Modeling and Querying Biomolecular Interaction Networks

Nathalie Chabrier-Rivier* Marc Chiaverini†

Vincent Danos‡ François Fages§ Vincent Schächter¶

December 4, 2003

Abstract

We introduce a formalism to represent and analyze protein-protein and protein-DNA interaction networks. We illustrate the expressivity of this language, by proposing a formal counterpart of Kohn's compilation on the mammalian cell cycle control. This effectively turns an otherwise static knowledge into a discrete transition system incorporating a qualitative description of the dynamics. We then propose to use the Computation Tree Logic CTL as a query language for querying the possible behaviours of the system. We provide examples of biologically relevant queries expressed in CTL about the mammalian cell cycle control and show the effectiveness of symbolic model checking tools to evaluate CTL queries in this context.

1 Introduction

In recent years, molecular biology has engaged in a large-scale effort to elucidate cellular processes in terms of their biochemical basis at the molecular level. Mass production of post genomic experimental results, such as mRNA expression data, protein expression or protein-protein interaction data, is following and completing the initial piecemeal catalog of elementary components – genes and proteins – of the sequencing and genomic analyses projects by progressively painting a global picture of the complex interactions that take place in a cell. Exploiting these experimental data to understand the underlying processes requires much more than database integration and storage: it calls for a strong parallel effort on the formal representation of biological *processes*.

Several formalisms have been proposed in recent years for the modeling of metabolic pathways, extracellular and intracellular signaling pathways, or gene regulatory networks: boolean networks [TT98], ordinary differential equations [SEJGM02], and more recently hybrid Petri nets [MDNM00, HT98] and hybrid automata [ABI$^+$01, GT01]. Formal concurrent languages were also considered, including hybrid concurrent constraint languages [BC02], or rewriting

*INRIA Rocquencourt, France.
†University of Paris 7, France.
‡CNRS Paris 7, France.
§INRIA Rocquencourt, France.
¶Genoscope Evry, France.

logics [EKL+02]. Regev and Shapiro [RSS01] were the first to propose the use of the π-calculus [MPW92].

Most formal approaches mentioned above proceed by wholesale importation of a language (*e.g.* Petri nets, the π-calculus) that emerged in answer to very specific design goals, some of which may be relevant to our present modeling task, and some of which may not. While the expected benefit is direct inheritance of preexisting methods and tools, this results in some contorted translations and the existence of useless constructs, and somewhat defeats the explanatory purpose of the formalization. We advocate a different approach: the *ab initio* design of formal languages to represent a chosen subset of biological phenomenology, along with adaptation or redesign of accompanying theoretical tools.[1]

This allows us greater freedom in coping with the essential tension always present in the design of a modeling language between *expressivity* and *analyzability*. The former is about how well the language can express a given phenomenon, and the latter about how well the obtained models will lend themselves to further investigations. If the model is too abstract, then none of what we learn from it will be significant; if it is too rich and concrete, then there is nothing which can be learnt beyond pure simulation. And what actually is learnt from pure simulation is sometimes questionable, if only because the model is often taught to behave as one expects in the first place.

It is the ambition of this paper to present a formalism that is both rich enough to describe interesting systems and simple enough to support formal methods. Its expressivity and simplicity are tested with examples drawn from Kohn's first molecular map of the cell cycle control [Koh99], so that we can be reasonably confident in the language representational value (it is also particularly legible so that the authors wish they had been taught molecular biology this way). The second focus of the paper is on the issue of providing automated methods for *querying* and validating models.

The current state-of-the-art in modeling is mostly based on simulation and graphical display [ABI+01, BC02, MB01, MDNM00], with some attempts towards stability and bifurcation analyses of dynamical behaviour on small systems [TT98, dJ01] described either by differential equations or by discretizations thereof. Our approach is markedly different and promotes symbolic manipulation and exploration of the model by means of computational logics which are commonplace, in hardware verification for instance. Formal methods extend the ways one can play with a given model and thus may second simulation and even replace it when quantitative information is sparse or innacurate.

This idea of introducing formal methods was mentioned as a prime motivation by early efforts at formal modeling [RSS01], and the specific prospects of using computational logics were clearly articulated in [EKL+02]. In the present paper, we give substance to this idea by proposing the use of the Computation Tree Logic CTL as a query language for biomolecular networks. We provide concrete examples of relevant biological queries expressed in CTL about the mammalian cell cycle control, and show the effectiveness of symbolic model checking techniques on some preliminary benchmarks obtained from a proof-of-concept implementation using the symbolic model-checker NuSMV [CCG+02].

[1] Process algebra specializing in the representation of protein-protein interactions [DL03a, DL03b, DK03] and membrane interactions [RPS+03, Car03] are being investigated, but our basic modeling language is somewhat simpler.

2 A core modeling language

2.1 A case for simplicity

We introduce below a simple and biologically legible formalism meant to represent molecular biology networks at the protein interaction level.

The formalism is quite expressive: one of our aims is to demonstrate this with a side-by-side comparison of standard biological subsystems described in natural language with their precise and concise rendering in the formalism. All our examples are taken from the cell cycle control reaction network after Kohn [Koh99] and we were able to complete the formalization of Kohn's first map, resulting in about 600 reactions [CD02]. A few ambiguities in Kohn's description were resolved in the process.

The formal set of reactions obtained can be complemented by different breeds of operational semantics: individual-level non-deterministic or stochastic dynamics, or population-level deterministic differential equation systems. *One advantage of having a core formalism is to stay agnostic regarding the operational semantics or dynamics* one wants to equip it with, a point which has gone largely unnoticed in the practice of biological modeling. Different operational semantics will probably support different analytic tools and be chosen depending on the application.

Another advantage of singling out a simple formalism is to stimulate the finding of better and/or richer ones. For instance, here, we choose not to represent domains (functional sub-units of proteins involved in bindings), and we are *a fortiori* not able to represent internal wirings in protein complexes. Some more involved biological narratives do take place at the domain-level and to account for these one needs domains in the language. We also choose to take complexes to be multisets of proteins, and we are consequently unable to express situations where the order in which a given complex is constructed will bear upon its interactive capabilities. Other choices made will be best commented with the examples in hand. For now, suffices it to say that in our language, the abstractions made are clear from the notation. In contrast, Kohn's formalism though extremely useful in displaying information, is not formal enough to be equipped with an operational semantics or to always allow unambiguous determination of reaction paths.

2.2 The formalism

We assume an infinite set of *protein names*, written $\mathcal{N}$, and ranged over by symbols such as A, B, ... and an *arity* function $\mathfrak{a}(.) : \mathcal{N} \to \mathbb{N}$ from protein names to integers, mapping a protein name to an integer representing its number of *sites*.

A *formal protein*, or simply a protein, is a pair $(A, \mathbf{x})$, written $A\langle\mathbf{x}\rangle$, where $A \in \mathcal{N}$ is a name and $\mathbf{x} \in \{0,1\}^{\mathfrak{a}(A)}$ is a vector of booleans representing the occupancy state of A's sites, or simply the *state* of A.

Protein-Protein Interaction. Proteins may be assembled into *protein complexes*, or simply *complexes* ranged over by C, D, ... and we write "$\cdot$" for composition. Furthermore, composition is assumed to be associative and commutative. In other words, the order of proteins inside the complexes is irrelevant. Here is an example, the following two expressions denote the same compound made of A_1, A_2, and A_3:

$$A_1\langle \mathbf{x}_1\rangle\cdot A_2\langle \mathbf{x}_2\rangle\cdot A_3\langle \mathbf{x}_3\rangle, \quad A_2\langle \mathbf{x}_2\rangle\cdot A_1\langle \mathbf{x}_1\rangle\cdot A_3\langle \mathbf{x}_3\rangle$$

Biologically, a complex is a bundle of proteins connected together mostly by low energy bonds. In the course of some interactions, members of the complex may exchange smaller molecules such as phosphate groups or be modified otherwise. This in turn induces different foldings in space and subsequent changes in the complex interaction capabilities. Now, at the level of abstraction of our formalism, all these interactions are grouped under the generic name of modification and are represented as state transformation.

Protein-DNA Interaction. Complexes can also modulate the rate of synthesis of proteins by binding to specific sites on DNA (small strings of DNA upstream of genes) having there a positive or negative effect on the synthesis of the protein(s) associated to the gene.

To express this, we use a map $\square. : \mathcal{N} \to \mathcal{P}$ (where $\mathcal{P}$ stands for the set of regulatory binding sites) associating to each $A \in \mathcal{N}$ a binding site. We'll keep with the same notation when describing a binding between a complex and such a binding site.

Solutions and Reactions. *Solutions*, ranged over by S, S', ... are multisets of proteins and complexes. *Reactions* are defined by rewriting rules which have the shape $\mathsf{S} \longrightarrow \mathsf{S}'$. Following the chemical metaphor further, we'll call complexes present in the left hand side of a given rule *reactants* and complexes present in the right hand side *products* of the rule.

We consider five kinds of reactions:

(MODIFICATION)
$$C\langle \mathbf{x}\rangle, D\langle \mathbf{y}\rangle \longrightarrow C\langle \mathbf{x}'\rangle, D\langle \mathbf{y}'\rangle$$
(COMPLEXATION)
$$C\langle \mathbf{x}\rangle, D\langle \mathbf{y}\rangle \longrightarrow C\langle \mathbf{x}'\rangle\cdot D\langle \mathbf{y}'\rangle$$
(DECOMPLEXATION)
$$C\langle \mathbf{x}'\rangle\cdot D\langle \mathbf{y}'\rangle \longrightarrow C\langle \mathbf{x}\rangle, D\langle \mathbf{y}\rangle$$
(SYNTHESIS)
$$F\langle \mathbf{x}\rangle\cdot\square\mathrm{A} \longrightarrow F\langle \mathbf{x}\rangle\cdot\square\mathrm{A}, A\langle \mathbf{y}\rangle$$
(DEGRADATION)
$$F\langle \mathbf{x}\rangle, C\langle \mathbf{y}\rangle \longrightarrow F\langle \mathbf{x}\rangle$$

Comments on the reactions. Modification and complexation were already commented on, and decomplexation is just the reaction inverse to complexation.

The synthesis reaction expresses that F is a transcription factor which, when bound to a regulatory binding site $\square$A, activates the synthesis of protein A.

In the synthesis and degradation reactions, the complex F, commonly known as a transcription or degradation *factor*, can be absent (or empty for the mathematically minded).

The first three reaction types are linear in that they preserve the number of components. The latter two are not, and even with this simple formal apparatus we see that they offer a mechanism for the cell to revise its own programming by renewing its stock of current "instructions".

Enzymatic notation. Many biochemical reactions require catalysis, that is, the presence of a type of protein called an *enzyme*, which is not modified by the reaction but enables it by lowering the free energy barrier and thus modifying the kinetics. To express conveniently these cases, we will use the following simplified "enzymatic" notations:

(MODIFICATION)
$$C\langle\mathbf{x}\rangle[D\langle\mathbf{y}\rangle \longrightarrow D\langle\mathbf{y}'\rangle]$$
(SYNTHESIS)
$$F\langle\mathbf{x}\rangle\cdot\Box\mathrm{A}[\longrightarrow A\langle\mathbf{y}\rangle]$$
(DEGRADATION)
$$F\langle\mathbf{x}\rangle[A\langle\mathbf{y}\rangle \longrightarrow]$$

Additionally, for synthesis, one may indicate the qualitative influence of F with a $+$ or $-$ as in:
$$F\langle\mathbf{x}\rangle\cdot\Box\mathrm{A}[\rightarrow A\langle\mathbf{y}\rangle]^{+}.$$

Dynamics. Note that for the $+$ and $-$ signs to be endowed with other than a purely descriptive meaning, a notion of reaction rate (discrete or continuous) is required, together with its interpretation in the operational semantics.

In general, any set of reactions defined following the rule schema above will generate a non-deterministic transition system on solutions in the obvious way, namely by repeatingly applying rules in any order. If fed with enough kinetic information, it is even possible to endow the same set of reactions with a structure of probabilistic transition system. It is equally possible to derive a classical differential system and this actually is the bulk of biological modeling (see for instance [SEJGM02]). We believe, however, that interesting analyses can already be led at the purely non-deterministic level. But one first needs to verify that a pretty good approximation of molecular biology fits within the formal picture.

3 Representing cell cycle regulation

The cell cycle is a central mechanism in the cell physiology which regulates cell division. Control over this fundamental biological activity is exerted by a family of interacting proteins known as the cyclins, or CYCs and the cyclin dependent kinases, or CDKs.

The cell cycle in eukaryotes is divided into four phases. Between two cell divisions, the cell is in a gap phase called G_1, which may contain a quiescent phase G_0. The cell can stay in phase G_0 for very long periods of time, without further division, in which case this phase can be construed as a steady state The synthesis phase S starts with the replication of the DNA. A second gap phase G_2 precedes the mitotic phase M during which the cell divides.

Each phase is characterized by the activity of two major types of proteins: cyclins and cyclin-dependent kinases (CDK). Experiments show a correlation between the phase and concentrations of cyclins of specific types. 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.

At any particular phase a given CDK·CYC complex will be dominant and busy verifying that some conditions are met and, if they are, will activate the global shift of the cell to the next phase.

Apart from being arguably the most important biological process, the cell cycle is extremely well documented at the molecular level. It is a significant challenge to understand how the higher-level functions emerge from this vast network of reactions. With the language introduced above, we are going to provide a few formal glimpses of our current knowledge of this mechanism.

To facilitate reading, we'll use an equivalent notation for complexes and write $A{\cdot}B\langle\mathbf{x};\mathbf{y}\rangle$ instead of $A\langle\mathbf{x}\rangle{\cdot}B\langle\mathbf{y}\rangle$. Another convention we take for the examples is that when a reaction occurs independently of the values of some internal states of the partners, we replace these with boolean variables written x, y, z.

3.1 Cyclin·CDK bindings

Let's first examine the formation of CDK·CYC complexes. CYCD can pair with CDK4:[2]

$$\mathrm{CDK4}\langle x,y\rangle, \mathrm{CYCD}\langle z\rangle \quad \longrightarrow \quad \mathrm{CDK4{\cdot}CYCD}\langle x,y;z\rangle$$

CYCA and CYCE compete in binding with CDK2:

$$\begin{array}{rcl} \mathrm{CDK2}\langle x,y\rangle, \mathrm{CYCA}\langle\rangle & \longrightarrow & \mathrm{CDK2{\cdot}CYCA}\langle x,y;\rangle \\ \mathrm{CDK2}\langle x,y\rangle, \mathrm{CYCE}\langle z\rangle & \longrightarrow & \mathrm{CDK2{\cdot}CYCE}\langle x,y;z\rangle \end{array}$$

Likewise, CYCA binds CDK1 in competition with CYCB. But here there is a slight twist, namely that CDK1's third phosphorylation site is required for the formation of a stable complex with CYCA:

$$\begin{array}{rcl} \mathrm{CDK1}\langle x,y,1\rangle, \mathrm{CYCA}\langle\rangle & \longrightarrow & \mathrm{CDK1{\cdot}CYCA}\langle x,y,1;\rangle \\ \mathrm{CDK1}\langle x,y,z\rangle, \mathrm{CYCB}\langle\rangle & \longrightarrow & \mathrm{CDK1{\cdot}CYCB}\langle x,y,z;\rangle \end{array}$$

3.2 Cyclin·CDK inhibition

Then specific Cyclin·CDK inhibitors step in. P16 inhibits by binding CDK4/6 in competition with CYCD:

$$\mathrm{CDK4}\langle x,y\rangle, \mathrm{P16}\langle\rangle \quad \longrightarrow \quad \mathrm{CDK4{\cdot}P16}\langle x,y;\rangle$$

whereas P21 binds the complex CDK4·CYCD and prevents its activity:

$$\mathrm{P21}\langle\rangle, \mathrm{CDK4{\cdot}CYCD}\langle x,y;z\rangle \quad \longrightarrow \quad \mathrm{P21{\cdot}CDK4{\cdot}CYCD}\langle ;x,y;z\rangle$$

What do we mean formally by saying that P16 and P21 are inhibitors? We mean that none of the reactions involve CDK4·P16 or CYCD·CDK4·P21; therefore, once P16 (resp. P21) reacts with CDK4 (resp. CYCD·CDK4), CDK4 (resp. CYCD·CDK4) becomes unavailable for any further reaction.

These are end-products. Take note that the formalism doesn't allow any distinction between not knowing that a reaction takes place and knowing that it does not. One could remedy this easily by adding 'non-reactions' such as:

$$\mathrm{CDK4{\cdot}P16}\langle x,y;\rangle, \mathrm{CYCD}\langle z\rangle \quad \not\longrightarrow \quad \mathrm{CDK4{\cdot}P16{\cdot}CYCD}\langle x,y;z\rangle.$$

[2]... and CDK6 but these two are indistinguishable as for their interactive properties so we don't make any further mention of CDK6.

3.3 Introducing MPF: CDK1·CYCB

The complex CDK1·CYCB, also known as MPF, is the one in charge of the actual division of the cell or mitosis. It is born inactive and is activated by other phosphatases (a protein taking a phosphate group) and kinases. All CDKs (and in particular CDK1) are activated by a phosphorylation of some specific amino-acid Thr160 (or Thr161), carried out by CYCH·CDK7 (also known as CAK).

$$\begin{array}{l} \mathrm{CDK7{\cdot}CYCH}\langle 0; \rangle[\mathrm{CDK1}\langle x, y, 0\rangle \quad \longrightarrow \quad \mathrm{CDK1}\langle x, y, 1\rangle] \\ \mathrm{CDK7{\cdot}CYCH}\langle 0; \rangle[\mathrm{CDK1{\cdot}CYCB}\langle x, y, 0; \rangle \quad \longrightarrow \quad \mathrm{CDK1{\cdot}CYCB}\langle x, y, 1; \rangle] \end{array}$$

CDK1 can be inhibited by a phosphorylation on amino-acids Thr14 and/or Tyr15 performed by WEE1 or MYT1. This phosphorylation is possible only when CDK1 is already bound to CYCB (or CYCA).

$$\begin{array}{l} \mathrm{WEE1}\langle 0\rangle[\mathrm{CDK1{\cdot}CYCB}\langle x, 0, y; \rangle \quad \longrightarrow \quad \mathrm{CDK1{\cdot}CYCB}\langle x, 1, y; \rangle] \\ \mathrm{MYT1}\langle\rangle[\mathrm{CDK1{\cdot}CYCB}\langle 0, x, y; \rangle \quad \longrightarrow \quad \mathrm{CDK1{\cdot}CYCB}\langle 1, x, y; \rangle] \end{array}$$

Then CDK1 inhibits its inhibitor:

$$\mathrm{CDK1.CYCB}\langle 0, 0, 1; \rangle[\mathrm{WEE1}\langle 0\rangle \quad \longrightarrow \quad \mathrm{WEE1}\langle 1\rangle]$$

A positive feedback loop involves CDC25C and CDK1·CYCB. CDC25C is activated in its N-terminal domain as follows:

$$\mathrm{CDK1{\cdot}CYCB}\langle 0, 0, 1; \rangle[\mathrm{CDC25C}\langle 0, x\rangle \quad \longrightarrow \quad \mathrm{CDC25C}\langle 1, x\rangle]$$

Once activated CDC25C dephosphorylates Thr14/Tyr15 of CDK1, which activates MPF, and the positive loop is closed (provided MPF third's site was set at 1 in a preceding interaction with CAK).

$$\begin{array}{l} \mathrm{CDC25C}\langle 1, x\rangle[\mathrm{CDK1{\cdot}CYCB}\langle x, 1, y; \rangle \quad \longrightarrow \quad \mathrm{CDK1{\cdot}CYCB}\langle x, 0, y; \rangle] \\ \mathrm{CDC25C}\langle 1, x\rangle[\mathrm{CDK1{\cdot}CYCB}\langle 1, x, y; \rangle \quad \longrightarrow \quad \mathrm{CDK1{\cdot}CYCB}\langle 0, x, y; \rangle] \end{array}$$

3.4 CYC·CDK vs. PRB

When a CYC·CDK complex becomes active it impacts indirectly on the synthesis of other proteins. CYCD·CDK4 begins by phosphorylating PRB at its first site:

$$\mathrm{CDK4{\cdot}CYCD}\langle 0, 1; x\rangle[\mathrm{PRB}\langle 0, x\rangle \quad \longrightarrow \quad \mathrm{PRB}\langle 1, x\rangle]$$

then CYCE·CDK2 can act on semi-phosphorylated PRB generating fully phosphorylated PRB:

$$\mathrm{CDK2{\cdot}CYCE}\langle 0, 1; x\rangle[\mathrm{PRB}\langle 1, 0\rangle \quad \longrightarrow \quad \mathrm{PRB}\langle 1, 1\rangle]$$

that cannot bind any longer to E2F1·DP.

$$\mathrm{PRB}\langle z, 0\rangle, \mathrm{E2F1{\cdot}DP1}\langle x; y\rangle \quad \longrightarrow \quad \mathrm{PRB{\cdot}E2F1{\cdot}DP1}\langle z, 0; x; y\rangle$$

PRB has a different behaviour in all its three possible states: when in state $\langle 0, 0\rangle$ it binds and inhibits the transcription of some proteins downstream; and when in the intermediate state $\langle 1, 0\rangle$ it has a weaker inhibiting effect; when completely phosphorylated, *i.e.* in state $\langle 1, 1\rangle$, it doesn't bind to E2F1·DP1 at all, and faster synthesis of different products including, of course, cyclins themselves happens.

We see that the formalism provides a straightforward representation of these highly constrained sequences of reactions. Talking about transcription, we finish our tour of protein interactions with an example, involving PRB again, of how protein regulate protein synthesis.

3.5 Transcriptional regulation

This example shows a simple and typical narrrative of protein synthesis. First protein JUN binds protein C-FOS:

$$\mathrm{JUN}\langle\rangle, \mathrm{C\text{-}FOS}\langle\rangle \quad \longrightarrow \quad \mathrm{JUN{\cdot}C\text{-}FOS}\langle;\rangle$$

Then JUN·CFOS stimulates the synthesis of ERCC1 (a protein involved in DNA repair):

$$\begin{array}{rcl} \mathrm{JUN{\cdot}CFOS}\langle;\rangle, \Box\mathrm{ERCC1} & \longrightarrow & \mathrm{JUN{\cdot}CFOS}\langle;\rangle{\cdot}\Box\mathrm{ERCC1} \\ \mathrm{JUN{\cdot}CFOS}\langle;\rangle{\cdot}\Box\mathrm{ERCC1}[& \longrightarrow & \mathrm{ERCC1}\langle\rangle]^{+} \end{array}$$

Unphosphorylated PRB binds JUN.

$$\mathrm{PRB}\langle 0, 0\rangle, \mathrm{JUN}\langle\rangle \quad \longrightarrow \quad \mathrm{PRB{\cdot}JUN}\langle 0, 0;\rangle$$

This enhances the binding of the JUN family members to C-FOS (but we cannot say this in the absence of a quantitative dynamics) and stimulates further — hence the ++ below — transcriptional activation by the JUN·C-FOS complex.

We observe that there are many ways leading to the construction of the tri-complex PRB·JUN·C-FOS formation and ultimately to its binding to the ERCC1 binding site:

$$\begin{array}{l} \mathrm{PRB}\langle 0, 0\rangle, \mathrm{JUN{\cdot}C\text{-}FOS}\langle\rangle \quad \longrightarrow \quad \mathrm{PRB{\cdot}JUN{\cdot}C\text{-}FOS}\langle 0, 0; ;\rangle \\ \mathrm{PRB{\cdot}JUN}\langle x, y;\rangle, \mathrm{C\text{-}FOS}\langle\rangle \quad \longrightarrow \quad \mathrm{PRB{\cdot}JUN{\cdot}C\text{-}FOS}\langle x, y; ;\rangle \\ \mathrm{PRB{\cdot}JUN{\cdot}C\text{-}FOS}\langle x, y; ;\rangle, \Box\mathrm{ERCC1} \quad \longrightarrow \quad \mathrm{PRB{\cdot}JUN{\cdot}C\text{-}FOS}\langle x, y; ;\rangle{\cdot}\Box\mathrm{ERCC1} \\ \mathrm{PRB{\cdot}JUN{\cdot}C\text{-}FOS}\langle x, y; ;\rangle{\cdot}\Box\mathrm{ERCC1}[\quad \longrightarrow \quad \mathrm{ERCC1}\langle\rangle]^{++} \end{array}$$

Again the language expresses this in a clear way. We also see that to do justice to transcription, it seems one needs a quantitative semantics, or at least quantitative enough to express a few different rates of synthesis.

4 Temporal logic as query language

The use of formal languages to represent complex molecular networks is motivated not only by an expected gain in descriptive and explanatory power, but also by the promise of biologically relevant analyses of the dynamical behaviour of these networks, both quantitative and qualitative. The latter type is especially important at this stage in the development of computational biology, for at least two reasons:

- regulatory, signaling and metabolic networks are very complex mechanisms which are far from being understood on a global scale ; qualitative analyses hold the promise of providing logical/computational interpretation of the role of biologically relevant subparts of these networks – abstracting away from their detailed dynamics – which in turn may help reverse-engineer through a modular approach;
- data on both the existence and the dynamics of molecular interactions is rare and unreliable; dynamical models which are too sensitive to the exact network structure or some parameter values (*e.g.* continuous ODE-based representations) may not be the best suited to analyze and predict behaviour in such settings.

In this part, we explore the use of automated methods for querying qualitative models of biomolecular networks. First, we give a brief introduction to the computation tree logic CTL which we propose to use to formulate queries about the dynamic properties of the system of interest. Next, we show how the set of reactions presented in the previous sections can be turned into an appropriate structure for CTL queries, namely concurrent transition systems. Finally, we discuss several examples of biologically relevant queries together with their formalization in CTL.

4.1 Computation Tree Logic

The Computation Tree Logic CTL is a logic for describing properties of computation trees and non-deterministic transition systems [CGP99]. 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* logic [Eme90] 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 ϕ can never happen, is expressed by the CTL formula $AG\neg\phi$, i.e. on all paths ϕ is always false. A "liveness" property, specifying that something good ψ will eventually happen, is expressed by the formula $AF\psi$. Note that by duality we have $EF\phi = \neg AG\neg\phi$ and $EG\phi = \neg AF\neg\phi$ for any formula ϕ.

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. This is summarized in the following grammar (ordinary boolean connectives are not shown):

$$\begin{array}{rcl} \phi & := & \alpha \in AP \mid E\psi \mid A\psi \\ \psi & := & \phi \mid X\psi \mid F\psi \mid G\psi \mid \psi U \psi \end{array}$$

Arbitrary state and path formulas are permitted in CTL* but not in CTL which is defined as the syntactic fragment of CTL* where temporal operators G, F, X and U must be immediately prefixed by a path quantifier A or E. For instance, $A(FG\phi)$ and $E(F\phi \wedge G\psi)$ are CTL* formula which are not in CTL. Another well-known fragment is LTL where only formulas of the form $A\phi$ where ϕ contains no path quantifier are allowed. For example, $AG(EF\phi)$ is not an LTL formula. Since the biological queries of interest, described in section 4.4, can be expressed in CTL but not all in LTL, our proposal is to use CTL as a query language for biochemical systems and we shall focus on this fragment in this paper.

The semantics of CTL and transition systems are 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 total relation (i.e. for any state $s \in S$ there exists a state $s' \in S$ such that $(s, s') \in R$), and $L : S \to 2^{AP}$ 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 π^i the suffix of π starting at s_i. Now, given a Kripke structure K, 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 $\pi \models \phi$, is as follows (clauses for ordinary boolean connectives are omitted):

- $s \models \alpha$ 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 $0 \leq j < k$.

Following [Eme90], assuming a Kripke structure K, it is convenient to identify a CTL formula ϕ to the set of states which satisfy it, i.e. $\{s \in S | s \models \phi\}$. In particular, we will write $init \in \phi$ (resp. $Init \subseteq \phi$) to express that a CTL formula ϕ holds in an initial state *init* (resp. in all initial states in the set *Init*).

4.2 Concurrent transition systems

Concurrent transition systems have been introduced in [Sha93] as a formal tool to reason about concurrent programs. They offer a simple language of reaction rules for modeling concurrent programs and specifying Kripke structures. We will thus use them to formalize the Kripke semantics of our core modeling language, and give a precise meaning to the idea of querying in CTL biochemical models.

A *concurrent transition system* (CTS for short) is a triple $(\vec{x}, I, R)$ where $\vec{x}$ is a tuple of state variables, I is an initial state (defined by the initial values of the state variables), and R is a set of condition-action rules. The rules have the following syntax:

$$\textit{condition } c \quad \textit{action } x'_1 = e_1, ..., x'_n = e_n$$

where c is a logical formula over state variables which denotes the condition under which the rule can be applied, and the primed version of the variables denotes the new values, given by expression e_i, 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 CTS defines a Kripke structure, where the set of states is the set of all tuples of values for the state variables, and the transition relation is the union[3] (i.e. disjunction) of the relations between the states of all instances of the condition-action rules.

Now the core modeling language described in the first part of this paper, can be turned in a CTS by a simple transformation in order to comply with the state variables format of CTSs. Indeed state patterns such as CDK4$\langle 0, x\rangle$ need be fully instantiated with (ground) terms such as CDK4$\langle 0, 0\rangle$ in order to be represented by state variables. The different internal states of a given protein are thus viewed in the associated CTS as different macromolecules and each is associated with a unique state variable.

State variables of the CTS can be chosen to take values in different domains corresponding to different levels of abstraction. In the refined notion of state of our core modeling language, the values of state variables are *integers* representing the multiplicity of each elementary component, protein or complexe in the cell. In more abstract models, we can choose as well to valuate state variables with *real* numbers representing concentrations, or with *boolean* values representing simply the presence or absence of the compounds in the cell. Whatever choice is made, it is worth noticing that the temporal evolution of the system is modeled in this setting by the transition steps[4] and the different transition paths model the non-deterministic behaviour of the system.

4.3 Example of the mammalian cell-cycle control

Now that we have developed enough of the logical aspect of affairs, we return to our benchmark example of the mammalian cell-cycle control.

As written above, the state of experimentally-derived knowledge on the dynamics of the mammalian cell-cycle (*e.g.* reaction rates), shows that it is somewhat premature to attempt reasoning with *quantities* of proteins in mammalian cells. Instead, it is more appropriate and interesting in this example to reason *qualitatively* on all possible behaviours of the system. Sets of states could be represented by partial information on the actual numerical values of state variables, using for instance intervals or constraints between variables. We shall choose however a simpler route, that of the boolean abstraction of the model, where state variables are *boolean* variables representing the presence/absence of proteins. In this case, the set of states is finite and its cardinality is 2^n where n is the number of state variables. In our example, we have $n = 532$ corresponding to the different forms of the 165 basic proteins and genes involved in the model.

[3]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.

[4]Hybrid dynamics combining discrete transition time with continuous time, can also be handled in this setting by turning a hybrid system into a concurrent transition system using Euler's method for discretizing differential equations [CF03b].

We consider accordingly, the concurrent transition system over boolean state variables defined by the following rule schemas:

1. Modification: $A, B \rightarrow C, B$,
   A is modified under the action of a catalyst B, and transformed into C, a phosphorylated form of A for example. Obviously, other state changes can be encoded in a similar manner.

2. Complexation: $A, B \rightarrow A.B$,
   A and B bind together to form a complex $A.B$;

3. Synthesis : $A \rightarrow A, B$,
   B is synthesized by the activated transcription factor A;

4. Degradation : $A, B \rightarrow A$,
   B is degraded by the degradation factor A.

In the complexation rule schema, $A.B$ stands for a *propositional variable* denoting the complex which results from the binding of the molecules denoted by A and B. An instance of this schema is the rule:

$$\text{CDK7}, \text{CYCH} \qquad \longrightarrow \qquad \text{CDK7}\cdot\text{CYCH}$$

where CYCH, CDK7 and CYCH.CDK7 are three boolean variables representing respectively, CYCH, CDK7 and the dimer CYCH.CDK7 each in a given internal state.

Likewise, one can introduce a variable named CDK1(pThr14).CYCB, to represent a phosphorylated form of the dimer CDK1.CYCB at site Thr14 of CDK1. The phosphorylation of this dimer by MYT1 is modeled by the modification rule:

$$\text{CDK1}\cdot\text{CYCB}, \text{MYT1} \qquad \longrightarrow \qquad \text{CDK1(pThr14)}\cdot\text{CYCB}\cdot\text{MYT1}$$

These boolean rules denote condition action rules with the following conventions. 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 or false in the action, with the convention that the variables which denote components with varying quantities (not promoters) and which appear in the left hand side and not in the right hand side of the schema, may take *arbitrary values*.

This is necessary to define a correct boolean abstraction that is an *over-approximation* of the set of all possible behaviours of the system, ignoring numerical quantities and reaction rates. For instance, the rule of complexation is a shorthand for the four condition-action rules:

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

The condition-action rules make explicit the possible disappearance of molecules A or B by complexation, and their combination guarantees the correctness of the boolean abstraction w.r.t. all possible behaviours of the system. This method is implemented in the system BIOCHAM currently under development [CF03a].

4.4 Biological queries

The biological queries one can consider about a boolean model of the cell cycle control are of different kinds. Below we enumerate a list of biological queries of interest and discuss their expression in CTL. Most of these queries are relative to an initial state or a set of initial states described by a logical formula.

About reachability :

1 Given an initial state *init*, is there a series of reactions that will produce some compound P?

This query translates into the formula $init \in \mathbf{EF}(P)$ where P is the boolean variable representing the product P.

2 Which are the states from which a set of products $P_1, \ldots, P_n$ can be produced simultaneously?

The query translates into the formula $\mathbf{EF}(P_1 \wedge \ldots \wedge P_n)$. Indeed, CTL formulas can be identified to the set of states which satisfy them, and the model checking tools described in the next section actually provide facilities for showing the set of states satisfied by a CTL formula.

About pathways :

3 Given an initial state *init*, can the cell reach a state s while passing by another state s_2?

$init \in \mathbf{EF}(s_2 \wedge \mathbf{EF}s)$.

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

$init \in \neg\mathbf{E}((\neg s_2)\ \mathbf{U}\ s)$. We express here the contrapositive of the query, that is there does not exist a path reaching s without passing by s_2.

5 Is it possible to produce P without creating nor using some Q?

$init \in \mathbf{E}(\neg Q\ \mathbf{U}\ P)$.

6 More generally, one can ask whether a state s is reachable under a certain constraint c from a set of initial states *Init*:

$Init \subseteq \mathbf{E}(c\ \mathbf{U}\ s)$.

About stability properties :

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

$s \in \mathbf{AG}(s)$ or $s \subseteq \mathbf{AG}(s)$ where s is a set of states described by a logical formula noted also s by abuse of notation.

A stable state in the strong sense is indeed a state in which the system stays indefinitely with no possibility of escaping; a steady state, in which the system might stay indefinitely but might also not, can be modeled by $s \subseteq \mathbf{EG}(s)$,

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

$init \in \mathbf{EF}(\mathbf{AG}(s))$. It is worth noticing that this query is not expressible in LTL.

9 Must the system reach a given stable state s from the initial state *init*?
$init \in \mathbf{AF}(\mathbf{AG}(s))$.

10 What are the stable states?
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. This is an interesting open problem that has been recently investigated in other contexts for the design of CTL query languages [Cha00, HS02, GCD03].

11 Can the system exhibit a cyclic behaviour w.r.t. the presence of a product P?
$init \in \mathbf{EG}((P \Rightarrow \mathbf{EF}\ \neg P) \wedge (\neg P \Rightarrow \mathbf{EF}\ P))$
This formula is not expressible in LTL. It 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.

About durations :

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

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

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

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 logic, it is 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 5.3 provides an automatic method for evaluating such queries. On the other hand, symbolic model checking techniques have also been extended to incorporate specifically duration data [LMS02].

About the correctness of the model :

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

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 biological experiments in order to validate or invalidate that property of the model. In biology (as in any natural science as opposed to computer science), the standard loop between modeling and model-validation becomes a threefold loop between modeling, querying the model and doing biological experiments.

It is worth noticing also that the boolean abstraction of the model introduces inaccuracies which would correspond to inaccurate quantities or inaccurate reaction rates in a quantitative model. The explored combinatorics of the underlying quantitative models may thus correspond to non standard situations which are biologically relevant. These inaccuracies can nevertheless be corrected by adding boolean conditions in the rules.

5 Automated query evaluation

5.1 Symbolic 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 ϕ, i.e. the set $\{s \in S \mid s \models \phi\}$. 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 subformulas of ϕ which are true in that state. First, the states are labeled with the atomic propositions of ϕ which are true in those states. Labeling with more complex formulas is done iteratively, following the syntax of the subformulas of ϕ. Formulas of the form $\neg\phi$ label those states which are not labeled by ϕ. Formulas of the form $\phi \vee \psi$ are added to the labels of the states labeled by ϕ or ψ. Formulas $EX\phi$ are added to the labels of the immediate predecessor states of the states labeled by ϕ. Formulas $E(\phi U \psi)$ are added to the hereditary predecessor states of ψ while they satisfy ϕ. Formulas $EG\phi$ involve the computation of the strongly connected components of the subgraph of transitions restricted to the states satisfying ϕ. 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 of transitions [CGP99].

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 [CGP99]. In our experiments reported below, we used the state-of-the-art symbolic model checker NuSMV [CCG$^+$02].

5.2 Computational results

Table 1 presents some performance figures concerning the evaluation of CTL queries in the boolean abstraction of the mammalian cell cycle control model desbribed in section 4.3.

The boolean model used in these experiments comprises 732 reaction rules over 165 proteins and genes, and 532 variables taking into account the different compounds of the system. The queries concern the pathways leading to the mitosis of the cell described in section 3.3 on the complex MPF, and on the cyclic presence of CYCA. The initial state corresponds to the gap phase G2 prior to the mitosis of the cell. The two first columns indicate the query and its type. The third column indicates the CPU time taken by NuSMV to answer the query. The fourth column indicates the CPU time taken for explaining the answer, that is for showing a pathway or a witness (the negation of the query is asked for no answers). The CPU times are given in seconds and have been measured on a processor Intel Pentium 3 at 600 Mhz under Linux.

Query type number	Query	NuSMV answer time in seconds	NuSMV show time in seconds
	compiling	47.5	-
2	**EF** SL1(p)	29	124
2	**EF** CYCE	2	22
2	**EF** CYCD	1.9	11.5
2	**EF** PCNA.CYCD	1.7	48.7
4	$\neg$**E**($\neg$ CDC25C(Nterm) **U** CDK1-CYCB(Thr161))	2.2	49.22
11	**EG** (CYCA $\Rightarrow$ **EF**$\neg$ CYCA $\wedge\neg$ CYCA $\Rightarrow$ **EF** CYCA)	31.8	-

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

When compared to a simpler Prolog based implementation of model checking, the timings obtained with the NuSMV model checker show the efficiency of the symbolic representation of states by Binary Decision Diagrams BDDs [CF03b]. These timings are however somewhat slower than what is usual in the program verification community for a model of only a few hundreds of rules and variables. One fundamental reason is the overall structure of transition graphs modeling biochemical networks. Such transition systems are indeed highly non-deterministic due to the “soup” aspect of molecular interactions and thus differ significantly in this respect from the transition graphs obtained from circuits or programs. It would be worth investigating further whether specific optimizations of model-checking algorithms are possible in this context, especially concerning the ordering of variables for the internal BDD representation of states [WCZK01].

5.3 Quantitative models and constraint-based model checking

As mentioned in previous sections, CTL queries can in principle be applied to quantitative models of molecular interactions. In particular, we have seen that the multiset rewriting core modeling of the cell-cyle control given in the first part of this paper, can be translated in a concurrent transition system over integer variables denoting the multiplicity of molecules in the cell.

Concurrent transition systems with variables ranging over unbounded or continuous numerical domains define however Kripke structures with an infinite set of states which cannot be handled by the symbolic model checking algorithm. On the other hand, constraint-based model checking applies to infinite state systems. In this approach, a constrained state is a finite representation using constraints, of a finite or infinite set of states. In the scheme of Delzanno and Podelski [DP99], 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 constraint logic programs.

Constraint-based model checking over integers (using constraint logic programs over finite domains technology) could thus be used to evaluate quantitative queries about the cell-cycle control. This has not been done in the mammalian cell-cycle example by lack of numerical data, but has been shown in [CF03b] on a simple quantitative model of gene expression using constraint based model checking over reals.

6 Conclusion

We have designed a simple language that has proven to be rich enough to describe biomolecular networks at the level of protein interactions. In particular, the combinatorics of complexation, activation, synthesis and degradation are easily expressed, as we have shown using Kohn's description of the mammalian cell-cycle control as our expressiveness benchmark.

We have also shown how symbolic model checking techniques could be applied to the querying and validation of boolean abstractions of networks of molecular interactions. First, we have shown that the temporal logic CTL is expressive enough to formalize a wide variety of biological queries of interest about a molecular network: ranging from pure reachability queries on the possibility of synthesizing a particular protein under pathway constraints, to the existence of checkpoints, and to the analysis of steady states and cyclic behaviours. Second, CTL querying applies to highly non-deterministic systems, by investigating all possible behaviours of the system, in situations where simulation can be ill-defined or unfeasible. Third, symbolic methods make it possible to group large sets of states into small state expressions which provide formal proofs of reachability, pathway, checkpoint and stability properties.

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.

Acknowledgments

This work has been done in the Cooperative Research Initiative ARC CPBIO[5]. We are grateful to our colleagues of the ARC CPBIO, especially to Magali Roux-Rouquié, Julien Renner and Grégory Sautejeau from Institut Pasteur, for interesting discussions on Systems Biology and relevant bits of Molecular Biology, Sylvain Soliman from the Contraintes group at INRIA Roqcuencourt

[5] `http://contraintes.inria.fr/cpbio`

and Alexander Bockmayr, Arnaud Courtois and Damien Eveillard from the ModBio group at LORIA Nancy, for fruitful discussions on quantitative models. We also acknowledge the support from Alessandro Cimatti and the NuSMV team from the IRST in Trento.

References

[ABI+01] 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.

[BC02] 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.

[Car03] L. Cardelli. Brane calculi. To Appear in the ENTCS Proceedings of Bio-Concur, 2003.

[CCG+02] 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.

[CD02] M. Chiaverini and V. Danos. A formalization of Kohn's molecular map. Available at www.pps.jussieu.fr/~bioconcu/kohn_map.html, July 2002.

[CF03a] N. Chabrier and F. Fages. The biochemical abstract machine BIOCHAM. In C. Christophe, H.P. Lenhof, and M. F. Sagot, editors, *Proceedings of the European Conference on Computational Biology, ECCB'03*, pages 597–599, Paris, France. System available at `http://contraintes.inria.fr/BIOCHAM`, September 2003.

[CF03b] N. Chabrier and F. Fages. Symbolic model checking of biochemical networks. In C. Priami, editor, *Proceedings of the International Workshop on Computational Methods in Systems Biology, CMSB'03*, Lecture Notes in Computer Science, pages 149–162, Rovereto, Italy, February 2003. Springer-Verlag.

[CGP99] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[Cha00] W. Chan. Temporal logic queries. In *Proceedings of the 12th International Conference on Computer Aided Verification CAV'00*, number 1855 in Lecture Notes in Computer Science, pages 450–463, Chicago, USA, 2000. Springer-Verlag.

[dJ01] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):69–105, 2001.

[DK03] V. Danos and J. Krivine. Formal molecular biology done in CCS. In *Proceedings of BIO-CONCUR'03, Marseille, France*, Electronic Notes in Theoretical Computer Science. Elsevier, 2003. To appear.

[DL03a] V. Danos and C. Laneve. Core formal molecular biology. In *Proceedings of the 12th European Symposium on Programming, ESOP'03*, volume 2618 of *LNCS*, pages 302–318. Springer-Verlag, April 2003.

[DL03b] V. Danos and C. Laneve. Graphs for formal molecular biology. In *Proceedings of the First International Workshop on Computational Methods in Systems Biology, CMSB'03*, volume 2602 of *LNCS*, pages 34–46. Springer-Verlag, February 2003.

[DP99] 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.

[EKL+02] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and K. Sonmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.

[Eme90] E.A. Emerson. *Temporal and Modal Logic*, pages 995–1072. J. van Leeuwen Ed., North-Holland Pub. Co./MIT Press, 1990.

[GCD03] A. Gurfinkel, M. Chechik, and B. Devereux. Temporal logic query checking: A tool for model exploration. *Special FSE'02 Issue of IEEE Transactions on Software Engineering*, 2003.

[GT01] 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.

[HS02] S. Hornus and Ph. Schnoebelen. On solving temporal logic queries. In *Proceedings of the 9th International Conference on Algebraic Methodology and Software Technology AMAST'02*, number 2422 in Lecture Notes in Computer Science, pages 163–177, Saint Gilles les Bains, Reunion Island, France, 2002. Springer-Verlag.

[HT98] R. Hofestädt and S. Thelen. Quantitative modeling of biochemical networks. In *In Silico Biology*, volume 1, pages 39–53. 1998.

[Koh99] 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.

[LMS02] F. Laroussinie, N. Marley, and Ph. Schnoebelen. On model checking durational Kripke structures. In *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures FOSSACS'02*, number 2303 in Lecture Notes in Computer Science, pages 264–279, Grenoble, France, 2002. Springer-Verlag.

[MB01] R. Maimon and S. Browning. Diagrammatic notation and computational structure of gene networks. In *Proceedings of the 2nd International Conference on Systems Biology*, 2001.

[MDNM00] 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.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes I and II. *Information and Computation*, 100:1–41, 42–78, 1992.

[RPS+03] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. Bioambients: An abstraction for biological compartments. *Theoretical Computer Science*, 2003. To Appear.

[RSS01] 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.

[SEJGM02] B. Schoeberl, C. Eichler-Jonsson, ED. Gilles, and G. Müller. Computational modeling of the dynamics of the map kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology*, 20:370–375, 2002.

[Sha93] U. A. Shankar. An introduction to assertionnal reasoning for concurrent systems. *ACM Computing Surveys*, 3(25):225–262, 1993.

[TT98] D. Thieffry and R. Thomas. Qualitative analysis of gene networks. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 3, pages 77–88, Singapore, 1998. World Scientific Press.

[WCZK01] D. Wang, E. Clarke, Y. Zhu, and J. Kukula. Using cutwidth to improve symbolic simulation and boolean satisfiability. In *In Proceeding of HLDVT'2001*, 2001.