A Stochastic Concurrent Constraint Based Framework to Model and Verify Biological Systems

Carlos Olarte and Camilo Rueda
Javeriana University, Dept. Computer Science.
{caolarte,crueda}@cic.puj.edu.co

Abstract

Concurrent process calculi are powerful formalisms for modelling concurrent systems. The mathematical style underlying process calculi allow to both model and verify properties of a system, thus providing a concrete design methodology for complex systems. `ntcc` , a constraints-based calculus for modeling temporal non-deterministic and asynchronous behaviour of processes has been proposed recently. Process interactions in `ntcc` can be determined by partial information (i.e. constraints) accumulated in a global store. `ntcc` has also an associated temporal logic with a proof system that can be conveniently used to formally verify temporal properties of processes. We are interested in using `ntcc` to model the activity of genes in biological systems. In order to account for issues such as the basal rate of reactions or binding affinities of molecular components, we believe that stochastic features must be added to the calculus. In this paper we propose an extension of `ntcc` with various stochastic constructs. We describe the syntax and semantics of this extension together with the new temporal logic and proof system associated with it. We show the relevance of the added features by modelling a non trivial biological system: the gene expression mechanisms of the λ virus. We argue that this model is both more elaborate and compact than the stochastic π calculus model proposed recently for the same system.

1 Introduction

We are interested in using soft computing techniques for modelling complex systems such as those arising frequently in biology. From a broad perspective we view soft computing as those techniques pertaining to systems that can best be described as a collection of processes dealing with partial information. What "partial" means depends on

the particular application. It can refer to being able to use partial knowledge of a state of affairs and to perform different kinds of guessing (bounded or unbounded non determinism, probabilistic choices, approximate answers). In this view, concurrency also belongs to this realm since it deals with partial information on the ordering of events. So does constraint programming which is based on the very idea of computing with *predicates* expressing different degrees of knowledge about variable values. Concurrent constraint (CC) process calculi [15] provides formal grounds to the integration of concurrency and constraints so that non trivial properties of concurrent systems can be expressed and proved. They are thus natural simulators to gain experience on different soft computing techniques.

We view biological phenomena at the molecular level as constructed from very complex interactions among a great number of concurrent processes acting at different biological scales.

Concurrent processes occurring in molecular biology exhibit a rich variety of synchronisation schemes, calling into play different degrees of precision (i.e. partial information) about temporal or chemical relations involving them. The complexity of biological phenomena poses a great challenge to any computational formalism. We think that a suitable CC process calculus should provide a convenient framework to get insights into the right models to cope with this challenge.

We thus borrow concepts and techniques from concurrent processes modelling to define suitable computational calculi and analyse their behaviour in real biological settings. What we gain from this *low level* approach is twofold. On the one hand, we are able to ground the development of simulation tools on a very precise formal foundation and by this means proposing coherent models of higher level biological structures and operations. On the other hand, our model can give us clues for constructing formal proofs of interesting properties of a given biological process.

Works modelling biological systems by means of process calculi have been proposed recently. Most of these works have been conducted using (extensions of) the π calculus ([12], [4]) and the Ambient calculus ([1] , [8]). Calculi devised for specific biological systems have also been proposed. For instance, calculi for modeling membranes ([7]), protein interaction ([16]) and reversibility in bio-molecular processes ([5]). We propose using a temporal non deterministic concurrent concurrent calculus (`ntcc` , see [11]) as a formal base to model timed gene activity processes in such a way that their biological properties can be formally proved. The novelty of our work is to be able to prove properties in those systems. Additionally, the notion of constraints allows us to model in a simpler way the behaviour of the genes (see section 4).

The `ntcc` calculus inherits ideas from the tcc model [14], a formalism for reactive concurrent constraint programming. In tcc time is

conceptually divided into *discrete intervals (or time-units)*. In a particular time interval, a deterministic ccp process receives a stimulus (i.e. a constraint) from the environment, it executes with this stimulus as the initial store, and when it reaches its resting point, it responds to the environment with the resulting store. Also the resting point determines a residual process, which is then executed in the next time interval.

The `ntcc` calculus is obtained from tcc by adding *guarded-choice* for modelling non-determinism and an *unbounded but finite delay* operator for asynchrony. Computation in `ntcc` progresses as in tcc, except for the non-determinism and asynchrony induced by the new constructs. The calculus allows for the specification of temporal properties, and for modelling and expressing constraints upon the environment both of which are useful in proving properties of timed systems.

However, `ntcc` does not provide stochastic constructs. These are fundamental to faithfully model aspects such as the effect of reactions on concentration of particular components, affinities or distances. We thus propose orthogonal extensions of `ntcc` to account for the stochastic behaviour of processes.

In this paper we are interested in showing how non trivial biological processes calling into action different forms of partial information can be modelled in `ntcc` extended with suitable stochastic constructs. We also investigate ways in which properties of a biological process can be formally proved. We are able to do this thanks to the logical nature of `ntcc` , which comes to the surface when we consider its relation with linear temporal logic: All the operators of `ntcc` correspond to temporal logic constructs. Since we extend `ntcc` , new linear temporal logic and proof system must also be provided. We propose both and use them to prove some properties of a gene regulation system called the lambda switch (see [3]). Our model using the stochastic extension is both simpler and more complete than the one recently proposed in [6].

The main contributions of this paper are: 1) to define an orthogonal extension adding stochastic constructs to `ntcc` , 2) to couple the extended calculus with a suitable temporal logic and proof system, 3) to show how the expressiveness of the extended `ntcc` model allows faithful and simple descriptions of complex systems of interacting biological processes, such as the lambda switch and 4) showing that by modelling a gene activities system in the extended stochastic `ntcc` one inherits a well defined logical inference system (also proposed here) that can be used to prove interesting temporal properties (or lack thereof) of the system.

2 Background

2.1 NTCC Calculus

In concurrent constraint calculi such as `ntcc` , process interactions can be determined by partial information (i.e. constraints) accumulated in a global store. The particular type of constraints is not fixed but specified in a *constraint system* that is considered a parameter of the calculus.

2.1.1 Constraint System

A constraint represents a piece of partial information over a set of variables. For example, in constraint $x > 3$, the value of x is unknown but we can assert that it is greater than 3.
A constraint system provides a signature from which constraints can be constructed. It also provides an entailment relation ($\models$) over constraints where $c_1 \models c_2$ holds iff the information of c_2 can be inferred from c_1.
Formally, a constraint system is a tuple $\langle \sum, \Delta \rangle$ where $\sum$ is a signature (i.e a set of constants, functions and predicate symbols) and Δ is a consistent first-order theory over $\sum$ (i.e a set of sentences over $\sum$ having at least one model). Constraints can be viewed as first-order formulae over $\sum$ and $c \models d$ holds if the implication $c \Rightarrow d$ is valid in Δ [11]. For practical reasons the entailment relation must be decidable.
A constraint *store* is a a set of variables and a conjunction of formulae (i.e constrains) between them. It is used to share information between process and for synchronisation purposes. The store is monotonically refined by adding information using *tell* operations of the calculus. For example, $tell(x < 2)$ adds constraint $x < 2$ to the store. Additionally, we can test if a constraint c can be entailed from the store by means of *ask* operations. For example, $ask(x < 5)$ tests whether $store \models x < 5$. The *ask* operation blocks when neither $store \models x < 5$ nor $store \models \neg(x < 5)$ holds.

2.1.2 `ntcc` Overview

`ntcc` [11] is a process calculus that extends tcc [14]. In both of them, processes share a common store of partial information [15]. Both `ntcc` and tcc have an explicit notion of (discrete) time. `ntcc` time is conceptually divided into *discrete intervals (or time-units)*. In a particular time interval, a deterministic ccp process receives a stimulus (i.e. a constraint) from the environment, it executes with this stimulus as the initial store, and when it reaches its resting point, it responds to the environment with the resulting store. Also the resting point determines a residual process, which is then executed in the next time interval.

`ntcc` has been successfully used to model many real life system such as reactive system, robot behaviour [10] and music composition [11].
Unlike tcc, `ntcc` includes constructs for modeling *nondeterminism* and *asynchrony*. A very important benefit of being able to specify non-deterministic and asynchronous behaviour arises when modelling the interaction among several components running in parallel, in which one component is part of the environment of the others. This is frequent in biological settings. These systems often need non-determinism and asynchrony to be modelled faithfully.

2.1.3 Process Syntax

In this section we describe briefly the syntax of `ntcc` processes. See [11] for further details.
`ntcc` provides the following constructors:

- **tell** : adds new information to the constraints store. For example, the process $P_1 \stackrel{def}{\equiv}$ **tell** $(c > 5)$ adds constraint $c > 5$ to the store.
- $\sum_{i \in 1..n}$ **when** c_i **do** P_i chooses non-deterministically a process P_i whose guard c_i is entailed by the store. For example, process $P_2 \stackrel{def}{\equiv}$ **when** $(c < 3)$ **do tell** $(d = 5)+$**when** $(e > 5)$ **do tell** $(d < 3)$ adds the information $d = 5$ when constraint $c < 3$ is entailed from the current store. On the other hand, if $e > 5$ is entailed, $d < 3$ is asserted. When both guards are entailed a non-deterministic choice is performed.
- Given two `ntcc` processes P and Q, process $P||Q$ represents the parallel composition between P and Q.
- **local** x **in** P behaves like P but the information of the variable x is local to P, i.e. P cannot see information about a global variable x and processes which are not part of P cannot see the information generated by P about x.
- **next** P executes process P in the next time unit (unit-delay)
- **unless** c **next** P executes P iff c cannot be entailed by the constraint store *in the current time unit*
- $!P$ executes P in all time units from the current one. It can be viewed as $P||$**next** $P||$**next next** $P||...$
- $\star P$ represents unbounded but finite delays, i.e P eventually will be executed. This process can be viewed as $P +$ **next** $P +$ **next next** P...**next** $^n P$ where n is a finite natural number.

The operators above lack of the ability to express stochastic behaviour that is quite common in biological systems. In section 3,

we propose two new operations in the calculus allowing us to express stochastic interaction between processes.

2.1.4 Rules of internal reduction

In this section we show the operational semantics of `ntcc` by giving reduction rules for each process. These rules will help us to understand how `ntcc` processes interact with each other until they reach a resting point. Recall that when this state is reached, another time units is created with an empty constraint store and the *residual* process. For a complete description of `ntcc` semantics refer to [11].
Reduction rules are based on *configurations*. A configuration $\langle P, d\rangle$ is composed of a `ntcc` process P and a store d.
For **tell** processes we have:

$$TELL \frac{}{\langle \mathbf{tell}\ c, d\rangle \rightarrow \langle \mathbf{skip}, d \wedge c\rangle}$$

where **skip** is the empty process. This reaction says that a **tell** process adds information (a constraint) to the constraint store d.

In **when** c **do** P processes the rule is as follows:

$$SUM \frac{}{\langle \sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i, d\rangle \rightarrow \langle P_j, d\rangle} if\ \ d \models c_j,\ \ j \in I$$

It means that a particular process P_j is non-deterministically chosen for execution among all those whose guard (c_i) can be entailed from the current store d.
For parallel composition we have:

$$PAR \frac{\langle P, c\rangle \rightarrow \langle P', d\rangle}{\langle P||Q, c\rangle \rightarrow \langle P'||Q, d\rangle}$$

It says that if P evolves to P', then the same transition will occur if we execute P in parallel with some process Q. Parallel composition is commutative.
For **unless** c **next** P processes:

$$UNLESS \frac{}{\langle \mathbf{unless}\ c\ \mathbf{next}\ P, d\rangle \rightarrow \langle \mathbf{skip}, d\rangle} if\ \ d \models c$$

The rule says that nothing is done when c is entailed by the store.
Finally, the rule for star processes is:

$$STAR \frac{}{\langle \star P, d\rangle \rightarrow \langle \mathbf{next}\ ^n P, d\rangle} if\ \ n \geq 0$$

It means that process P will be run in the (undetermined) future.

The above rules define so-called *internal* transitions. In addition to these, `ntcc` defines an *observable* transition which is the one that goes from one time unit to the next. At the end of a time unit the resulting store can be observed by the environment. Then, processes contained in **next** constructs are scheduled for the next time unit. This includes those defined by **unless** processes whose guard cannot be entailed from the current store (see [11] for details).

2.2 Linear Temporal Logic in `ntcc`

`ntcc` can be used to verify properties over timed systems. It provides for this a linear temporal logic in which temporal properties over infinite sequences of constraints can be stated [11]. The syntax of this logic is as follows:

$$A, B, ... : c \mid A \dot{\Rightarrow} A \mid \dot{\neg} A \mid \dot{\exists}_x A \mid \circ A \mid \Diamond A \mid \Box A$$

c (an atom) is a constraint. $\dot{\Rightarrow}$, $\dot{\neg}$ and $\dot{\exists}_x$ represent the linear-temporal logic implication, negation and existential quantification, respectively [9]. These symbols should not be confused with their counterpart in the constraint system (i.e $\Rightarrow$, $\neg$ and $\exists$). Symbols $\circ$, $\Box$ and $\Diamond$ denote the temporal operators *next*, *always* and *eventually*.

The interpretation structures of formulae in this logic are infinite sequences of states [9]. In `ntcc` , states are replaced by constraints. Given the set C of constraints in the constraint system, let $\alpha \in C^\infty$ be an infinite sequence of constraint and $\alpha(i)$ the $i-th$ element of α. We say that $\alpha \in C^\infty$ is a model of (or that it satisfies) A, notation $\alpha \models A$, if $\langle \alpha \ , \ 1 \rangle \models A$ where:

$\langle \alpha \ , \ i \rangle \models c$	*iff*	$\alpha(i) \models c$
$\langle \alpha \ , \ i \rangle \models \dot{\neg} A$	*iff*	$\langle \alpha \ , \ i \rangle \not\models \ A$
$\langle \alpha \ , \ i \rangle \models A_1 \dot{\Rightarrow} A_2$	*iff*	$\langle \alpha \ , \ i \rangle \models A_1$ *implies* $\langle \alpha \ , \ i \rangle \models A_2$
$\langle \alpha \ , \ i \rangle \models \circ A$	*iff*	$\langle \alpha \ , \ i+1 \rangle \models A$
$\langle \alpha \ , \ i \rangle \models \Box A$	*iff*	$\forall_{j \geq i} \langle \alpha \ , \ j \rangle \models A$
$\langle \alpha \ , \ i \rangle \models \Diamond A$	*iff*	$\exists_{j \geq i}$ *s.t.* $\langle \alpha \ , \ j \rangle \models A$
$\langle \alpha \ , \ i \rangle \models \dot{\exists}_x A$	*iff*	*there is an* x*-variant* α' *of* α *s.t.* $\langle \alpha' \ , \ i \rangle \models A$

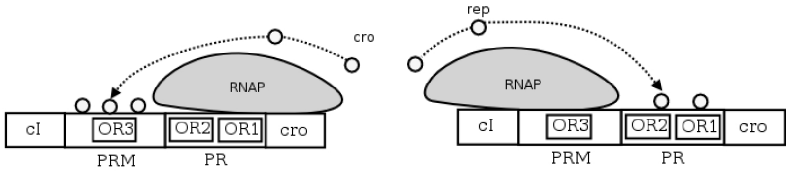
In the last expression, d and α' are $x-$variants of c and α , respectively, if they are the same except for the information about x.

In [11] a proof system is built on the top of this logic. Given a process P and a formula A, a proof of $P \models A$ can be obtained by following a set of inference rules. Nevertheless, we are interested in proving properties with probabilistic statements such as "*The concentration of component c will eventually become 0 with probability* ρ".

In section 3.3 we provide an inference system to deal with such as properties.

2.3 Lambda Switch

In this section we give a brief description of a biological system called the λ switch that we model in section 4 using our proposed extension of `ntcc` . For additional details see [6] or [3].
Bacteriophage λ is a virus that infects the *Escherichia coli* bacterium. As we will see, this biological system exhibits cooperativity relationships and non-deterministic behavior. When the virus injects its genome into the bacteria, there are two states that the bacteria can reach: (1)*lytic growth* in which the virus produces new viruses and (2) *lysogeny* in which the viral genome is passed to new generations in a passive way. The switching between states is determined by processes in a region of the virus genome called the λ switch (see figure 1). In this switch there are two promoter regions called PRM and PR where production of *rep* and *cro* proteins, respectively, takes place. The *lytic growth* state is characterised by a high concentration of *cro* proteins whereas *lysogeny* is characterised by a high concentration of *rep*.



(a) Lytic growth (b) Lysogeny

Figure 1: Lambda switch states

Promoters are overlapped by three regions (binding sites) called $OR1$, $OR2$ and $OR3$. Region $OR1$ exhibits a high affinity for *rep* and a low affinity for *cro*. On the other hand, $OR3$ exhibits a high affinity for *cro* and a low affinity for *rep*.

In *lysogeny*, *rep* proteins usually bind $OR1$ and $OR2$. When $OR1$ is bound by *rep*, $OR2$ affinity for *rep* increases. This is a cooperation relation between bindings at different sites. On the other hand, $OR3$ and the promoter PRM are usually vacant but eventually bound by the polymerase $RNAP$. When this occurs, the transcription of the gene cI starts and new instances of *rep* proteins are produced. When $OR1$ is not vacant, binding of $RNAP$ to PR is inhibited, stopping in this way the production of *cro*. Another cooperation relationship is

present in the lambda switch: since PRM is a weak promoter, when $OR2$ is bound by *rep*, *rep* cooperates with $RNAP$ and more frequent transcriptions of gene cI happen. It implies that more *rep* proteins are produced thus increasing the chance of maintaining the *lysogeny* state. *Lysogeny* state is maintained until an environmental signal turns the switch to the *lytic growth*. This process is called *Induction* and it occurs with very low probability. In this state the concentration of *rep* decreases dramatically and then $RNAP$ has the chance to bind to $OR1$, that is now vacant most of the time. PR, a stronger promoter than PRM, starts the transcription of gene *cro* and thus new instances of *cro* are produced. Because $OR3$ has a high affinity for *cro*, it is bound by this protein avoiding $RNAP$ bindings to PRM, inhibiting in this way production of *rep* proteins.
This biological system has been successfully modelled in [6] by using the π calculus [13] with stochastic behaviour [12]. However, we believe that notions such as cooperation and the effect of distance (i.e reactions may occur depending on the distance between molecules) can be expressed in a more straightforward manner by using the notion of constraint. We also introduce some additional details that were left aside in the model in [6].

3 Introducing Stochastic Behaviour in `ntcc`

In this section we introduce two new operators in `ntcc` for modelling stochastic behaviour. We also propose an extension of the `ntcc` linear-temporal logic that can be used to prove properties involving probabilistic statements.

3.1 ${}_\rho P$

Informally, this construct allows executing processes (i.e P) according to a given probability $\rho \in [0, 1]$. For example, if we observe 100 time-units generated from the process: $!_\rho P$, we will observe $100 \times \rho$ times the execution of P.

The inclusion of this construct in the calculus can be orthogonally done by adding a probabilistic function $\Phi : \mathsf{R} \in [0, 1] \rightarrow \mathit{Bool}$ into the constraint systems. Φ is computed by generating pseudo-random numbers following a binomial distribution with probability ρ and 1 as number of events. If $\Phi(\rho) = 1$ we say that the process will be executed, otherwise it will not.
${}_\rho P$ can be expressed in terms of the rest of the standard constructs of `ntcc` as follows:

$$ {}_\rho P \equiv \textbf{local } x \textbf{ in tell } (x = \Phi(\rho)) || \textbf{when } x = 1 \textbf{ do } P \qquad (1) $$

Notice that P will be activated only if $\Phi(\rho) = 1$ as is described by rule $RHOP$:

$$RHOP \frac{}{\langle {}_\rho P, d\rangle \rightarrow \langle P, d\rangle} if\ \Phi(\rho) = 1$$

This construct will be useful to model events such as rates of gene transcription. For example, in the lambda switch, when $RNAP$ binds PRM, one will observe the production of a new *rep* protein. Nevertheless, this transcription depends on the cooperativity relationship between $RNAP$ and $OR2$. In this way, we will observe the increase of *rep* with some probability ρ_1 when there is no *rep*s proteins binding $OR2$ and with some other probability ρ_2 otherwise, having $\rho_2 > \rho_1$ (see section 4.3 for further details).

3.2 $(\star_\rho)P$ Processes

Star processes are used to express eventually in `ntcc` . However, we should be able to differentiate between two processes that eventually occur with different probabilities. For example, the *induction* process in the lambda switch occurs eventually but with a very low probability, while bindings between *rep* and region $OR3$ are quite frequent in *lysogeny*. We propose the constructor $(\star_\rho)P$ in which P eventually occurs with probability ρ. Operationally:

$$STAR_\rho \frac{}{\langle \star_\rho P, d\rangle \rightarrow \langle \mathbf{next}^n\ {}_\rho P, d\rangle} if\ n \geq 0$$

It is easy to verify that this process can be viewed as a ${}_\rho P$ process:

$$(\star_\rho)P \stackrel{def}{\equiv} \mathbf{local}\ x\ \mathbf{in\ tell}\ (x = \Phi(\rho))||\mathbf{when}\ x = 1\ \mathbf{do}\ \star P$$

and that commutativity holds, i.e., $(\star_\rho)P \equiv \star(\rho P)$

3.3 Stochastic parameters in the linear-temporal logic

To prove properties such as "*The concentration of rep will eventually become 0 with probability* ρ" we change the structure of formulae by adding probabilities, i.e formulae will be tuples $\langle A\ ,\ \rho\rangle$ where A is a formula in the `ntcc` lineal-temporal logic. The syntax is:

$$A', B', ... = \langle c, \rho\rangle \mid A' \stackrel{\circ}{\Rightarrow} B' \mid \stackrel{\circ}{\neg} A' \mid \stackrel{\circ}{\exists}_x A' \mid \circ A',\ \mid \Diamond A' \mid \Diamond A'$$

Probability ρ should not be confused with some notion of *degree of validity* of the formula. This probability refers to the occurrence of events (time). For example, the formula $\langle c < 3, 0.5\rangle$ expresses that

with a probability of 0.5 constraint $c < 3$ will be asserted.
The semantics of our new logic is as follows:

$$
\begin{array}{lll}
\langle \alpha\ ,\ i\rangle \models \langle c\ ,\ \rho\rangle & \textit{iff} & \alpha(i) \models \langle c\ ,\ \rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \overset{\circ}{\neg}\langle A\ ,\ \rho\rangle & \textit{iff} & \langle \alpha\ ,\ i\rangle \not\models\ \langle A\ ,\ 1-\rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \langle A_1\ ,\ \rho_1\rangle \overset{\circ}{\Rightarrow} \langle A_2\ ,\ \rho_2\rangle & \textit{iff} & \langle \alpha\ ,\ i\rangle \models \langle A_1\ ,\ \rho_1\rangle \textit{ implies} \\
 & & \langle \alpha\ ,\ i\rangle \models \langle A_2\ ,\ \rho_2\rangle \\
\langle \alpha\ ,\ i\rangle \models \circ\langle A\ ,\ \rho\rangle & \textit{iff} & \langle \alpha\ ,\ i+1\rangle \models \langle A\ ,\ \rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \Box\langle A\ ,\ \rho\rangle & \textit{iff} & \forall_{j\geq i}\langle \alpha\ ,\ j\rangle \models \langle A\ ,\ \rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \Diamond\langle A\ ,\ \rho\rangle & \textit{iff} & \exists_{j\geq i}\ s.t. \langle \alpha\ ,\ j\rangle \models \langle A\ ,\ \rho\rangle \\
\langle \alpha\ ,\ i\rangle \models \overset{\circ}{\exists}_x\langle A\ ,\ \rho\rangle & \textit{iff} & \textit{there is an } x-\textit{variant} \\
 & & \alpha' \textit{ of } \alpha \textit{ s.t. } \langle \alpha'\ ,\ i\rangle \models \langle A\ ,\ \rho\rangle
\end{array}
$$

where $\alpha(i) \models \langle c\ ,\ \rho\rangle$ is defined as: $\langle c_1\ ,\ \rho_1\rangle \models \langle c_2\ ,\ \rho_2\rangle$ iff c_1 entails c_2 in the constraint system and $\rho_2 \leq \rho_1$.

Since formulae such as $A' \overset{\circ}{\Rightarrow} B'$ are not in the expected form $\langle A, \rho\rangle$, we can express equivalences between linear-temporal implication and negation operators in our new logic and their counterpart in `ntcc` logic by using properties of probability theory:

$$
\begin{array}{c}
\overset{\circ}{\neg}\langle A, \rho\rangle = \langle \dot{\neg} A, 1-\rho\rangle \\
\langle A, \rho_1\rangle \overset{\circ}{\wedge} \langle B, \rho_2\rangle = \langle A \dot{\wedge} B, \rho_1 \times \rho_2\rangle \\
\langle A, \rho_1\rangle \overset{\circ}{\Rightarrow} \langle B, \rho_2\rangle = \langle A \dot{\wedge} B, 1-\rho_1+\rho_1 \times \rho_2\rangle
\end{array}
$$

Third expression is obtained by converting $\langle A, \rho_1\rangle \overset{\circ}{\Rightarrow} \langle B, \rho_2\rangle$ into $\overset{\circ}{\neg}(\langle A, \rho_1\rangle \overset{\circ}{\wedge} \overset{\circ}{\neg}\langle B, \rho_2)$ and then applying two first rules.

3.3.1 Inference System

We extend the inference system proposed in [11] with inferences rules taking into account the new form of formulae and the probabilistic operators ${}_\rho P$ and $(\star_\rho)P$. The rules permit build a proof if some process P satisfies some formula $\langle A, \rho\rangle$ in the logic, i.e, if $P \vdash \langle A, \rho\rangle$

$$LTELL : \textbf{tell}\ c \vdash \langle c\ ,\ 1.0\rangle \tag{2}$$

$$LPAR : \frac{P \vdash \langle A\ ,\ \rho\rangle \quad Q \vdash \langle B\ ,\ \rho_2\rangle}{P||Q \vdash \langle A\ ,\ \rho\rangle \overset{\circ}{\wedge} \langle B\ ,\ \rho_2\rangle} \tag{3}$$

i.e. the parallel execution of two process satisfies the conjunction of the formulae of each process.

$$LLOC : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\textbf{local}\ x\ \textbf{in}\ P \vdash \overset{\circ}{\exists}_x\langle A\ ,\ \rho\rangle} \tag{4}$$

$$LNEXT : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\textbf{next}\ P \vdash \circ\langle A\ ,\ \rho\rangle} \tag{5}$$

$$LUNL : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\textbf{unless } c \textbf{ next } P \vdash \langle c\ ,\ 1.0\rangle \overset{\circ}{\vee} \circ\langle A\ ,\ \rho\rangle} \tag{6}$$

$$LCONS : \frac{P \vdash \langle A\ ,\ \rho\rangle}{P \vdash \langle B\ ,\ \rho_2\rangle}\ if\ \langle A\ ,\ \rho\rangle \overset{\circ}{\Rightarrow} \langle B\ ,\ \rho_2\rangle \tag{7}$$

$$LSTAR : \frac{P \vdash \langle A\ ,\ \rho\rangle}{\star P \vdash \Diamond\langle A\ ,\ \rho\rangle} \tag{8}$$

$$LSUM : \frac{\forall i \in I \quad P_i \vdash \langle A_i\ ,\ \rho_i\rangle}{\sum_{i\in I} \textbf{when } c_i \textbf{ do } P_i \vdash \overset{\circ}{\bigvee}_{i\in I}(\langle c_i\ ,\ 1\rangle \overset{\circ}{\wedge} \langle A_i\ ,\ \rho_i\rangle) \overset{\circ}{\vee} \overset{\circ}{\bigwedge}_{i\in I} \overset{\circ}{\neg}\langle c_1\ ,\ 1.0\rangle} \tag{9}$$

$$PPRO : \frac{P \vdash \langle A\ ,\ \rho\rangle}{{}_{\rho_2}P \vdash \langle A\ ,\ \rho \times \rho_2\rangle} \tag{10}$$

The previous equation represents the inference rule for ρP processes. Since the probability ρ and the probability ρ_2 are independent, the probability of both occurrences is $\rho \times \rho_2$.
Finally, the rule for the $(\star_\rho)P$ process is:

$$PSTAR : \frac{P \vdash \langle A\ ,\ \rho\rangle}{{}_{\star_{\rho_2}}P \vdash \Diamond\langle A\ ,\ \rho \times \rho_2\rangle} \tag{11}$$

4 Modeling the λ-Switch with the stochastic `ntcc` extension

Recall that our objective is to use concurrent calculi to faithfully model biological systems. We test the appropriateness for this task by constructing a (somewhat) detailed model of the lambda-switch using the extended calculus defined above. We use *process definition* constructs of the form $\textbf{PROCESS}(\mathbf{x}) \overset{def}{\equiv} P$ that do not formally belong to the calculus. These, however, can be easily defined in terms of the standard calculus constructs (see [11]).

4.1 REP and CRO Protein Control

In our model production of *rep* proteins is controlled by 3 extended calculus processes: (1) Induction, that reduces the concentration of *rep* thus switching the system to a *lytic growth* state, (2) a process supervising that the concentration of *rep* does not exceed a given threshold and (3) a process that increases the concentration of *rep* as a result of the expression of gene *cI*.

The induction process (see section 2.3), activated by environment signals that are at present little understood, causes a strong reduction of **rep** proteins and therefore switching to a *lytic growth* state. The occurrence probability for this process is very low. By using $(\star_\rho)P$ processes we can model this fact as:

$$\mathbf{INDUCTION}(\rho_{\mathbf{ind}}) \stackrel{def}{\equiv} \star_{\rho_{ind}}(\mathbf{tell}\ reset\_rep_c) \tag{12}$$

Eventually, under the probability ρ_{ind}, this process reduces the concentration of *rep* to 0 (see $reset\_rep_c$ in equation 19).
The following process avoids a negative feedback by controlling the concentration of *rep* proteins. This represents the situation in which there are so many *rep* proteins floating around that even $OR3$ will get bound to *rep* thus inhibiting gene *cI* expression. The process inhibits the production of *rep* when concentration reaches $rep_{threshold}$.

$$\begin{array}{l}\mathbf{REPTHOLD}(\mathbf{rep_{threshold}}) \stackrel{def}{\equiv} \\ \mathbf{when}\ C_{rep} > rep_{threshold}\ \mathbf{do\ next\ tell}\ inhibit_{rep}\end{array} \tag{13}$$

4.2 Gene transcription

As mentioned before, PRM is a weaker promoter than PR . This means that production of *rep* takes more time than production of *cro* once $RNAP$ binds to PRM (resp. PR). Sometimes no *rep* protein is produced at all when the binding occurs, i.e $RNAP$ falls off without transcribing gene *cI*.

We model the gene transcription at the PR promoter as follows:

$$\mathbf{CROtrans} \stackrel{def}{\equiv} \mathbf{when}\ pr_{bound}\ \mathbf{do}\ _{\rho_{crotrans}}(\mathbf{tell}\ inc\_cro_c) \tag{14}$$

where predicate pr_{bound} is true iff $RNAP$ is binding PR . Constraint inc_cro_c causes a new (higher) value for concentration of *cro* proteins to be asserted in the next time unit (see equation 19).
Modeling gene transcription at PRM is more difficult because the probability of transcription may vary according to the presence or absence of *rep* at $OR2$. When $OR2$ is bound by *rep* , the probability of gene *cI* transcription at PRM is higher and in consequence higher is also the probability of producing more *reps*. Equation 15 models gene transcription at PRM :

$$\begin{array}{c}\mathbf{CItrans}(\rho_{high}, \rho_{low}) \stackrel{def}{\equiv} \mathbf{when}\ \neg inhibit_{rep} \wedge prm_{bound}\ \mathbf{do\ local}\ p\ \mathbf{in} \\ (\mathbf{when}\ or2\_bound\_rep\ \mathbf{do\ tell}\ p = \rho_{high} \\ + \mathbf{when}\ or2\_bound\_cro \vee or2\_vacant\ \mathbf{do}\ p = \rho_{low}) \\ || \quad _p(\mathbf{tell}\ inc\_rep_c)\end{array} \tag{15}$$

where cooperativity is modelled by means a **when** c **do** P process that chooses between ρ_{high} and ρ_{low} in order to execute **tell** $inc(rep_c)$

with the right probability. Notice that synchronisation is guaranteed by the use of constraints: process ${}_p(\textbf{tell}\ inc\_rep_c)$ blocks until the value of p is known.

4.3 Operator Regions

In what follows we model the behaviour of $OR1, OR2$ and $OR3$ binding sites. Since each operator has different affinities for *rep* and *cro* , ${}_\rho P$ type processes are needed. Additionally, we have to take in account cooperativity relationship between $OR1$ and $OR2$: $OR2$ increases its affinity for *rep* when $OR1$ is bound by *rep* . The following equations model the operator regions in the lambda switch:

$$\begin{array}{l}
\textbf{OR1} \stackrel{def}{\equiv} \\
\quad \textbf{when}\ or1_unbound\ \textbf{do} \\
\qquad\quad \textbf{when}\ rep_c > 0\ \textbf{do}\ {}_{\rho_{or1rep}}(\textbf{next tell}\ (or1_rep_bound \wedge dec_rep_c)) \\
\qquad\quad + \textbf{when}\ cro_c > 0\ \textbf{do}\ {}_{\rho_{or1cro}}(\textbf{next tell}\ (or1_cro_bound \wedge dec_cro_c)) \\
\quad + \textbf{when}\ or1_rep_bound\ \textbf{do}\ {}_{1.0-\rho_{or1rep}}(\textbf{next tell}\ (or1_unbound \wedge inc_rep_c)) \\
\quad + \textbf{when}\ or1_cro_bound\ \textbf{do}\ {}_{1.0-\rho_{or1cro}}(\textbf{next tell}\ (or1_unbound \wedge inc_cro_c))
\end{array} \tag{16}$$

$$\begin{array}{l}
\textbf{OR2} \stackrel{def}{\equiv} \textbf{local}\ or2rep\ \textbf{in} \\
\quad \textbf{when}\ or2_unbound\ \textbf{do} \\
\qquad\quad \textbf{when}\ rep_c > 0\ \textbf{do}\ {}_{\rho_{or2rep}}(\textbf{next tell}\ (or2_rep_bound \wedge dec_rep_c)) \\
\qquad\quad + \textbf{when}\ cro_c > 0\ \textbf{do}\ {}_{\rho_{or2cro}}(\textbf{next tell}\ (or2_cro_bound \wedge dec_cro_c)) \\
\qquad\quad + \textbf{when}\ \neg or1_bound_rep\ \textbf{do tell}\ (\rho or2rep = \rho or2rep_low) \\
\quad + \textbf{when}\ or2_rep_bound\ \textbf{do}\ {}_{1.0-\rho_{or2rep}}(\textbf{next tell}\ (or2_unbound \wedge inc_rep_c)) \\
\quad + \textbf{when}\ or2_cro_bound\ \textbf{do}\ {}_{1.0-\rho_{or2cro}}(\textbf{next tell}\ (or2_unbound \wedge inc_cro_c)) \\
\|(\textbf{when}\ or1_bound_rep\ \textbf{do tell}\ (\rho or2rep = \rho or2rep_high)
\end{array} \tag{17}$$

$$\begin{array}{l}
\textbf{OR3} \stackrel{def}{\equiv} \\
\textbf{when}\ or3_unbound\ \textbf{do} \\
\qquad\quad \textbf{when}\ rep_c > 0\ \textbf{do}\ {}_{\rho_{or3rep}}(\textbf{next tell}\ (or3_rep_bound \wedge dec(rep_c)) \\
\qquad\quad + \textbf{when}\ cro_c > 0\ \textbf{do}\ {}_{\rho_{or3cro}}(\textbf{next tell}\ (or3_cro_bound \wedge dec(cro_c)) \\
\quad + \textbf{when}\ or1_rep_bound\ \textbf{do}\ {}_{1.0-\rho_{or3rep}}(\textbf{next tell}\ (or3_unbound \wedge inc(rep_c)) \\
\quad + \textbf{when}\ or1_cro_bound\ \textbf{do}\ {}_{1.0-\rho_{or3cro}}(\textbf{next tell}\ (or3_unbound \wedge inc(cro_c))
\end{array} \tag{18}$$

where $\rho or1rep$, $\rho or1cro$, $\rho or2rep\_high$, $\rho or2rep\_low$, $\rho or3rep$ and $\rho or3cro$ are the probabilities (affinities) of each binding site w.r.t *rep* and *cro*. Constraints dec_rep_c and dec_cro_c will cause decrementing of the concentration of *rep* (resp. *cro*) in the environment (see equation 19). In the above processes, when the operator is unbound (i.e vacant) and *reps* or *cros* are available in the environment, eventually the protein (i.e *rep* or *cro*) binds the operator and consequently also decreases the protein concentration in the environment.

Finally, equation 19 controls the concentration of *rep* and *cro* for the next time unit according to signal controls (constraints) posted by pre-

vious processes (e.g. inc_rep_c):

$$\begin{array}{l}\textbf{CONCCTR} \stackrel{def}{\equiv} \\ (\textbf{when } reset\_rep_c \textbf{ do next tell } rep_c = 0+ \\ \quad \textbf{when } inc\_rep_c \wedge \neg reset\_rep_c \textbf{ do next tell } (rep_c = rep'_c + 1)+ \\ \quad \textbf{when } dec\_rep_c \wedge \neg reset\_rep_c \textbf{ do next tell } (rep_c = rep'_c - 1))|| \\ (\textbf{when } inc\_cro_c \textbf{ do next tell } (cro_c = cro'_c + 1)+ \\ \quad \textbf{when } dec\_cro_c \textbf{ do next tell } (cro_c = cro'_c - 1))\end{array} \tag{19}$$

In previous equation, variables with prime symbol (e.g. cro'_c) are used to denote the state of the variables in the previous time-unit. This is not part of the `ntcc` calculus syntax but it can be modeled by means of a process making "*persistent*"the state of the variables between two time-units . This process can be defined as follows [2]:

$$State_i(v_i) \quad \stackrel{def}{\equiv} \quad \textbf{tell } m'_i = v_i || \textbf{next } State_i(m_i) \tag{20}$$

where v_i represents the value of the variable in the first time-unit and m'_i is the value of the variable m_i in the previous time-unit.

4.4 RNAP binding

Equations 14 and 15 depend on bindings between $RNAP$ and promoters PR and PRM , respectively. The behaviour of $RNAP$ has two components: (1) when $RNAP$ is binding PR (resp PRM), there is a probability of $1.0 - \rho rnap\_pr$ (resp. $1.0 - \rho rnap\_prm$) of falling off thus interrupting gene transcription. And (2) when promoters are unbound, $RNAP$ may bind to PR (resp PRM) if $OR1$ (resp. $OR3$) is vacant, with probability $rnap\_pr$ (resp. $rnap\_prm$). Equation 21 models this fact:

$$\begin{array}{l}\textbf{RNAPCTR} \stackrel{def}{\equiv} \\ \textbf{when } rnap\_unbound \textbf{ do } (\\ \textbf{when } or1\_unbound \textbf{ do } {}_{rnap\_pr}(\textbf{next tell } (or1\_rnap\_bound \wedge pr\_bound)) \\ \quad + \textbf{when } or3\_unbound \textbf{ do } {}_{rnap\_prm}(\textbf{next tell } (or3\_rnap\_bound \wedge prm\_bound)) \\ + \textbf{when } rnap\_bound \textbf{ do } (\\ \quad + \textbf{when } or1\_rnap\_bound \textbf{ do } {}_{1.0-rnap\_pr}(\textbf{next tell } or1\_unbound) \\ \quad + \textbf{when } or3\_rnap\_bound \textbf{ do } {}_{1.0-rnap\_prm}(\textbf{next tell } or3\_unbound))\end{array} \tag{21}$$

Using parallel composition among equations 12 to 21 we can de-

scribe the overall lambda-switch system as follows:

$$\begin{array}{l}\lambda - \mathbf{PROC}(\rho\_ind, rep\_thr, \rho or1rep, \rho or2rep\_high, \rho or2rep\_low, \rho or3rep, \\ \qquad \rho or1cro, \rho or2cro, \rho or3cro, \rho crotrans, \rho citrans\_high, \rho citrans\_low) \stackrel{def}{\equiv} \\ \quad \mathbf{local}\ rep_c(0), cro_c(0), pr_{bound}, prm_{bound} inhibit_{rep} \\ \quad\quad or1\_bound\_rep, or1\_bound\_cro, or2\_bound\_rep, or2\_bound\_cro, \\ \quad\quad or2\_bound\_rep, or2\_bound\_cro, or1\_unbound, or2\_unbound, or3\_unbound, \\ \quad\quad rnap\_bound, rnap\_unbound, or1\_bound\_rnap, or2\_bound\_rnap, \\ \quad\quad or3\_bound\_rnap, \rho_{or1rep}, \rho_{or2rep}, \rho_{or3rep}, \rho_{or1cro}, \rho_{or2cro}, \rho_{or3cro} \\ \quad \mathbf{in} \\ \quad\quad INDUCTION(\rho\_ind)||!REPTHOLD(rep\_thr)||!CROtrans||!CItrans|| \\ \quad\quad !OR1||!OR2||!OR3||!RNAPCTR||!CONCCTR\end{array} \tag{22}$$

This equation defines the stochastic parameters of the model and then *executes* concurrently the processes needed to control the behavior of the lambda switch.

4.5 Simulating Biological Systems

The sections above showed how we can model a biological system by means of constructs in the `ntcc` calculus. This models come from a functional description of the system and they are easily mapped to process definition describing the expected behavior. In this way, models must be viewed as *runnable specification* since we can obtain the values of the variables in each time-unit representing in our case of study, changes in the concentration of proteins during the time.
To accomplish this task, we developed a simulator for our calculus. This was built in the top of the Oz system (`www.mozart-oz.org`), a multi-paradigm programming language including constraint-based libraries and support for concurrent programming. This simulator takes as input process definition such as those presented in above equation and returns the stores in each time-unit. The store contains the value of each variable obtaining in this way quantitative measures of the system.
As example , we show the definition of *INDUCTION* process in figure 2. Constructs in the calculus are defined by means of Oz records. For example $rep()$ represents replication (!) operator, $rho(P)$ stands for ${}_\rho P$ and $IndRho$ is the probability of induction process. For readers non familiarised with Oz syntax, $proc\{\$Root\}...end$ is the standard mechanism to define procedures that can be injected as new constraints in the constraint store. This mechanism is used by *tell* processes. In this way, all the processes can be defined into the simulator. Next, a procedure takes as input all processes definition and generates a given number of time-units respecting the operational semantic of the calculus.

Stores in each time-unit can be used to plot for example concentration of *rep* and *cro* proteins as showed in figure 3. Notice that *rep*

```
Induction = rep(rho(tell(proc{$ Root} Root.reset_rep =:1
end) IndRho))
```

Figure 2: Induction process definition in the simulator

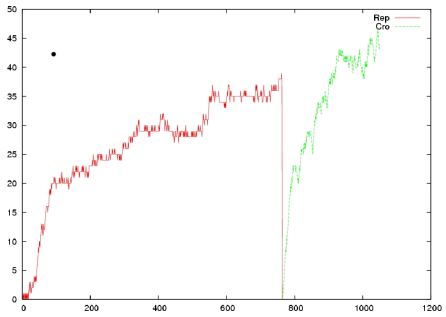


Figure 3: Rep and Cro proteins concentrations.

concentration growths at the beginning while *cro* concentration remains around zero. After induction, the system behaves in a opposite way.

Tools such this will allow biologist to observe the behaviour of the system and study how the system evolves if one change some parameters such as affinities between components (i.e probabilities).

4.6 Proving temporal properties of the λ switch system

Simulators can be useful to observe the system in an finite interval of time, but they are not enough to predict behaviour in the future. An advantage of using process calculi such as `ntcc` with a well defined semantic operation and an underlying logic, is that we can reason about the system by means of an inference systems. Proofs will allow us to check our model and establish if we can expect some behaviour in the system or not. In this section we show proofs for two properties in the lambda switch system:

- *Eventually, with probability ρ (a very low probability in this case) the concentration of rep proteins drops to zero .*
- *If $OR1$ and $OR2$ are bound by reps and $OR3$ is bound by $RNAP$, a new instance of rep will eventually be produced (i.e. the protein concentration will be incremented) with probability ρ_{ci_high}.* Re-

call that when $OR2$ is bound by rep, the rate of transcription of gene cI is incremented because of the cooperativity relationship between $OR2$ and PRM .

The first one shows how induction process will be observed in the future. Since induction process has a very low probability, it is quiet difficult to observe this phenomenon in a simulation. It may be necessary run the simulation too many times or generate too many time-units. In fact, figure 3 was generated increasing the probability of the induction phenomenon. The second one shows how it is possible to perform model checking over our models and proof, in this case, if some property of the system (cooperativity) can be deduced from the model.

In order to proof the first property, we star with the definition of the overall system:

$$LPAR : \frac{INDUCTION \vdash \langle A\ ,\ \rho\rangle\ OR1 \vdash \langle A_2\ ,\ \rho_2\rangle\ ...\ RNAPCTR \vdash \langle A_n\ ,\ \rho_n\rangle}{\lambda - PROC \vdash \langle A\ ,\ \rho\rangle \overset{\circ}{\wedge} \Box\langle A_2\ ,\ \rho_2\rangle ... \Box \overset{\circ}{\wedge} \langle A_n\ ,\ \rho_n\rangle} \tag{23}$$

Notice that we use the temporal $always$ operator because processes are replicated (i.e. use the "!" prefix) in the definition of $\lambda - PROC$.

By using $LCONS$ in equation 23 we get:

$$LPAR : \frac{}{\lambda - PROC \vdash \langle A\ ,\ \rho\rangle} \tag{24}$$

The $INDUCTION$ process satisfies the formula $\langle A\ ,\ \rho\rangle$. Now, we are going to find out the structure of A:

$$PSTAR : \frac{\mathbf{tell}\ reset(rep_c) \vdash \langle reset(rep_c)\ ,\ 1.0\rangle}{\star_{\rho\_ind}\mathbf{tell}\ reset(rep_c)||INDUCTION \vdash \Diamond\langle reset(rep_c)\ ,\ \rho\_ind\rangle} \tag{25}$$

As $INDUCTION \overset{def}{\equiv} \star_{\rho\_ind}\mathbf{tell}\ reset(rep_c)||INDUCTION$ (omitting the local hiding), we can affirm that $\lambda - PROC$ satisfies the property "*eventually under a probability* ρ_ind the concentration of rep will be zero" , i.e, $\Diamond\langle reset(rep_c)\ ,\ \rho\_ind\rangle$.

To prove the second property we star from the definition of $\lambda - PROC$ and then use the definitions of $CItrans$ and $RNAPCTR$:

$$LPAR : \frac{RNAPCTR \vdash \langle A\ ,\ \rho\rangle \quad CITRANS \vdash \langle A_2\ ,\ \rho_2\rangle\ ...}{\lambda - PROC \vdash \Box\langle A\ ,\ \rho\rangle \overset{\circ}{\wedge} \Box\langle A_2\ ,\ \rho_2\rangle ...} \tag{26}$$

Using the definition of $CITRANS$ we get:

$$LSUM: \frac{\textbf{tell}\ p = \rho\_high \vdash \langle p = \rho\_high\ ,\ 1.0\rangle\ \textbf{tell}\ p = \rho\_low \vdash \langle p = \rho\_low\ ,\ 1.0\rangle}{\begin{array}{c}\textbf{when}\ or2\_bound\_rep\ \textbf{do tell}\ p = \rho\_high+\\ \textbf{when}\ or2\_vacant\ \textbf{do tell}\ p = \rho\_low \vdash\\ \langle or2\_bound\_rep \dot{\wedge} p = \rho\_high\ ,\ 1.0\rangle \mathring{\vee} \langle or2\_vacant \dot{\wedge} p = \rho\_low\ ,\ 1.0\rangle \mathring{\vee}\\ \langle \dot{\neg} or2\_bound\_rep \dot{\wedge} \dot{\neg} or2\_vacant\ ,\ 1.0\rangle\end{array}} \tag{27}$$

Since the premise is the formula $\langle or1\_bound\_rep \dot{\wedge} or2\_bound\_rep \dot{\wedge} or3\_bound\_rnap\ ,\ 1.0\rangle$ we can use $LCONS$ as follows:

$$LCONS: \frac{\langle or1\_bound\_rep \dot{\wedge} or2\_bound\_rep \dot{\wedge} or3\_bound\_rnap\ ,\ 1.0\rangle}{\begin{array}{c}\textbf{when}\ or2\_bound\_rep\ \textbf{do tell}\ p = \rho\_high+\\ \textbf{when}\ or2\_vacant\ \textbf{do tell}\ p = \rho\_low \vdash\\ \langle p = \rho\_high\ ,\ 1.0\rangle\end{array}} \tag{28}$$

Finally, by using $PPRO$ we can verify that $CITRANS$ satisfies the following formula:

$$\frac{\begin{array}{c}\textbf{when}\ or2\_bound\_rep\ \textbf{do tell}\ p = \rho\_high+\\ \textbf{when}\ or2\_vacant\ \textbf{do tell}\ p = \rho\_low \vdash \langle p = \rho\_high\ ,\ 1.0\rangle\end{array}}{\begin{array}{c}\textbf{when}\ \ldots\ \textbf{do}\ \ldots + \textbf{when}\ \ldots\ \textbf{do}\ \ldots ||\\ {}_{p}\_high(\textbf{tell}\ inc(rep_c)) \vdash \langle inc(rep_c)\ ,\ \rho\_high\rangle\end{array}} \tag{29}$$

Equation 29 says that new *rep*s will appear with a probability ρ_high verifying the cooperative behavior between $OR2$ and PR .

5 Concluding remarks and Future Work

We have given orthogonal extensions of `ntcc` for modelling stochastic behaviour of processes. In particular, we proposed two new operators: ${}_{\rho}P$ and $(\star_{\rho})P$. The first one expresses that P is executed in the current time unit with probability ρ and the second one that P is eventually executed with probability ρ. We showed that both can be expressed in terms of existing `ntcc` constructs by including in the signature of the underlying constraint system of `ntcc` a probabilistic function $\Phi(\rho) : [0, 1] \rightarrow bool$ following a binomial distribution. Additionally, an inference system to prove probabilistic properties in the calculus was provided. This inference system is built on an `ntcc` linear-temporal logic extension by adding probabilities to each formula. For each process construction including our new processes $(\star_{\rho})P$ and ${}_{\rho}P$, we defined an inference rule to proof if a process P satisfies an specification (i.e a formula in the logic).

Using this new stochastic non-deterministic calculus we were able to provide a model of a biological system called the lambda switch that is both simpler and more complete than models previously proposed based on the stochastic π-calculus. This approach has several advantages: 1)The notion of constraints allows to define relation between components in a declarative way leading to a straightforward

representation of the functional description of the system. 2) Notion of partial information in the constraint store allows to represent the system even though we do not have a complete knowledge of all the components and relations involved. 3) Thanks to the parallel composition operator, the system is described in terms of well defined processes running concurrently. 4) Models in this calculus can be simulated following the operational semantic of the calculus. Finally, 5) We can build a proof with the inference system provided to know if some behaviour can be observed in the system without simulating the system infinitely.
As future work, we expect to use our proof system to verify properties of relevance to biologists. Adapting or implementing a new automatic theorem-prover will help us in this task. We also plan to construct reasonably complete models of some other biological systems.

References

[1] Regev A and Panina E. Bioambients: An abstraction for biological compartments. In *Theoretical Computer Science 325(1)*, 2003.

[2] Alejandro Arbelez, Julian Gutirrez, Carlos Olarte, and Camilo Rueda. A generic framework to model, simulate and verify genetic regulatory networks. In *XXXII Conferencia Latinoamericana de Informática, CLEI 2006*, 2006.

[3] Adam Arkin, John Rossb, and Harley H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. In *Genetics*, 1998.

[4] Ciobanu G. and Ciubotariu V. A pi-calculus model of the na pump. In *Genome Informatics: 469-472*, 2002.

[5] Krivine J. and Danos V. Formal molecular biology done in ccs-r. In *BioConcur 2003 - Workshop on Concurrent Models in Molecular Biology*, 2003.

[6] Celine Kuttler, Joachim Niehren, and Ralf Blossey. Gene regulation in the pi calculus: Simulating cooperativity at the lambda switch. In *Bio-CONCUR 2004*, 2004.

[7] Cardelli L. Brane calculi. In *CMSB - Lecture Notes in Computer Science*, 2004.

[8] Cardelli L. and Gordon A. Mobile ambients. In *FoSSaCS - Lecture Notes in Computer Science*, 1998.

[9] Z. Manna and A. Penueli. *The Temporal Logic of Reactive and Concurrent Systems, Sepecification*. Springer, 1991.

[10] Pilar Munoz and Andres Hurtado. Programming robot devices with a timed concurrent constraint programming. In *Principles*

and Practice of Constraint Programming - CP2004. LNCS 3258, page 803. Springer, 2004.

[11] Mogens Nielsen, Catuscia Palamidessi, and Frank D. Valencia. Temporal concurrent constraint programming: Denotation, logic and applications. In *Special Issue of Selected Papers from EXPRESS'01, Nordic Journal of Computing*, 2001.

[12] C. Priami. Stochastic pi-calculus. In *Computer Journal*, 2004.

[13] J. Parrow R. Milner and D. Walker. A calculus of mobile processes, Parts I and II. *Journal of Information and Computation*, 100:1–77, September 1992.

[14] V. Saraswat, R. Jagadeesan, and V. Gupta. Fundation of timed concurrent constraint programming. In *IEEE Symposium on Logic in Computer Science*. IEEE press, 1994.

[15] V. A. Saraswat. *Concurrent Constraint Programming*. The MIT Press, Cambridge, MA, 1993.

[16] Danos V. and Laneve C. Formal molecular biology. In *Theoretical Computer Science 325(1)*, 2004.