Using Stochastic NTCC to Model Biological Systems

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

Abstract
Concurrent process calculi are powerful formalisms for modeling 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 behavior 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 modeling 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.

Keywords: NTCC, Lambda-Switch, biological systems, concurrent process calculus, concurrent constraint programming

1 Introduction
We are interested in using soft computing techniques for modeling 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 [11] 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 synchronization 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 modeling to define suitable computational calculi and analyze their behavior 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.

We propose using a temporal non deterministic concurrent calculus (*ntcc*, see [6]) as a formal base to model timed gene activity processes in such a way that their biological properties can be formally proved. This goes in the same direction as the concurrent process calculi models of biological systems proposed recently ([9], [2], [3]).

The *ntcc* calculus inherits ideas from the tcc model [10], 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 modeling 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 modeling 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 behavior 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 modeled 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. Our model using the stochastic extension is both simpler and more complete than the one recently proposed in [3].

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 modeling 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 \((\Sigma, \Delta)\) where \( \Sigma \) is a signature (i.e a set of constants, functions and predicate symbols) and \( \Delta \) is a consistent first-order theory over \( \Sigma \) (i.e a set of sentences over \( \Sigma \) having at least one model). Constraints can be viewed as first-order formulae over \( \Sigma \) and \( c \models d \) holds if the implication \( c \Rightarrow d \) is valid in \( \Delta \) [6]. 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 synchronization purposes. The store is monotonically refined by adding information using tell operations of the calculus. For example, \( \text{tell}(x < 5) \) adds constraint \( x < 5 \) 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 Overview

ntcc [6] is a process calculus that extends tcc [10]. In both of them, processes share a common store of partial information [11]. 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 behavior [5] and music composition [6]. Unlike tcc, ntcc includes constructs for modeling nondeterminism and asynchrony. A very important benefit of being able to specify non-deterministic and asynchronous behavior arises when modeling 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 modeled faithfully.

2.1.3 Process Syntax

In this section we describe briefly the syntax of ntcc processes. See [6] for further details.

ntcc provides the following constructors:

- \textbf{tell} : adds new information to the constraints store. For example, the process \( P_1 \overset{\text{def}}{=} \text{tell} (c > 5) \) 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 \( P_2 \overset{\text{def}}{=} \text{when } (c < 3) \text{ do tell } (d = 5) + \text{when } (e > 5) \text{ do tell } (d < 3) \) 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 \( Q \).

• \textbf{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 \( x \).

• \textbf{next} \( P \) executes process \( P \) in the next time unit (unit-delay)

• \textbf{unless} \( c \) next \( P \) executes \( P \) if \( c \) cannot be entailed by the constraint store in the current time unit

• \( !P \) executes \( P \) in all time units from the current one on. It can be viewed as \( P || \text{next } P || \text{next next } P || ... \)

• \( *P \) represents unbounded but finite delays, i.e \( P \) eventually will be executed. This process can be viewed as \( P + \text{next } P + \text{next next } P + \text{next } nP \) where \( n \) is a finite natural number.

Two new operators (\( \rho P \) and \( *P \)) will be introduced in section 3.1 to express stochastic behavior. They will be illustrated in the model of the lambda switch.

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 quiescent 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 [6]. Reduction rules are based on configurations. A configuration \( (P, d) \) is composed of a ntcc process \( P \) and a store \( d \).

For tell processes we have:

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

where \text{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:

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

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:

\[
\text{PAR}^{\langle P, c \rangle \rightarrow \langle P', d \rangle \text{ if } P || \langle Q, c \rangle \rightarrow \langle P', [Q, d] \rangle}
\]

It says that if \( P \) evolves to \( P' \), then the same transition can occur if we execute \( P \) in parallel with some process \( Q \). Parallel composition is commutative.

For unless \( c \) next \( P \) processes:

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

The rule says that nothing is done when \( c \) is entailed by the store.

Finally, the rule for star processes is:

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

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 include those defined by unless processes whose guard cannot be entailed from the current store (see [6] for details).
2.2 Linear-temporal Logic in nttc

nttc 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 [6]. The syntax of this logic is as follows:

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

\(c\) is a constraint. \(\Rightarrow\), \(\neg\) and \(\exists\) represent the linear-temporal logic implication, negation and existential quantification, respectively [4]. 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 [4]. In nttc, 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 \(\alpha\) satisfies) \(A\), notation \(\alpha \models A\), if \(\alpha(1) \models A\) where:

\[
\begin{align*}
\langle \alpha , i \rangle &\models c \quad \iff \quad \alpha(i) \models c \\
\langle \alpha , i \rangle &\models \neg A \quad \iff \quad \langle \alpha , i \rangle \not\models A \\
\langle \alpha , i \rangle &\models A_1 \Rightarrow A_2 \quad \iff \quad \langle \alpha , i \rangle \models A_1 \implies \langle \alpha , i \rangle \models A_2 \\
\langle \alpha , i \rangle &\models \circ A \quad \iff \quad \langle \alpha , i+1 \rangle \models A \\
\langle \alpha , i \rangle &\models \Box A \quad \iff \quad \forall j \geq i \langle \alpha , j \rangle \models A \\
\langle \alpha , i \rangle &\models \Diamond A \quad \iff \quad \exists j \geq i \exists \text{s.t.} \langle \alpha , j \rangle \models A
\end{align*}
\]

(1)

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 [6] 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 prove these kind of properties.

2.3 Lambda Switch

In this section we give a brief description of a biological system (called the \(\lambda\) switch) that we model in section 4 using our proposed extension of nttc. For additional details see [3] or [1].

Bacteriophage \(\lambda\) 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 \(\lambda\) switch (see figure 1). In this switch there are two promoter regions called PRM and PR where production of rep and cro proteins, respectively, take place. The lytic growth state is characterized by a high concentration of cro proteins whereas lysogeny is characterized by a high concentration of rep.

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 cl 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 cl happen. It implies that more rep
proteins are produced thus increasing the chance of maintaining the \textit{lysogeny} state. \textit{Lysogeny} state is maintained until an environmental signal turns the switch to the \textit{lytic growth}. This process is called \textit{Induction} and it occurs with very low probability. In this state the concentration of \textit{rep} decreases dramatically and so \textit{RNAP} has the chance to bind to \textit{OR1}, that is now vacant most of the time. \textit{PR}, a stronger promoter than \textit{PRM}, starts the transcription of gene \textit{cro} and thus new instances of \textit{cro} are produced. Because \textit{OR3} has a high affinity for \textit{cro}, it is bound by this protein avoiding \textit{RNAP} bindings to \textit{PRM}.

This biological system has been successfully modeled in [3] by using the \textit{\pi} calculus [8] with stochastic behavior [7]. 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 [3].

3 Introducing Stochastic Behavior in \textit{ntcc}

In this section we introduce two new operators in \textit{ntcc} for modeling stochastic behavior. We also propose an extension of the \textit{ntcc} linear-temporal logic that can be used to prove properties involving probabilistic statements.

3.1 $\star P$ Processes

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

$$\text{STAR}_{\rho}(\star P, d) \rightarrow (\text{next}^n P, d) \quad \text{if } n \geq 0 \land \Phi(\rho) = 1$$

where $\Phi : \mathbb{R} \rightarrow \text{bool}$. $\Phi$ can be 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.

This operator can be expressed as combinations of the original ones:

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

by adding the function $\Phi$ to the signature of the constraint system.
3.2 $\rho P$

The previous constructor operates over an arbitrary time interval. We are interested in expressing processes that occur in a single time unit but with a probability $\rho$. Those processes, denoted $\rho P$, can be expressed as follows:

$$\rho P \equiv \text{local } x \text{ in } \text{tell } (x = \Phi(\rho))|\text{when } x = 1 \text{ do } P$$  \hspace{1cm} (2)

Operationally:

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

3.3 Stochastic parameters in the linear-temporal logic

To prove properties such as “The concentration of 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 ntcc linear-temporal logic. The new operators are as follows:

$$A', B', ..., = \langle c, \rho \rangle \mid A' \Rightarrow A' \mid \frac{\rho}{2} A' \mid \exists_x A' \mid \circ A', \mid \diamond A' \mid \Box A'$$

Probability $\rho$ 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:

$$\langle \alpha, i \rangle \models \langle c, \rho \rangle \iff \alpha(i) \models \langle c, \rho \rangle$$

$$\langle \alpha, i \rangle \models \frac{\rho}{2} \langle A, \rho \rangle \iff \langle \alpha, i \rangle \models \langle A, 1 - \rho \rangle$$

$$\langle \alpha, i \rangle \models \frac{\rho}{2} \langle A_1, \rho_1 \rangle \Rightarrow \frac{\rho}{2} \langle A_2, \rho_2 \rangle \iff \langle \alpha, i \rangle \models \langle A_1, \rho_1 \rangle \Rightarrow \langle A_2, \rho_2 \rangle$$

$$\langle \alpha, i \rangle \models \circ \langle A, \rho \rangle \iff \langle \alpha, i \rangle \models \langle A, i + 1 \rangle \models \langle A, \rho \rangle$$

$$\langle \alpha, i \rangle \models \Box \langle A, \rho \rangle \iff \forall j \geq i \langle \alpha, j \rangle \models \langle A, \rho \rangle$$

$$\langle \alpha, i \rangle \models \exists_x \langle A, \rho \rangle \iff \text{there is an } x \text{-variant } \alpha' \text{ of } \alpha \text{ s.t. } \langle \alpha', i \rangle \models \langle A, \rho \rangle$$

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

3.3.1 Inference System

We extend the inference system proposed in [6] with inferences rules taking into account the new form of formulae and the probabilistic operators $\rho P$ and $\ast \rho P$. The rules are as follows:

$$\text{LTTLL : tell } c \vdash \langle c, 1.0 \rangle$$  \hspace{1cm} (4)

$$\text{LPAR : } P \vdash \langle A, \rho \rangle \quad Q \vdash \langle B, \rho_2 \rangle \quad P || Q \vdash \langle A, \rho \rangle \wedge \langle B, \rho_2 \rangle$$  \hspace{1cm} (5)

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

$$\text{LLOC : } \frac{P \vdash \langle A, \rho \rangle}{\text{local } x \text{ in } P \vdash \exists_x \langle A, \rho \rangle}$$  \hspace{1cm} (6)

$$\text{LNEXT : } \frac{P \vdash \langle A, \rho \rangle \quad \text{next } P \vdash \circ \langle A, \rho \rangle}{\text{next } P \vdash \langle A, \rho \rangle}$$  \hspace{1cm} (7)

$$\text{LUNL : } \frac{P \vdash \langle A, \rho \rangle \quad \text{unless } c \text{ next } P \vdash \langle c, 1.0 \rangle \vee \circ \langle A, \rho \rangle}{P \vdash \langle A, \rho \rangle}$$  \hspace{1cm} (8)

$$\text{LCONS : } \frac{P \vdash \langle A, \rho \rangle \quad \text{if } \langle A, \rho \rangle \Rightarrow \langle B, \rho_2 \rangle}{P \vdash \langle B, \rho_2 \rangle}$$  \hspace{1cm} (9)
LSTAR: \[ P \vdash \langle A, \rho \rangle \]

\[ \forall i \in I \ P_i \vdash \langle A_i, \rho_i \rangle \]

(10)

LSUM: \[ \sum_{i \in I} \text{when } c_i \text{ do } P_i \vdash \langle \sum_{i \in I} (c_i \cdot 1 \cdot \langle A_i, \rho_i \rangle) \cup \sum_{i \in I} \langle c_1, 1.0 \rangle \rangle \]

(11)

PPRO: \[ P \vdash \langle A, \rho \rangle \]

\[ P_p \vdash \langle A \cdot \rho \times \rho_2 \rangle \]

(12)

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 \( \ast_P \) process is:

PSTAR: \[ P \vdash \langle A, \rho \rangle \]

\[ \ast_P \vdash \langle A \cdot \rho \times \rho_2 \rangle \]

(13)

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 \( \text{PROCESS}(x) \equiv P \)

that do not formally belong to the calculus. These, however, can be easily defined in terms of the standard calculus constructs (see [6]).

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 \( \ast_P \) processes we can model this fact as:

INDUCTION(\( \rho_{\text{ind}} \)) \( \equiv \ast_P \text{tell } reset\text{rep}_c \)

(14)

Eventually, under the probability \( \rho_{\text{ind}} \), this process reduces the concentration of rep to 0 (see \( \text{reset}\text{rep}_c \) in equation 21).

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 \( \text{rep}_\text{threshold} \).

REPTHOLD(\( \text{rep}_\text{threshold} \)) \( \equiv \text{when } \rho > \text{rep}_\text{threshold} \text{ do next tell inhibit}_\text{rep} \)

(15)

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:

\( \text{CROtrans} \equiv \text{when } \rho_{\text{bound}} \text{ do } \rho_{\text{crotrans}} \text{tell } inc\text{cro}_c \)

(16)
where predicate \( p_{\text{bound}} \) 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 21).

Modeling gene transcription at \( \text{PRM} \) 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 \( \text{PRM} \) is higher and in consequence higher is also the probability of producing more \( \text{reps} \). Equation 17 models gene transcription at \( \text{PRM} \):

\[
\text{CITrans}(\rho_{\text{high}}, \rho_{\text{low}}) \overset{\text{def}}{=} \text{when } \neg \text{inhibit\_rep} \land p_{\text{bound}} \text{ do local } p \text{ in }
\begin{align*}
& \text{(when } \text{or}^2 \text{bound\_rep do tell } p = \rho_{\text{high}} \\
& \text{+ when } \text{or}^2 \text{bound\_cro} \lor \text{or}^2 \text{vacant do } p = \rho_{\text{low}}) \\
& \| p(\text{tell inc\_rep}_c) \quad (17)
\end{align*}
\]

where cooperativity is modeled by means a \text{when } c \text{ do } P process that chooses between \( \rho_{\text{high}} \) and \( \rho_{\text{low}} \) in order to execute \text{tell inc\_rep}_c with the right probability. Notice that synchronization is guaranteed by the use of constraints: process \( p(\text{tell inc\_rep}_c) \) blocks until the value of \( p \) is known.

### 4.3 Operator Regions

In what follows we model the behavior of the OR1, OR2, and OR3 binding sites. Since each operator has different affinities for rep and cro, type processes are needed. OR2 and OR1 exhibit another cooperativity relationship: 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} \overset{\text{def}}{=} \text{when } \text{or}^1 \text{unbound do }
\begin{align*}
& \text{when rep}_c > 0 \text{ do } \rho_{\text{or}^1\text{rep}} \text{(next tell } (\text{or}^1 \text{rep\_bound} \land \text{dec\_rep}_c)) \\
& \text{+ when cro}_c > 0 \text{ do } \rho_{\text{or}^1\text{cro}} \text{(next tell } (\text{or}^1 \text{cro\_bound} \land \text{dec\_cro}_c)) \\
& \| (\text{when } \text{or}^1 \text{bound\_rep do tell } (\text{or}^1 \text{rep = por}_2\text{rep\_low}))
\end{align*} 

\text{OR2} \overset{\text{def}}{=} \text{local } \text{or}^2\text{rep in }
\begin{align*}
& \text{when or}^2\text{unbound do }
\begin{align*}
& \text{when rep}_c > 0 \text{ do } \rho_{\text{or}^2\text{rep}} \text{(next tell } (\text{or}^3 \text{rep\_bound} \land \text{dec\_rep}_c)) \\
& \text{+ when cro}_c > 0 \text{ do } \rho_{\text{or}^2\text{cro}} \text{(next tell } (\text{or}^3 \text{cro\_bound} \land \text{dec\_cro}_c)) \\
& \| (\text{when } \text{or}^1 \text{bound\_rep do tell } (\text{or}^2 \text{rep = por}_2\text{rep\_high}))
\end{align*}
\end{align*}

\text{OR3} \overset{\text{def}}{=} \text{when } \text{or}^3\text{unbound do }
\begin{align*}
& \text{when rep}_c > 0 \text{ do } \rho_{\text{or}^3\text{rep}} \text{(next tell } (\text{or}^3 \text{rep\_bound} \land \text{dec\_rep}_c)) \\
& \text{+ when cro}_c > 0 \text{ do } \rho_{\text{or}^3\text{cro}} \text{(next tell } (\text{or}^3 \text{cro\_bound} \land \text{dec\_cro}_c)) \\
& \| (\text{when } \text{or}^1 \text{bound\_rep do tell } (\text{or}^3 \text{rep = por}_3\text{rep}\_})
\end{align*}

where \( \rho_{\text{or}1\text{rep}}, \rho_{\text{or}1\text{cro}}, \rho_{\text{or}2\text{rep}\_high}, \rho_{\text{or}2\text{rep}\_low}, \rho_{\text{or}3\text{rep}} \) and \( \rho_{\text{or}3\text{cro}} \) are the probabilities (affinities) of each binding site w.r.t rep and cro. Constraints \( \text{dec\_rep}_c \) and \( \text{dec\_cro}_c \) will cause decrementing of the concentration of rep (resp. cro) in the environment (see equation 21). In the above processes, when the operator is unbound (i.e. vacant) and \( \text{reps} \) or \( \text{cros} \) 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 21 controls the concentration of rep and cro for the next time unit:

\[
\text{CONCCTR} \overset{\text{def}}{=} \text{(when } \text{reset\_rep}_c \text{ do next tell } \text{rep}_c = 0 \text{ + when inc\_rep}_c \land \neg \text{reset\_rep}_c \text{ do next tell } (\text{rep}_c = \text{rep}_c + 1) + \\
\text{when dec\_rep}_c \land \neg \text{reset\_rep}_c \text{ do next tell } (\text{rep}_c = \text{rep}_c - 1) \| (\text{when inc\_cro}_c \text{ do next tell } (\text{cro}_c = \text{cro}_c + 1) + \text{when dec\_cro}_c \text{ do next tell } (\text{cro}_c = \text{cro}_c - 1))
\]

(21)
4.4 RNAP binding

Equations 16 and 17 depend on bindings between RNAP and promoters PR and PRM, respectively. The behavior of RNAP has two components: (1) when RNAP is binding PR (resp PRM), there is a probability of 1.0 - ρ_{rnap,pr} (resp. 1.0 - ρ_{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 ρ_{rnap,pr} (resp. ρ_{rnap,prm}). Equation 22 models this fact:

\[
\text{RNAPCTR} \overset{\text{def}}{=} \begin{cases} \\
\text{when } \text{rnap\_unbound} \text{ do } ( \\
\text{when } \text{or1\_unbound} \text{ do } \text{rnap\_pr} (\text{next tell } (\text{or1\_rnap\_bound } \land \text{pr\_bound})) \\
+ \text{when } \text{or3\_unbound} \text{ do } \text{rnap\_prm} (\text{next tell } (\text{or3\_rnap\_bound } \land \text{prm\_bound})) \\
+ \text{when } \text{rnap\_bound} \text{ do } ( \\
\text{when } \text{or1\_rnap\_bound} \text{ do } 1.0 - \text{rnap\_pr} (\text{next tell } \text{or1\_unbound}) \\
+ \text{when } \text{or3\_rnap\_bound} \text{ do } 1.0 - \text{rnap\_prm} (\text{next tell } \text{or3\_unbound})) \\
\end{cases}
\]

With equations 14 to 22 we can describe the overall lambda-switch system as follows:

\[
\lambda - \text{PROC}(\rho_{\text{ind}}, \text{rep\_hr}, \text{por1rep}, \text{por2rep\_high}, \text{por2rep\_low}, \text{por3rep}, \\
\text{por1cro}, \text{por2cro}, \text{por3cro}, \text{pcrotrans}, \text{pcrans\_high}, \text{pcrans\_low}) \overset{\text{def}}{=} \begin{cases} \\
\text{local } \text{rep\_0}, (0), \text{cro\_0}, (0), \text{pr\_bound}, \text{prm\_bound\_inhibit\_rep} \\
\text{or1\_bound\_rep}, \text{or1\_bound\_cro}, \text{or2\_bound\_rep}, \text{or2\_bound\_cro}, \text{or1\_unbound}, \text{or2\_unbound}, \text{or3\_unbound}, \\
\text{rnap\_bound}, \text{rnap\_unbound}, \text{or1\_bound\_rnap}, \text{or2\_bound\_rnap}, \text{or3\_bound\_rnap} \\
\rho_{\text{or1rep}}, \rho_{\text{or2rep}}, \rho_{\text{por1cro}}, \rho_{\text{por2cro}}, \rho_{\text{por3cro}} \\
\text{in } \text{INDUCTION}(\rho_{\text{ind}}) || \text{REPTHOLD}((\text{rep\_hr}) || \text{CROtrans} || \text{CItrans} || \\
\text{OR1} || \text{OR2} || \text{OR3} || \text{RNAPCTR} || \text{CONCCTR})
\end{cases}
\]

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

4.5 Proving temporal properties of the λ switch system

In this section we show proofs for two properties the lambda switch system must satisfy:

- 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_{c\_high}. Recall 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.

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

\[
\text{LPar} : \quad \text{INDUCTION} \vdash (A, \rho) \quad \text{OR1} \vdash (A_2, \rho_2) \ldots \text{RNAPCTR} \vdash (A_n, \rho_n)
\]

Notice that we use the temporal always operator because processes are replicated (i.e. use the “!!” prefix) in the definition of λ - PROC.

By using LCONS in equation 24 we get:

\[
\text{LPar} : \quad \lambda - \text{PROC} \vdash (A, \rho)
\]

The INDUCTION process satisfies the formula (A, ρ). Now, we are going to find out the structure of A.
\[ P_{\text{STAR}} : \frac{\text{tell } \text{reset}(\text{rep}) \vdash \langle \text{reset}(\text{rep}) , 1.0 \rangle}{\text{\textit{INDUCTION}} \vdash \Diamond \langle \text{reset}(\text{rep}) \rangle , \rho_{\text{ind}}} \] (26)

As \text{INDUCTION} \overset{\text{def}}{=} \text{\textit{INDUCTION}} \text{ (omitting the local hiding), we can affirm that } \lambda - \text{PROC} \text{ satisfies the property "eventually under a probability } \rho_{\text{ind}} \text{ the concentration of } \text{rep} \text{ will be zero", i.e, } \Diamond \langle \text{reset}(\text{rep}) \rangle , \rho_{\text{ind}}.

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

\[ \text{LPAR} : \frac{\text{RNAPCTR} \vdash \langle A , \rho \rangle \quad \text{CITrans} \vdash \langle A_2 , \rho_2 \rangle \ldots}{\lambda - \text{PROC} \vdash \Box \langle A , \rho \rangle \land \Box \langle A_2 , \rho_2 \rangle \ldots} \] (27)

Using the definition of \text{CITrans} we get:

\[ \text{tell } p = \rho_{\text{high}} \vdash \langle p = \rho_{\text{high}} , 1.0 \rangle \quad \text{tell } p = \rho_{\text{low}} \vdash \langle p = \rho_{\text{low}} , 1.0 \rangle \]

\[ \text{LSUM} : \frac{\text{when or}_2 \text{\textit{bound}} \text{\textit{rep}} \text{ do tell } p = \rho_{\text{high}} + \text{when or}_2 \text{\textit{vacant}} \text{ do tell } p = \rho_{\text{low}} \vdash}{\langle \text{or}_2 \text{\textit{bound}} \text{\textit{rep}} \land p = \rho_{\text{high}} , 1.0 \rangle \lor \langle \text{or}_2 \text{\textit{vacant}} \land p = \rho_{\text{low}} , 1.0 \rangle \lor}{\langle \neg \text{or}_2 \text{\textit{bound}} \text{\textit{rep}} \land \neg \text{or}_2 \text{\textit{vacant}} , 1.0 \rangle} \] (28)

Since the premise is the formula \( \langle \text{or}_1 \text{\textit{bound}} \text{\textit{rep}} \land \text{or}_2 \text{\textit{bound}} \text{\textit{rep}} \land \text{or}_3 \text{\textit{bound}} \text{\textit{rnap}} , 1.0 \rangle \) we can use \text{LCONS} as follows:

\[ \text{LCONS} : \frac{\text{when or}_2 \text{\textit{bound}} \text{\textit{rep}} \text{ do tell } p = \rho_{\text{high}} + \text{when or}_2 \text{\textit{vacant}} \text{ do tell } p = \rho_{\text{low}} \vdash}{\langle p = \rho_{\text{high}} , 1.0 \rangle} \] (29)

Finally, by using \text{PPRO} we can verify that \text{CITrans} satisfies the following formula:

\[ \text{when or}_2 \text{\textit{bound}} \text{\textit{rep}} \text{ do tell } p = \rho_{\text{high}} + \text{when or}_2 \text{\textit{vacant}} \text{ do tell } p = \rho_{\text{low}} \vdash \langle p = \rho_{\text{high}} , 1.0 \rangle \]

\[ \text{when ... do ... + when ... do ...} \big| \rho_{\text{high}}(\text{\textit{tell inc}(\text{rep})}) \vdash \langle \text{inc}(\text{rep}) \rangle , \rho_{\text{high}} \] (30)

Equation 30 says that new \text{reps} will appear with a probability \( \rho_{\text{high}} \) verifying the cooperative behavior between \text{OR2} \text{ and } \text{PR}.

5 Concluding remarks and Future Work

We have given orthogonal extensions of \text{ntcc} for modeling stochastic behavior of processes. In particular, we proposed two new operators: \( \ast_{\rho}P \) and \( \rhoP \). The first one expresses that \( P \) is eventually executed with probability \( \rho \) and the second one that \( P \) is executed in the current time unit with probability \( \rho \). We showed that both can be expressed in terms of existing \text{ntcc} constructs by including in the signature of the underlying constraint system of \text{ntcc} a probabilistic function \( \Phi(\rho) : [0,1] \rightarrow \text{bool} \) following a binomial distribution. The syntax and semantics associated to the temporal logic and proof system of \text{ntcc} were modified accordingly to account for the new stochastic constructs. We have shown that the proposed extension provides enough expressiveness to faithfully model a non trivial stochastic system. 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.

Finally, an inference system to prove probabilistic properties in the calculus was provided. This inference system is built on an \text{ntcc} linear- temporal logic extension by adding probabilities to each formula. For each process construction including our new processes \( \ast_{\rho}P \) and \( \rhoP \), we defined an inference rule to proof if a process \( P \) satisfies an specification (i.e a formula in the logic). We showed the use of this system by proving two properties in our lambda switch model.
In the short term we plan to implement a \texttt{ntcc} stochastic processes simulator to better visualize process behavior. We will build it up from a \texttt{ntcc} framework previously developed by our group ([5]). This will allow us to trace the evolution of protein concentrations so as to assess our model w.r.t. existing biological data for the lambda switch. Additionally, 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 such as the $N_A$ pump [2].

References


