Timed Concurrent Constraint Programming in Systems Biology

Alejandro Arbeláez, Julian Gutiérrez and Jorge A. Pérez

AVISPA Research Group*, Department of Science and Engineering of Computing

Pontificia Universidad Javeriana, Cali, Colombia

{aarbelaez,jg,japerez}@cic.puj.edu.co

3rd November 2006

Abstract

Systems biology aims at getting a higher-level understanding of living matter, building on the available data at the molecular level. In this field, theories and methods from computer science have proven very useful, mainly for system modeling and simulation. Here we argue that languages based on *timed* concurrent constraint programming (timed ccp) —a well-established model for concurrency based on the idea of partial information— have a place in systems biology. We summarize some works in which our group has tried to assess the possibilities/limitations of one such formalisms in this domain. Our base language is `ntcc`, a non-deterministic, timed ccp process calculus that provides a *unified framework* for modeling, simulating and *verifying* several kinds of biological systems. We discuss how the interplay of the operational and logic perspectives that `ntcc` integrates greatly favors biological systems analysis.

1 Introduction

Recent years have seen an extraordinary progress in the field of molecular biology. The enormous amount of biological data gathered in the last years has generated a paradigm shift, in which giving such data a coherent meaning is now a growing necessity for researchers. The interest is then to identify and understand *biological functions* building on the available knowledge on *basic elements* such as proteins and genes. This requires following a *system-level approach* where isolated data is structured as to make up interactions that, in turn, will constitute more complex interactions at a higher level of abstraction. This is, broadly speaking, the goal and motivations of what it is commonly referred to as *systems biology*.

Process calculi are abstract specification languages in which the notions of *process* and *interaction* prevail in the formalization of systems exhibiting concurrent behavior. There is a natural correspondence between the kinds of interactions provided by process calculi and those present in biological systems. The simplicity of such correspondence has captured the attention of experts in both domains. As a matter of fact, calculi for mobile processes have been used in the biological context (see, e.g., [24]), and new calculi focusing on particular aspects of biological interactions have been proposed (see [19] for a survey). This research direction is sometimes called the *language approach* for systems biology. Briefly, the idea consists in defining some *working analogies* between both biological entities and phenomena and the elements of the calculus. Evolution of biological systems can be then formalized by means of some (operational) semantics provided by the calculus. In this context, most process calculi only offer modeling and simulation capabilities.

Although this language approach has shed light on the nature of several biological systems, we believe that logic-based reasoning techniques could effectively complement biological systems analysis. More precisely, we argue for process calculi based on *timed* concurrent constraint programming (ccp) [27, 26] for analyzing biological processes. Since process terms in ccp can be viewed at the same time as computing

*`http://avispa.puj.edu.co`

agents and logic formulas, it can constitute a unified framework where biological systems can be described, simulated and also *verified*. This paper aims at supporting this claim by summarizing our initial efforts in this direction. Our base language is `ntcc` [16], a non-deterministic, discrete time process calculus based on ccp. Below we elaborate on how two salient features of `ntcc`—the interplay of partial information and non-determinism and its logic-based reasoning techniques— can be convenient in the biological context.

Partial information arises naturally in the description of biological systems. Biologists usually count with partial information obtained from experimental measurements over the systems of interest; such information should be exploited as much as possible so that working hypothesis can be refined and, at the end, new experiments can be better oriented [15]. In the spirit of the language approach, and from a more abstract level, partial information can be classified as *quantitative* and *behavioral*. While *partial quantitative information* usually involves incomplete information on the *state of the system* (e.g., the set of possible values that a variable can take), *partial behavioral information* refers to the uncertainty in the behavior of interactions (e.g., the unknown relative speeds on which two systems interact).

These two kinds of partial information are naturally captured in `ntcc`. On the one hand, partial quantitative information is captured by the notion of *constraint system*, a structure that defines logic inference capabilities over constraints. Constraint systems are *parametric* to `ntcc`, which allows to state several kinds of conditions by choosing the appropriate constraint system(s). On the other hand, partial behavioral information is represented by *non-deterministic* and *asynchronous operators* available in `ntcc`. We shall see how the interplay of these operators in the discrete time of `ntcc` allows to explicitly describe and reason about the uncertainty in the time occurrence of many biological phenomena.

Reasoning techniques in `ntcc` allow to prove whether a given process P satisfy a given property F, using a linear-temporal specification logic and its corresponding proof system. The symbolic flavor conveyed by logic-based verification can effectively complement conventional simulations when analyzing biological systems involving partial behavioral information. In fact, conventional simulations of biological components which, e.g., act at unknown or unpredictable times, might not faithfully reflect the possible behavior of the system. This kind of inaccuracies could be observed in simulations independently of how powerful simulation tools are. This is but one situation where counting with logic-based reasoning tools would come in handy for complementing analysis of biological systems.

We shall take advantage of these features by modeling biological systems as processes and their properties as linear-temporal formulas, all *in a single framework* in which non-determinism and partial information are essential. An additional advantage of using `ntcc` for the study of biological systems consists in the possibilities of turning this theoretical framework into software tools. As a matter of fact, our group has built ntccSim [4, 2], a simulation tool that admits the description of biological systems as `ntcc` processes and allows to observe their behavior over time. Formalisms, methods and tools from timed ccp therefore constitute a real alternative for biological systems analysis.

Plan of the document Section 2 further discusses the intuitions underlying systems biology outlined above. Section 3 introduces the `ntcc` calculus in a biological context. Section 4 discusses some biological systems that have been analyzed with our approach. Some related work is reviewed in Section 5. Section 6 gives some concluding remarks and proposes directions for future work.

2 Systems Biology

Recent progresses in *molecular biology* have allowed to describe the structure of many components making up biological systems (e.g., genes and proteins) as *isolated* entities. Instead of being alone, these entities are part of complex biological networks present at the cellular environment (such as, e.g., genetic regulatory networks) which define and regulate cellular processes. The current challenge is to move from molecular biology to *systems biology* [14, 15], in order to understand how these individual components and entities *integrate* to each other in the networks they shape. Once this integration has been understood, it will be then possible to discover how these entities perform their tasks.

Systems biology then aims at studying the mechanisms by which genes and proteins integrate and interact among them inside an organism. That is, systems biology studies in an integrated way both the structure and expression of a gene or a set of genes. The notions of *system* and *multilevel interaction* are crucial in this study. The former is justified by the need of considering the interactions within the given system, under the assumption that components of a system are not isolated and therefore influence each other. The latter refers to the capability of analyzing the same biological system, observing and abstracting its essential properties, at different levels of detail.

The complexity and size of biological systems motivates the use of computational techniques that allow to build models of these systems that *abstract* their behavior and make their study easier. More precisely, in a hypothesis-driven research approach [15], computing techniques are at the start of a four-stage research cycle that is complemented by analysis, technology and genomics phases. The idea is to use computer-based techniques to simulate biological models representing contradictory issues of biological significance. These so-called *dry experiments* should reveal inadequacies of the assumptions embedded in models and, after a phase where simulation results are analyzed and theories formulated, they are the basis for *wet (real) experiments*. Finally, the successful experiments will be those that eliminate inadequate models.

In this context, where the interest is to *refine* models by progressive simulations, existing languages and formalisms from concurrency theory can be convenient. Notice that the above-described hypothesis-driven approach heavily depends on the appropriate use of partial information in simulations. Moreover, counting with hypothesis suggest that the use of logic methods for their analysis is reasonable. We now enter to describe a suitable framework for carrying out this kind of analysis.

3 Timed Concurrent Constraint Programming

Here we give a concise, informal introduction to `ntcc`, the process calculus that we have used to model and verify biological systems. Based on [12], we focus on how `ntcc` constructs can be convenient in the biological context. The interested reader is referred to [16] for an in-depth presentation of `ntcc`.

We start by briefly discussing some basic notions of concurrent constraint programming (ccp), a well-established formalism for concurrency which generalizes Logic Programming [25]. One of the most appealing and distinctive features of ccp is that it combines the traditional *operational* view of process calculi with a *declarative* one of processes based upon logic. This combination allows ccp to benefit from the large body of techniques of both process calculi and logic.

In ccp the knowledge about the system is expressed in terms of *constraints*, or statements defining the possible values a variable can take (e.g., $x + y \geq 7$). These pieces of partial information are *monotonically* accumulated in shared medium, so-called *store*. Processes (or agents) then interact with each other by *telling* and *asking* constraints to the store. They synchronize according to the information in the store.

One fundamental notion in ccp is that of a *constraint system*. Informally, a constraint system provides a signature from which constraints can be constructed, and an entailment relation which specifies the inter-dependencies among them. For operational reasons, we shall require this relation to be decidable. A practical example of a constraint system is FD [13]. In FD variables are assumed to range over finite domains and, in addition to equality, we may have predicates that restrict the possible values of a variable to some finite set.

3.1 The `ntcc` process calculus

The `ntcc` process calculus [16] is a *temporal* extension of ccp. Its process constructs naturally capture the main features of timed and reactive systems. In particular, `ntcc` allows to model:

- *non-determinism* to express diverse execution alternatives for a system from the same initial conditions.
- *asynchrony* to represent unbounded but finite delays in the execution of a system.
- *unit-delays* to explicitly model pauses in system execution.
- *time-outs* to express the possibility of default behavior, reasoning about the absence of information.
- *synchrony* to control and coordinate the concurrent execution of multiple systems.
- *infinite behavior* to represent the persistent execution of a system.

`ntcc` formalizes discrete, reactive computation. In `ntcc`, time is conceptually divided into *discrete intervals* (or *time units*). In a particular time unit, a process P gets an input c from the environment (an item of information represented as a constraint), it executes with this input as the initial store, and when it reaches its resting point, it outputs the resulting store d to the environment. The resting point determines a residual process P', which is then executed in the next time unit. Notice that information is not automatically transferred from one time unit to the following.

Process Syntax

In `ntcc`, processes $P, Q, \ldots \in \mathit{Proc}$ are built from constraints $c \in \mathcal{C}$ and variables $x \in \mathcal{V}$ in the underlying constraint system by:

$$\begin{array}{rclllll} P, Q, \ldots & ::= & \mathbf{tell}(c) & \mid \sum\limits_{i\in I} \mathbf{when}\; c_i \;\mathbf{do}\; P_i & \mid P \parallel Q & \mid \mathbf{local}\; x \;\mathbf{in}\; P \\ & \mid & \mathbf{next}\,(P) & \mid \mathbf{unless}\; c \;\mathbf{next}\; P & \mid \star\, P & \mid\, !\, P \end{array}$$

Below we provide some intuitions regarding the behavior of `ntcc` processes.

Including and Querying (Partial) Information Process $\mathbf{tell}(c)$, the simplest operation to express *partial information*, includes a constraint c into the current store, thus making it available to other processes in the same time interval.

In the biological context, **tell** operations allow to represent at least two kinds of *partial information* statements: so-called *ground rules* and *state definition* statements. The first ones precisely state certain conditions that apply during the life of the biological system. These conditions can easily exploit the available (possibly incomplete) knowledge. Complementary, *state definition* statements refer to those constraints intended to define the exact values for the variables in the system. This is particularly useful when one exactly knows the set of possible states for the system at a given time; series of such statements (for different time units) thus constitute a detailed view of the behavior of the system. Remarkably, the *declarative flavor* in both kinds of statements could favor the definition of essential properties in (biological) models.

Guarded operations of the form $\mathbf{when}\; c \;\mathbf{do}\; P$ complement **tell** operations and constitute the basic means for *querying* (or *asking*) information about the state of a system. Intuitively, a $\mathbf{when}\; c \;\mathbf{do}\; P$ process queries the current constraint store: if the guard c is present in such a store then the execution of P is enabled. The "presence" of c depends on the inference capabilities associated with the store. That is, a particular constraint could not be explicitly present in the store, but it could be inferred from the available information. It is straightforward to interpret **when** operations as a way of expressing the *preconditions* for reaching a particular state of the system. The behavior of the system can be precisely stated in this way.

Non-deterministic Choices Non-determinism allows to represent several possible courses of action from the same initial state, without providing any information on how one of such courses is selected. In `ntcc`, non-deterministic behavior is obtained by generalizing processes of the form $\mathbf{when}\; c \;\mathbf{do}\; P$: a *guarded-choice summation* $\sum_{i\in I} \mathbf{when}\; c_i \;\mathbf{do}\; P_i$, where I is a finite set of indexes, represents a process that, in the current time interval, must non-deterministically choose one of the P_j $(j \in I)$ whose corresponding constraint c_j is entailed by the store. The chosen alternative, if any, precludes the others. If no choice is possible then the summation is precluded. We use $\sum_{i\in I} P_i$ as an abbreviation for the "blind-choice" process $\sum_{i\in I} \mathbf{when}\; \mathtt{true} \;\mathbf{do}\; P_i$. We use **skip** as an abbreviation of the empty summation and "+" for binary summations.

In the biological context, the combination of guarded choices and partial information represents an appropriate mechanism to formalize the inherent *unpredictability* in system interactions. In this sense, non-deterministic choices allows to explicitly represent *partial behavioral information*.

Communication Process $P \parallel Q$ represents the parallel composition of P and Q. In one time unit P and Q operate concurrently, "communicating" via the common store by adding and querying information. We use $\prod_{i\in I} P_i$, where I is a finite set of indexes, to denote the parallel composition of all P_i.

Local Information In `ntcc`, processes of the form $\mathbf{local}\ x\ \mathbf{in}\ P$ behave like P, except that all the information on x produced by P can only be seen by P and the information on x produced by other processes cannot be seen by P.

Although the conventional spirit of this kind of operators is to restrict the interface through which a process can interact with each other, in the context of partial information local information may represent a valuable help in the analysis of systems. When performing overall analyzes of complex systems, local variables may help to "hide" the behavior of those components that are irrelevant in the interactions to be analyzed. The interplay of hiding and partial information may allow to analyze systems at different levels of detail.

Basic Timed Behavior The basic time operator in `ntcc` is $\mathbf{next}\ (P)$, which represents the activation of P in the next time interval. That is, $\mathbf{next}\ (P)$ models a *unit-delay* of process P. It can be also considered as the simplest way of expressing dynamic behavior over time. This is fundamental in `ntcc`, since information is *not automatically* transferred from one time interval to the next. Based on $\mathbf{next}\ (P)$, more sophisticated delay constructs can be defined: we use $\mathbf{next}^n\ (P)$ as an abbreviation for $\mathbf{next}\ (\mathbf{next}\ (\ldots \mathbf{next}\ (P)) \ldots))$, where $\mathbf{next}$ is repeated n times.

Absence of Information / Unexpected Behavior In the biological setting, to be able of reasoning about *absence* of information is both important and necessary. Although sometimes it is possible to predict some of the possible future states for a system, usually there is a strong need of expressing *unexpected behavior*. In this kind of scenarios, processes of the form $\mathbf{unless}\ c\ \mathbf{next}\ P$ may come in handy: P will be activated only if c cannot be inferred from the current store. The "unless" processes thus add (weak) time-outs to the calculus, i.e., they wait one time unit for a piece of information c to be present and if it is not, they trigger activity in the next time interval.

Asynchrony The $\star$ operator allows to express asynchronous behavior through the time intervals. Process $\star P$ represents an arbitrary long but finite delay for the activation of P.

This kind of asynchronous behavior therefore constitutes another instance of partial behavioral information: in addition to the partial information *on the variables* that are part of the state of the system (and that can be expressed by the operators discussed above), the $\star$ operator allows to express partial information *on the time units* where processes are executed. This is particularly interesting when describing (biological) processes that interact at *unknown relative speeds*.

The partial information spirit of the asynchronous behavior in `ntcc` is strengthened by the following derived operator, expressing *bounded eventuality*:

$$\star_{[n,m]}\ P = \mathbf{next}^n\ (P) + \mathbf{next}^{n+1}\ (P) + \cdots + \mathbf{next}^{m-1}\ (P) + \mathbf{next}^m\ (P).$$

This temporal operator thus represents an additional amount of partial information, as it ensures that P will be activated at some point within the time units in the closed interval of naturals $[n, m]$. As in the original operator, there is no additional information of when this restricted eventuality will take place.

Persistent Behavior Somehow opposed to the eventual behavior enforced by asynchronous behavior, *persistent* (or infinite) behavior serves to express conditions that are valid during every possible state of the system. The *replication* operator $!P$ represents $P \parallel \mathbf{next}\ (P) \parallel \mathbf{next}^2(P) \parallel \ldots$, i.e. unboundedly many copies of P but one at a time. As such, persistent behavior is an appropriate way of enforcing conditions stating ground rules of the systems of interest. It also can also be understood as a mechanism that allows to move from *static* descriptions or conditions (valid only in one state of the system) to *dynamic* statements that are always valid.

As in the asynchronous case, it is possible to derive a bounded version of the persistent operator:

$$!_{[n,m]}\ P = \mathbf{next}^n\ (P) \parallel \mathbf{next}^{n+1}\ (P) \parallel \cdots \parallel \mathbf{next}^{m-1}\ (P) \parallel \mathbf{next}^m\ (P).$$

This operator represents the fact that P is always active during all the time units in the interval $[n, m]$. As its eventual counterpart, this derived operator (known as *bounded invariance*) may come in handy when certain additional information regarding the (persistent) execution of P is available.

LTELL	$\mathbf{tell}(c) \vdash c$	LSUM	$\dfrac{\forall i \in I \;\; P_i \vdash A_i}{\sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i \vdash \dot{\bigvee}_{i\in I}(c_i \mathrel{\dot\wedge} A_i) \mathrel{\dot\vee} \dot{\bigwedge}_{i\in I} \dot\neg\, c_i}$		
LPAR	$\dfrac{P \vdash A \;\; Q \vdash B}{P \parallel Q \vdash A \mathrel{\dot\wedge} B}$	LUNL	$\dfrac{P \vdash A}{\mathbf{unless}\ c\ \mathbf{next}\ \ P \vdash c \mathrel{\dot\vee} \circ A}$		
LREP	$\dfrac{P \vdash A}{!P \vdash \Box A}$	LLOC	$\dfrac{P \vdash A}{\mathbf{local}\ x\ \mathbf{in}\ P \vdash \dot\exists_x\, A}$		
LSTAR	$\dfrac{P \vdash A}{\star P \vdash \Diamond A}$	LNEXT	$\dfrac{P \vdash A}{\mathbf{next}\ (P) \vdash \circ A}$	LCONS	$\dfrac{P \vdash A}{P \vdash B}$ if $A \mathrel{\dot\Rightarrow} B$

Table 1: A proof system for (linear-temporal) properties of `ntcc` processes

A Logic Approach for Property Verification

`ntcc` is associated with a linear-temporal logic, which is defined as follows. Formulas $A, B, \ldots \in \mathcal{A}$ are defined by the grammar:

$$A, B, \ldots := c \mid A \mathrel{\dot\Rightarrow} A \mid \dot\neg\, A \mid \dot\exists_x\, A \mid \circ A \mid \Box A \mid \Diamond A.$$

Here c denotes an arbitrary constraint which acts as an atomic proposition. Symbols $\dot\Rightarrow$, $\dot\neg$ and $\dot\exists_x$ represent linear-temporal logic implication, negation and existential quantification. These symbols are not to be confused with the logic symbols $\Rightarrow$, $\neg$ and $\exists_x$ of the constraint system. Symbols $\circ$, $\Box$ and $\Diamond$ denote the linear-temporal operators *next*, *always* and *eventually*. We use $A \mathrel{\dot\vee} B$ as an abbreviation of $\dot\neg\, A \mathrel{\dot\Rightarrow} B$ and $A \mathrel{\dot\wedge} B$ as an abbreviation of $\dot\neg(\dot\neg\, A \mathrel{\dot\vee} \dot\neg\, B)$. The standard interpretation structures of linear temporal logic are infinite sequences of states. In `ntcc`, states are represented with constraints, thus we consider as interpretations the elements of $\mathcal{C}^\omega$. When $\alpha \in \mathcal{C}^\omega$ is a model of A, we write $\alpha \models A$.

We shall say that P satisfies A if every infinite sequence that P can possibly output satisfies the property expressed by A. A relatively complete proof system for assertions $P \vdash A$, whose intended meaning is that P satisfies A, is given in Table 1. We shall write $P \vdash A$ if there is a derivation of $P \vdash A$ in this system.

4 Using `ntcc` for Analyzing Biological Systems

In this section we describe how we have applied our approach for biological systems analysis in several kinds of systems. These include: mechanisms for active transport of substances through cellular membranes, genetic regulatory networks (GRNs), and mutations over a GRN. We briefly comment on the nature of the modeled systems and describe their `ntcc` models.

4.1 Active Transport in Cellular Membranes

In [12] we have used `ntcc` to model and verify an *ion pump*, a natural channel connecting the two sides of a membrane. These pumps move ions across the membrane in a process called *transport*. Depending on the source of the required energy, the transport can be either *passive* or *active*. In passive transport ions freely move across the membrane following an electrochemical gradient, so the cell does not need to provide energy for the transport. Since in active transport ions move against the direction of the gradient, the cell has to supply energy (usually in form of ATP) to accomplish this movement.

The Sodium-Potassium pump [28] (SP-pump in the sequel) is a system for active transport in animal cells. It exchanges Sodium ions inside the cell with Potassium ions outside of it. The pump is composed of two proteins known as the alpha and beta subunits. The purpose of the pump is to keep the concentration of sodium inside the cell lower than outside. This difference of concentrations generates an electrochemical gradient that leads the passive transport of Sodium ions towards the cytoplasm in the cell. If the pump does not work well then the gradient becomes weak for transport, thus affecting the entrance of required substances into the cell.

The pumping process in the SP-pump can be divided in six phases. At the beginning there is a pump conformation with high affinity for Sodium ions inside the cell (1). This conformation encourages the binding of three Sodium ions with the pump. Then the alpha subunit is phosphorylated by ATP hydrolysis (2), leaving a residual ADP molecule in the cytoplasm. This chemical reaction provides the needed energy for the pumping process. Once this occurs, the pump conformation changes and then the Sodium ions can leave the cell (3). At this point, there is a pump conformation with high affinity for Potassium ions outside the cell (4). This results in the binding of two Potassium ions with the pump. Hence, the alpha subunit is dephosphorylated (5) and the pump conformation returns to the initial state. At this moment Potassium ions can enter the cell (6). The pumping process is always performed regulating the concentration of Sodium in the cell.

In parallel to this active transport movement, there is a *passive* transport movement that allows Potassium and Sodium ions to move against the direction of the active transport. This complementary movement is induced by an electrochemical gradient present in the cell.

Elements of an `ntcc` model of the SP-pump

Here we describe the main principles underlying the `ntcc` model of the SP-pump. We use non-deterministic and asynchronous behavior for modeling partial behavioral information regarding temporal responses of certain components. We use mutable entities (*cells*) and recursive definitions in some of our models. Cells can be easily encoded in `ntcc`; see [16] for more details.

The model assumes a constraint system over finite domains of integers, considering three places for interaction: inside and outside the cell, and an intermediate place where ions stay before entering or flowing out of the cell (i.e., the pump). The model involves a series of cells that store useful quantities about the pumping process. We use notations $x : v$ and $x := v$ to represent the *initialization* and the *assignment* of a cell x with value v, respectively. Output and input operations of the pump are then modeled as modifications over variables representing the number of ions both inside and outside the cell. In this way, for instance, variables Na_O and Na_I represent the amount of Sodium ions placed outside and inside the cell, respectively. In addition, a certain amount of each kind of ion needed for the correct functioning of the cell is assumed. Such amounts are denoted by Na_{IDEAL} and K_{IDEAL}. Moreover, some additional variables capture other details of the pump: $OPump$ represents the orientation of the pump (either inside or outside the cell), $Alpha$ denotes the current binding of the alpha subunit and $Pump$ represents the current content of the pump. These three variables will be instantiated with constants that can be encoded by integers: for instance, possible values for $Alpha$ are `P`, `free` and `null` (note the special font style given to constants). Finally, integer variables ATP and ADP represent the presence of ATP and ADP inside the cell, respectively.

The complete model for the SP-pump (denoted as the $NaKPump$ process) reflects the complementary nature of active and passive transport in the SP-pump, and is represented by the integration of $ActiveTrans$ and $PassiveTrans$ processes. From this $NaKPump$ process it is then possible to assume some environment in which the pump is placed. This is the intuition behind process $System$. We now proceed to explain in a greater detail the ideas behind these processes. For the sake of space, we only include fragments of the model; the interested reader is referred to [12] for complete details.

Active Transport Phases Process $ActiveTrans$ integrates sub-processes for the six phases described before; these processes invoke each other. Some processes include recursive calls to themselves. This intends to represent the possibility that the system remains stuck in certain phases, even if all the conditions needed to evolve are given. That is, we are trying to model “reversible” phases, a behavior that is represented by non-deterministic choices. As a result, those phases could be executed several times therefore delaying system execution in at least one time unit. Such a delay occurs because the system waits for the presence of some substances at a specific place of the pump. In fact, those substances could be available but not in the required place. Figure 1 presents a fragment of $ActiveTrans$ in which the phases where the Sodium leaves out the cell are represented. Process $NaPhase2$ is the only reversible phase.

The above-described non-deterministic and asynchronous behavior could represent other conditions on component binding, such as an appropriate physical contact among elements that (chemically) react with components of the pump. Similarly, non-deterministic behavior can also represent some kind of malfunction.

$$
\begin{array}{lcl}
NaPhase1 & \stackrel{\text{def}}{=} & \textbf{when } (Na_I > Na_{IDEAL} \vee K_I < K_{IDEAL}) \wedge Pump = \texttt{Empty} \wedge OPump = \texttt{In} \textbf{ do} \\
& & \quad (\textbf{next } (Na_I := Na_I - 3 \parallel Pump := \texttt{Na} \parallel \textbf{tell}(unchangedK = 1) \parallel NaPhase2) \; + \\
& & \qquad \textbf{next } (NaPhase1 \parallel \textbf{tell}(unchangedK = 1) \parallel \textbf{tell}(unchangedNa = 1))) \\
NaPhase2 & \stackrel{\text{def}}{=} & \textbf{when } Pump = \texttt{Na} \wedge Alpha = \texttt{free} \wedge ATP > 0 \textbf{ do} \\
& & \quad (\textbf{next } (OPump := \texttt{Out} \parallel Alpha := \texttt{P} \parallel ADP := 1 \parallel \\
& & \qquad \qquad \textbf{tell}(unchangedK = 1) \parallel \textbf{tell}(unchangedNa = 1) \parallel NaPhase3) \\
& & \qquad + \textbf{next } (NaPhase2 \parallel \textbf{tell}(unchangedK = 1) \parallel \textbf{tell}(unchangedNa = 1))) \\
NaPhase3 & \stackrel{\text{def}}{=} & \textbf{when } Pump = \texttt{Na} \wedge OPump = \texttt{Out} \textbf{ do} \\
& & \quad \textbf{next } (Na_O := Na_O + 3 \parallel Pump := \texttt{Empty} \parallel \textbf{tell}(unchangedK = 1) \parallel KPhase1)
\end{array}
$$

Figure 1: Fragment of the ntcc model for the active transport phases of the Sodium-Potassium pump

$$
\begin{array}{lcl}
PassiveNa & \stackrel{\text{def}}{=} & \textbf{unless } \; Na_O = Na_I \; \textbf{ next} \\
& & \quad (\textbf{next}^5 \; (PassiveNa) \parallel \\
& & \qquad \star_{[0,5]}(\textbf{unless } \; unchangedNa = 1 \; \textbf{ next } \; (Na_I := Na_I + 3 \parallel Na_O := Na_O - 3) \parallel \\
& & \qquad \qquad \textbf{when } unchangedNa = 1 \textbf{ do } \; (Na_I := Na_I + 3 \parallel Na_O := Na_O - 3))) \\
PassiveTrans & \stackrel{\text{def}}{=} & PassiveNa \parallel PassiveK
\end{array}
$$

Figure 2: Fragment of the ntcc model for the passive transport phases of the Sodium-Potassium pump

For instance, in phase $NaPhase2$ the phosphate could not bind to the alpha subunit, which would result in a malfunction of the system that could be directly observed from the evolution of the pump in time.

Passive Transport Phases Passive transport is represented by process $PassiveTrans$, which defines two sub-processes: one for the entrance of Sodium ions and another for the output of Potassium ions. In the modeling of these sub-processes we consider partial behavioral information on the actual time when the ion movement really occurs, which is represented by a bounded asynchronous operator. Figure 2 describes a fragment of $PassiveTrans$.

Additional Processes The integration of the above processes as in the $NakPump$ process is straightforward. There is an additional process (i.e., $Control$) which governs the global behavior of the pump w.r.t. the equilibrium of the ions amounts; in the case an equilibrium on the amount of one of the ions is reached, a general system malfunction (denoted as $death = 1$) is established. As the other processes, the structure of this control process makes it possible the inclusion of additional features. Process $Start$, which receives a group of six parameters (denoted as $\sigma_{1\ldots6}$), sets up the variables used in the model. Figure 3 shows a fragment of the complete model.

Remarkably, our models can be parametrized with actual quantitative values extracted from experimentation. Indeed, ion concentrations depend on *parameters* which make it more accurate; more detailed models involving other biological components (such as, e.g., the electrochemical gradients governing the dynamics of the passive transport) would then require the inclusion of more sophisticated numerical parameters. In this sense, considering a constraint system over real numbers would not only allow to include such parameters but also would allow to perform analyzes at different levels of detail.

Verifying the SP-pump

We now briefly describe how a non-trivial biological property can be verified over the sketched ntcc model of the SP-pump. Assume an *inhibition process* over the SP-pump that is enforced by a malicious drug that

$$\begin{aligned} NaKPump &\stackrel{\text{def}}{=} \textbf{local}\ Na_I, Na_O, K_I, K_O, Alpha, ADP, Pump, OPump\ \textbf{in} \\ &\qquad Start(\sigma_{1\ldots6}) \parallel ActiveTrans \parallel PassiveTrans \parallel Control \\ System &\stackrel{\text{def}}{=} NaKPump \parallel Environment \end{aligned}$$

Figure 3: Integrated `ntcc` model for the Sodium-Potassium pump (Fragment)

is present in the environment surrounding the pump. The goal of this drug is to take control of the alpha subunit, thus preventing the phosphate from inducing a conformational change in the pump. Such an this obstruction will lead to a complete inhibition of the active transport mechanism of the pump. We express this in our model by specifying the $Environment$ process as follows:

$$Environment \stackrel{\text{def}}{=} Drug \star_{[m,n]} \ \textbf{when}\ Alpha = \texttt{free}\ \textbf{do}\ !Alpha := \texttt{null} \qquad (n > m) \qquad (1)$$

It is easy to see that the actual time unit where $Drug$ will be active is undetermined, because of the uncertainty induced by the $\star$ operator. Notice that we are focusing on the drug-related part of $Environment$: other aspects of it could be easily specified.

Clearly, by inhibiting the active transport component of the pump, the cell will reach an equilibrium between the internal and external Sodium concentrations. Such an *irreversible* equilibrium causes the death of the cell and will occur in an undetermined future. These facts suggest us the following assertion to be verified:

$$NaKPump \parallel Drug \vdash \Diamond\Box\ death = 1 \qquad (2)$$

where $death = 1$ represents the death of the cell. Intuitively, we want to formally verify that in the presence of the drug described above the cell will die in an undetermined future, with no chance of returning to a previous state.

In [12] we use the inference system of `ntcc` to derive a proof for (2). Informally, the idea is to restrict the attention to the interaction among $Control$, $PassiveNa$ and $Drug$. Due to the absence of the active transport mechanism the passive transport will introduce sodium ions into the cell until reaching an equilibrium (i.e., $Na_I = Na_O$). Once that occurs, $Control$ (that has been awaiting the equilibrium) emits $equilNa = 1$ to the environment. Such a signal is enough to determine the death of the cell.

4.2 Genetic Regulatory Networks

Here we discuss how genetic regulatory networks (GRNs) can be modeled in `ntcc`. We propose a group of "building blocks" for modeling: each block represents a particular behavior that is frequent in GRNs. Some of these blocks are *generic processes* that can be parametrized according to the specific GRN, while others are *templates* that give guidelines on how to define actual `ntcc` processes. They have been used in [3, 2] to model and simulate regulation processes (repression and induction) of the *lac operon*, a genetic cluster that participates in the the transport and metabolism of lactose in bacteria such as the *E. Coli*.

Building Blocks for Modeling GRNs

GRNs are one of the most studied systems, mainly because of its importance at the cellular level. They control (or regulate) cellular processes according to the information provided by the ADN of each organism. At the molecular level, GRNs depend on many factors which make them particularly difficult to understand. Finding concise mathematical models describing behavior of GRNs is challenging as they are composed of elements that can be related to both discrete and continuous systems. In spite of this, it is possible to abstract some features that are common to GRNs at the molecular level. We now describe `ntcc` models for such features as *blocks* that might help to better formalize GRNs.

Continuity Regulation in GRNs is determined by the concentration levels of different biological entities along time. This motivates to consider two different kinds of continuity: persistence in the values of the variables and continuous time.

To model persistence of a single variable it is easy to think in a process $State_i$ that, for a variable m_i, explicitly transfers the current value of m_i to the next time unit. More precisely, the idea is, in the current time unit, to schedule a process $State_i(v_i)$ that will set the variable m'_i (which represents the value of m_i in the previous time unit) with v_i, the current value of m_i. This idea can be extended to a group of values in a straightforward manner:

$$State(\rho_1, \ldots, \rho_n) \stackrel{\text{def}}{=} \prod_{i \in I} (\ \mathbf{tell}(m'_i = \rho_i) \parallel \mathbf{next}\ (State_i(\rho_i))\ )$$

where I is the set of indexes of variables in the biological system and ρ_i is the current value of m_i. $State$ can be used to configure system simulations with parameters coming from actual biological measurements.

Temporal continuity is achieved by considering many `ntcc` time units as "samples" of one system unit:

$$Time_{Dt}(t) \stackrel{\text{def}}{=} \mathbf{tell}(Ts = t) \parallel \mathbf{next}\ (Time(t + Dt))$$

where Ts is the *continuous* time value of the system in the current time unit. Constant Dt represents the *resolution* of the system: it gives an idea of how fine the sampling is. As such, we can expect a trade-off between precision and efficiency: lower values of Dt give better approximations of real continuous systems but will demand more resources in system simulations. Process $Dynamic$ below can represent the continuous behavior of the whole system.

$$Dynamic \stackrel{\text{def}}{=} State(\rho_1, ..., \rho_n) \parallel Time_{Dt}(0.0)$$

Molecular Events Molecular systems involve several events that have to be considered, such as, e.g., the detection of when a group of molecules interacts with others or performs a specific task. We shall use discrete variables to indicate either presence or absence of molecular events in models. Such variables will be called *signaling variables*. The following is a generic `ntcc` process representing molecular behavior:

$$Signal \stackrel{\text{def}}{=} !\prod_{e \in E,\ svar \in S} (\ \mathbf{when}\ e\ \mathbf{do}\ \ \mathbf{next}\ (\mathbf{tell}(svar = 1)) \parallel \mathbf{unless}\ \ e\ \ \mathbf{next}\ \ \ \mathbf{tell}(svar = 0)\ )$$

where E is the set of constraints expressing molecular events and S the set of signaling variables in the system. Some readers might relate this process with an if-then-else construct. Nevertheless, $Signal$ provides a more sophisticated behavior as it can reason about *absence of information* on the conditions in E.

Regulation and status values Most of the processes used to represent dynamic behavior of biological entities share a similar structure. They can be modeled as processes controlled by signaling variables. The parametric process $Regulate_i$ models the behavior of an entity i which is under the control of a signaling variable $svar$. The value of $svar$ determines the execution of either P_i or N_i; this is represented as an exclusive choice.

$$Regulate_i(svar, P_i, N_i) \stackrel{\text{def}}{=} \mathbf{when}\ svar = 1\ \mathbf{do}\ \ P_i\ +\ \mathbf{when}\ svar = 0\ \mathbf{do}\ \ N_i$$

To model *status* (or level) of gene transcription, we use process $Status_i$ below as a *template* to define a wide variety of situations in which we want to determine particular conditions in/of a biological entity.

$$Status_i \stackrel{\text{def}}{=} !\ (\ (\sum_{c \in C} \mathbf{when}\ cond_c\ \mathbf{do}\ \ \mathbf{next}\ (\mathbf{tell}(m_i = fc_i(m'_i)))\ )\ \parallel \mathbf{unless}\ \bigvee_{c' \in C} cond_{c'}\ \ \mathbf{next}\ \ \mathbf{tell}(m_i = m'_i)\ )$$

The above process assumes that conditions for changes in the status are indexed by the set C, so for two different i, j, $cond_i$ and $cond_j$ are two different conditions. The new value is defined by a control function fc_i. When no condition for change holds, the state of the system remains unchanged in the next time unit.

Genes Process Gen_x below is a parametric specification representing the structure and behavior of a single gene. It is defined using the generic process $Regulate_i$ and the template $Status_i$. The considered parameters represent the degradation and production rates of mRNAs as well as the proteins produced in the transcription and translation of a gene. We consider three entities: level of transcription and concentration of both mRNAs and proteins produced by the gene.

$$\begin{array}{rcl} GenStatus_i & \stackrel{\text{def}}{=} & !\ (\ (\ \mathbf{when}\ tbegin = 1 \wedge tend = 0\ \mathbf{do}\ \ \mathbf{next}\ (\mathbf{tell}(m_i = m_i' + 1))\ + \\ & & \qquad \mathbf{when}\ tbegin = 0 \wedge tend = 1\ \mathbf{do}\ \ \mathbf{next}\ (\mathbf{tell}(m_i = m_i' - 1))\ )\ \parallel \\ & & \qquad \mathbf{unless}\ \ tbegin\ \neq\ tend\ \ \mathbf{next}\ \ \mathbf{tell}(m_i = m_i')\ ) \\ MRNA_j(p_j, d_j) & \stackrel{\text{def}}{=} & Regulate_j(tbegin, \mathbf{next}\ (\mathbf{tell}(m_j = m_j' + p_j - Dt \times (d_j \times m_j'))), \\ & & \qquad \mathbf{next}\ (\mathbf{tell}(m_j = m_j' - Dt \times (d_j \times m_j')))) \\ PROTEIN_k(p_k, d_k) & \stackrel{\text{def}}{=} & Regulate_k(mrnah, \mathbf{next}\ (\mathbf{tell}(m_k = m_k' + Dt \times (p_k \times m_j' - d_k \times m_k'))), \\ & & \qquad \mathbf{next}\ (\mathbf{tell}(m_k = m_k' - Dt \times (d_k \times m_k')))) \\ Gen_x(p_j, d_j, p_k, d_k) & \stackrel{\text{def}}{=} & GenStatus_i \parallel !\ MRNA_j(p_j, d_j) \parallel !\ PROTEIN_k(p_k, d_k) \end{array}$$

In Gen_x, m_i, m_j and m_k are variables representing the status of gene expression, mRNA concentration and protein concentration, respectively. Moreover, d_j and d_k represent the rate of molecular degradation of mRNAs and proteins, respectively. The production rate of these entities is determined by the constants p_j and p_k and by two signaling variables $tbegin$ and $tend$. These denote the starting and ending time of RNA polymerase gene transcription. Signaling variable $mrnah$ is used to indicate when the mRNA concentration is "high enough" to start protein translation.

In order to model when RNA polymerase is placed between two genes an additional process is required. Such a process should control when each gene starts and finishes transcription. In [3] this process is modeled using the $Status_i$ template.

4.3 Modeling Biological Mutations

In this example we are interested in modeling the control system of a GRN. Below we define three `ntcc` processes: $StartControl$, $MutatedGene$ and $WildGene$. The first process indicates the number of molecules interacting with the control region at the start of the study of the system. The second one defines the system behavior under mutated conditions. The last one represents the system behavior in wild or normal conditions. Variable x represents the cellular concentration of molecules interacting with the control region of the set of genes.

$$\begin{array}{rcl} StartControl & \stackrel{\text{def}}{=} & \mathbf{tell}(x = n) \\ MutatedGene & \stackrel{\text{def}}{=} & \star\ !\ (\mathbf{tell}(mut = 1) \parallel \mathbf{next}\ (\mathbf{tell}(x = f_m))) \\ WildGene & \stackrel{\text{def}}{=} & !\ \mathbf{unless}\ \ mut = 1\ \ \mathbf{next}\ \ \mathbf{tell}(x = f_w) \\ ControlRegion & \stackrel{\text{def}}{=} & Start \parallel MutatedGene \parallel WildGene \end{array}$$

In the above definitions, process $MutatedGene$ establishes that a mutation will eventually occur in the gene in an undetermined future time unit and, as a consequence, the behavior of the system will be defined by the constraint $x = f_m$, where f_m is a function determining an incorrect behavior in the gene control region. In addition, process $WildGene$ states that the behavior of the control region is represented by the constraint $x = f_w$ unless the mutation occur (which is represented by constraint $mut = 1$). Function f_w represents the behavior of the system in wild conditions. Figure 4, obtained using ntccSim, illustrates the behavior of the system parametrized with value $n = 0$.

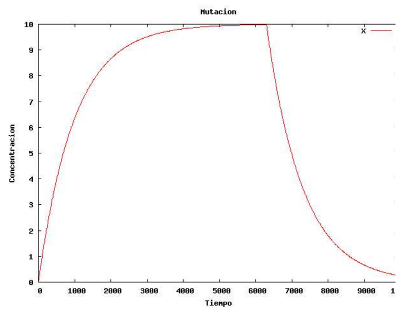

Figure 4: Molecular concentration in a DNA region of a mutated gene

A Complementary Proof In this section we will verify a system property using the inference system associated with `ntcc`. As a case of study, we will verify that when the mutation occur, variable x will be determined only by function f_m. Formally, we wish to verify the following formula:

$$ControlRegion \vdash \Diamond\Box x = f_m$$

The formulas for processes $StartControl$, $MutatedGene$ and $WildGene$ are:

$$\begin{array}{lcl} StartControl & \vdash & x = n \\ MutatedGene & \vdash & \Diamond\Box(mut = 1 \,\dot{\wedge} \circ x = f_m) \\ WildGene & \vdash & \Box(mut = 1 \,\dot{\vee} \circ x = f_w) \end{array}$$

$$\dfrac{\dfrac{}{StartControl \vdash x = n}\ \textsf{LTELL} \qquad \dfrac{}{MutatedGene \vdash \Diamond\Box(mut = 1 \,\dot{\wedge} \circ x = f_m)}\ \textsf{LRULES1}}{StartControl \parallel MutatedGene \vdash \ (\ x = n\ )\ \dot{\wedge}\ (\ \Diamond\Box(mut = 1 \,\dot{\wedge} \circ x = f_m)\ )}\ \textsf{LPAR}$$

where LRULES1 denotes the systematic application of rules LSTAR, LREP, LPAR, LNEXT and LTELL of the proof system over process $MutatedGene$. For the sake of space, we assume the following abbreviations: $SC = StartControl$ and $MG = MutatedGene$.

$$\dfrac{\dfrac{}{WildGene \vdash \ (\ \Box(mut = 1 \,\dot{\vee} \circ x = f_w)\ )}\ \textsf{LRULES2} \qquad \dfrac{}{SC \parallel MG \vdash \ (\ x = n\ )\ \dot{\wedge}\ (\ \Diamond\Box(mut = 1 \,\dot{\wedge} \circ x = f_m)\ ))}}{WildGene \parallel SC \parallel MG \vdash \ (\ \Box(mut = 1 \,\dot{\vee} \circ x = f_w)\ )\ \dot{\wedge}\ (\ x = n\ )\ \dot{\wedge}\ (\ \Diamond\Box(mut = 1 \,\dot{\wedge} \circ x = f_m)\ )}\ \textsf{LPAR}$$

where LRULES2 represents the application of rules LREP, LUNL and LTELL over process $WildGene$. Finally, we can perform the following deduction:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{ControlRegion \vdash \ (\ \Box\,(mut = 1 \,\dot{\vee} \circ x = f_w)\ )\ \dot{\wedge}\ (\ x = n\ )\ \dot{\wedge}\ (\ \Diamond\Box\,(mut = 1 \,\dot{\wedge} \circ x = f_m)\ )}}{ControlRegion \vdash \Box\,(mut = 1 \,\dot{\vee} \circ x = f_w)\,\dot{\wedge}\,\Diamond\Box\,(mut = 1 \,\dot{\wedge} \circ x = f_m)}\ \textsf{LCONS}}{ControlRegion \vdash \Diamond\Box\,(\ \ (mut = 1 \,\dot{\vee} \circ x = f_w)\,\dot{\wedge}\,(mut = 1 \,\dot{\wedge} \circ x = f_m)\ )}\ \textsf{LCONS}}{ControlRegion \vdash \Diamond\Box\,(\ mut = 1 \,\dot{\wedge}\,(mut = 1 \,\dot{\wedge} \circ x = f_m)\ )}\ \textsf{LCONS}}{ControlRegion \vdash \Diamond\Box\circ x = f_m}\ \textsf{LCONS}}{ControlRegion \vdash \Diamond\Box x = f_m}\ \textsf{LCONS}$$

The above logical expression verify that the constraint $x = f_m$ will define the behavior of the system in an undetermined future time, and that this behavior will continue forever.

We have shown how the behavior of a system can be analyzed by two formal ways: (i) by following the steps of the operational semantics in a mechanical way, using ntccSim (Figure 4) and (ii) by verifying temporal properties using the ntcc inference system. A remarkable aspect to consider here is that it is possible that we may not see the mutation by simulations, since this could occur in a very long time. As a consequence, in this case the logical proof can be regarded as being more effective, as it can reveal the actual behavior of the system.

5 Related Work

Some of the main representative calculi within the so-called language approach for systems biology are the π-calculus [22, 23], BioAmbients [21], the Brane calculus [8], Beta binders [20] and the κ-calculus [9]. The use of these calculi as as description languages for Biology has been studied in recent years and, as mentioned in the introduction, little work has been done on relating them with logic-based reasoning techniques. Some of such works have explored the use of constraints and/or logic in the biological context, see, e.g., [11,5,6,1]. Two of them ([5,6]) are most related to our approach and deserve special mention. We review them separately.

Stochastic CCP Stochastic CCP (sCCP) [5] is an untimed, stochastic extension of the ccp model. The main difference wrt the original model proposed in [27] is the addition of a λ function to ask and tell operations as well as to procedure calls. The intuitive meaning of this function is twofold. In fact, it can be understood either as a priority within a probability distribution or as the speed associated with performing each operation. From a practical perspective, there is an interpreter of sCCP processes, built in SICStus Prolog, that allows for simulation of biological systems.

In order to model biochemical networks, the work in [5] offers parameterizable processes to describe reversible and irreversible reactions as well as reactions described by Michaelis-Mentel and Hill equations. The definition of similar processes in ntcc this is also possible. Moreover, to model genetic regulatory networks, three basic processes (or *logical gates*) are proposed to model regulation. More precisely, processes *pos_gate*, *neg_gate* and *null_gate*, intended to model positive, negative and absence of regulation, respectively, are proposed. This kind of sCCP processes can be easily modeled in ntcc.

Clearly, the use of stochastic parameters is the main difference between sCCP and ntcc. We have already started to work on equipping ntcc with probabilistic/stochastic constructs (see Section 6). We feel that the combination of probabilistic behavior with the discussed advantages of ntcc in the biological context (time, partial information, logic reasoning techniques) will constitute a strong framework where biological systems can be better studied.

Temporal Logic with Constraints The works [6,7] propose BIOCHAM, a biochemical abstract machine. In BIOCHAM biological systems are modeled using a rule-based language. This approach is, according to the authors, more natural to the biologists and well-suited for applying model checking techniques. This is perhaps the main difference wrt our approach, as processes in ntcc have a natural relationship with the temporal logic associated to the language. Furthermore, we think that the explicit time representation inherent to ntcc can, in combination with the non-deterministic and asynchronous constructs, be intuitive enough for experts when describing the behavior of (possibly partially known) biological systems.

Reasoning techniques include three independent semantic structures (each one with an associated logic), which are used depending on the desired level of detail. The simplest semantics is a *boolean* one that associates a boolean variable to each biological entity, with the possibility of checking *qualitative* properties using Computational Tree Logic (CTL). In the *concentration* semantics each entity is associated to a real number representing its concentration. Reaction rules are interpreted as kinetic rules and a fragment of LTL is used for verification. Finally, in the *stochastic* semantics an integer is used to model the number of molecules of each entity in the system. Notice how for each level of abstraction there is a different meaning for the modeling language and different verification approaches. We believe that by the appropriate use of constraint systems in the description of systems, analysis at several levels of detail are possible, preserving the *same unified framework*.

6 Concluding Remarks and Future Work

In this paper we have shown how ntcc, a timed, non-deterministic process calculus based on constraints, can be convenient for the analysis of different kinds of biological systems. We have seen how the interplay of the operational and logic perspectives of processes —a distinctive feature of ccp languages— serves as a unified framework upon which expressive models of biological systems can be described, observed and, unlike most similar works, verified using a temporal logic.

The discussed biological systems serve to illustrate the advantages of using ntcc in the biological context. In fact, ntcc allows to take advantage of the natural use of *processes* as independent agents to model biological entities, *discrete time* constructs as flexible mechanisms to describe dynamic properties of systems, *constraints* as a way of representing incomplete information about the state of a system (i.e., partial *quantitative* information), and *asynchronous and non-deterministic* constructs to formally model unpredictable behavior in the evolution of a system (i.e., partial *behavioral* information). Moreover, these advantages in modeling are complemented by both practical and theoretical opportunities for simulating and verifying biological models. On the one hand, it is possible to run ntcc specifications in ntccSim in order to know a possible execution path showing the behavior of a system, given a set of particular conditions (e.g., initial number of molecules in a system). On the other hand, the use of an LTL inference system to prove temporal properties about ntcc models allows to discover non-trivial behavior patterns, including those encompassing asynchronous and non-deterministic nature. This tight relationship between operational and logic reasoning tools is rarely seen in other formalisms, even in those also based on the ccp model.

All these appealing features certainly motivate us to further work on the applications of ntcc to the biological context. A current work direction pertains to the use of quantitative information in models of biological systems. Particularly important is the inclusion of probabilistic/stochastic information both in ntcc models and ntccSim. We have already obtained some preliminary results. In fact, in [17] a version of ntcc in which the signature of the constraint system is extended with a probability function is proposed. Intuitively, the role of such a function is to return “true” or “false”, taking a real number as a parameter. This adds a significant flexibility to process definitions, as one could devise processes that are executed depending on the outcome of such a function. The advantages of using this extended language in the biological context were described in [18], where cooperativity in a genetic regulation network is formally studied. Moreover, we count with preliminary results on the design of a *probabilistic* extension of ntcc with probabilistic choice. We expect to refine these theoretical extensions by modeling, simulating and verifying more complex biological systems than the ones analyzed so far.

From a more practical point of view, the development of efficient mechanisms for including ordinary differential equations (ODEs) in models and simulations is also compulsory. Although we have defined some encodings of ODEs using ntcc, we plan to implement a constraint system over ODEs in Mozart. Such a system, in combination with the existing constraint systems over real intervals and finite domains, will allow to take advantage of the knowledge currently held by biologists about the structure and behavior of molecular systems, and consequently, to contribute to fill the gap that prevents computer scientists from straightforwardly using some well studied models of biological networks.

References

[1] M. Antoniotti, C. Piazza, A. Policriti, M. Simeoni, and B. Mishra. Taming the complexity of biochemical models through bisimulation and collapsing: theory and practice. *Theor. Comput. Sci.*, 325(1):45–67, 2004.

[2] A. Arbeláez and J. Gutiérrez. Estudio Exploratorio de la Aplicación de la Programación Concurrente por Restricciones en Bioinformática (Applying Concurrent Constraint Programming in Bioinformatics: An Exploratory Study). BSc Thesis – Engineering Degree in Computer Science, Pontificia Universidad Javeriana - Cali (Colombia)., 2006.

[3] A. Arbeláez, J. Gutiérrez, C. Olarte, and C. Rueda. A generic framework to model, simulate and verify genetic regulatory networks. In *Proc. of 32nd Latin-American Conference on Informatics (CLEI 2006)*, 2006.

[4] AVISPA Research Group. ntccSim: A simulation tool for timed concurrent processes, 2006. More information available at `http://avispa.puj.edu.co`.

[5] L. Bortolussi and A. Policriti. Modelling Biological Systems in Stochastic Concurrent Constraint Programming. In *Proc. of WCB06 - Workshop on Constraint-Based Methods in Bioinformatics*, 2006.

[6] L. Calzone, N. Chabrier-Rivier, F. Fages, and S. Soliman. Machine learning biochemical networks from temporal logic properties. *Transactions on Computational Systems Biology*, 2006. CMSB'05 Special Issue (to appear).

[7] L. Calzone, F. Fages, and S. Soliman. BIOCHAM: An Environment for Modeling Biological Systems and Formalizing Experimental Knowledge. *Bioinformatics*, 22:1805–1807, 2006.

[8] L. Cardelli. Brane Calculi. In Danos and Schächter [10], pages 257–278.

[9] V. Danos and C. Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.

[10] V. Danos and V. Schächter, editors. *Computational Methods in Systems Biology, International Conference CMSB 2004, Paris, France, May 26-28, 2004, Revised Selected Papers*, volume 3082 of *LNCS*. Springer, 2005.

[11] D. Eveillard, D. Ropers, H. de Jong, C. Branlant, and A. Bockmayr. A multi-scale constraint programming model of alternative splicing regulation. *Theor. Comput. Sci.*, 325(1):3–24, 2004.

[12] J. Gutiérrez, J. A. Pérez, C. Rueda, and F. D. Valencia. Timed Concurrent Constraint Programming for Analysing Biological Systems. In *Proc. of the Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'06)*, 2006. To appear in the ENTCS (Electronic Notes in Theoretical Computer Science) series.

[13] P. Van Hentenryck, V. Saraswat, and Y. Deville. Design, Implementation, and Evaluation of the Constraint Language cc(FD). In *Constraint Programming*, volume 910 of *LNCS*, pages 293–316. Springer, 1994.

[14] H. Kitano, editor. *Foundations of Systems Biology*. MIT Press, 2001.

[15] H. Kitano. Systems Biology: A Brief Overview. *Science*, 295:1662–1664, 2002.

[16] M. Nielsen, C. Palamidessi, and F. Valencia. Temporal concurrent constraint programming: Denotation, logic and applications. *Nordic Journal of Computing*, 9:145–188, 2002.

[17] C. Olarte and C. Rueda. A Stochastic Non-deterministic Temporal Concurrent Constraint Calculus. In *Proc. of International Conference of the Chilean Computer Science Society (SCCC 2005)*. IEEE-CS, 2005.

[18] C. Olarte and C. Rueda. Using stochastic ntcc to model biological systems. In *Proc. of the 31st Latin American Conference on Informatics (CLEI'05)*, 2005.

[19] D. Prandi, C. Priami, and P. Quaglia. Process calculi in a biological context. *Bulletin of the EATCS*, February 2005.

[20] C. Priami and P. Quaglia. Beta Binders for Biological Interactions. In Danos and Schächter [10], pages 20–33.

[21] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. Bioambients: an abstraction for biological compartments. *Theor. Comput. Sci.*, 325(1):141–167, 2004.

[22] A. Regev and E. Shapiro. Cells as Computation. *Nature*, 419:343, September 2002.

[23] A. Regev and E. Shapiro. *Modelling in Molecular Biology*, chapter The π-calculus as an abstraction for biomolecular systems, pages 219–266. Natural Computing Series. Springer, 2004.

[24] Aviv Regev, William Silverman, and Ehud Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[25] V. Saraswat. *Concurrent Constraint Programming*. The MIT Press, 1993.

[26] V. Saraswat, R. Jagadeesan, and V. Gupta. Foundations of timed concurrent constraint programming. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 71–80. IEEE, 1994.

[27] V. Saraswat, M. Rinard, and P. Panangaden. The semantic foundations of concurrent constraint programming. In *POPL '91*, pages 333–352, Jan 1991.

[28] G. Scheiner-Bobis. The sodium pump: Its molecular properties and mechanics of ion transport. *Euro. J. Biochem.*, 269:2424–2433, 2002.