of complete subsystems such as protein synthesis by DNA transcription without introducing RNAs in the model.

Syntax:

```
reaction = name: reaction
         | solution => solution
         | solution = [object] => solution
         | solution <=> solution
         | solution <= [object] => solution
solution =  _| object| solution + solution |(solution)
```

A solution is thus a sum of objects, the character `_` denotes the empty solution. The order and multiplicity of molecules in a solution are ignored, only the presence or absence of objects are considered.

The following abbreviations can be used for reaction rules: `A<=>B` for the two symmetrical rules, and `A=[C]=>B` for the rule A+C→B+C with catalyst molecule C.

Example:

`cdk1+ cycB => cdk1- cycB` is a complexation rule. `cdk1-cycB= [Myt1]=>cdk1~{thr14}-cycB` is a phosphorylation rule with catalyst Mytosine 1. This rule is equivalent to `cdk1- cycB + Myt1 => Myt1 + cdk1~{thr14} - cycB`.

A reaction transforms one solution matching the left hand side of the rule, into another solution in which the objects of the right hand side have been added. The molecules in the left-hand side of the rule which do not appear in the right hand side may be non-deterministically present or consumed in the resulting solution. This convention defines the *Boolean abstraction* of stoichiometric models used in Biocham. It reflects the capability of Biocham to reason about all possible behaviours of the system with unknown concentration values and unknown kinetic parameters [1, 3].

Following the uniqueness assumption, molecule parts 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 they are deterministically consumed in the form in which they appear on the left hand side of the rule.

Biocham has also a rich pattern language with constraints 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 `$` to denote unspecified parts of a molecule. These variables can be constrained with simple set constraints. The description of Biocham patterns [6] is however beyond the scope of this paper. The appendix contains the Biocham model of the MAPK cascade of the introductory example written with a set of 16 reaction rule patterns which expand into 30 rule instances.

3.3 Kripke semantics

A Biocham model is a set of reaction rules given with an initial state. The formal semantics of a Biocham model is a Kripke structure that is a mathematical structure which provides a firm ground for :

(i) Comparing different modelling formalisms and languages;

(ii) Comparing different models of the same biological system;

(iii) Importing models from other sources;

(iv) Designing and implementing automated reasoning tools.

A *Kripke structure* K is a triplet (S, R, L) where S is the set of states, $R \subseteq S \times S$ is a total relation (i.e. for any state $s \in S$ there exists a state $s' \in S$ such that $(s, s') \in R$) called transition, and $L : S \to 2^A$ is a labelling function over the set of atomic propositions A, which associates to each state the set of atomic propositions true in that state. A path in K starting from a state s_0 is an infinite sequence of states $\pi = s_0, s_1, \ldots$ such that $(s_i, s_{i+1}) \in R$ for all $i \geq 0$.

Clearly, one can associate to a Biocham model a Kripke structure, where the set of states S is the set of all tuples of Boolean values denoting the presence or absence of the different biochemical compounds (molecules, genes and abstract processes), the transition relation R is the union (i.e. disjunction) of the relations associated to the reaction rules, and the labelling function L simply associates to a given state the set of biochemical compounds that are present in the state. Reaction rules in Biocham are asynchronous in the sense that one reaction rule is fired at a time (interleaving semantics), hence the transition relation is the union of the relations associated to the reaction rules. On the other hand, in a synchronous semantics for Biocham, the transition relation would have been defined by intersection. The choice of synchronous semantics was rejected in Biocham as it would bias fundamental biological phenomena such as the masking of a relation by another one and the resulting inhibition or activation of biological processes. Note that as explained in the previous section, the Boolean abstraction of enzymatic reactions used in Biocham associates several transitions to a single Biocham reaction rule, one for each case of possible consumption of the molecules in the left hand side of the rule.

The Kripke structure defines the semantics of a Biocham model as a non-deterministic transition system where the temporal evolution of the system is modelled by the succession of transition steps, and the different possible behaviours of the system are obtained by the non-deterministic choice of reactions.

3.4 Importing biochemical models from other formalisms

Since the basic building block of a Biocham model is an (enzymatic) reaction, it is quite easy to import any model based on such reactions into Biocham. This is the case of most graphical map-based models, but also of some ODE models, derived from the mass-action law or Michaelis-Menten kinetics. A well known source of such models is KEGG [17], which provides (graphical) maps of metabolic and signalling pathways. Biocham has been designed to provide such maps with a simple yet precise semantics.

In this respect, the Biocham project is part of the workpackage entitled "Towards a Bioinformatics SemanticWeb" in the EU network REWERSE.[1] In parallel to this effort, the CMBSlib [28] web site[2] has been created as an open repository of computational models of biological systems, in order to:
(i) Compare different *models* expressed in the same formalism;
(ii) Compare different *formalisms* and *tools* for a same model;
(iii) Cross-fertilize modelling experience and language issues between designers.
This library currently includes models of biological processes obtained from the literature and by translation from KEGG maps or ODE models into different formalisms. It is open to all contributions in any (ASCII) format and in most exotic formalisms.

4. QUERYING BIOCHAM MODELS IN TEMPORAL LOGIC CTL

Thanks to its simple Kripke semantics, Biocham supports the use of Computation Tree Logic (CTL) [11] as a query language for querying the temporal properties of Biocham models. This methodology introduced in [4, 5] is implemented in Biocham with an interface to the state-of-the-art symbolic model checker NuSMV [10].

CTL basically extends propositional logic used for describing states, with operators for reasoning over time and non-determinism. Several temporal operators are introduced in CTL: $X\phi$ meaning ϕ is true at the 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. For reasoning about non-determinism, two path quantifiers are introduced: $A\phi$ meaning ϕ is true on all paths, $E\phi$ 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).

CTL is expressive enough to express a wide range of biological queries:
About reachability: Is there a pathway for synthesizing a protein P, $EF(P)$?
About pathways: Can the cell reach a state s while passing by another state s_2, $EF(s_2 \wedge EF(s))$? Is state s_2 a necessary checkpoint for reaching state s, $\neg E((\neg s_2)\ U\ s)$? 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 or using protein Q, $E(\neg Q\ U P)$?
About stability: Is a certain (partially described) state s of the cell a steady state, $s => EG(s)$? A permanent state, $s => AG(s)$? 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

[1] The 6th EU Framework Programme Network of Excellence REWERSE stands for REasoning on the WEb with Rules and SEmantics, see http://www.rewerse.net
[2] http://contraintes.inria.fr/CMBSlib

cyclic behaviour with respect to the presence of a product P, EG((*P* => *EF* ¬*P*)(¬*P* => *EF P*))? The latter 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. This formula is not expressible in LTL [11], where formulas are of the form $A\phi$ with ϕ containing no path quantifier.

The formal semantics of CTL in a fixed Kripke structure K are given in Table 1, as the inductive definition of the truth relation stating that a CTL formula ϕ is true at state s, written $s \models \phi$, or true along path π, written $\pi \models \phi$ (the clauses for ordinary Boolean connectives are omitted). π^i denotes the suffix of π starting at s_i.

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

$s \models \alpha$	iff	$\alpha \in L(s)$,
$s \models E\psi$	iff	there is a path π from s such that $\pi \models \psi$,
$s \models A\psi$	iff	for every path π from s, $\pi \models \psi$,
$\pi \models \phi$	iff	$s \models \phi$ where s is the starting state of π,
$\pi \models X\psi$	iff	$\pi^1 \models \psi$,
$\pi \models F\psi$	iff	there exists $k \geq 0$ such that $\pi^k \models \psi$,
$\pi \models G\psi$	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 $0 \leq j < k$.

5. THE CHALLENGE OF THE VIRTUAL CELL

High-throughput technologies addressing cell functions at a whole genome scale are revolutionizing cell biology. The challenge of virtual cell projects is to map molecular interactions within the cell, and to build virtual cell models predicting the effects of a drug on a given cell.

Virtual cell environments, like for instance the Virtual Cell project [26] or Cellerator [27], maintain a library of models of different parts of the cell, among different living organisms. ODE models typically range from about ten variables to 50 variables like in the budding yeast cell cycle model of [8]. On the other hand, qualitative models represented by interaction maps allow for the global modelling of a large number of interacting subsystems.

Symbolic model checking techniques used in Biocham are efficient enough to automatically evaluate CTL queries about biochemical networks of several hundreds or thousands of rules and variables [4, 5]. It is worth noting however that this is far below the size of digital circuits that the same model-checking algorithms can treat. The reason for this discrepancy in performance comes from the high level of non-determinism which results from competition between reaction rules and the soup aspect of biochemical solutions.

Combining ODE models with purely qualitative models like current Biocham models is an important issue for managing the complexity of concurrent interacting models. This combination is under investigation within the framework of non-deterministic hybrid systems.

6. LEARNING REACTION RULES FROM TEMPORAL PROPERTIES

With such a simple syntax and semantics for describing reaction rules in Biocham, it is possible to apply learning techniques to reaction rules discovery. We have done some preliminary experiments using the inductive logic programming system Progol [23] for the automatic discovery of missing Biocham reaction rules in a simple model of the cell cycle with 10 variables, given a set of accessibility properties. The basic experiment consists in furnishing a set of examples of accessibility relations and a set of counterexamples, and letting the inductive logic program search for a set of reaction rules satisfying the accessibility properties of the system. In the first phase of validation of the learning technique, where we are now, the models we use are known models from which we compute a set of temporal properties, and remove one or more reaction rules to check whether the missing rules can be recovered by learning from the temporal properties.

More generally, the basic idea is to specify the intended or observed temporal properties of the system with CTL formulae, and apply learning techniques such as inductive logic programming, in order to correct the model by suggesting the addition or modification of Biocham rules in the model.[3]

[3] This approach is investigated in the 6th PCRD EU project APRIL 2 "Applications of Probabilistic Inductive Logic Programming", http://www.aprill.org.

7. CONCLUSION AND PERSPECTIVES

Biocham is a free software[4] for modelling biochemical processes and querying those models in temporal logic. The largest example treated so far is a model of the mammalian cell cycle control [4] after Kohn's diagram [18]. Other models have been imported from interaction maps available on the Web and ODE models. This shows the simplicity of the scheme and the flexibility of this approach.

The pathway logic of [12] is close to Biocham for the algebraic representation of cell compounds and the representation of molecular interactions by rewriting rules. However, the Boolean abstraction used in Biocham and the state-of-the-art symbolic model checker NuSMV permit the handling of potentially larger models. The choice of CTL for expressing biological queries provides also more expressiveness than LTL, which is used in pathway logic. Much can be gained by exchanging Biocham and pathway logic models, cross-fertilizing our modelling experiences and comparing language issues, in particular with respect to the pattern language. The CMBSlib open repository [28] has been created for this purpose as well as for comparison with very different formalisms.

Currently, Biocham is primarily oriented towards the qualitative modelling of biochemical processes and the querying of the temporal properties of Boolean models. This approach can be generalized however to numerical models by relying on constraint-based model checking techniques [5]. In this extension, called Biocham2, variables can denote real values expressing the concentrations of molecules, and rules are extended with constraints to denote the relationship between the old and the new values of the variables. In particular, biochemical systems described by differential equations can be handled in this framework using time discretization methods, and can be combined with Boolean models. The modelling power of such non-deterministic hybrid systems is under investigation.

ACKNOWLEDGMENTS

This work benefitted from various discussions with our colleagues of the ARC CPBIO, in particular with Alexander Bockmayr, Vincent Danos and Vincent Schächter, and of the European project APRIL 2, in particular with Stephen Muggleton.

REFERENCES

1. R. Alur, C. Belta, F. Ivanicic, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin & J. Schug. Hybrid modelling and simulation of biomolecular networks. In: *Proc. 4th Int. Workshop on Hybrid Systems: Computation and Control,* (Rome, Italy) *HSCC'01*, vol. 2034 of *Lecture Notes in Computer Science*, pp. 19–32. Berlin: Springer (2001).
2. ARC CPBIO. Process Calculi and Biology of Molecular Networks, 2002–2003. http://contraintes.inria.fr/cpbio/.
3. A. Bockmayr & A. Courtois. Using hybrid concurrent constraint programming to model dynamic biological systems. In: *Proc. Int. Conf. on Logic Programming*, pp. 85–99, (Copenhagen) *ICLP'02*. Berlin: Springer (2002).
4. N. Chabrier, M. Chiaverini, V. Danos, F. Fages & V. Schächter. Modelling and querying biochemical networks. *Theor. Computer Sci.* To appear, 2004.
5. N. Chabrier & F. Fages. Symbolic model checking of biochemical networks. In: *CMSB'03: Proc. 1st Workshop on Computational Methods in Systems Biology* (ed. C. Priami), vol. 2602 of *Lecture Notes in Computer Science*, pp. 149–162 (Rovereto, Italy) March 2003. Berlin: Springer (2003).
6. N. Chabrier, F. Fages & S. Soliman. *BIOCHAM's user manual*. INRIA, 2003–2004. http://contraintes.inria.fr/BIOCHAM/
7. N. Chabrier-Rivier, F. Fages & S. Soliman. The biochemical abstract machine BIOCHAM. In: *CMSB'04: Proc. 2nd Workshop on Computational Methods in Systems Biology* (eds V. Danos & V. Schächter), Lecture Notes in Computer Science. Springer-Verlag (2004).
8. K.C. Chen, A. Csikász-Nagy, B. Györffy, J. Val, B. Novák & J.J. Tyson. Kinetic analysis of a molecular model of the budding yeast cell cycle. *Mol. Biol. Cell* **11** (2000) 396–391.
9. M. Chiaverini & V. Danos. A core modelling language for the working molecular biologist. In: *CMSB'03: Proc. 1st Workshop on Computational Methods in Systems Biology* (ed. C. Priami), vol. 2602 of *Lecture Notes in Computer Science*, p. 166 (Rovereto, Italy) March 2003. Berlin: Springer (2003).
10. A. Cimatti, E. Clarke, F. Giunchiglia, E. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani & A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In: *Proc. Int. Conf. on Computer-Aided Verification, CAV'02*, Copenhagen, Denmark, July 2002.
11. E.M. Clarke, O. Grumberg & D.A. Peled. *Model Checking*. Cambridge, Mass.: MIT Press (1999).
12. S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer & M. K. Sönmez. Pathway logic: symbolic analysis of biological signalling. In: *Proc. 7th Pacific Symp. on Biocomputing*, pp. 400–412, January 2002.
13. M. Hucka et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19** (2003) 524–531. http://sbml.org
14. R. Ghosh & C. Tomlin. Lateral inhibition through delta-notch signaling: a piecewise affine hybrid model. In: *Proc. 4th Int. Workshop on Hybrid Systems: Computation and Control* (Rome, Italy) *HSCC'01*, vol. 2034 of *Lecture Notes in Computer Science*, pp. 232–246. Berlin: Springer (2001).

[4]The Biocham system can be downloaded from http://contraintes.inria.fr/BIOCHAM

15. R. Hofestädt & S. Thelen. Quantitative modelling of biochemical networks. In: *In Silico Biology*, vol. 1, pp. 39–53. Netherlands: IOS Press (1998).
16. T. Ideker, T. Galitski & L. Hood. A new approach to decoding life: systems biology. *A. Rev. Genomics Human Genetics* **2** (2001) 343–372.
17. M. Kanehisa & S. Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **28** (2000) 27–30.
18. K.W. Kohn. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Mol. Biol. Cell* **10** (1999) 703–2734.
19. W. Kolch, A. Kotwaliwale, K. Vass & P. Janosch. The role of raf kinases in malignant transformation. In: *Expert Rev. Mol. Medicine*, vol. 25. Cambridge: University Press (2002). http://www.expertreviews.org/02004386h.htm
20. A. Levchenko, J. Bruck & P.W. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *Proc. Natl Acad. Sci. USA* **97** (2000) 5818–5823.
21. R. Maimon & S. Browning. Diagrammatic notation and computational structure of gene networks. In: *Proc. 2nd Int. Conf. on Systems Biology* (eds T.-M. Yi, M. Hucka, M. Morohashi & H. Kitano). Online Proceedings (2001). http://www.icsb2001.org/toc.html
22. H. Matsuno, A. Doi, M. Nagasaki & S. Miyano. Hybrid Petri net representation of gene regulatory network. In: *Proc. 5th Pacific Symp. on Biocomputing*, pp. 338–349 (2000).
23. S.H. Muggleton. Inverse entailment and progol. *New Generation Computing* **13** (1995) 245–286.
24. M. Nagasaki, S. Onami, S. Miyano & H. Kitano. Bio-calculus: its concept, and an application for molecular interaction. In: *Currents in Computational Molecular Biology*, vol. 30 of *Frontiers Science Series*. Japan: Universal Academy Press (2000). This book is a collection of poster papers presented at the RECOMB 2000 Poster Session.
25. A. Regev, W. Silverman & E.Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In: *Proc. 6th Pacific Symp. Biocomputing*, pp. 459–470 (2001).
26. J. Schaff & L.M. Loew. The virtual cell. In: *Proc. 4th Pacific Symp. Biocomputing*, pp. 228–239 (1999).
27. B.E. Shapiro, A.Levchenko, E.M. Meyerowitz, B.J. Wold & E.D. Mjolsness. Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. *Bioinformatics* **19** (2003) 677–678. http://wwwaig.jpl.nasa.gov/public/mls/cellerator/
28. S. Soliman & F. Fages. CMBSlib: a library for comparing formalisms and models of biological systems. In: *CMSB'04: Proc. 2nd Workshop on Computational Methods in Systems Biology*, (eds V. Danos & V. Schächter), Lecture Notes in Computer Science. Springer-Verlag (2004).

APPENDIX

BIOCHAM MODEL OF THE MAPK SIGNALLING CASCADE

Here is the full code of the MAPK example given in section 2. The phosphorylation sites for `MEK` and `MAPK` are declared first, and then the Biocham rules are given, sometimes with pattern variables (noted `$P`) which are constrained in the `where` part of the rules. In this model, the first rules are reversible, the other ones are directional. The initial state is partially defined.

```
% MAPK cascade in solution (no scaffold)
%
% adapted from:
% www-aig.jpl.nasa.gov/public/mls/cellerator/notebooks/MAPK-in-solution.html
% by Sylvain Soliman <Sylvain.Soliman@inria.fr>
% Nov. 26, 2003
%
% original source:
% Levchenko, A., Bruck, J., Sternberg, P.W. (2000) .Scaffold proteins may
% biphasically affect the levels of mitogen- activated protein kinase
% signaling and reduce its threshold properties. Proc. Natl. Acad. Sci. USA
% 97( 11):5818-5823. http://www.pnas.org/cgi/content/abstract/97/11/5818

declare MEK~parts_of({p1,p2}).
declare MAPK~parts_of({p1,p2}).

RAF + RAFK <=> RAF-RAFK.

RAF~{p1} + RAFPH <=> RAF~{p1}-RAFPH.

MEK~$P + RAF~{p1} <=> MEK~$P-RAF~{p1}
where p2 not in $P.

MEKPH + MEK~{p1}~$P <=> MEK~{p1}~$P-MEKPH.

MAPK~$P + MEK~{p1,p2} <=> MAPK~$P-MEK~{p1,p2}
where p2 not in $P.

MAPKPH + MAPK~{p1}~$P <=> MAPK~{p1}~$P-MAPKPH.

RAF-RAFK => RAFK + RAF~{p1}.

RAF~{p1}-RAFPH => RAF + RAFPH.

MEK~{p1}-RAF~{p1} => MEK~{p1,p2} + RAF~{p1}.
MEK-RAF~{p1} => MEK~{p1} + RAF~{p1}.

MEK~{p1}-MEKPH => MEK + MEKPH.
MEK~{p1,p2}-MEKPH => MEK~{p1} + MEKPH.

MAPK-MEK~{p1,p2} => MAPK~{p1} + MEK~{p1,p2}.
MAPK~{p1}-MEK~{p1,p2} => MAPK~{p1,p2} + MEK~{p1,p2}.

MAPK~{p1}-MAPKPH => MAPK + MAPKPH.
MAPK~{p1,p2}-MAPKPH => MAPK~{p1} + MAPKPH.
```

```
% Are present in the initial state, the following molecules:
present({
RAFK,
RAF,
MEK,
MAPK,
MAPKPH,
MEKPH,
RAFPH
}).
% All the other ones, which are complexed forms or phosphorylated forms
% are absent
absent({?-?,?~{p1}~?}).
```

The last line of the file uses patterns to declare absent from the initial state all molecules that are complexes (`?-?`) or phosphorylated at `p1` (`?~{p1}~?`). It is equivalent to the following sequence:

```
absent(RAF-RAFK).
absent(RAFPH-RAF~{p1}).
absent(MEK-RAF~{p1}).
absent(MEK~{p1}-RAF~{p1}).
absent(MEKPH-MEK~{p1}).
absent(MEKPH-MEK~{p1,p2}).
absent(MAPK-MEK~{p1,p2}).
absent(MAPK~{p1}-MEK~{p1,p2}).
absent(MAPKPH-MAPK~{p1}).
absent(MAPKPH-MAPK~{p1,p2}).
absent(RAF~{p1}).
absent(MEK~{p1}).
absent(MEK~{p1,p2}).
absent(MAPK~{p1}).
absent(MAPK~{p1,p2}).
```

When loading the model into BIOCHAM, one can see the expanded rules and the rule numbering used to explain the answer to the CTL query of section 2.

```
BIOCHAM 1.1 (C) 2003, 2004 INRIA, France,
by N. Chabrier-Rivier, F. Fages and S. Soliman.
http://contraintes.inria.fr/BIOCHAM
biocham: load_biocham('EXAMPLES/MAPK/mapk.bc').
biocham: expand_rules.
1  RAF+RAFK=>RAF-RAFK.
2  RAF-RAFK=>RAF+RAFK.
3  RAF~{p1}+RAFPH=>RAFPH-RAF~{p1}.
4  RAFPH-RAF~{p1}=>RAF~{p1}+RAFPH.
5  MEK+RAF~{p1}=>MEK-RAF~{p1}.
6  MEK-RAF~{p1}=>MEK+RAF~{p1}.
7  MEK~{p1}+RAF~{p1}=>MEK~{p1}-RAF~{p1}.
8  MEK~{p1}-RAF~{p1}=>MEK~{p1}+RAF~{p1}.
9  MEKPH+MEK~{p1}=>MEKPH-MEK~{p1}.
10  MEKPH-MEK~{p1}=>MEKPH+MEK~{p1}.
11  MEKPH+MEK~{p1,p2}=>MEKPH-MEK~{p1,p2}.
12  MEKPH-MEK~{p1,p2}=>MEKPH+MEK~{p1,p2}.
13  MAPK+MEK~{p1,p2}=>MAPK-MEK~{p1,p2}.
14  MAPK-MEK~{p1,p2}=>MAPK+MEK~{p1,p2}.
15  MAPK~{p1}+MEK~{p1,p2}=>MAPK~{p1}-MEK~{p1,p2}.
16  MAPK~{p1}-MEK~{p1,p2}=>MAPK~{p1}+MEK~{p1,p2}.
17  MAPKPH+MAPK~{p1}=>MAPKPH-MAPK~{p1}.
18  MAPKPH-MAPK~{p1}=>MAPKPH+MAPK~{p1}.
19  MAPKPH+MAPK~{p1,p2}=>MAPKPH-MAPK~{p1,p2}.
20  MAPKPH-MAPK~{p1,p2}=>MAPKPH+MAPK~{p1,p2}.
21  RAF-RAFK=>RAFK+RAF~{p1}.
22  RAFPH-RAF~{p1}=>RAF+RAFPH.
23  MEK~{p1}-RAF~{p1}=>MEK~{p1,p2}+RAF~{p1}.
24  MEK-RAF~{p1}=>MEK~{p1}+RAF~{p1}.
25  MEKPH-MEK~{p1}=>MEK+MEKPH.
26  MEKPH-MEK~{p1,p2}=>MEK~{p1}+MEKPH.
27  MAPK-MEK~{p1,p2}=>MAPK~{p1}+MEK~{p1,p2}.
28  MAPK~{p1}-MEK~{p1,p2}=>MAPK~{p1,p2}+MEK~{p1,p2}.
29  MAPKPH-MAPK~{p1}=>MAPK+MAPKPH.
30  MAPKPH-MAPK~{p1,p2}=>MAPK~{p1}+MAPKPH.
```

It is possible to export a `.dot` file of the rules, to use with the Graphviz[5] visualization suite.
The generated map is depicted in Figure 4.

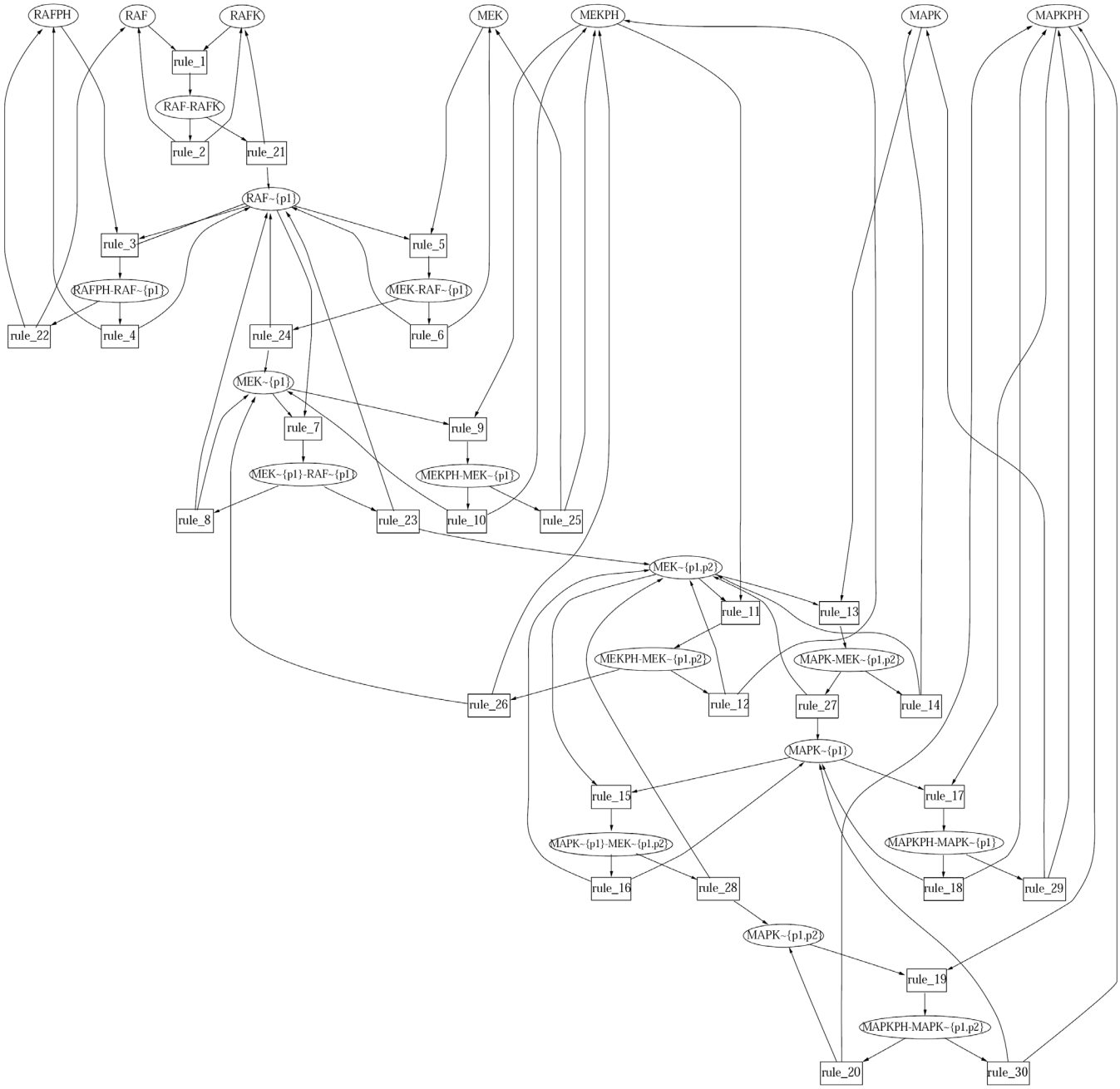

Figure 4. Interaction map generated from BIOCHAM rules for the MAPK cascade.

[5] http://www.research.att.com/sw/tools/graphviz/