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] provide 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 \textit{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 \textit{guarded-choice} for modelling non-determinism and an \textit{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 if 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 $\Delta$ 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 $\Delta$ [11]. For practical reasons the entailment relation must be decidable.

A constraint store is 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, $\text{tell}(x < 2)$ 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, $\text{ask}(x < 5)$ tests whether $\text{store} \models x < 5$. The ask operation blocks when neither $\text{store} \models x < 5$ nor $\text{store} \models \neg(x < 5)$ 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 non-determinism 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 \( P_1 \triangleq \text{tell} (c > 5) \) adds constraint \( c > 5 \) to the store.

- \( \sum_{i \in \{1..n\}} \text{when } c_i \text{ do } P_i \) chooses non-deterministically a process \( P_i \) whose guard \( c_i \) is entailed by the store. For example, process \( P_2 \triangleq \text{when } (c < 3) \text{ do tell} (d = 5) + \text{when } (e > 5) \text{ do tell} (d < 3) \) 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 \( P || \text{next } P || \text{next next } P || ... \)

- **\( \star P \)** represents unbounded but finite delays, i.e \( P \) eventually will be executed. This process can be viewed as \( P + \text{next } P + \text{next next } P + \ldots \)

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:

\[
TELL \quad \langle \text{tell } c, d \rangle \rightarrow \langle \text{skip}, d \land c \rangle
\]

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:

\[
SUM \quad \langle \sum_{i \in I} \text{when } c_i \text{ do } P_i, d \rangle \rightarrow \langle P_j, d \rangle \quad \text{if } d \models c_j, \quad j \in I
\]

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:

\[
PAR \quad \langle P, c \rangle \rightarrow \langle P', d \rangle
\]

\[
\langle P || Q, c \rangle \rightarrow \langle P' || Q, d \rangle
\]

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:

\[
UNLESS \quad \langle \text{unless } c \text{ next } P, d \rangle \rightarrow \langle \text{skip}, d \rangle \quad \text{if } d \models c
\]

The rule says that nothing is done when $c$ is entailed by the store.

Finally, the rule for star processes is:

\[
STAR \quad \langle *P, d \rangle \rightarrow \langle \text{next } \#P, d \rangle \quad \text{if } n \geq 0
\]

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:

\[
A, B, \ldots : c \mid A \Rightarrow A \mid \neg A \mid \exists x A \mid \circ A \mid \Diamond A \mid \Box A
\]

- \(c\) (an atom) is a constraint. \(\Rightarrow\), \(\neg\) and \(\exists x\) 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 \(\alpha\). We say that \(\alpha \in C^\infty\) is a model of (or that it satisfies) \(A\), notation \(\alpha \models A\), if \(\langle \alpha , i \rangle \models A\) where:

- \(\langle \alpha , i \rangle \models c\) if and only if \(\alpha(i) \models c\)
- \(\langle \alpha , i \rangle \models \neg A\) if and only if \(\langle \alpha , i \rangle \models \lnot A\)
- \(\langle \alpha , i \rangle \models A_1 \Rightarrow A_2\) if and only if \(\langle \alpha , i \rangle \models A_1\) implies \(\langle \alpha , i \rangle \models A_2\)
- \(\langle \alpha , i \rangle \models \circ A\) if and only if \(\langle \alpha , i+1 \rangle \models A\)
- \(\langle \alpha , i \rangle \models \Box A\) if and only if \(\forall j \geq i \langle \alpha , j \rangle \models A\)
- \(\langle \alpha , i \rangle \models \Diamond A\) if and only if \(\exists j \geq i \langle \alpha , j \rangle \models A\)
- \(\langle \alpha , i \rangle \models \exists x A\) if and only if there is an \(x\)-variant \(\alpha'\) of \(\alpha\) s.t. \(\langle \alpha' , i \rangle \models A\)

In the last expression, \(d\) and \(\alpha'\) are \(x\)–variants of \(c\) and \(\alpha\), 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 \(\rho\)."
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.

![Lambda switch states](image)

(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 \emph{lysogeny} state. \emph{Lysogeny} state is maintained until an environmental signal turns the switch to the \emph{lytic growth}. This process is called \emph{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 \( \pi \) 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: \( 1 \rho P \), 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 \( \Phi : \mathbb{R} \in [0, 1] \to \text{Bool} \) into the constraint systems. \( \Phi \) is computed by generating pseudo-random numbers following a binomial distribution with probability \( \rho \) 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:

\[
\rho P \equiv \text{local } x \text{ in tell } (x = \Phi(\rho))\|\text{when } x = 1 \text{ do } P
\]  \hspace{1cm} (1)
Notice that \( P \) will be activated only if \( \Phi(\rho) = 1 \) as is described by rule \( \text{RHOP} \):

\[
\text{RHOP}_{(\rho P, d)} \rightarrow (P, d)^{i f \ \Phi(\rho) = 1}
\]

This construct will be useful to model events such as rates of gene transcription. For example, in the lambda switch, when \( \text{RNAP} \) binds \( \text{PRM} \), one will observe the production of a new \( \text{rep} \) protein. Nevertheless, this transcription depends on the cooperativity relationship between \( \text{RNAP} \) and \( \text{OR2} \). In this way, we will observe the increase of \( \text{rep} \) with some probability \( \rho_1 \) when there is no \( \text{reps} \) proteins binding \( \text{OR2} \) and with some other probability \( \rho_2 \) 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 \( \text{ntcc} \). 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 \( \text{rep} \) and region \( \text{OR3} \) are quite frequent in \textit{lysogeny}. We propose the constructor \((\star \rho)P \) in which \( P \) eventually occurs with probability \( \rho \). Operationally:

\[
\text{STAR}_{\rho_{\star \rho P, d}} \rightarrow (\text{next}^n \rho P, d)^{i f \ n \geq 0}
\]

It is easy to verify that this process can be viewed as a \( \rho P \) process:

\[
(\star \rho)P \overset{\text{def}}{=} \text{local } x \text{ in } \text{tell } (x = \Phi(\rho)) || \text{when } x = 1 \text{ do } \star P
\]

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 \( \text{rep} \) will eventually become 0 with probability \( \rho \)” 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 \( \text{ntcc} \) lineal-temporal logic. The syntax is:

\[
A', B', ..., = \langle c, \rho \rangle | A' \overset{\diamond}{\rightarrow} B' | \odot A' | \exists_x A' | \Diamond A' | \Diamond A'
\]

Probability \( \rho \) should not be confused with some notion of \textit{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:

\[
\begin{align*}
\langle \alpha, i \rangle \models (c, \rho) & \quad \text{iff} \quad \alpha(i) \models (c, \rho) \\
\langle \alpha, i \rangle \models o(A, \rho) & \quad \text{iff} \quad \langle \alpha, i \rangle \models (A, 1 - \rho) \\
\langle \alpha, i \rangle \models (A_1, \rho_1) \Rightarrow (A_2, \rho_2) & \quad \text{iff} \quad \langle \alpha, i \rangle \models (A_1, \rho_1) \Rightarrow \langle A_2, \rho_2 \rangle \\
\langle \alpha, i \rangle \models \Box(A, \rho) & \quad \text{iff} \quad \forall j \geq i, \langle \alpha, j \rangle \models (A, \rho) \\
\langle \alpha, i \rangle \models \Diamond(A, \rho) & \quad \text{iff} \quad \exists j \geq i, s.t. \langle \alpha, j \rangle \models (A, \rho) \\
\langle \alpha, i \rangle \models \Box_x(A, \rho) & \quad \text{iff} \quad \text{there is an } x \text{- variant } \\
\langle \alpha', i \rangle \models (A, \rho) & \quad \text{of } \alpha \text{ s.t. }\langle \alpha', i \rangle \models (A, \rho)
\end{align*}
\]

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

Since formulae such as $A' \Rightarrow B'$ 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 \texttt{ntcc} logic by using properties of probability theory:

\[
\begin{align*}
\Box(A, \rho) &= \neg A, 1 - \rho \\
(A, \rho_1) \land (B, \rho_2) &= (A \land B, \rho_1 \times \rho_2) \\
(A, \rho_1) \Rightarrow (B, \rho_2) &= (A \land B, 1 - \rho_1 + \rho_1 \times \rho_2)
\end{align*}
\]

Third expression is obtained by converting $\langle A, \rho_1 \rangle \Rightarrow (B, \rho_2)$ into $\Box((A, \rho_1) \land \neg (B, \rho_2)$ 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 $\rho$ and $(\ast_P)$. The rules permit build a proof if some process $P$ satisfies some formula $\langle A, \rho \rangle$ in the logic, i.e, if $P \vdash \langle A, \rho \rangle$

**LET**: $\text{tell } c \vdash (c, 1.0)$ \hspace{1cm} (2)

**LPAR**: $P \vdash \langle A, \rho \rangle, Q \vdash \langle B, \rho_2 \rangle \quad \frac{}{P || Q \vdash \langle A, \rho \rangle \land (B, \rho_2)}$ \hspace{1cm} (3)

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

**LLOC**: $\quad \frac{P \vdash \langle A, \rho \rangle}{\text{local } x \text{ in } P \vdash \exists x (A, \rho)}$ \hspace{1cm} (4)

**LNEXT**: $\quad \frac{P \vdash \langle A, \rho \rangle}{\text{next } P \vdash o(A, \rho)}$ \hspace{1cm} (5)
LUNL: \[
\begin{align*}
P \vdash (A, \rho) \\
\text{unless } c \text{ next } P \vdash (c, 1.0) \lor o(A, \rho)
\end{align*}
\] (6)

LCONS: \[
\begin{align*}
P \vdash (A, \rho) & \quad \text{if } P \vdash (B, \rho_2) - (B, \rho_2) \\
\end{align*}
\] (7)

LSTAR: \[
\begin{align*}
P \vdash (A, \rho) & \quad \text{if } P \vdash (\circ(A, \rho)) \\
\end{align*}
\] (8)

LSUM: \[
\begin{align*}
\forall i \in I & \quad P_i \vdash (A_i, \rho_i) \\
\sum_{i \in I} \text{ when } c_i \text{ do } P_i \vdash \frac{1}{\Lambda_i(1)} (A_i, \rho_i) \lor \Lambda_i(1) \\
\end{align*}
\] (9)

PPRO: \[
\begin{align*}
P \vdash (A, \rho) & \quad \text{if } P \vdash (A, \rho \times \rho_2) \\
\end{align*}
\] (10)

The previous equation represents the inference rule for \( \rho P \) processes. Since the probability \( \rho \) and the probability \( \rho_2 \) are independent, the probability of both occurrences is \( \rho \times \rho_2 \).

Finally, the rule for the \((\star)P\) process is:

PSTAR: \[
\begin{align*}
P \vdash (A, \rho) & \quad \text{if } P \vdash (\circ(A, \rho)) \\
\end{align*}
\] (11)

4 Modeling the \( \lambda \)-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 \( \text{PROCESS}(x) \defeq P \) 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 \((\star_p)P\) processes we can model this fact as:

\[
\text{INDUCTION}(\rho_{\text{ind}}) \overset{\text{def}}{=} \star_{\rho_{\text{ind}}} (\text{tell reset rep})
\]

Eventually, under the probability \(\rho_{\text{ind}}\), this process reduces the concentration of rep to 0 (see \(\text{reset rep}\) 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 \(\text{rep threshold}\).

\[
\text{REPTHOLD}(\text{rep threshold}) \overset{\text{def}}{=} \begin{cases} \text{when } C_{\text{rep}} > \text{rep threshold} \text{ do next tell inhibit rep} \end{cases}
\]

### 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:

\[
\text{CROtrans} \overset{\text{def}}{=} \begin{cases} \text{when } pr_{\text{bound}} \text{ do } \rho_{\text{crotrans}} (\text{tell inc cro}) \end{cases}
\]

where predicate \(pr_{\text{bound}}\) is true iff RNAP is binding \(PR\). Constraint \(\text{inc cro}\) 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 \(PRM\) :

\[
\text{Cltrans}(\rho_{\text{high}}, \rho_{\text{low}}) \overset{\text{def}}{=} \begin{cases} \text{when } \neg \text{inhibit rep } \land \text{prm bound} \text{ do local p in } \text{tell inc rep} \end{cases}
\]

where cooperativity is modelled by means a \(\text{when c do P}\) process that chooses between \(\rho_{\text{high}}\) and \(\rho_{\text{low}}\) in order to execute \(\text{tell inc(rep)}\).
with the right probability. Notice that synchronisation is guaranteed by the use of constraints: process \( p(tell \ inc_{rep_c}) \) 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 \( cro \), \( P \) type processes are needed. Additionally, we have to take into 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:

\[
\text{OR1 def} \equiv \\
\text{when or1_unbound do} \\
\text{when rep_c > 0 do } \rho_{or1rep} (\text{next tell (or1_rep_bound and dec_rep_c)}) \\
+ \text{ when cro_c > 0 do } \rho_{or1cro} (\text{next tell (or1_cro_bound and dec_cro_c)}) \\
+ \text{ when or1_rep_bound do } 1.0 - \rho_{or1rep} (\text{next tell (or1_unbound and inc_rep_c)}) \\
+ \text{ when or1_cro_bound do } 1.0 - \rho_{or1cro} (\text{next tell (or1_unbound and inc_cro_c)})
\]

\( (16) \)

\[
\text{OR2 def} \equiv \text{local or2rep in} \\
\text{when or2_unbound do} \\
\text{when rep_c > 0 do } \rho_{or2rep} (\text{next tell (or2_rep_bound and dec_rep_c)}) \\
+ \text{ when cro_c > 0 do } \rho_{or2cro} (\text{next tell (or2_cro_bound and dec_cro_c)}) \\
+ \text{ when ~or1_bound_rep do tell (or2rep = or2rep_low)} \\
+ \text{ when or2_rep_bound do } 1.0 - \rho_{or2rep} (\text{next tell (or2_unbound and inc_rep_c)}) \\
+ \text{ when or2_cro_bound do } 1.0 - \rho_{or2cro} (\text{next tell (or2_unbound and inc_cro_c)}) \\
\text{|| (when or1_bound_rep do tell (or2rep = or2rep_high)})
\]

\( (17) \)

\[
\text{OR3 def} \equiv \\
\text{when or3_unbound do} \\
\text{when rep_c > 0 do } \rho_{or3rep} (\text{next tell (or3_rep_bound and dec(rep_c)}) \\
+ \text{ when cro_c > 0 do } \rho_{or3cro} (\text{next tell (or3_cro_bound and dec(cro_c)}) \\
+ \text{ when or1_rep_bound do } 1.0 - \rho_{or3rep} (\text{next tell (or3_unbound and inc(rep_c)}) \\
+ \text{ when or1_cro_bound do } 1.0 - \rho_{or3cro} (\text{next tell (or3_unbound and inc(cro_c)})
\]

\( (18) \)

where \( \rho_{or1rep, or1cro, or2rep_high, or2rep_low, or3rep and or3cro} \) 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 \( rep_s \) or \( cro_s \) 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<sub>c</sub>):

\[
\text{CONCCTR def} \equiv \\
\begin{align*}
& \text{when reset_rep, do next tell rep<sub>c</sub> = 0+} \\
& \quad \text{when inc_rep, } \neg \text{reset_rep, do next tell (rep<sub>c</sub> = rep<sub>c</sub>' + 1)+} \\
& \quad \text{when dec_rep, } \neg \text{reset_rep, do next tell (rep<sub>c</sub> = rep<sub>c</sub>' - 1))} \\
& \text{when inc_cro, do next tell (cro<sub>c</sub> = cro<sub>c</sub>' + 1)+} \\
& \quad \text{when dec_cro, do next tell (cro<sub>c</sub> = cro<sub>c</sub>' - 1))}
\end{align*}
\]

(19)

In previous equation, variables with prime symbol (e.g. cro<sub>c</sub>') 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]:

\[
\text{State}_{i}(v_i) \overset{\text{def}}{=} \text{tell } m'_i = v_i |\text{next State}_{i}(m_i)
\]

(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 \(\rho_{rnap,pr}\) (resp. \(\rho_{rnap,prm}\)). Equation 21 models this fact:

\[
\text{RNAPCTR def} \equiv \\
\begin{align*}
& \text{when rnap_unbound do (} \\
& \quad \text{when or1_unbound do rnap_pr(next tell (or1_rnap_bound \land pr_bound))} \\
& \quad \quad \text{+ when or3_unbound do rnap_prm(next tell (or3_rnap_bound \land prm_bound))} \\
& \quad \text{+ when rnap_bound do (} \\
& \quad \quad \text{+ when or1_rnap_bound do 1.0 – rnap_pr(next tell or1_unbound)} \\
& \quad \quad \quad \text{+ when or3_rnap_bound do 1.0 – rnap_prm(next tell or3_unbound))}
\end{align*}
\]

(21)

Using parallel composition among equations 12 to 21 we can de-
scribe the overall lambda-switch system as follows:

\[ \lambda \rightarrow \text{PROC}(\rho_{\text{ind}}, \rho_{\text{thr}}, \rho_{\text{or}1, \text{high}}, \rho_{\text{or}1, \text{low}}, \rho_{\text{or}2, \text{high}}, \rho_{\text{or}2, \text{low}}, \rho_{\text{or}3, \text{high}}, \rho_{\text{or}3, \text{low}}, \rho_{\text{cro}}, \rho_{\text{cro}1, \text{high}}, \rho_{\text{cro}1, \text{low}}, \rho_{\text{cro}2, \text{high}}, \rho_{\text{cro}2, \text{low}}, \rho_{\text{cro}3, \text{high}}, \rho_{\text{cro}3, \text{low}}) \]

\[ \text{def} \quad \text{local } \rho_{\text{rep}}(0), \rho_{\text{cro}}(0), \rho_{\text{bound}}, \rho_{\text{bound}1, \text{rep}}, \rho_{\text{bound}2, \text{rep}}, \rho_{\text{bound}3, \text{rep}}, \rho_{\text{bound}1, \text{cro}}, \rho_{\text{bound}2, \text{cro}}, \rho_{\text{bound}3, \text{cro}}, \rho_{\text{unbound}}, \rho_{\text{unbound}1, \text{rep}}, \rho_{\text{unbound}2, \text{rep}}, \rho_{\text{unbound}3, \text{rep}}, \rho_{\text{unbound}1, \text{cro}}, \rho_{\text{unbound}2, \text{cro}}, \rho_{\text{unbound}3, \text{cro}}, \rho_{\text{rnap}}, \rho_{\text{rnap}1, \text{bound}}, \rho_{\text{rnap}2, \text{bound}}, \rho_{\text{rnap}3, \text{bound}}, \rho_{\text{rnap}1, \text{unbound}}, \rho_{\text{rnap}2, \text{unbound}}, \rho_{\text{rnap}3, \text{unbound}}, \rho_{\text{citrans}}, \rho_{\text{citrans}1, \text{bound}}, \rho_{\text{citrans}2, \text{bound}}, \rho_{\text{citrans}3, \text{bound}}, \rho_{\text{citrans}1, \text{unbound}}, \rho_{\text{citrans}2, \text{unbound}}, \rho_{\text{crotrans}}, \rho_{\text{crotrans}1, \text{bound}}, \rho_{\text{crotrans}2, \text{bound}}, \rho_{\text{crotrans}3, \text{bound}}, \rho_{\text{crotrans}1, \text{unbound}}, \rho_{\text{crotrans}2, \text{unbound}}, \rho_{\text{crotrans}3, \text{unbound}} \]

\[ \text{INDUCTION}(\rho_{\text{ind}}) \quad | \quad \text{REPT HOLD}(\rho_{\text{thr}}) \quad | \quad \text{OR1} \quad | \quad \text{OR2} \quad | \quad \text{OR3} \quad | \quad \text{RNAPCTR} \quad | \quad \text{CONCCTR} \]

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 on 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 ρP 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 cl 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:

$$\text{LPAR : } \frac{\text{INDUCTION }\vdash \langle A, \rho \rangle \ OR1 \vdash \langle A_2, \rho_2 \rangle \ ... \ \text{RNAPCTR } \vdash \langle A_n, \rho_n \rangle}{\lambda - \text{PROC } \vdash \langle A, \rho \rangle \land \langle A_2, \rho_2 \rangle \land \cdots \land \langle A_n, \rho_n \rangle}$$

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

By using LCONS in equation 23 we get:

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

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

$$\text{PSTAR : } \frac{\text{tell } \text{reset(rep)} \vdash \langle \text{reset(rep)} , 1.0 \rangle}{\star_{\rho_{\text{ind}}} \text{tell } \text{reset(rep)}}$$

As INDUCTION $\text{def} \equiv \star_{\rho_{\text{ind}}} \text{tell } \text{reset(rep)} \| \text{INDUCTION}$ (omitting the local hiding), we can affirm that $\lambda - \text{PROC}$ satisfies the property “eventually under a probability $\rho_{\text{ind}}$ the concentration of rep will be zero”, i.e, $\Diamond \langle \text{reset(rep)} , \rho_{\text{ind}} \rangle$.

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

$$\text{LPAR : } \frac{\text{RNAPCTR } \vdash \langle A, \rho \rangle \ \text{CITrans } \vdash \langle A_2, \rho_2 \rangle \ ...}{\lambda - \text{PROC } \vdash \langle A, \rho \rangle \land \langle A_2, \rho_2 \rangle \land \cdots}$$
Using the definition of CITRANS we get:

\[
\text{LSUM :}
\begin{align*}
\text{tell } p = \rho_{\text{high}} \vdash (p = \rho_{\text{high}}, 1.0) & \quad \text{tell } p = \rho_{\text{low}} \vdash (p = \rho_{\text{low}}, 1.0) \\
\text{when or2\_bound\_rep do tell } p = \rho_{\text{high}} & \\
\text{when or2\_vacant do tell } p = \rho_{\text{low}} & \\
(\text{or2\_bound\_rep } \land p = \rho_{\text{high}}, 1.0) \lor (\text{or2\_vacant } \land p = \rho_{\text{low}}, 1.0) & \\
(\lnot \text{or2\_bound\_rep } \land \lnot \text{or2\_vacant }, 1.0) & \\
\end{align*}
\]

(27)

Since the premise is the formula \(\langle \text{or1\_bound\_rep } \land \text{or2\_bound\_rep } \land \text{or3\_bound\_rnap }, 1.0 \rangle\) we can use LCONS as follows:

\[
\text{LCONS :}
\begin{align*}
\langle \text{or1\_bound\_rep } \land \text{or2\_bound\_rep } \land \text{or3\_bound\_rnap }, 1.0 \rangle & \\
\text{when or2\_bound\_rep do tell } p = \rho_{\text{high}} & \\
\text{when or2\_vacant do tell } p = \rho_{\text{low}} & \\
(p = \rho_{\text{high}}, 1.0) & \\
\end{align*}
\]

(28)

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

\[
\begin{align*}
\text{when or2\_bound\_rep do tell } p = \rho_{\text{high}} & \\
\text{when or2\_vacant do tell } p = \rho_{\text{low}} & \\
(p = \rho_{\text{high}}, 1.0) & \\
\end{align*}
\]

(29)

Equation 29 says that new reps will appear with a probability \(\rho_{\text{high}}\) 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 \(\rho\) and the second one that \(P\) is eventually executed with probability \(\rho\). 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 \(\Phi(\rho) : [0, 1] \rightarrow \text{bool}\) 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 \(\pi\)-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


