The Biochemical Abstract Machine BIOCHAM

Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman
{Nathalie.Chabrier-Rivier,Francois.Fages,Sylvain.Soliman}@inria.fr

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

Abstract. In this article we present the Biochemical Abstract Machine BIOCHAM and advocate its use as a formal modeling environment for networks biology. Biocham provides a 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 behaviors. We present the main features of Biocham, provide details on a simple example of the MAPK signaling cascade and prove some results on the equivalence of models w.r.t. their temporal properties.

1 Introduction

In networks 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 [1]. Several formalisms have been proposed in recent years for modeling biochemical processes either qualitatively [2–4] or quantitatively [5–9]. 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 properties about them. Our focus in Biocham has been on the design of a biochemical rule language and a query language of the model in temporal logic, that are intended to be used by biologists.

Biocham has been designed in the framework of the ARC CPBIO on "Process Calculi and Biology of Molecular Networks" [10] which aims at pushing forward a declarative and compositional approach to modeling languages in Systems Biology. Biocham is a language and a programming environment for modeling biochemical systems, making simulations, and checking temporal properties. It is composed of :

1. a rule-based language for modeling biochemical systems, allowing patterns and constraints in the definition of rules;
2. a simple simulator;
3. a powerful query language based on Computation Tree Logic CTL;
4. an interface to the NuSMV [11] model checker for automatically evaluating CTL queries.

The use of Computation Tree Logic (CTL) [12] 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 to CTL automate reasoning on all the possible behaviors of the system modeled in a purely qualitative way. The semantics of Biocham ensures that the set of possible behaviors of the model over-approximates the set of all behaviors of the system corresponding to different kinetic parameters.

Biocham shares several similarities with the Pathway Logic system [4] 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 wide variety of biological queries, and the use of a state-of-the-art symbolic model checker for handling the complexity of highly non-deterministic models.

The first experimental results of this approach for querying models of biochemical networks in temporal logic have been reported in [13, 14], on a qualitative model of the mammalian cell cycle control [15, 16] and in [14] on a quantitative model of gene expression [9]. In this paper we describe the Biocham system which provides a modeling environment supporting this methodology.

The next section defines the syntax of Biocham objects, rules and patterns, and their semantics. The following section describes the CTL query language and the expression of biological queries. Then we detail Biocham functionalities on a simple model of the MAPK signaling cascade. In section 5 we discuss the comparison of different models of given biological systems and show two equivalence results w.r.t. CTL properties. Section 6 reports our on-going experience in applying inductive logic programming techniques to learning reaction rules from temporal properties, and learning rule patterns from a given set of reaction rules. Finally we compare our approach with related work and conclude on the perspectives of this work.

2 Syntax and Semantics

2.1 A simple Algebra of Biochemical Compounds

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.

Syntax:

object = molecule | abstract
molecule = name | molecule-molecule | molecule~{name,...,name}
| gene | (molecule)
gene = #name
abstract = @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 [4] and other systems [17]. 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 1. `RAF`, `RAF-RAFK` and `RAF~{p1}-MEK` are valid Biocham notations for, respectively, a Raf protein, its complex with a Raf-kinase, and an activated form of Raf in the complex with the MAPK/ERK kinase. `(RAF-MEK)~{p1}` is another notation for the same phosphorylated form of the complex without making precise the constituent which is phosphorylated. Precising or not the phosphorylated constituent defines two formally different complexes. It is also possible to specify more precisely the phosphorylation, for instance by naming its sites, as in `RAF~{ser338,tyr341}`, but then again, this defines a formally different molecule.

The fourth syntactical form is used to denote genes or gene promotors, 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 2. `DMP1-#p19ARF` can be used to denote the binding of protein DMP1 on the promotor of the gene producing protein p19ARF noted `#p19ARF`.

The same assumption of uniqueness is made on abstract objects that are noted with a '@'. Abstract objects can be used to represent particular phases of a process, complete subsystems or abstract biological processes.

Example 3. `@UbiPro` can be used to denote the Ubiquitine/Proteasome subsystem and write this formal object as a catalyst in degradation reaction rules.

2.2 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]=> 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, `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`.

Example 4. `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}`.

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 behaviors of the system with unknown concentration values and unknown kinetics 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 in the left-hand side of the rule. The same goes for control variables, noted with a '@', which are deterministically consumed.

Biocham has also a rich pattern language with constraints which is used to specify molecules and sets of reaction rules in a concise manner, it is detailed in the next section.

2.3 Patterns

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.

Example 5. `list_rules(RAF~?-? + ? => ?).`

This command contains a pattern matching all rules reacting with any form (phosphorylated or complexed) of RAF.

```
list_rules(? =[RAFK]=> ?).
```

This pattern is matching 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.

With some restrictions, patterns can be used to define reaction rules. This is not the case of the above example as the patterns can match unspecified (unconstrained) molecules. Two constructs are provided in Biocham to specify the domains of variables either globally or locally to a rule:

– `declare`: allowing to define all the phosphorylation sites of a molecule;
– `where`: imposing local constraints on variables in a rule declaration.

Example 6. `declare MEK~{{},{p1},{p2},{p1,p2}}.`
```
declare MAPK~parts_of({p1,p2}).
...
MEK~$P + RAF~{p1} <=> MEK~$P-RAF~{p1}
where p2 not in $P.
and   $cycA in {cycA, cks1-cycA}
```
In this context, MEK is declared to have two phosphorylation sites `p1, p2` and that all combinations are possible, MAPK has two sites too, with again all combinations allowed.

Then appears a reaction pattern which specifies the complexation or decomplexation of some forms of MEK not already phosphorylated on p2. This pattern is used to define reactions. There are 2 possible forms (not containing p2) for `$P` to combine with the 2 possibilities for the direction of the rule. This reaction pattern thus expands into 4 reaction rules.

Section 4 contains a Biocham model of the MAPK signaling cascade written with a set of 16 reaction rule patterns which expand into 30 rule instances.

2.4 Kripke Semantics

A Biocham model is a set of reaction rules given with an initial state. As any rule pattern can be expanded into a set of reaction rules, there is no loss of generality in considering only reaction rules. The formal semantics of a Biocham model is a Kripke structure that is a mathematical structure which provides a firm ground for :

– comparing different modeling formalisms and languages,
– comparing different models of a same biological system,
– importing models from other sources,
– and designing and implementing automated reasoning tools.

A *Kripke structure* K is a triple (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 labeling 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 $\pi = s_0, s_1, ...$ 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 (which will be noted $\rightarrow$), and the labeling function L simply associates to a given state the set of biochemical compounds which 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 a synchronous semantics has been 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 Sect. 2.2, 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.

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

3 Querying Biocham Models in Temporal Logic CTL

Thanks to its simple Kripke semantics, Biocham supports the use of the Computation Tree Logic CTL [12] as a query language for querying the temporal properties of Biocham models. This methodology introduced in [13, 14] is implemented in Biocham with an interface to the state-of-the-art symbolic model checker NuSMV [11].

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 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. Simplest queries are *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 with-

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 $0 \leq j < k$.

out creating nor using protein Q, $I \Rightarrow E(\neg Q\ U\ P)$? *About steady states and permanent states*: 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, $EG((P \Rightarrow EF\ \neg P) \wedge (\neg P \Rightarrow EF\ P))$? 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 [12], 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 is 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.

4 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 [18].

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 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 [19] or the SBML repository page [20], both coming from [21]. This cascade was also the first example treated by Regev, Silverman and Shapiro [2] 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 out 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 concentrations axis has been simply split and rescaled to the maximum value for each compound.

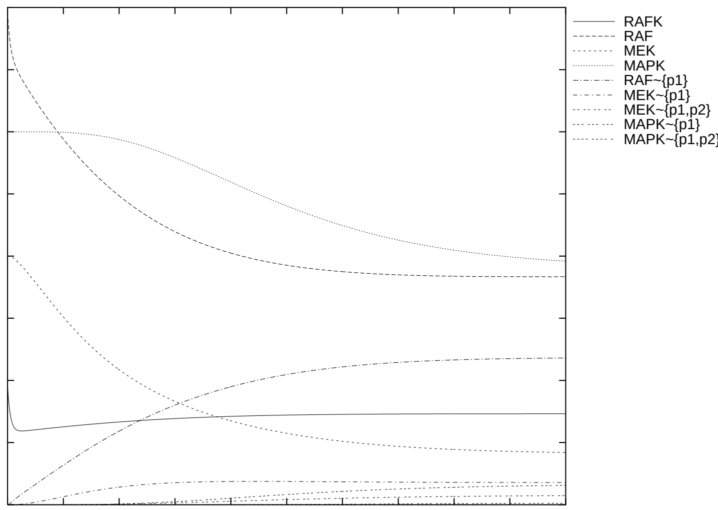

Fig. 1. Simulation result of an ODE model of the MAPK cascade.

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 sensibility of the system to the values of the parameters can be checked by running different simulations with different values of the parameters.

4.1 The MAPK cascade in Biocham Syntax

Our aim in Biocham is to introduce complementary techniques to automate reasoning on all possible behaviors of the system modeled 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. 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 signaling cascade. The semantics of Biocham (explained in Sects. 2.2 and 2.4) ensures that the set of the possible behaviors of the boolean model over-approximates the set of all behaviors of the system for all kinetic parameters' values.

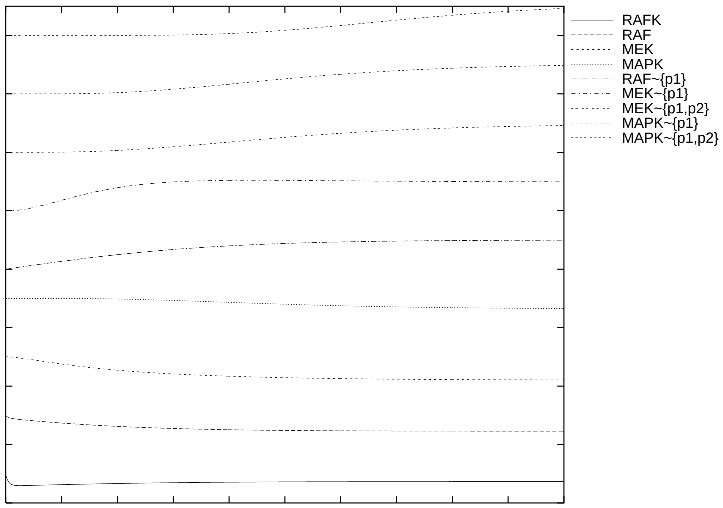

Fig. 2. Same simulation as figure 1, side by side rescaled view.

Here is the full code of the MAPK example[1] in Biocham syntax. 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.

```
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}.
```

[1] adapted from the SBML model `http://www-aig.jpl.nasa.gov/public/mls/cellerator/notebooks/MAPK-in-solution.html`

```
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.
```

These rule patterns define the following set of expanded reaction rules:

```
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.
```

The Biocham rules can be visualized in graphical form by exporting them to a `.dot` file to use with the Graphviz[2] visualization suite. The generated map is depicted in Figure 3.

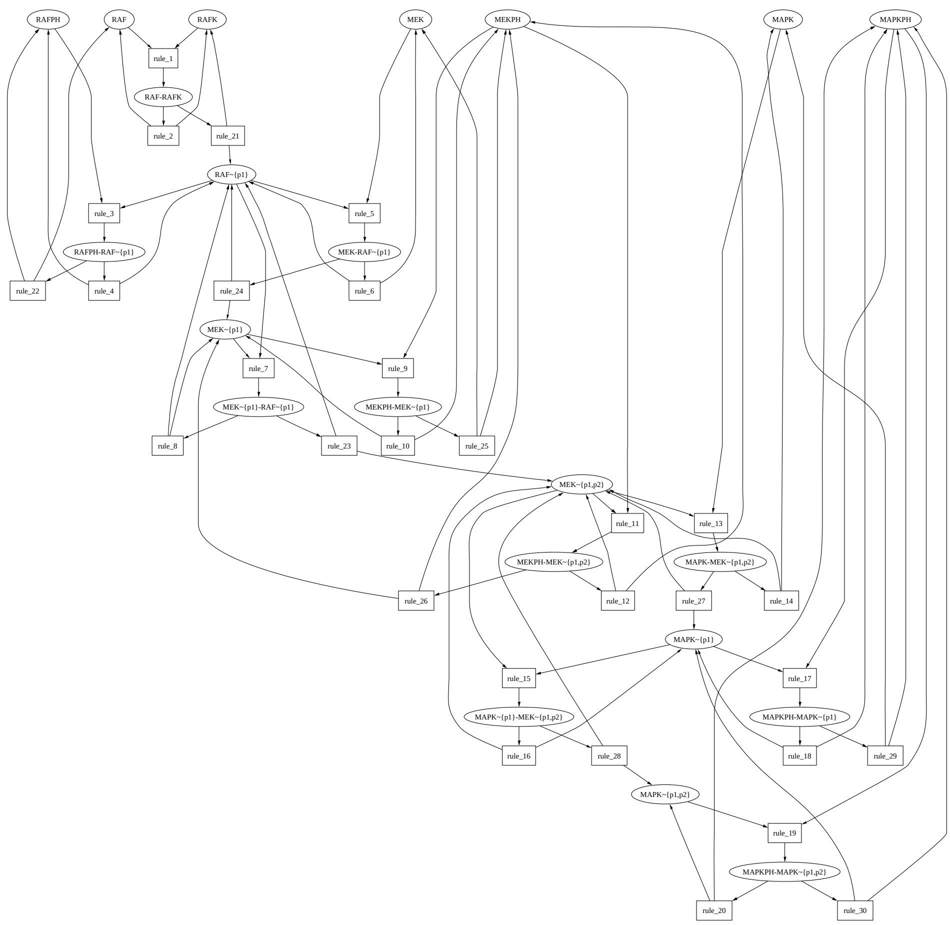

Fig. 3. Interaction map generated from BIOCHAM rules for the MAPK cascade.

4.2 Simulation

Since a Biocham model is highly non-deterministic, simulations are randomized, which means that at each time step, one of the possible reactions happens. An initial state can be defined by taking present the following molecules:

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

```
present({
RAFK,
RAF,
MEK,
MAPK,
MAPKPH,
MEKPH,
RAFPH
}).
```

and absent the following ones:

```
absent({?-?,?~{p1}~?}).
```

The last pattern declares the complexes (`?-?`) and the molecules phosphorylated at `p1` (`?~{p1}~?`) as absent from the initial state. 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}).
```

Figure 4 depicts one random simulation of the MAPK cascade, that is one but only one possible behavior 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.

4.3 CTL Queries

Biocham uses the Computation Tree Logic (CTL) [12] as a query language for querying the temporal properties of the system under all possible conditions. Querying a Biocham model in CTL temporal logic provides a mean to analyze exhaustively all possible behaviors of the system from first principles of enzymatic reactions, in particular when numerical data are not available.

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 to the production

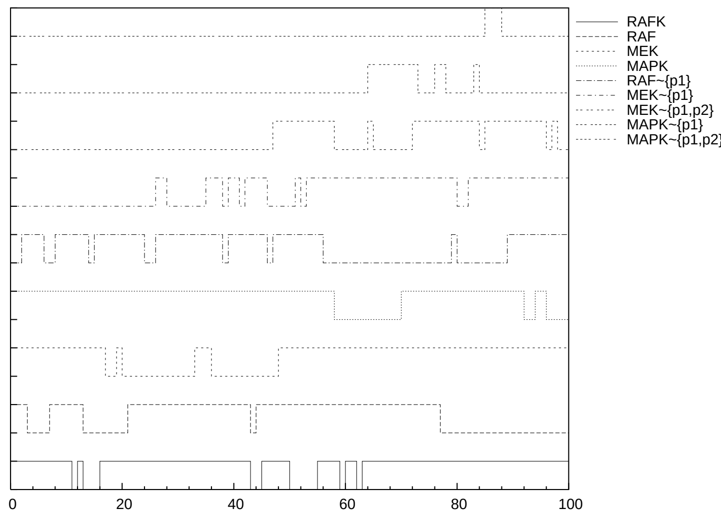

Fig. 4. Random simulation of the Biocham model of the MAPK cascade

of the activated MAPK, noted `MAPK~{p1,p2}`, which is the output of the cascade, that is 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.

The largest example treated so far with Biocham is a model of the Mammalian cell cycle control [13] developed after Kohn's map [16], involving 500 proteins and genes and 147 rule patterns which expand into 2733 rule instances. The computational results reported in [13] 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.

5 Comparing Models

5.1 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 [22], which provides (graphical) maps of metabolic and signaling 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 Semantic Web" in the EU network REWERSE[3].

5.2 CTL-based Equivalence of Models

There are usually lots of different possible models for a same system, depending on the level of detail and of the available knowledge of that system. Even at the level of a single enzymatic reaction, there are already two common ways to specify it into Biocham:

either by detailing the formation of an unstable complex:

```
A + B <=> D
D => A + C
```

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

or directly, as

`A + B => A + C` (which can also be written `B =[A]=> C`)

The first model is a direct translation of common ODE numerical models. However when converting to Biocham, the second model may be more natural as it is simpler. We provide below some equivalence results w.r.t. the CTL properties of both modelings.

We suppose that the complex `D` only appears in the above rule, as otherwise the simplification is obviously not correct. Let us write K_1 for the Kripke structure associated with the first modeling, and K_2 for that associated with the second. These structures define two truth relationships: $\models_1$ and $\models_2$.

Proposition 1 (Reachability). *Let ϕ be an atomic CTL formula, if $I \models_2 EF(\phi)$ then $I \models_1 EF(\phi)$.*

Moreover, if `A` and `B` do not appear negatively (i.e. under an odd number of negations) in ϕ and `D` does not appear at all in ϕ, then $I \models_1 EF(\phi)$ implies $I \models_2 EF(\phi)$.

Proof. Let us first detail the transitions associated with the Biocham rules. In K_1 we have:

1. `A + B` $\rightarrow$ `A + B + D`
2. `A + B` $\rightarrow$ `A + D`
3. `A + B` $\rightarrow$ `B + D`
4. `A + B` $\rightarrow$ `D`
5. `D` $\rightarrow$ `A + B + D`
6. `D` $\rightarrow$ `A + B`
7. `D` $\rightarrow$ `A + C + D`
8. `D` $\rightarrow$ `A + C`

In K_2:

1. `A + B` $\rightarrow$ `A + B + C`
2. `A + B` $\rightarrow$ `A + C`

The first implication is straightforward, since whenever there exists a path in K_2 there exists a path in K_1, which is identical except for the transitions using `A + B` $\rightarrow$ `A + C` that can be replaced by two transitions: `A + B` $\rightarrow$ `D` and `D` $\rightarrow$ `A + C`, and those using `A + B` $\rightarrow$ `A + B + C` that can be replaced by `A + B` $\rightarrow$ `B + D` and `D` $\rightarrow$ `A + C`.

For the second implication, the only difference between K_1 and K_2 w.r.t reachability is the appearance of intermediate states containing a `D` (it is easy to check that all other paths of K_1 can be mimicked in K_2); however `D` does not appear in ϕ and those states only differ from the preceding states by the presence of `D` and the possible absence of `A` and `B`, which cannot be checked in ϕ either. □

Proposition 2 (Checkpoints). *Let $\neg E(\neg\phi U \psi)$ be a checkpoint formula, i.e. ϕ and ψ are atomic formulae describing states,* *if* A *and* B *do not appear negatively in* ψ *and* D *do not appear positively in* ψ*, then*

$I \models_2 \neg E(\neg\phi U \psi)$ *implies* $I \models_1 \neg E(\neg\phi U \psi)$.

if A *and* B *do not appear negatively in* ϕ *and* D *do not appear positively in* ϕ*, then*

$I \models_1 \neg E(\neg\phi U \psi)$ *implies* $I \models_2 \neg E(\neg\phi U \psi)$.

Proof. Let first remind the meaning of $I \models \neg E(\neg\phi U \psi)$: it means that there exist no path π leading to ψ such that ϕ is false in each state of π.

For the first implication, let us consider a path π leading to ψ in K_1; if π ends in a state where D is absent, then one can consider a path π' in K_2, obtained in the same way as in the previous proposition. This path is still leading to ψ in K_2, contains fewer states, ϕ is true in at least one of those states, and thus in one of the states of π. If π ends in a state where D is present, the only difference between that state and the previous (or next) state where D was not present amounts to the appearance of D and possible disappearance of A and B and none of those can be checked by ψ (since it is not positive in D nor negative in A or in B), thus ψ was already true in the state where D was not present and we can apply the above reasoning to get a state where ϕ is true.

For the reverse, let π be a K_2 path leading to ψ, there exists a K_1 path π' leading to the same state, obtained as in the previous proposition. We know that some state s of π' makes ϕ true, and if s is also in π then we are done. Otherwise s is a state such that D is present and once again the only differences with the previous state without D amount to D, A and B and because of the hypotheses made for ϕ, ϕ was already true in the state without D, and thus ϕ is true in a state of π. □

5.3 Enriching Models

To go one step further, there is the need to encompass models written in different formalisms and languages. The CMBSlib [23] web site[4] has been created as an open repository of computational models of biological systems, in order to:

- compare different *models* expressed in the same formalism,
- compare different *formalisms* and *tools* for a same model,
- cross-fertilize modeling 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] `http://contraintes.inria.fr/CMBSlib`

6 Learning

6.1 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 [24] 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, 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 formulas, and apply learning techniques such as inductive logic programming, in order to correct the model by suggesting to add or modify Biocham rules in the model[5].

6.2 Learning Patterns as Generalizations of Existing Rules

The same kind of learning techniques, namely inductive logic programming, can also be applied to the search of generalizations of existing rules, or even of appearing compounds, by the means of Biocham's pattern language.

Since the patterns allow basically rules with variables and constraints on these variables, it is quite straightforward to try and learn such patterns from existing models. Here again the status is that of preliminary experiments, but there is much hope in using this technique to complete partial models.

7 Related Work

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 [25] or Cellerator [19], maintain a library of models of different parts of the cell, among different living organisms. ODE models typically range from a tenth of variables to 50 variables like in the Budding Yeast cell Cycle model of [26]. On the other hand, qualitative models represented by interaction maps allow for the global modeling of a large number of interacting subsystems.

[5] We investigate this approach in the 6th PCRD EU project APRIL 2 "Applications of Probabilistic Inductive Logic Programming", `http://www.aprill.org`.

A formal model of the Mammalian cell cycle control has been developed in the ARC CPBIO [13, 10] after Kohn's map [16]. This model transcribed in Biocham involves 500 proteins and genes and 147 reaction rule patterns which expand into 2733 reaction rule instances. Performance results of CTL querying in this model are reported in [13]. 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 [13, 14]. 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 the competition between reaction rules and the soup aspect of biochemical solutions.

The Pathway Logic of [4] 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 modeling experiences and comparing language issues in particular w.r.t. the pattern language. The CMBSlib open repository [23] has been created for this purpose as well as for comparison with very different formalisms.

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.

8 Conclusion and Perspectives.

Biocham is a free software[6] for modeling biochemical processes and querying these models in temporal logic. The largest example treated so far is a model of the mammalian cell cycle control [13] after Kohn's diagram [16]. 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.

Our first experiments for learning reaction rules from a partial model and reachability properties of the system are encouraging. We are still in the phase of validating the learning method based on Inductive Logic Programming. The next phase will be, in collaboration with biologists, to try to apply learning techniques to the discovery of new reaction rules.

Currently, Biocham is primarily oriented towards the qualitative modeling 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 [14]. In this extension, called Biocham2, variables can denote real values expressing the concentrations of

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

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 modeling power of such non-deterministic hybrid systems is under investigation.

Acknowledgments

This work benefited 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. Ideker, T., Galitski, T., Hood, L.: A new approach to decoding life: Systems biology. Annual Review of Genomics and Human Genetics **2** (2001) 343–372
2. 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
3. Nagasaki, M., Onami, S., Miyano, S., Kitano, H.: Bio-calculus: Its concept, and an application for molecular interaction. In: Currents in Computational Molecular Biology. Volume 30 of Frontiers Science Series. Universal Academy Press, Inc. (2000) This book is a collection of poster papers presented at the RECOMB 2000 Poster Session.
4. 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
5. Matsuno, H., Doi, A., Nagasaki, M., Miyano, S.: Hybrid petri net representation of gene regulatory network. In: Proceedings of the 5th Pacific Symposium on Biocomputing. (2000) 338–349
6. Hofestädt, R., Thelen, S.: Quantitative modeling of biochemical networks. In: In Silico Biology. Volume 1. IOS Press (1998) 39–53
7. Alur, R., Belta, C., Ivanicic, F., Kumar, V., Mintz, M., Pappas, G.J., Rubin, H., Schug, J.: Hybrid modeling and simulation of biomolecular networks. In Springer-Verlag, ed.: Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01. Volume 2034 of Lecture Notes in Computer Science., Rome, Italy (2001) 19–32
8. Ghosh, R., Tomlin, C.: Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. In Springer-Verlag, ed.: Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01. Volume 2034 of Lecture Notes in Computer Science., Rome, Italy (2001) 232–246
9. Bockmayr, A., Courtois, A.: Using hybrid concurrent constraint programming to model dynamic biological systems. In Springer-Verlag, ed.: Proceedings of ICLP'02, International Conference on Logic Programming, Copenhagen (2002) 85–99
10. ARC CPBIO: Process calculi and biology of molecular networks (2002–2003) http://contraintes.inria.fr/cpbio/.

11. 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)
12. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
13. Chabrier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biochemical networks. Theoretical Computer Science **To appear** (2004)
14. 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
15. Chiaverini, M., Danos, V.: A core modeling language for the working molecular biologist. 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) 166
16. Kohn, K.W.: Molecular interaction map of the mammalian cell cycle control and DNA repair systems. Molecular Biology of Cell **10** (1999) 703–2734
17. Maimon, R., Browning, S.: Diagrammatic notation and computational structure of gene networks. In Yi, T.M., Hucka, M., Morohashi, M., Kitano, H., eds.: Proceedings of the 2nd International Conference on Systems Biology, Online Proceedings (2001) http://www.icsb2001.org/toc.html.
18. Kolch, W., Kotwaliwale, A., Vass, K., Janosch, P.: The role of raf kinases in malignant transformation. In: Expert Reviews in Molecular Medicine. Volume 25. Cambridge University Press (2002) http://www.expertreviews.org/02004386h.htm.
19. Shapiro, B.E., Levchenko, A., Meyerowitz, E.M., Wold, B.J., Mjolsness, E.D.: Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. Bioinformatics **19** (2003) 677–678 http://www-aig.jpl.nasa.gov/public/mls/cellerator/.
20. et al., M.H.: The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. Bioinformatics **19** (2003) 524–531 http://sbml.org.
21. 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
22. Kanehisa, M., Goto, S.: KEGG: Kyoto encyclopedia of genes and genomes. Nucleic Acids Research **28** (2000) 27–30
23. Soliman, S., Fages, F.: CMBSlib: a library for comparing formalisms and models of biological systems. In Danos, V., Schächter, V., eds.: CMSB'04: Proceedings of the second Workshop on Computational Methods in Systems Biology. Lecture Notes in Computer Science, Springer-Verlag (2004)
24. Muggleton, S.H.: Inverse entailment and progol. New Generation Computing **13** (1995) 245–286
25. Schaff, J., Loew, L.M.: "the virtual cell". In: Proceedings of the fourth Pacific Symposium on Biocomputing. (1999) 228–239
26. Chen, K.C., Csikász-Nagy, A., Györffy, B., Val, J., Novàk, B., Tyson, J.J.: Kinetic analysis of a molecular model of the budding yeast cell cycle. Molecular Biology of the Cell **11** (2000) 396–391