# Symbolic Model Checking of Biochemical Networks

Nathalie Chabrier, François Fages*

Projet Contraintes, INRIA-Rocquencourt,
BP105, 78153 Le Chesnay Cedex, France,
Nathalie.Chabrier@inria.fr, Francois.Fages@inria.fr

**Abstract.** Model checking is an automatic method for deciding if a circuit or a program, expressed as a concurrent transition system, satisfies a set of properties expressed in a temporal logic such as CTL. In this paper we argue that symbolic model checking is feasible in systems biology and that it shows some advantages over simulation for querying and validating formal models of biological processes. We report our experiments on using the symbolic model checker NuSMV and the constraint-based model checker DMC, for the modeling and querying of two biological processes: a qualitative model of the mammalian cell cycle control after Kohn's diagrams, and a quantitative model of gene expression regulation.

## 1 Introduction

In recent years, Biology has clearly engaged an elucidation work of high-level biological processes in terms of their biochemical basis at the molecular level. The mass production of post genomic data, such as ARN expression, protein production and protein-protein interaction, raises the need of a strong parallel effort on the formal representation of biological *processes*. Metabolism networks, extracellular and intracellular signaling pathways, and gene expression regulation networks, are very complex dynamical systems. Annotating data bases with qualitative and quantitative information about the dynamics of biological systems, will not be sufficient to integrate and efficiently use the current knowledge about these systems. The design of formal tools for *modeling* biomolecular processes and for *reasoning* about their dynamics seems to be a mandatory research path to which the field of formal verification in computer science may contribute a lot.

Several formalisms have been proposed in recent years for the modeling of biochemical networks. Regev and Shapiro [22] were the first to propose the use of a formal concurrent language, namely Milner's $\pi$-calculus, for the modeling of a biochemical processes such as the RTK/MAPK pathway. The bio-calculus of [21] introduces a more biology-oriented syntax for a similar calculus. More

* This work has been done in the framework of the INRIA Cooperative Research Action "Process Calculi and Biology of Molecular Networks", ARC CPBIO, http://contraintes.inria.fr/cpbio

recently, quantitative modelings of biochemical processes have been developed with hybrid Petri nets [19, 16], hybrid concurrent constraint languages [4], and hybrid automata [1, 14].

In this paper we propose to go beyond simulation and to focus on the issue of providing automated methods for *querying and validating formal models* in systems biology. More specifically, we propose,

– first, the use of the temporal logics CTL as a query language for models of biological processes,
– second, the use of concurrent transition systems for the modeling of biological processes.
– and third, the use of symbolic model checking techniques for automatically evaluating CTL queries in both qualitative and quantitative models,

Our approach will be illustrated by two examples: a qualitative model of the mammalian cell cycle control after Kohn's diagrams [6, 17], and a quantitative model of gene expression.

### 1.1 Example 1: The Mammalian Cell Cycle Control

In this example, the main actors are genes, proteins with their phosphorylation sites, multimolecular complexes, and membranes. The molecules interact together to produce new proteins (synthesis), form multimolecular complexes (complexation), modify proteins (phosphorylation and dephosphorylation) degrade or transport molecules.

The cell cycle in eukaryotes is divided into four phases. Between two cell divisions, the cell is in a gap phase called $G_1$. The synthesis phase $S$ starts with the replication of the nucleus. A second gap phase $G_2$ precedes the fourth phase: the mitose phase $M$ during which the cell divides into two cells. The gap phase $G_1$ is mainly responsible for the duration of the cell cycle, it is in fact a growing phase of the cell and may contain a quiescent phase $G_0$ in which the cell can stay for long period of time or forever (stable state) without further division. Each phase is characterized by the activity of two major types of proteins: cyclins and cyclin-dependent kinases (Cdk). Cdk activity requires binding to a cyclin, and is controlled by specific inhibitors and by stimulatory or inhibitory phosphorylations by several kinases or phosphatases which in turn may produce positive feedback loops.

A state of the cell is defined by the values of the actors: either the presence or absence of molecules, or their number, or their concentration in each part of the cell, and by general data like the pH and the temperature. Note that a set of states can be just represented by partial information on the actual values of state variables, like for instance intervals or constraints between variables.

The biological queries one can consider about the cell cycle control are of different kinds:

**About reachability** :

1 Given an initial state *init*, is there a pathway for synthesizing a protein $P$ ?

2 Which are the initial states from which another set of proteins $S$ can be produced ?

**About pathways** :

3 Can the cell reach a state $s$ while passing by another state $s_2$?

4 Is state $s_2$ a necessary checkpoint for reaching state $s$?

5 Can the cell reach a state $s$ without violating certain constraints $c$?

6 From an initial state *init*, is it possible to synthesize a protein $P$ without creating nor using protein $Q$ ?

**About stable states** :

7 Is a certain (partially described) state $s$ of the cell a stable state ?

8 Can the cell reach a given stable state $s$ from the initial state *init*?

9 Must the cell reach a given stable state $s$ from the initial state *init*?

10 What are the stable states?

**About durations** :

11 How long does it take for a molecule to become activated?

12 In a given time, how many Cyclins A can be accumulated?

13 What is the duration of a given cell cycle's phase?

**About the correctness of the model** :

14 *Can one see the inaccuracies of the model, and correct them?*

### 1.2 Example 2: Regulation of Gene Expression

As noted in [24, 9], the dynamics of gene regulatory networks can be modeled by a system of differential equations of the form

$$\dot{x}_i = f_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) * x_i,\ x_i \geq 0,\ 1 \leq i \leq n,$$

where $\boldsymbol{x}$ is a vector of exogenous variables and cellular concentrations of gene products (proteins and mRNAs), $g_i(\boldsymbol{x})$ is the rate of degradation of protein $x_i$, and $f_i$ is a highly non-linear function which expresses the effect of the other variables on the synthesis of $x_i$. Exogenous variables are defined by setting $\dot{x}_i = 0$. The other variables may participate to complex positive or negative feedback loops.

We are interested in answering the following types of queries:

**About activation** :

15 Can protein $x$ reach a concentration greater than a given value $\tau$?

16 Which states may produce a concentration for $x$ greater than some value $\tau$?

**About invariants** :

17 Is a given relationship $c$ between concentrations always satisfied?

### 1.3 Plan of the paper

The next section presents the temporal logic CTL that we propose to use as a query language for biochemical systems. This approach is illustrated by the formalization of the biological queries given above in the two case studies of this paper. Some limits of CTL are discussed w.r.t. biological queries which do not translate directly in CTL.

In Section 3, we focus on the simple formalism of concurrent transition systems for the rule-based modeling of biochemical networks. This is illustrated with a transition system over boolean variables for the cyclin box of the mammalian cell cycle control, and with a transition system over real numbers and linear constraints for a simple example of gene interaction.

Section 4 presents the basic model checking algorithm and the symbolic and constraint-based variants of model checking used in our experiments for querying our biological models in CTL. This section provides some performance figures that show the feasibility of the approach.

Section 5 provides some extra information and references to related work. The last section presents our conclusion.

## 2 The Temporal Logic CTL as a Query Language for Biochemical Models

### 2.1 Preliminaries on CTL

The Computation Tree Logic CTL is a logic for describing properties of computation trees and (non-deterministic) transition systems [8]. CTL is a temporal logic which abstracts from duration values and describes the occurrence of events in the two dimensions of the system: time and non-determinism. CTL basically extends either *propositional* or *first-order* (FO) logic [13], with two path quantifiers for non-determinism: $A$, meaning "for all transition paths", and $E$, meaning "for some transition path", and with several temporal operators: $X$ meaning "next time", $F$ meaning "eventually in the future", $G$ meaning "always", $U$ meaning "until".

A "safety" property, specifying that some situation described by a formula $\phi$ can never happen, is expressed by the CTL formula $AG\neg\phi$, i.e. on all paths $\phi$ is always false. A "liveness" property, specifying that something good $\psi$ will eventually happen, is expressed by the formula $AF\psi$. Note that by duality we have $EF\phi = \neg AG\phi$ and $EG\phi = \neg AF\neg\phi$ for any formula $\phi$.

Formally, CTL formulas are divided into state formulas and path formulas. Let $AP$ be a set of atomic propositions, describing states. A *state formula* is either an atomic proposition, or a path formula prefixed by a path quantifier, or a logical combination of such formulas. The set of *path formulas* is the closure of the set of state formula by the temporal operators and logical connectives. Arbitrary state and path formulas form CTL$^*$ formula. CTL logic is a syntactic fragment of CTL$^*$ in which the temporal operators must be immediately prefixed by a path quantifier. For example, $A(FG\phi)$ and $E(F\phi \wedge G\psi)$ are CTL$^*$ formula which are not expressible in CTL. In this paper we shall only be concerned with the fragment of CTL formulas.

The semantics of CTLis given by Kripke structures. A *Kripke structure* $K$ is a triple $(S, R, L)$ where $S$ is a set of states, $R \subseteq S \times S$ is a (transition) relation and $L : S \rightarrow 2^{AP}$ is a function that associates to each state the set of atomic propositions true in that state. A path in $K$ 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 $i \geq 0$. We denote by $\pi^i$ the suffix of $\pi$ starting at $s_i$. Now the inductive definition of the truth relation stating that a CTL formula $\phi$ is true in $K$ at state $s$, noted $K, s \models \phi$, or true in $K$ along path $\pi$, noted $K, \pi \models \phi$, is the following (the standard rules for logical connectives are omitted):

- $K, s \models \phi$ iff $s \models \phi$, if $\phi$ is a state formula,
- $K, s \models E\phi$ iff there is a path $\pi$ from $s$ such that $K, \pi \models \phi$,
- $K, s \models A\phi$ iff for every path $\pi$ from $s$, $K, \pi \models \phi$,
- $K, \pi \models \phi$ iff $s \models \phi$ where $s$ is the starting state of $\pi$, if $\phi$ is a state formula,
- $K, \pi \models X\phi$ iff $K, \pi^1 \models \phi$,
- $K, \pi \models F\phi$ iff there exists $k \geq 0$ such that $K, \pi^k \models \phi$,
- $K, \pi \models G\phi$ iff for every $k \geq 0$, $K, \pi^k \models \phi$,
- $K, \pi \models \phi U \psi$ iff there exists $k \geq 0$ such that $K, \pi^k \models \psi$ and $K, \pi^j \models \phi$ for all $0 \leq j < k$.

Following [13], assuming a Kripke strucutre $K$, we shall identify a CTL formula $\phi$ to the set of states which satisfy it, i.e. $\{s \in S | K, s \models \phi\}$. Thus, by abuse of notation, we will write $s \in \phi$ if $\phi$ is true in state $s$ in $K$.

### 2.2 Example 1: The Mammalian Cell Cycle Control

The examples of biological questions listed in the previous section translate into CTL as follows.

**About reachability** :

1 $init \in EF(P)$,

2 $EF(S)$, the CTL formula is indeed a representation of all states satisfying it, model checking tools provide facilities for enumerating explicitly these states.

**About pathways** :

3 $EF(s_2 \wedge EFs)$,

4 $\neg E((\neg s_2)\ U\ s)$,

5 $E(c\ U\ s)$,

6 $init \in E(\neg Q\ U\ P)$,

**About stable states** :

7 $s \in AG(s)$, a stable state in the strong sense is a state in which the cell stays indefinitely with no possibility of escaping; a state in which the cell can stay indefinitely but can escape from, can be modeled by $s \in EG(s)$,

8 $init \in EF(AGs)$,

9 $init \in AF(AGs)$.

10 The set of stable states of the system cannot be represented by a CTL query. In CTL, it is only possible to check whether a given (partially described) state is a stable state. One approach to computing the set of stable states (or checkpoints, etc.) of a biochemical network would be to combine model checking methods with search methods, that is an interesting open problem.

**About durations** :
Time in temporal logic CTL is a purely qualitative notion, based on a single precedence relation. Reasoning about durations is thus not expressible with the temporal operators of CTL. Nevertheless, if the state description logic underlying CTL is not propositional but first-order, it is always possible in FO to model time intervals by adding to all atomic propositions extra numerical arguments representing their starting time and duration. Constraint-based model checking presented in section 4.1 provides an automatic method for evaluating such queries.

**About the correctness of the model** : When an intended property is not verified, the pathways leading to a counterexample help the user to refine the model. Similarly, when an unintended property is satisfied, the pathway leading to a witness helps the user to refine his model by enforcing extra conditions in rules, or, if the property is not known to be biologically true or false, the witness may suggest to do biological experiments in order to validate or invalidate that property of the model. In biology, the standard loop between modeling and model-validation is in fact a three fold loop between modeling, querying the model and doing biological experiments.

### 2.3 Example 2: Regulation of Gene Expression

The second series of questions for the example of gene regulation can be translated in CTL as follows. It is worth noting that in this second series of examples, the setting of first-order logic is useful to express the constraints in CTL queries [10, 13].

**About activation** :
15 $init \in EF(x > \tau)$,
16 $EF(x > \tau)$,

**About invariants** :
17 $init \in AG(c)$.

The same remark as above about the tools for correcting the model or suggesting biological experiments, applies as well.

## 3 Modeling Biochemical Networks with Concurrent Transition Systems

Concurrent transition systems have been introduced in the scheme of [23] for reasoning about concurrent programs. Concurrent transition systems offer a direct way of specifying a Kripke structure by reaction rules and we shall use them for this reason for the modeling of biochemicals networks.

A *concurrent transition system* is a Kripke structure presented as a triple $(\boldsymbol{x}, I, R)$ where $\boldsymbol{x}$ is a tuple of (data and control) variables, $I$ is a formula on $\boldsymbol{x}$ expressing the initial condition as a set of values for all variables, and $R$ is a set of condition-action rules. The rules have the following syntax:

$$\textit{condition } \phi(\boldsymbol{x}) \quad \textit{action } \boldsymbol{x}' = \rho(\boldsymbol{x})$$

where $\phi(\boldsymbol{x})$ denotes the condition under which the rule can be applied, and the primed version of the variables denotes the new values $\rho(\boldsymbol{x})$ of the variables after the rule is applied. By convention, the variables which are not modified in the right hand side of the rule keep their value unchanged.

Clearly, a concurrent transition system defines a Kripke structure, where the set of states is the set of all tuples of values for the variables, the initial state is the tuple of values satisfying the initial condition, and the transition relation is the union[1] (i.e. disjunction) of the relations between the states of all instances of the condition-action rules.

In the following, we shall associate a data variable to each molecule (protein or gene). The value of a variable will be either a numerical value expressing the concentration of the molecule, or a boolean value expressing simply the presence or absence of the molecule. The temporal evolution of the system will be modeled by the transition steps. The different transition paths will model the non-deterministic behavior of the system.

### 3.1 Example 1 : The Mammalian Cell Cycle Control

In [17], Kohn provided an annotated diagrammatic representation of the molecular interaction map of the mammalian cell cycle control and DNA repair systems. The part concerning the cell cycle control, and with more details the Cyclin box and the E2F box, have been modeled in the ARC CPBIO by M. Chiaverini and V. Danos [6], as a set of 732 reaction rules over 165 molecules, and 532 variables taking into account the different forms of a molecule. The beauty of this model is that each rule is an instance of one of the following five rule schemas:

1. Complexation : $A \wedge B \rightarrow AB$,
   two molecules $A$ and $B$ bind together to form a multimolecular complex $AB$;
2. Phosphorylation : $A \wedge B \rightarrow Ap \wedge B$,
   molecule $A$ is modified under the action of a catalyst $B$, $A$ is transformed in a phosphorylated form $Ap$,
3. Dephosphorylation : $Ap \wedge B \rightarrow A \wedge B$,
   the phosphorylated molecule $Ap$ is dephosphorylated by catalyst $B$;
4. Synthesis : $A \rightarrow A \wedge B$,
   molecule $B$ is synthesized by the gene with the activated promotor $A$;

[1] Concurrent transition systems are asynchronous in the sense that one rule is executed at a time (interleaved semantics), hence the transition relation is the union of the relations associated to the rules. On the other hand, synchronous programs, that are not considered in this paper, have their transition relation defined by intersection.

5. Degradation : $A \wedge B \rightarrow A$,
   molecule $B$ is degraded by molecule $A$.

In this model, each type of molecule is modeled by a variable. By lack of quantitative data, the variables associated to molecules are all boolean. They simply express the presence or absence of the molecule in the cell. This model of the mammalian cell cycle control is thus a purely logical model. For example, CycH, Cdk7 and CycH-Cdk7 are three variables representing respectively, Cyclin H, Cdk 7 and the dimer Cyclin H-Cdk 7. In the complexation rule schema, $AB$ stands for a propositional variable denoting the multimolecular complex which results from the binding of the molecules denoted by $A$ and $B$. An instance of this schema is the rule CycH $\wedge$ Cdk7 $\rightarrow$ CycH-Cdk7. Trimers, tetramers and more generally polymers can be formed by applying the complexation schema to dimers, etc. Note that it is possible to distinguish between the complexes (AB)C and A(BC) if there are biological reasons to make this distinction.

Similarly, one can introduce a variable named Cdk1(phosphorylated at Thr14)-cyclinB, to represent a phosphorylated form of the dimer Cdk1-Cyclin B at site Thr 14 of Cdk 1. The phosphorylation of this dimer by Mytosine 1 is modeled by the rule instance: Cdk1-CycB $\wedge$ Myt1 $\rightarrow$ Cdk1(phosphorylated at Thr14)-CycB $\wedge$ Myt1. An instance of the dephosphorylation rule is: Cdk1(phosphorylated at Thr14 and Tyr15)-CycB $\wedge$ Cdc25C(phosphorylated in N-terminal domain) $\rightarrow$ Cdk1-cyclinB $\wedge$ Cdc25C(phosphorylated in N-terminal domain).

For the sake of conciseness, we have used the following convention in the rule schemas for denoting condition-action rules. The left hand side of a rule is just its condition. The right hand side is a formula which expresses which variables are made true in the action, with the convention that the variables which do not appear in the schema remain unchanged, and the variables which appear in the left hand side and not in the right hand side of the schema may take *arbitrary values*. The rule schema of complexation is thus a short-hand for the four condition-action rules:

$\textit{condition } A \wedge B \quad \textit{action } AB' = true, A' = true, B' = true$
$\textit{condition } A \wedge B \quad \textit{action } AB' = true, A' = false, B' = true$
$\textit{condition } A \wedge B \quad \textit{action } AB' = true, A' = true, B' = false$
$\textit{condition } A \wedge B \quad \textit{action } AB' = true, A' = false, B' = false$

The condition-action rules make explicit the possible disappearance of molecules $A$ and $B$ by complexation.

### 3.2 Example 2 : Regulation of Gene Expression

Following Euler's method for solving differential equations numerically, one can associate a discrete, yet infinite state, transition system to a system of differential equations.

We shall use the following pedagogical example of interaction between two genes taken from [4]:

$\dot{y} = 0.01 * x$,
$\dot{x} = 0.01 - 0.02 * x$ if $y < 0.08$,

$\dot{x} = -0.02 * x$ if $y \geq 0.08$.
Gene $x$ activates gene $y$, but above a certain threshold, gene $y$ inhibits expression of gene $x$.

A discretization by one time unit $dt$ (e.g. $dt = 1$) leads to the following simple transition system:

*condition* $y < 0.8$ *action* $x' = x + (0.01 - 0.02 * x) * dt,\ y' = y + 0.01 * x * dt$

*condition* $y \geq 0.8$ *action* $x' = x - 0.02 * x * dt,\ y' = y + 0.01 * x * dt$

The transition system in this example is deterministic but it is worth noting that this is not required by the scheme. Note that the derivatives can be added to the states of the system in order to reason or express queries about them. Dynamic discretizations are possible by adding the time step $dt$ as a state variable, similarly to multirate simulation in hybrid systems.

# 4 Model Checking for Systems Biology

## 4.1 Preliminaries on Model Checking

Model checking is an algorithm for computing, in a given Kripke structure $K$, the set of states which satisfy a given CTL formula $\phi$, i.e. the set $\{s \in S | K, s \models \phi\}$. For the sake of simplicity, we consider only the CTL fragment of CTL, and use the fact that (by duality) any CTL formula can be expressed in terms of $\neg$, $\vee$, $EX$, $EU$ and $EG$.

When $K$ has a finite set of states, the model checking algorithm, in its simplest form, works with an explicit representation of $K$ as a transition graph, and labels each state with the set of subformula of $\phi$ which are true in that state. First, the states are labeled with the atomic propositions of $\phi$ which are true in those states. The labeling of more complex formula is done iteratively, following the syntax of the subformula of $\phi$. Formulas of the form $\neg\phi$ label those states which are not labeled by $\phi$. Formulas of the form $\phi \vee \psi$ are added to the labels of the states labeled by $\phi$ or $\psi$. Formulas $EX\phi$ are added to the labels of the predecessor states of the states labeled by $\phi$. Formulas $E(\phi U \psi)$ are added to the predecessor states of $\psi$ while they satisfy $\phi$. Formulas $EG\phi$ involve the computation of the strongly connected components of the subgraph of transitions restricted to the states satisfying $\phi$. The states labeled by $EG\phi$ are the states in this subgraph for which there exists a path leading to a state in a non trivial strongly connected component. The complexity of this algorithm is $O(|\phi| * (|S| + |R|))$ where $|\phi|$ is the size of the formula, $|S|$ is the number of states, and $|R|$ is the number transitions [8].

Symbolic model checking is a more efficient algorithm that uses a symbolic representation of finite Kripke structures with boolean formulas. In particular, the whole transition relation is encoded as a single (disjunctive) boolean formula, sets of states are encoded by boolean formulas, and ordered binary decision diagrams (OBDDs) are used as canonical forms for the boolean formulas. The symbolic model checking algorithm computes an OBDD representing the set of states satisfying a given CTL formula. The computation involves the iterative computation of the least fixed point (for $EF$) and the greatest fixed point (for $EG$) of simple predicate transformers associated to the temporal connectives [8]. In our experiments reported below, we used the state-of-the-art symbolic model checker NuSMV [7].

Constraint-based model checking applies to infinite state Kripke structures, such as Kripke structures with variables ranging over unbounded or continuous numerical domains. A constrained state is a finite representation using constraints, of a finite or infinite set of states. In the scheme of Delzanno and Podelski [10], infinite state Kripke structures are represented by constraint logic programs, and the CTL formulas, that are based on a fragment of first-order logic, are identified to the least fixed point and greatest fixed point of such programs. In our experiments reported below about the quantitative model of gene expression regulation, we used the implementation in Sicstus Prolog with constraints over finite domains and real numbers (simplex algorithm) of the model checker DMC [11].

### 4.2 Symbolic Model Checking of Logical Models

| Type | Query | Pathway length | Number of DMC states | DMC time in seconds | NuSMV time in seconds |
|---|---|---|---|---|---|
| | compiling | - | - | - | 47.5 |
| 2 | EF(cycE) | 6 | 279 | 1320 | 16.5 |
| 2 | EF(SL1_1) | 10 | 2107 | 29970 | 57.8 |
| 2 | EF(cycA) | 6 | 1072 | 23161 | 16.8 |
| 2 | EF(PCNA) | 6 | 245 | 2524 | 23.7 |
| 4 | ¬E( (¬(Cdc25-active)) U Cdk1-CycB-active ) | - | - | - | 112 |

**Table 1.** Evaluation of CTL queries in the mammalian cell cycle control model with DMC and NuSMV.

The Table 1 provides some performance figures about the evaluation of CTL queries in the mammalian cell cycle control model. The two first columns indicate the query and its type . The third column indicates the length of the pathway leading to a counterexample or to a witness. The fourth column indicates the number of state expressions computed by DMC. The two last columns indicate the CPU time in seconds measured on a Pentium 4 at 660 Mhz for DMC and NuSMV. The NuSMV timings (which include the reconstruction of a pathway) show the efficiency of the OBDD representation of states compared to the simple representation of states in Prolog (without boolean constraints) used in DMC.

### 4.3 Constraint-based Model Checking of Quantitative Models

Constraint-based model checking of quantitative models must be contrasted with symbolic model checking techniques which use finite domain abstraction techniques to deal with quantitative models [25]. The constraint-based model checker DMC [11] performs a backward reachability analysis, starting from a constrained state expressing the CTL property to prove, up to the computation of a state expression containing the initial state. Safety (resp. liveness) properties involve the computation of the least (resp. greatest) fixpoint of a constraint logic program.

It is worth noting that this kind of reasoning mixes symbolic computation on set of states described by *numerical constraints*, with a form of reasoning by induction. In the simple example of gene regulation, the query $EF(x \geq 0.5)$ immediately evaluates to false, as any predecessor state of a state described by the constraint $x \geq 0.5$ again satisfies that constraint and is thus subsumed.

The table below show some performance figures in the Prolog implementation of DMC. Better performance results could be obtained by using other discretizations of the problem, other translations involving Runge-Kutta method, and other implementations of the constraints.

| Type | Request | Pathway length | Number of states | Computing time (sec) |
|---|---|---|---|---|
| 16 | EF(x≥0.5) | 0 | 1 | 0.02 |
| 16 | EF(x≥0.2) | 28 | 29 | 3.65 |
| 16 | EF(x≥0.45) | 116 | 117 | 59 |
| 16 | EF(y≥0.8) | 212 | 213 | 256 |
| 16 | EF(x+y≥1.3) | 178 | 179 | 173 |
| 16 | EF(x+y≥1.2) | 194 | 195 | 206 |
| 17 | AG(x>0.5) | 1 | 2 | 0.03 |
| 17 | AG(x<0.1) | 14 | 127 | 8 |
| 17 | AG(y<0.8) | 211 | 421 | 7 |

**Table 2.** Examples of CTL queries in the example of gene interaction.

## 5 Discussion and Related Work

The experiments reported in this paper should be read as a proof-of-concept rather than as providing an already usable accurate modeling of biological systems. Some errors or ambiguities were corrected with the biologists of the ARC CPBIO, but more work with biologists is needed to validate further the formal modeling of Kohn's diagram as a concurrent transition system, and incorporate more knowledge in the model.

The pathway logic of S. Eker et al. [12] is tightly related to our approach. The modeling of biochemical networks with concurrent transition systems is of a somewhat lower level than with pathway logic. Pathway logic is indeed more expressive as it can express algebraic properties of the components, such as the commutativity and associativity of complexation. This capability can be used to infer the possible reactions of molecules from their logical structure. It is worth considering however, that the interaction capabilities of a protein are often not related to the ones of its components, as they depend on the 3D structure of the proteins which is obviously impractical to take into account in a global modeling [3].

One limitation to the modeling of biological systems with concurrent transition systems is the necessity to encode all the parameters of the system in a finite vector of data and control variables. In order to get rid of this difficulty, we are currently investigating the use of other model checkers based on linear logic or multiset rewriting, that dont't have this restriction and make it possible to reason about systems within an arbitrary context [5].

Another obvious limitation in the experiments reported here is the absence of stochastic data. There are however stochastic model checking methods [18, 20] which can be investigated to circumvent this limitation.

The performances of our model checker can be improved in many ways as we have already shown with the use of DMC and NuSMV for the logical model of the mammalian cell cycle control. Similar improvements will be necessary to show the scalability of this approach for quantitative models. In this respect, constraint-based model checking is thightly related to hybrid systems methods and to hybrid verification tools such as for example Hytech [15] or $d/dt$ [2]. Approximation techniques coming from hybrid automata can be imported in constraint-based model checking with the framework of abstract interpretation. On the other hand, constraint-based model checking provides a method for generalizing hybrid verification tools, going from pure reachability analysis towards more general CTL query evaluation.

## 6 Conclusion

We have applied symbolic model checking techniques to the querying and validation of both quantitative and qualitative models of biomolecular systems. Our first experiments show some advantages over simulation. Constraint reasoning makes it possible to group large or infinite sets of states into small constrained state expressions which provide formal proofs of reachability, pathway, checkpoint and stability properties. In some cases the properties can be checked by computing a fewer number of states than by simulation. It is also possible to reason with infinite sets of initial states finitely represented by constraints. Moreover the proof method applies to non-deterministic systems, for which simulations may be unfeasible.

We have also shown that constraint-based model checking can be applied in quantitative models described by differential equations. In our experiments we used a symbolic variant of Euler's method, but the use of the more accurate

Runge-Kutta method, of non-linear constraint solving by interval propagation, and the use of abstraction techniques open many ways for improvement.

For all these reasons, we believe that, beyond simulation, verification tools such as model checking will become indispensable for querying and validating complex models in systems biology.

**Acknowledgement :** We gratefully acknowledge the interactions we had with our colleagues of the ARC CPBIO, especially with Magali Roux-Rouquié, Julien Renner and Grégory Sautejeau from Institut Pasteur, for interesting discussions on Systems Biology and relevant bits of Biomolecular Biology, Vincent Danos and Marc Chiaverini from CNRS PPS Lab. for their beautiful transcription of Kohn's diagrams in their core modeling language, Vincent Schächter at Genoscope Evry, for his insights on the validation of biological models, Alexander Bockmayr, Arnaud Courtois and Damien Eveillard from the ModBio group at LORIA Nancy, for fruitful discussions on quantitative models.

## References

1. R. Alur, C. Belta, F. Ivancic, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In Springer, editor, *Hybrid Systems: Computation and Control*, LNCS 2034, pages 19–32, Rome, Italy, 2001.
2. E. Asarin, T. Dang, and O. Maler. d/dt: A verification tool for hybrid systems. In *Invited session "New Developments inVerification Tools for Hybrid Systems", in Proceedings of the Conference on Decision and Control*, Florida, USA, July 2001.
3. R. Backofen, S. Will, and E. Bornberg-Bauer. Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics*, 3(15):234–242, 1999.
4. A. Bockmayr and A. Courtois. Using hybrid concurrent constraint programming to model dynamic biological systems. In Springer, editor, *18th International Conference on Logic Programming*, pages 85–99, Copenhagen, 2002.
5. M. Bozzano, G. Delzanno, and M. Martelli. Model checking linear logic specifications. Technical report, Technical report,University di Genova, March 2002.
6. M. Chiaverini and V. Danos. A core modeling language for the working molecular biologist. Technical report, CNRS, PPS, Paris 7, November 2002.
7. A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *In Proceeding of International Conference on Computer-Aided Verification, CAV'2002*, Copenhagen, Danmark, July 2002.
8. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
9. H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):69–105, 2001.
10. G. Delzanno and A. Podelski. Model checking in clp. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems TACAS'99*, volume 1579 of *LNCS*, pages 223–239. Springer-Verlag, January 1999.
11. G. Delzanno and A. Podelski. *DMC user guide*, 2000.

12. S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and K. Sonmez. Pathway logic: Symbolic analysis of biological signaling. In *the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.
13. E.A. Emerson. *Temporal and Modal Logic*, pages 995–1072. J. van Leeuwen Ed., North-Holland Pub. Co./MIT Press, 1990.
14. R. Ghosh and C. Tomlin. Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. In Springer, editor, *Hybrid Systems: Computation and Control*, LNCS 2034, pages 232–246, Rome, Italy, 2001.
15. T. Henzinger, J. Preusig, and H.Wong-Toi. Some lessons from the hytech experience. In *Proceedings of the 40th Annual IEEE Conference on Decision and Control, CDC'2001*, 2001.
16. R. Hofestädt and S. Thelen. Quantitative modeling of biochemical networks. In *In Silico Biology*, volume 1, pages 39–53. 1998.
17. K.W. Kohn. Molecular interaction map of the mammalian cell cycle control and dna repair systems. *Molecular Biology of Cell*, 10(8):703–2734, August 1999.
18. S. Laplante, R. Lassaigne, and F. Magniez. Probabilistic model checking: an approach based on property testing. In *Proc. of the 7th annual IEEE symposium on Logic in Computer Science LICS'02*, Copenhagen, 2002.
19. H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid petri net representation of gene regulatory network. In *Pacific Symposium on Biocomputing (5)*, pages 338–349, 2000.
20. D. Monniaux. *The analysis of probabilistic programs by abstract interpretation*. PhD thesis, Ecole Normale Supérieure, Paris, France, 2001.
21. M. Nagasaki, S. Onami, S. Miyano, and H. Kitano. Bio-calculus: Its concept, and an application for molecular interaction. In *Currents in Computational Molecular Biology.*, volume 30 of *Frontiers Science Series*. 2000.
22. A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Proceedings of the Pacific Symposium of Biocomputing*, pages 6:459–470, 2001.
23. U. A. Shankar. An introduction to assertionnal reasoning for concurrent systems. *ACM Computing Surveys*, 3(25):225–262, 1993.
24. R. Thomas and d'Ari. *Biological feedback*. CRC press, 1990.
25. A. Tiwari and P. Lincoln. Automated technique for stability analysis of delta-notch lateral inhibition mechanism. Technical report, SRI, Stanford USA, 2002.