ESTUDIO EXPLORATORIO DE LA APLICACIÓN DE LA PROGRAMACIÓN CONCURRENTE POR RESTRICCIONES EN BIOINFORMÁTICA

ALEJANDRO ARBELÁEZ RODRÍGUEZ
JULIAN EDUARDO GUTIÉRREZ SANTIAGO

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
SANTIAGO DE CALI
2006

ESTUDIO EXPLORATORIO DE LA APLICACIÓN DE LA PROGRAMACIÓN CONCURRENTE POR RESTRICCIONES EN BIOINFORMÁTICA

ALEJANDRO ARBELÁEZ RODRÍGUEZ
JULIAN EDUARDO GUTIÉRREZ SANTIAGO

Tesis de grado para optar al título de
Ingeniero de Sistemas y Computación

Director
CAMILO RUEDA CALDERÓN
Ingeniero de Sistemas y Computación

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
SANTIAGO DE CALI
2006

Santiago de Cali, Mayo 22 de 2006

Doctor
JORGE FRANCISCO ESTELA
Decano Académico de la Facultad de Ingeniería
Pontificia Universidad Javeriana
Ciudad

Certifico que el presente trabajo de grado, titulado "ESTUDIO EXPLORATORIO DE LA APLICACIÓN DE LA PROGRAMACIÓN CONCURRENTE POR RESTRICCIONES EN BIOINFORMÁTICA" realizado por ALEJANDRO ARBELÁEZ RODRÍGUEZ y JULIAN EDUARDO GUTIÉRREZ SANTIAGO, estudiantes de Ingeniería de Sistemas y Computación, se encuentra terminado y puede ser presentado para sustentación.

Atentamente,

Ing. CAMILO RUEDA CALDERÓN
Director del Proyecto

Santiago de Cali, Mayo 22 de 2006

Doctor
JORGE FRANCISCO ESTELA
Decano Académico de la Facultad de Ingeniería
Pontificia Universidad Javeriana
Ciudad

Por medio de ésta, presentamos a usted el trabajo de grado titulado "ESTUDIO EXPLORATORIO DE LA APLICACIÓN DE LA PROGRAMACIÓN CONCURRENTE POR RESTRICCIONES EN BIOINFORMÁTICA" para optar el título de Ingeniero de Sistemas y Computación.

Esperamos que este trabajo reúna todos los requisitos académicos y cumpla el propósito para el cual fue creado, y sirva de apoyo para futuros proyectos en la Universidad Javeriana relacionados con la materia.

Atentamente,

Alejandro Arbeláez

ALEJANDRO ARBELÁEZ RODRÍGUEZ

Julian Gutierrez

JULIAN EDUARDO GUTIÉRREZ SANTIAGO

ARTICULO 23 de la Resolución No. 13 del 6 de Julio de 1946
del Reglamento de la Pontificia Universidad Javeriana.

"La Universidad no se hace responsable por los conceptos emitidos por sus alumnos en sus trabajos de Tesis. Sólo velará porque no se publique nada contrario al dogma y a la moral católica y porque las Tesis no contengan ataques o polémicas puramente personales; antes bien, se vea en ellas el anhelo de buscar la Verdad y la Justicia"

Nota de Aceptación:

Aprobado por el comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero de Sistemas y Computación.

JORGE FRANCISCO ESTELA
Decano Académico de la Facultad de Ingeniería

MARÍA CONSTANZA PABÓN
Directora de la Carrera de Ingeniería de Sistemas y Computación

CAMILO RUEDA CALDERÓN
Director de Tesis

CARLOS OLARTE VEGA
Jurado

JUAN FRANCISCO DÍAZ
Jurado

Alejandro Arbeláez Rodríguez

A mis padres y hermanos por su apoyo incondicional.

Julian Eduardo Gutiérrez Santiago

A mi madre y a mi novia por su apoyo incondicional.

Agradecimientos

Los autores expresan sus agradecimientos:

- A Camilo Rueda, nuestro director de tesis, quien nos guió en el desarrollo de este trabajo de grado y permitió trabajar activamente como asistentes de investigación del Grupo AVISPA. Esta experiencia definitivamente colaboró en nuestra formación en la labor de investigación.
- A nuestros asesores, Guillermo Barreto y Andrés Jaramillo, por su colaboración desinteresada en la solución de dudas relacionadas con los fundamentos biológicos de este trabajo de grado.
- A Carlos Olarte, por su tiempo y asistencia. Sus sugerencias mejoraron significativamente la presentación de los resultados teóricos expuestos en este documento.
- A Gustavo Gutiérrez, quien fue un apoyo permanente cuando surgieron inconvenientes relacionados con el diseño e implementación de los programas en Mozart.
- A Jorge Pérez, quien desde un principio estuvo interesado en los aspectos teóricos de este trabajo de grado. Su colaboración fue fundamental en la investigación inicial sobre cálculos de procesos presentada en el capítulo 2.
- A Carlos Olarte, Rafael Jordán, Guillermo Barreto, Jorge Pérez y Hugo López por leer las primeras versiones de algunos capítulos de este documento.
- A todas aquellas personas que de una u otra forma nos colaboraron en el desarrollo de este trabajo de grado.

Agradecemos también a nuestra institución, la Pontificia Universidad Javeriana - Cali, ya que la realización de este trabajo de grado fue financiada por la Facultad de Ingeniería a través del proyecto de investigación "Modelamiento de Problemas en Ciencia y Tecnología usando Cálculos de Procesos Concurrentes". Este proyecto en sus fases I y II fue desarrollado entre los meses Enero de 2005 y Febrero de 2006 por el Grupo de Investigación AVISPA.

Índice general

Índice de cuadros

Índice de figuras

Resumen

La comprensión del *funcionamiento* de muchos sistemas biológicos normalmente está sujeta a una serie de suposiciones hechas a partir del comportamiento observable de los componentes que los conforman. Este problema es debido a que muchos fenómenos biológicos aún son entendidos *parcialmente*, o incluso desconocidos por completo. Como consecuencia, muchos modelos computacionales propuestos para su estudio no ofrecen los resultados esperados por la comunidad científica interesada en comprender el funcionamiento de este tipo de sistemas.

Este hecho motiva la búsqueda de modelos computacionales para describir el funcionamiento de los sistemas biológicos en los que sea transparente la representación de *información parcial*. Una posibilidad poco estudiada es el uso de *cálculos de procesos* pertenecientes al modelo de Programación Concurrente por Restricciones (CCP por sus siglas en inglés) para abordar este problema. Como consecuencia, en este trabajo se explora el uso de formalismos basados en *restricciones* para el estudio de sistemas biológicos. En particular, se utiliza el cálculo `ntcc` para *modelar* y *verificar* propiedades en la red de regulación genética del operón de la lactosa. Además, las características operacionales del cálculo sugieren la posibilidad de llevar a la práctica los modelos propuestos en la teoría mediante el uso de un lenguaje de programación basado en restricciones como Mozart-Oz.

Se proponen dos modelos en `ntcc` para la red de regulación estudiada, uno que describe su comportamiento a nivel *celular* y otro a nivel *molecular*. Dado que el modelo biológico a nivel celular está dado en términos de un sistema de ecuaciones diferenciales, se propone un esquema de ejecución de procesos `ntcc` y un conjunto de *encodings* para representar este tipo de estructuras matemáticas. Por otro lado, el modelo computacional a nivel molecular es construido haciendo uso de un conjunto de procesos *genéricos* y *paramétricos* propuestos por los autores para la representación del funcionamiento de redes de regulación genética. El modelo incluye la descripción de una mutación usando los operadores asíncronos y no deterministas de `ntcc`. En ambos casos los modelos son simulados para observar casos particulares de su comportamiento mediante el uso de una herramienta de simulación de procesos `ntcc`. Además, se verifica una propiedad de estabilidad sobre una de las entidades que conforman la red de regulación con el uso del sistema de pruebas asociado a `ntcc`.

Basados en los resultados obtenidos, se presenta un análisis que trata los principales conceptos estudiados en este trabajo. Además, se presentan ejemplos para exponer algunos problemas que pueden surgir al implementar los modelos hechos en `ntcc`. El análisis finaliza con la presentación de un conjunto de conclusiones que describen las ventajas y desventajas identificadas en el uso del modelo CCP, y en particular del cálculo `ntcc`, para el estudio de sistemas biológicos. Al final se proponen algunas direcciones de trabajo futuro. Por tanto, esta tesis se presenta como un trabajo en el *estado del arte* del estudio de sistemas biológicos desde la óptica de la *teoría de la concurrencia*, haciendo uso de un *cálculo de procesos* basado en restricciones.

Abstract

The study of the *behavior* of many biological systems is usually accomplished having presumptions coming from the observable conduct of their components or sub-systems. This problem arises due to a lot of biological phenomena are either *partially* understood or completely unknown. As a consequence, several computational models proposed for studying them do not provide the expected results to the scientific community interested in understanding the behavior of this kind of systems due to the assumptions made could be wrong.

This fact encourages the search of computational models to express as clear and easy as possible *partial information* in specifications describing the behavior of biological systems. An option slightly explored is the use of *process calculi* derived from the Concurrent Constraint Programming model to tackle this problem. Thus, in this work the employ of formalisms based on *constraints* for studying biological systems is investigated. In particular, the `ntcc` calculus is used for *modeling* the lactose operon genetic regulatory network and for *verifying* a property over it. Moreover, the `ntcc` operational features allow to *implement* the specifications proposed by means of a concurrent constraint programming language such as Mozart-Oz.

The regulatory network studied is modeled at both *cellular* and *molecular* levels of detail, resulting in two different specifications. On the one hand, since the biological model in cellular level is defined with a differential equations system, an `ntcc` processes running scheme and a set of *encodings* are proposed for representing this kind of mathematical structures. On the other hand, the computational specification in molecular level is built by means of a set of *generic* and *parametric* `ntcc` processes for describing the behavior of genetic regulatory networks. This specification includes the model of a genetic mutation described using asynchronous and non deterministic `ntcc` operators. Both specifications, the cellular and molecular one, are simulated to observe and analyze their behavior with an `ntcc` processes simulation tool. Additionally, a stability property in a biological entity that is part of the regulatory network is verified using the proof system associated with `ntcc`.

Based on the obtained results, an analysis of the main concepts discussed in this work is presented. Furthermore, two examples are presented to explain some issues that can occur when implementing `ntcc` specifications. The analysis finishes presenting a set of conclusions related with the advantages and disadvantages identified in the employ of the Concurrent Constraint Programming model and in particular the `ntcc` process calculus, in the study of biological systems. Finally, some activities and research directions are proposed as a future work. Thus, this thesis is presented as a work in the *state of the art* in the study of biological systems from the *concurrency theory* point of view, by using a constraint based *process calculus*.

1 Introducción

En este trabajo de grado se explora el uso de formalismos computacionales basados en el modelo de Programación Concurrente por Restricciones (CCP) para el estudio de sistemas biológicos. Más específicamente, se analiza el rol de los cálculos de procesos basados en restricciones en el contexto biológico utilizando la red de regulación genética del operón de la lactosa como caso de estudio.

1.1. Motivación

El progreso reciente de la biología molecular ha permitido describir la estructura de muchos de los componentes que conforman los sistemas biológicos (e.g., genes y proteínas) como entidades *aisladas*. Pero estas entidades no se encuentran solas, sino que hacen parte de complejas redes biológicas presentes en las células (e.g., las redes de regulación genética) cuyo propósito es definir y regular los procesos celulares. El reto ahora es pasar de la biología molecular a la biología sistémica (*systems biology* [Kit01a]) para comprender como estos componentes individuales (i.e., genes y proteínas) se integran dentro de las redes que conforman, y de esta forma poder inferir la manera en que ellos se *comportan* e *interactúan*.

Debido al gran tamaño y complejidad de estos sistemas, se ha considerado el uso de modelos computacionales que permitan realizar *abstracciones* de su *funcionamiento* para facilitar su estudio. En ese sentido, el comportamiento inherentemente concurrente de los sistemas biológicos ha hecho que se explore el uso de los *cálculos de procesos* como una posible opción para describir su funcionamiento. Razones que justifican lo anterior pueden ser inferidas a partir de las características de los cálculos. A continuación se presentan algunas de éstas características, las cuales han sido aprovechadas tanto en el contexto computacional como en el biológico.

Los cálculos de procesos son formalismos computacionales comúnmente usados para la descripción de sistemas *concurrentes*, donde las ideas de *proceso* e *interacción* son la base de los modelos. Los sistemas son entendidos como procesos complejos compuestos de otros más simples, siguiendo un enfoque de construcción *composicional*. Tal enfoque es complementado con las construcciones *matemáticas* provistas por los cálculos, las cuales definen la manera en que los procesos interactúan entre ellos y con su ambiente. De esta forma, los fundamentos matemáticos subyacentes a los cálculos permiten *modelar* y *verificar* propiedades de un sistema, haciendo posible una mejor comprensión de su *comportamiento*.

Estas características de los cálculos han sido una motivación para usarlos en una amplia gama de áreas del conocimiento. Por ejemplo, para el modelado de sistemas distribuidos [Mil89] y sistemas reactivos [MPV02], la composición musical [RV05, RV02], la descripción de protocolos de comunicación segura [AG99] y la construcción de lenguajes de programación [DRV$^+$01] por nombrar algunos ejemplos. El interés de la comunidad científica en los cálculos de procesos también se ha evidenciado en el desarrollo de extensiones de los cálculos existentes, con el objetivo de poder razonar sobre una gran variedad de conceptos comunes en los sistemas reales, tales como tiempo [SJG94, Val03], movilidad [Mil99] y comportamiento estocástico [Pri95, GJP99] y probabilístico [SGJ97, HP00].

En biología los cálculos de procesos también han sido usados. La mayoría de estos trabajos han sido realizados usando formalismos tales como el cálculo π [Mil99] y el cálculo Ambient [CG98], o algunas de sus extensiones. En [RS04, RPS$^+$04, PRSS01] pueden encontrarse algunos de los trabajos más representativos. En otros casos, con el objetivo de modelar de forma más precisa ciertos fenómenos biológicos, se han propuesto nuevos cálculos diseñados específicamente para la descripción de procesos biológicos, por ejemplo, interacciones entre membranas [Car04], interacciones entre proteínas [DL04] y la reversibilidad de los procesos biomoleculares [KD03].

Sin embargo, ninguno de estos cálculos ha atacado el problema del modelado de sistemas biológicos tendiendo en cuenta que sólo se posee *información parcial* de su comportamiento a *nivel de sistema*. Esto es debido a que muchos fenómenos biológicos aún son materia de estudio en la biología sistémica. Por tal motivo, en este trabajo de grado se explora el uso de la Programación Concurrente por Restricciones como un posible modelo computacional para la representación de este tipo de información. En ese sentido, nuestro interés está centrado en el estudio de cálculos de procesos basados en *restricciones*, y como ellos presentan ventajas y desventajas que facilitan y limitan el estudio de sistemas biológicos.

En particular, consideramos que dentro del modelo de Programación Concurrente por Restricciones, el cálculo `ntcc` [MPV02] presenta un conjunto de características interesantes para *describir*, *simular* y *verificar* propiedades de sistemas biológicos complejos como las redes de regulación genética. Por ejemplo, el uso natural de *agentes concurrentes* (i.e., procesos) para modelar entidades biológicas, la noción explícita de *tiempo* para describir la evolución de sistemas biológicos dinámicos, *restricciones* como una forma de expresar información parcial sobre el valor de las variables, *comportamiento asíncrono y no determinista* para modelar información parcial sobre el funcionamiento del sistema y la posibilidad de incluir *infomación cuantitativa* con el objetivo de parametrizar los modelos con valores reales provenientes de la experimentación.

Además, este esquema de computación *teórico*, construido a partir de un cálculo de procesos basado en el modelo de Programación Concurrente por Restricciones, puede ser llevado a la práctica siguiendo un enfoque de *programación declarativo* mediante el uso de un lenguaje de programación como Mozart-Oz [Smo95], para observar el comportamiento a *nivel de sistema* de los modelos propuestos en el cálculo.

1.2. Objetivos y alcances

Este trabajo de grado presenta los siguientes objetivos y alcances:

General

Explorar el uso de la Programación Concurrente por Restricciones para el modelamiento de problemas en Bioinformática.

Específicos

1. Modelar con Programación Concurrente por Restricciones una red de regulación genética de un operón.
2. Implementar el modelo de la red de regulación para observar el funcionamiento de casos particulares.
3. Analizar ventajas y desventajas del uso de la Programación Concurrente por Restricciones en el modelamiento de redes de regulación genética bajo el enfoque de la biología sistémica.

Alcances

1. Un modelo discreto en CCP de una red de regulación genética del operón de la lactosa.
2. Un programa en Mozart, usando los sistemas de restricciones existentes, que entregue las señales de la red de regulación genética a partir de unas condiciones iniciales.
3. Una serie de conclusiones relacionadas con las ventajas y desventajas del uso de la Programación Concurrente por Restricciones basadas en el modelo de la red de regulación genética implementado en Mozart.

1.3. Contribuciones

Las principales contribuciones de este trabajo de grado se presentan a continuación:

1. Un esquema de ejecución de procesos y un conjunto de *encodings* de ecuaciones diferenciales de primer orden en `ntcc`, los cuales permiten representar el comportamiento de *sistemas continuos* usando las características temporales discretas del cálculo.
2. Un conjunto de herramientas genéricas que permiten *modelar*, *simular* y *verificar* propiedades en redes de regulación genética. Estas herramientas incluyen:
   - Un conjunto de procesos *genéricos* y *paramétricos* en `ntcc` que *abstraen* los principales elementos de las redes de regulación genética para poder modelar su *comportamiento*.
   - **ntccSim**, una herramienta multiplataforma de simulación de procesos `ntcc` para *observar* el *funcionamiento* de casos particulares en la redes de regulación genética. Esta herramienta presenta una serie de modificaciones y mejoras del simulador de procesos `ntcc` estocásticos usado en [AGOR06].
   - Un enfoque teórico de *verificación* de propiedades biológicas basado en una *lógica temporal lineal* usando el sistema de pruebas de `ntcc`.
3. Un estudio del *estado del arte* sobre el modelado de sistemas biológicos usando cálculos de procesos. Los principales resultados de tal estudio se encuentran publicados en [GPR05].

Gracias al desarrollo de las contribuciones 1 y 2 se plantearon *dos* modelos de la red de regulación genética del operón de la lactosa, uno a nivel *celular* y otro a nivel *molecular*. Ambos modelos fueron implementados en Mozart y simulados con **ntccSim**. Adicional a los resultados relacionados con el modelado e implementación (i.e., simulación) de la red de regulación genética del operón de la lactosa, establecidos al inicio de este trabajo de grado, se realizó una *prueba formal* basada en la *lógica* asociada a `ntcc`. Finalmente, la contribución 3 constituye el punto de partida para la definición de las conclusiones relacionadas con las ventajas y desventajas identificadas en el uso de la Programación Concurrente por Restricciones para el estudio de sistemas biológicos.

Publicaciones asociadas con este trabajo

Algunos de los resultados presentados en este trabajo de grado han sido publicados previamente, nacional e internacionalmente, como contribuciones arbitradas (*peer-reviewed*). Estas publicaciones dan fe, por un lado, de la validez de los resultados obtenidos, y por otro, de la profundidad con que fueron desarrollados los temas tratados en este trabajo de grado.

- Julian Gutiérrez, Jorge A. Pérez y Camilo Rueda. *Modelamiento de Sistemas Biológicos usando Cálculos de Procesos Concurrentes*. Revista Epiciclos, 4, 2005. [GPR05].
- Julian Gutiérrez, Jorge A. Pérez, Camilo Rueda y Frank D. Valencia. *Using a Timed Concurrent Constraint Process Calculus for Modeling and Verifying Biological Systems*. MeCBIC'06 (Parte de ICALP'06). Electronic Notes in Theoretical Computer Science (ENTCS) series. Italia. 2006. [GPRV06].
- Alejandro Arbeláez, Julian Gutiérrez, Carlos A. Olarte y Camilo Rueda. *A Generic Framework to Model, Simulate and Verify Genetic Regulatory Networks*. CLEI'06. Memorias de la Conferencia. Chile. 2006. [AGOR06].

1.4. Estructura del documento

Capítulo 2. Este capítulo inicia presentando la sintaxis, semántica operacional y lógica asociada del cálculo `ntcc`. Además, se describen las principales características de **ntccSim**, una herramienta de simulación de procesos `ntcc` en Mozart. Después se explican brevemente algunos conceptos sobre biología molecular y biología sistémica y se propone una clasificación de problemas biológicos estudiados desde la informática. El capítulo concluye con un estudio del estado del arte sobre el uso de algunos modelos de computación formal distintos a `ntcc` para el estudio de sistemas biológicos.

Capítulo 3. Este capítulo está dedicado a la descripción a nivel de sistema de la red de regulación genética del operón de la lactosa. Se describe su estructura y funcionamiento a nivel poblacional, celular y molecular. Además, se describen algunas mutaciones conocidas de este sistema biológico. Una de ellas es modelada más adelante en el capítulo 5 haciendo uso de los operadores asíncronos y no deterministas de `ntcc`. Al final del capítulo se referencian algunos modelos biológicos del operón de la lactosa, los cuales están basados en características lógicas, continuas e híbridas (i.e., discretas y continuas) identificadas en el comportamiento regulatorio de esta red de control celular.

Capítulo 4. En este capítulo se propone un esquema de ejecución de procesos `ntcc` y un conjunto de *encodings* que permiten modelar el comportamiento de sistemas continuos descritos con ecuaciones diferenciales ordinarias (ODEs). Además, se presentan algunos ejemplos de aplicación para clarificar el uso de los *encodings* propuestos. Posteriormente, el modelo para representar este tipo de sistemas se extiende para poder describir otros con dependencias temporales más complejas. Este modelo extendido es usado para describir el comportamiento a nivel celular de la red de regulación genética del operón de la lactosa. El capítulo termina presentado la implementación del modelo en Mozart y gráficas de su simulación con **ntccSim**.

Capítulo 5. En este capítulo se presentan un conjunto de procesos genéricos y paramétricos en `ntcc` para modelar el funcionamiento de las redes de regulación genética. Estos procesos son utilizados para describir el comportamiento a nivel molecular de la red de regulación del operón de la lactosa. Además, se propone un modelo en `ntcc` de una posible mutación en el sistema de regulación. Posteriormente, se muestran algunas gráficas de la simulación del sistema con y sin mutación. Al final del capítulo se presenta una prueba formal relacionada con el comportamiento de una entidad biológica de la red de regulación usando la lógica temporal lineal y el sistema de pruebas asociados con `ntcc`.

Capítulo 6. Este capítulo está dedicado a resaltar los aspectos más relevantes relacionados con este trabajo de grado. Por lo tanto, el capítulo inicia con un análisis del rol de los cálculos de procesos en el modelado y verificación de propiedades de sistemas biológicos. Posteriormente, se destacan la importancia del tiempo como elemento fundamental para la descripción de sistemas dinámicos y de las características del modelo de Programación Concurrente por Restricciones para modelar información parcial. Después, algunas consideraciones prácticas, fruto de la implementación de los modelos en Mozart, son descritas con la ayuda de ejemplos. El capítulo finaliza con la presentación de un conjunto de conclusiones y algunas direcciones de posible trabajo futuro.

2 Fundamentos Computacionales y Biológicos

Este capítulo es un marco de referencia para la comprensión de los principales tópicos tratados en este trabajo de grado. En él se presentan los conceptos fundamentales sobre el modelo de Programación Concurrente por Restricciones, los cálculos de procesos, la bioinformática, la biología molecular y la biología sistémica. Además, con el ánimo de promover una lectura crítica y sobre todo constructiva por parte del lector, se presenta un estudio del estado del arte sobre el uso de cálculos de procesos para el modelado de sistemas biológicos.

2.1. Programación Concurrente por Restricciones (CCP)

La Programación Concurrente por Restricciones (CCP) [SRP91] es un poderoso formalismo computacional para modelar sistemas concurrentes. En CCP los agentes concurrentes (i.e., *procesos*) que componen los sistemas interactúan entre si por medio de operaciones *tell* y *ask* que adicionan y consultan *información parcial* de las variables que representan los sistemas. Tal información está dada en forma de *restricciones* (e.g., $x + y > 10$) sobre los posibles valores que una variable puede tomar.

Una de las características más importantes de CCP es que este formalismo para la concurrencia combina en el mismo modelo dos visiones distintas de los procesos que componen un sistema: la visión *operacional* de los *cálculos de procesos* con una visión *declarativa* basada en la *lógica*. Esta combinación le permite al modelo CCP beneficiarse de la gran cantidad de técnicas existentes para modelar y razonar de los cálculos de procesos y la lógica.

2.1.1. De CCP a ntcc

El cálculo de procesos `ntcc` [MPV02, Val03] es un formalismo particular perteneciente al modelo CCP, que extiende este último con las nociones de *tiempo discreto*, *comportamiento asíncrono* y *no determinismo*. Los operadores del cálculo están definidos de tal forma que sea posible capturar, de forma natural, las principales características de sistemas temporales.

En particular, `ntcc` permite modelar: (i) *retardos unitarios* para forzar la ejecución de un proceso a la siguiente unidad de tiempo, (ii) *time-outs* para esperar la ejecución de una unidad de tiempo y de esta forma poder determinar la ausencia o presencia de información (parcial) que

permita decidir si se ejecuta o no un proceso en la siguiente unidad de tiempo, (iii) *sincronía* para controlar o coordinar la ejecución concurrente de múltiples procesos, (iv) *comportamiento infinito* para representar la ejecución persistente de un proceso, (v) *asincronía* para retardar de forma finita pero indeterminada la ejecución de un proceso,y (vi) *no determinismo* para formalizar el hecho de que un sistema puede evolucionar de formas diversas dadas las mismas condiciones iniciales.

Estas características o comportamientos que `ntcc` permite modelar se traducen en ventajas reales en diversos ámbitos. Por un lado, la visión *operacional* de los procesos en `ntcc`, soportada por las semánticas que tiene definido el cálculo (una operacional y otra denotacional), permiten determinar el comportamiento de los procesos que interactúan en el sistema modelado. Esto se traduce en posibilidades reales de llevar a la práctica, en forma de programas en un lenguaje de programación, especificaciones hechas con el cálculo. Por otro lado, la visión *declarativa* de los procesos permite razonar sobre ellos haciendo uso de la lógica temporal lineal asociada a `ntcc` [Val05] y de un sistema de inferencia. De esta manera, es posible formular especificaciones temporales de procesos `ntcc` y probar que un proceso dado satisface una fórmula en la lógica.

2.1.2. ntcc, un cálculo de procesos por restricciones

En esta sección se presenta una descripción de la sintaxis, semántica operacional y lógica asociada a `ntcc`. Dado que el cálculo es paramétrico a un *sistema de restricciones*, se inicia con una definición de esta estructura matemática.

Sistema de restricciones

Un sistema de restricciones es una pareja (Σ,Δ), donde Σ es un conjunto de constantes, funciones y predicados y Δ es una teoría de primer orden sobre Σ, es decir, Σ es un conjunto de sentencias de primer orden con al menos un modelo. De esta forma, las restricciones pueden ser pensadas como fórmulas de primer orden sobre Σ.

Dado un sistema de restricciones (Σ,Δ), se define $\mathcal{L}$ como el lenguaje de primer orden (Σ,$\mathcal{V}$,$\mathcal{S}$), donde $\mathcal{V}$ es un conjunto de variables y $\mathcal{S}$ es un conjunto que contiene los símbolos $\neg, \wedge, \Rightarrow, \exists$, `true` y `false`, los cuales denotan las operaciones lógicas negación, conjunción, implicación, cuantificación existencial y los predicados verdadero y falso respectivamente. Las restricciones, denotadas por $c, d, \ldots$ son fórmulas de primer orden sobre el lenguaje $\mathcal{L}$. Se dice que c deduce d en Δ, escrito $c \models_\Delta d$, si $c \Rightarrow d$ es cierto en todos los modelos posibles de Δ. Por motivos operacionales se requiere que $\models$ sea decidible. Además, las restricciones se acumulan en un *almacén* que contiene toda la información del sistema; dicho almacén de restricciones o *store* es *monotónico* en cuanto la información conocida del sistema únicamente crece con el tiempo. Por ejemplo, si en un momento dado $store \models (x < 7)$, más adelante sólo serán posibles valores para x tales que x sea menor que 7 (e.g., $store \models x = 3$).

Sintaxis

A continuación se enumeran brevemente las principales características de `ntcc`, proporcionando detalles de su sintaxis.

Comunicación La comunicación en `ntcc` sigue las ideas del modelo CCP: es posible añadir y obtener información parcial de las variables del sistema en términos de operaciones *ask* y *tell*. Más precisamente, `ntcc` proporciona los siguientes procesos:

- **tell**(c), que añade la restricción c al almacén de restricciones.
- **when** c **do** P, que ejecuta el proceso P solo si c puede deducirse a partir de la información en el almacén de restricciones (i.e., $store \models c$).

Alternativas de Ejecución Dado que la información parcial permite representar diferentes valores para las variables de un sistema, resulta conveniente poder modelar diversas escogencias y alternativas de acción. En `ntcc` es posible expresar escogencias no deterministas extendiendo el proceso **when** c **do** P de la siguiente manera:

$$\sum_{i \in I} \textbf{when } c_i \textbf{ do } P_i$$

Esta expresión representa un proceso que debe escoger de forma no determinista un P_j $(j \in I)$ cuya guarda c_j se deduce del almacén de restricciones (i.e., $store \models c_j$). Esta escogencia es excluyente. Por motivos de claridad, se usa la abreviación $\sum_{i \in I} P_i$ para representar procesos de la forma $\sum_{i \in I} \textbf{when } true \textbf{ do } P_i$. Además, se usa el operador "+" para escogencias binarias y se denota el proceso nulo (i.e., cuando $I = \emptyset$) mediante **skip**.

Concurrencia La ejecución concurrente de dos procesos P y Q se representa por la *composición paralela* $P \parallel Q$. Se usa la notación $\prod_{i \in I} P_i$, donde I es un conjunto de índices de procesos, para expresar la composición paralela de todos los procesos P_i.

Información Local El proceso **local** x **in** P se comporta como el proceso P, excepto que toda la información de x producida por P únicamente puede ser vista por P y la información de x producida por otros procesos no puede ser vista por P. Se usa la abreviación **local** $x_1, x_2, ..., x_n$ **in** P para denotar el proceso **local** x_1 **in** (**local** x_2 **in** (... (**local** x_n **in** P) ...)).

Dependencias Temporales Esta es una de las características más interesantes de `ntcc`, pues el cálculo provee construcciones que permiten representar dependencias temporales (e.g., retardos unitarios o *time-outs*) entre procesos. Es decir, se toma una determinada acción dependiendo de la presencia o ausencia previa de restricciones. Existen dos operadores de este tipo en `ntcc`:

- **next** P, que representa la activación del proceso P en la siguiente unidad de tiempo.
- **unless** c **next** P, que activa P en la siguiente unidad de tiempo siempre y cuando no se pueda deducir c del almacén de restricciones actual.

Se usa la notación $\mathbf{next}^n\ (P)$ para abreviar el proceso $\mathbf{next}\,(\mathbf{next}\,(\ldots(\mathbf{next}\,P)\ldots))$, donde **next** se repite n veces.

Comportamiento Asíncrono e Infinito Con el operador "$\star$" es posible extender de manera arbitraria (pero finita) la espera o retardo en la ejecución de un proceso. De este modo, es posible modelar *comportamiento asíncrono* en un proceso. Así por ejemplo, $\star\,\mathbf{tell}(c)$ representa la eventual adición de la restricción c, sin información adicional con respecto al instante de tiempo en que esto ocurrirá. Por otra parte, el operador "!" define la ejecución infinita de un proceso dado, ejecutando una copia del proceso en cuestión en toda unidad de tiempo futura. $!P$ puede ser representado por $P \parallel \mathbf{next}\,P \parallel \mathbf{next}^2 P \parallel \ldots$, es decir, la ejecución de infinitas copias de P, una por cada unidad de tiempo.

A continuación, en el cuadro 2.1, se presenta de forma concreta la sintaxis de `ntcc`, la cual condensa la ideas expuestas anteriormente.

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

Cuadro 2.1: Sintaxis de ntcc

Procedimientos y Recursión Usando `ntcc` es posible codificar, con los operadores del cálculo presentados en el cuadro 2.1, definiciones de procesos como procedimientos y recursión. En el primer caso se usa una definición de la forma $A(x) \stackrel{\text{def}}{=} P_x$ donde P_x es un proceso que usa la variable x. De esta forma, por ejemplo, un llamado al procedimiento A con el parámetro c, denotado por $A(c)$, puede ejecutar el proceso P_x una vez la variable x es sustituida por c. Por otro lado, es posible definir llamados a procedimientos recursivos de la forma $q(x) \stackrel{\text{def}}{=} P_q$, donde q es el nombre de un proceso y P_q llama a q solo dentro del alcance de un "**next**". Tal como se define en [Val02], se usa *paso de parámetros por valor* en los llamados a procedimientos recursivos. Finalmente, las definiciones dadas pueden ser extendidas fácilmente al caso de definiciones con un número arbitrario de parámetros (ver [Val02] para más detalles).

Semántica Operacional

El comportamiento de los procesos en `ntcc` es formalizado mediante su semántica operacional, la cual considera *transiciones* entre *configuraciones* de parejas de la forma $\langle P, c\rangle$, donde P es un proceso `ntcc` y c es un almacén de restricciones (i.e., *store*).

Las *transiciones* de la semántica operacional están dadas por las relaciones $\longrightarrow$ y $\Longrightarrow$ definidas en el cuadro 2.2. Por un lado, una transición *interna* (no observable) $\langle P, d\rangle \longrightarrow \langle P', d'\rangle$ debe ser leída como "P con *store* d reduce, en un paso interno, a P' con *store* d'". Por otro, una transición *externa* (observable) $P \xRightarrow{(c,d)} R$ debe ser leída como "P con una entrada c del ambiente reduce, en una unidad de tiempo, a R con salida d al ambiente". Las transiciones observables son obtenidas a partir de un conjunto de transiciones internas.

Interpretación de la ejecución de procesos `ntcc` Considere una secuencia infinita de transiciones observables (o *ejecuciones*) $P = P_1 \xRightarrow{(s_1,r_1)} P_2 \xRightarrow{(s_2,r_2)} P_3 \xRightarrow{(s_3,r_3)} \ldots$. Esta secuencia puede ser interpretada como una *interacción* entre el sistema P y un ambiente, donde en el tiempo i el ambiente da un estimulo s_i y P_i produce r_i como respuesta. Si $\alpha = s_1.s_2.s_3\ldots$ y $\alpha' = r_1.r_2.r_3\ldots$, entonces la interacción es representada como $P \xRightarrow{(\alpha,\alpha')} \omega$.

Alternativamente, si $\alpha = \mathtt{true}^\omega$, se puede interpretar la ejecución como una interacción entre los componentes paralelos en P sin la influencia de un ambiente externo, es decir, cada componente es parte del ambiente de los otros. En este caso α es llamada la secuencia de entrada *vacía* y α' es considerada como una *observación temporal* de tal interacción en P. Se dice que la *postcondición más fuerte* de un proceso P, denotada por $sp(P)$, describe el conjunto todas las secuencias infinitas que P puede producir. Más precisamente, $sp(P) = \{\alpha' \mid \text{para algún } \alpha : P \xRightarrow{(\alpha,\alpha')} \omega\}$.

$$\text{TELL}\ \frac{}{\langle \mathbf{tell}(c), d\rangle \longrightarrow \langle \mathbf{skip}, d \wedge c\rangle} \qquad \text{SUM}\ \frac{d \models c_j\ j \in I}{\langle \sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i, d\rangle \longrightarrow \langle P_j, d\rangle}$$

$$\text{PAR}\ \frac{\langle P, c\rangle \longrightarrow \langle P', d\rangle}{\langle P \parallel Q, c\rangle \longrightarrow \langle P' \parallel Q, d\rangle} \qquad \text{LOC}\ \frac{\langle P, c \wedge \exists_x d\rangle \longrightarrow \langle P', c'\rangle}{\langle (\mathbf{local}\ x, c\ \mathbf{in}\ P), d\rangle \longrightarrow \langle (\mathbf{local}\ x, c'\ \mathbf{in} P'), d \wedge \exists_x c'\rangle}$$

$$\text{UNL}\ \frac{}{\langle \mathbf{unless}\ \ c\ \ \mathbf{next}\ \ P, d\rangle \longrightarrow \langle \mathbf{skip}, d\rangle}\ \text{if } d \models c$$

$$\text{REP}\ \frac{}{\langle !\, P, d\rangle \longrightarrow \langle P \parallel \mathbf{next}\, !\, P, d\rangle} \qquad \text{STAR}\ \frac{}{\langle \star\, P, d\rangle \longrightarrow \langle \mathbf{next}^{\,n} P, d\rangle}\ \text{if } n \geq 0$$

$$\text{STR}\ \frac{\gamma_1 \longrightarrow \gamma_2}{\gamma_1' \longrightarrow \gamma_2'}\ \text{if } \gamma_1 \equiv \gamma_1' \text{ and } \gamma_2 \equiv \gamma_2'$$

$$\text{OBS}\ \frac{\langle P, c\rangle \longrightarrow^* \langle Q, d\rangle \not\longrightarrow}{P \xRightarrow{(c,d)} R}\ \text{if } R \equiv F(Q)$$

Cuadro 2.2: Semántica Operacional de ntcc

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

Cuadro 2.3: Sistema de pruebas de propiedades temporales lineales de procesos ntcc

Lógica Temporal Lineal en ntcc

`ntcc` tiene asociada una lógica temporal lineal que permite expresar propiedades de los procesos. Sean $A, B, \ldots \in \mathcal{A}$ fórmulas en esta lógica definidas por la gramática:

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

donde c es una restricción cualquiera. Los símbolos $\dot{\Rightarrow}$, $\dot{\neg}$ y $\dot{\exists}$ representan los operadores lógicos temporales implicación, negación y cuantificación existencial respectivamente. No confundir estos símbolos con los operadores lógicos (no temporales) $\Rightarrow$, $\neg$ y $\exists$ del sistema de restricciones (ver capítulo 5 de [Val02] para más detalles). Los símbolos $\bigcirc$, $\Box$ y $\Diamond$ representan los operadores temporales *next*, *always* y *sometime*. Por ejemplo, dada una propiedad A (e.g., $x > 7$) el significado de las fórmulas $\bigcirc A$, $\Box A$ y $\Diamond A$ es que la propiedad se mantiene o tiene lugar, en la *siguiente* unidad de tiempo, *siempre* y *eventualmente* de forma respectiva. Además, se usan las abreviaciones $A \dot{\vee} B$ y $A \dot{\wedge} B$ para representar $\dot{\neg} A \dot{\Rightarrow} B$ y $\dot{\neg}(\dot{\neg} A \dot{\vee} \dot{\neg} B)$ respectivamente.

Se dice que el proceso P satisface A si toda secuencia infinita de estados (i.e., restricciones en `ntcc`) que P pueda posiblemente producir satisface la propiedad expresada por A. Un sistema de pruebas para aserciones de la forma $P \vdash A$, cuyo significado es que P satisface A, es dado en el cuadro 2.3. Finalmente, se presenta el *Lema* 1 usado en las derivaciones de las pruebas hechas con la lógica.

Lema 1 (Nielsen et al. [Val02]) *Para todo proceso P se cumple:*

1. $P \vdash \dot{\mathtt{true}}$, 2. $P \nvdash \dot{\mathtt{false}}$, 3. $\dfrac{P \vdash A}{P \parallel Q \vdash A}$ 4. $\dfrac{P \vdash A \qquad P \vdash B}{P \vdash A \dot{\wedge} B}$.

2.1.3. Simulación de Procesos ntcc en Mozart-Oz

Aprovechando las características *operacionales* de los cálculos de procesos basados en el modelo CCP, y en particular de los procesos en `ntcc`, el grupo de investigación AVISPA [Gro06] desarrollo un simulador de procesos `ntcc` llamado **ntccSim** en el lenguaje de programación Oz. De esta forma, usando **ntccSim** es posible ejecutar procesos `ntcc` en Mozart [Con06], la implementación de Oz [Smo95].

ntccSim esta construido de tal forma que, siguiendo las reglas de la *semántica operacional* de `ntcc`, es posible simular el comportamiento de la mayoría de los procesos básicos del cálculo, además de poder definir procedimientos y recursión. Una característica importante de **ntccSim** es que permite la definición de especificaciones en `ntcc` usando múltiples sistemas de restricciones sobre el mismo modelo, mejorando sustancialmente la precisión en la representación de los datos. Otro aspecto a destacar es que la implementación o simulación de especificaciones en `ntcc` usando **ntccSim** sigue un estilo de programación declarativa.

En el cuadro 2.4 se muestra la manera de escribir procesos `ntcc` en **ntccSim** según las siguientes definiciones. Sea $[\![.]\!]$ una función que asocia a cada elemento de `ntcc` (e.g., una restricción, un proceso, un procedimiento, etc.) una implementación en Mozart-Oz. Además, sea P un proceso `ntcc`, c una restricción cualquiera, A un nombre de un procedimiento y Q un nombre de un procedimiento recursivo. Finalmente sea $< Sentencias - Oz >$ sentencias propias del lenguaje que permiten imponer restricciones en Mozart-Oz. Nótese que falta la descripción del proceso **local** x **in** P.

Consideraciones Prácticas Es importante destacar que algunos de los procesos `ntcc` en Mozart-Oz tienen como parámetro la variable Root (**tell**, **when** y **unless**), la cual es un registro de Mozart que representa las variables del sistema. La variable Root tiene dos campos: *current* y *residual*. El primero representa las variables en la unidad de tiempo actual, mientras que el segundo las variables en la unidad de tiempo anterior. El siguiente procedimiento en Mozart-Oz permite simular procesos `ntcc` usando **ntccSim**:

```
{NTCC.simulate +ListaProc +DescVar +TotalUnidades +Proc}
```

donde `ListaProc` es una lista de procesos `ntcc` que se ejecutan (i.e., simulan) concurrentemente, `DescVar` es la representación de las variables en Mozart-Oz (e.g., una tupla, un registro, etc.), `TotalUnidades` el número total de unidades a simular y `Proc` un procedimiento en Mozart-Oz con la variable Root como parámetro para realizar acciones de usuario (e.g., imprimir mensajes, guardar en archivos, etc.) entre la simulación de dos unidades de tiempo en `ntcc`.

Para mayor claridad se presenta a continuación un ejemplo del uso de **ntccSim**:

```
declare
  [NTCC] = {Module.link ['x-ozlib://ntccsim/ntccSim.ozf']}
  TimeUnits = 10
  Vars = var(x:_ y:_ z:_)
```

$[\![c]\!] = < Sentencias - Oz >$
$[\![\textbf{tell}(c)]\!] = \text{tell}(\textbf{proc}\ \{\$\ \text{Root}\}\ [\![c]\!]\ \textbf{end})$
$[\![\textbf{when}\ c\ \textbf{do}\ P]\!] = \text{when}(\textbf{proc}\ \{\$\ \text{Root}\}\ [\![c]\!]\ \textbf{end}\ [\![P]\!])$
$[\![\sum_{i=1}^{n} \textbf{when}\ c_i\ \textbf{do}\ P_i]\!] = \text{sum}([\![\textbf{when}\ c_1\ \textbf{do}\ P_1]\!] \ldots [\![\textbf{when}\ c_n\ \textbf{do}\ P_n]\!])$
$[\![P \parallel Q]\!] = \text{par}([\![P]\!]\ \ [\![Q]\!])$
$[\![\prod_{i=1}^{n} P_i]\!] = \text{par}([\![P_1]\!] \ldots [\![P_n]\!])$
$[\![\textbf{next}\ P]\!] = \text{next}([\![P]\!])$
$[\![\textbf{unless}\ \ c\ \ \textbf{next}\ \ \ P]\!] = \text{unless}(\textbf{proc}\{\$\ \text{Root}\}\ [\![c]\!]\ \textbf{end}\ [\![P]\!])$
$[\![!P]\!] = \text{rep}([\![P]\!])$
$[\![\star P]\!] = \text{star}([\![P]\!])$
$[\![\mathbf{A}(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P]\!] = \textbf{fun}\ \{\text{A X1} \ldots \text{Xn}\}\ [\![P]\!]\ \textbf{end}$
$[\![\mathbf{Q}(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P_Q]\!] = \textbf{fun lazy}\ \{\text{Q X1} \ldots \text{Xn}\}\ [\![P_Q]\!]\ \textbf{end}$

Cuadro 2.4: Procesos ntcc en Mozart-Oz

```
Vars:::0#100
P = when(proc{$ Root} skip end tell(proc{$ Root} Root.current.x =: 1 end))
Q = when(proc{$ Root} Root.current.y =: 1 end P)
R = sum(P Q)
S = par(P Q)
T = par(P Q R S)
U = next(P)
V = unless(proc{$ Root} Root.current.x =: 1 end P)
W = rep(P)
J = star(P)
fun{A X1} tell(proc{$ Root} Root.current.y =: X1 end) end
fun lazy{B X1}
   par(tell(proc{$ Root} Root.current.z =: X1 end) next({B X1+1}) )
end
PrintVars = proc{$ Root} {Browse Root.x#Root.y#Root.z} end
{NTCC.simulate [T U V W J {A 7} {B 3}] Vars TimeUnits PrintVars}
```

2.2. Problemas biológicos en informática

Los problemas biológicos estudiados desde la óptica de la informática son tratados por la *bioinformática* [Les02], una área de investigación que ha avanzado con extrema rapidez en los últimos años. Esto se ha debido a los desarrollos, tanto teóricos como prácticos, de la *biología* y las *ciencias de la computación*. Como tal, la bioinformática usa técnicas propias de las ciencias de la computación para modelar, simular y razonar sobre problemas provenientes de la biología.

Una área de estudio particularmente compleja en la que se han concentrado muchos de los esfuerzos en bioinformática es la biología sistémica (*systems biology*) [Kit01a]. Esta área de estudio, fruto de la evolución de la *biología molecular*, sienta sus bases en la concepción de los sistemas biológicos como entidades en las que existe de forma integrada *interacción multinivel* de los elementos que los componen. Por tal razón, desarrollar técnicas de modelado que permitan precisar tales interacciones en un sistema biológico a partir del comportamiento de sus subsistemas es un reto inmediato.

A continuación se presenta una breve descripción de los fundamentos sobre los que se basa la biología molecular. Posteriormente, se da una clasificación de problemas según los principales focos de estudio de esta área del conocimiento y al final se exponen las características más relevantes relacionadas con el estudio de sistemas desde la óptica de la biología sistémica.

2.2.1. La biología molecular

Los procesos moleculares dentro de la célula siguen un curso natural, una regla biológica denominada *el dogma central* de la biología molecular [Lew00]. Este dogma establece que la información biológica no puede ser transferida de proteínas a proteínas ni de proteínas a ácidos nucleicos (ADN y ARN). Pocas excepciones son conocidas en la naturaleza. Una de ellas se da en los *priones*, en los que es posible el flujo de información de proteínas a proteínas.

Así, la transferencia *estándar* de información biológica puede darse de tres formas distintas según el material inicial y final de cada proceso (ver Figura 2.1). Estas formas configuran los procesos de replicación, transcripción y traducción. En la *replicación* la información biológica pasa del ADN al ADN, en la *transcripción* del ADN al ARN y en la *traducción* del ARN a proteínas.

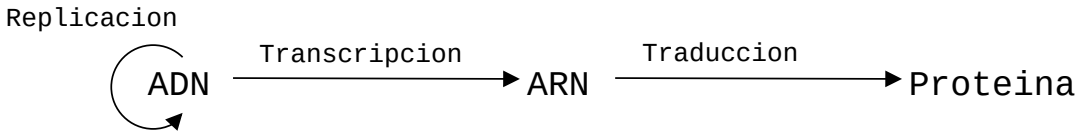


Figura 2.1: Dogma Central de la Biología Molecular

Es necesario anotar que en algunos organismos puede existir flujo de información desde el ARN hacia el ADN. Esta transferencia de información es perfectamente posible según el dogma, ya que tiene lugar entre ácidos nucleicos. Sin embargo, esa ruta no es la más común en los organismos vivos.

Replicación Este proceso consiste en generar dos cadenas de ADN hijas a partir de una cadena madre. La replicación de la información genética es fundamental en el proceso evolutivo ya que es una de las principales tareas en la división celular. Se dice que este proceso es *semi-conservativo* porque cada una de las cadenas nuevas conserva (o está constituida por) una de las hebras de la madre. La replicación, así como la transcripción y la traducción, presenta diferencias entre los organismos eucariotas y procariotas. Esto se debe, entre otras razones, al hecho de que los organismos eucariotas tienen membrana nuclear mientras que los procariotas no.

Transcripción Este proceso consiste en copiar la información genética en forma de ADN en ARN. La transcripción en organismos eucariotas incluye un proceso adicional (posterior a la síntesis de ARN) conocido como *splicing*. En el splicing se fragmenta la cadena de ARN inicial (conocida como ARN precursor) en subunidades llamadas *exones* e *intrones*. A partir de estas subunidades se genera una nueva cadena de ARN (llamada ARN mensajero) con algunos o todos los exones. Los intrones son eliminados completamente.

Traducción Este proceso consiste en producir o sintetizar proteínas a partir de un molde de ARN. Las proteínas son cadenas de *aminoácidos*. En teoría, es posible la existencia de una gran variedad de aminoácidos distintos, pero solamente 20 tipos diferentes se utilizan para construir las proteínas. El organelo encargado de hacer esta "traducción"de ácidos nucleicos a proteínas es el ribosoma.

2.2.2. Una clasificación de problemas

Cuando se intenta entender (a nivel molecular) la composición, funcionamiento y evolución de los distintos componentes biológicos se encuentran varios problemas. Su naturaleza permite identificar al menos tres grandes grupos. Primero, problemas relacionados con *secuencias* de nucleótidos y aminoácidos. Segundo, problemas relacionados con la predicción de la *estructura* espacial de macromoléculas. Tercero, problemas relacionados con el *funcionamiento* de los sistemas. A continuación se describe cada uno de estos problemas y se hace referencia a trabajo previo en cada uno de ellos usando cálculos de procesos.

Problemas relacionados con secuencias

A pesar de que este tipo de problemas es uno de los más estudiados por la biología molecular computacional [Pev00], es al mismo tiempo el menos tratado con cálculos de procesos. Esto se debe a que aquí lo que se busca es comparar, alinear o identificar nuevos patrones en secuencias de información biológica (i.e., ácidos nucleicos y proteínas). Así, se busca identificar por ejemplo, secuencias específicas en el ADN para el reconocimiento de genes o la presencia de motivos en proteínas. Por esta razón, la posición de los componentes de tales secuencias (y no su interacción) es el elemento fundamental en este tipo de problemas, haciendo inconveniente el uso de cálculos de procesos para su estudio.

Problemas relacionados con la estructura

Uno de los retos para los investigadores en biología y ciencias de la computación es poder definir la conformación espacial de un polímero (e.g., el ADN, el ARN o una proteína) a partir de la secuencia de los monómeros (i.e., nucleótidos o aminoácidos) que lo constituyen. Adicionalmente, un problema importante en este grupo consiste en la identificación de las leyes que gobiernan las interacciones entre dos o más moléculas a través de sus *motivos*, lugares donde las moléculas pueden interactuar con otras. De forma concreta, la programación por restricciones y el cálculo π [Mil99] han sido utilizados en [Bac97,PDF04] y [RS04] respectivamente, para proponer soluciones aproximadas y hacer abstracciones de problemas de este tipo.

Problemas relacionados con la función

En la célula se pueden identificar tres tipos de tareas que definen todo su funcionamiento. Cada una de ellas es responsabilidad de un grupo de redes o rutas particulares. Estas son: las encargadas del *control celular*, las encargadas del transporte y transformación de sustancias *dentro de la célula* y las encargadas del transporte y transformación de sustancias *a través de la membrana celular*. Cada una se describe a continuación.

Redes de regulación genética Son las encargadas de todo el control celular. Este control puede estar a nivel de procesos o sistemas. Como esta regulación se fundamenta en la producción de *enzimas* que catalizan o inhiben los procesos celulares, la concentración de éstas dentro de la célula resulta fundamental. Algunos modelos de estas redes están definidos en términos de sistemas de ecuaciones diferenciales que relacionan las concentraciones de los componentes involucrados en los procesos de control. La Programación Concurrente por Restricciones ha sido usada en [BC02, ERdJ$^+$04, BC01] para resolver problemas de este tipo utilizando el cálculo hcc [GJSB94].

Rutas metabólicas Son el conjunto de todos los procesos en los que interactúan componentes biológicos para realizar el transporte y la transformación de sustancias en el *interior* de la célula. Por componente biológico se entiende toda conformación molecular y macromolecular como el agua, el oxígeno, los nutrientes y los organelos intracelulares (e.g., las mitocondrias, los ribosomas y los cloroplastos) entre otros. Normalmente los procesos de una red metabólica son de tipo bioquímico. Estos procesos o *reacciones químicas* generalmente se expresan en términos de ecuaciones diferenciales. En Mozart-Oz se han desarrollado algunas herramientas para encontrar rutas metabólicas con características restringidas [DDD04a, DDD05, DDD04b].

Redes de transducción de señales Son las encargadas de todos los procesos de interacción y transporte de sustancias a través de la *membrana* celular. El transporte de sustancias puede ser activo o pasivo, dependiendo si ellas se mueven a favor o en contra de gradientes electroquímicos

existentes entre el interior y el exterior de la célula. Existen varios cálculos de procesos que han sido especialmente desarrollados para modelar problemas de este tipo [Car04, RPS+04].

2.2.3. Biología sistémica (systems biology)

La biología sistémica (o *systems biology* [Kit01a]) es un área de investigación multidisciplinaria, cuyo propósito es el estudio de los mecanismos presentes en los sistemas biológicos por medio de los cuales genes y proteínas se integran e interactúan dentro de un organismo. Es decir, la biología sistémica estudia de forma integrada la *estructura* y *expresión* de un gen o un conjunto de genes.

Este estudio, basado en las ideas de sistema e interacción multinivel [Kit01b], presenta algunos aspectos característicos. Primero, se deben tener en cuenta las *interacciones* entre los componentes que conforman el sistema. Como consecuencia, se asume que los elementos que lo constituyen no se encuentran aislados del entorno, sino que influyen y son influidos por él. Segundo, se debe poder analizar, modelar y simular, con mayor o menor detalle, el mismo sistema biológico. Esto implica poder identificar y abstraer sus propiedades fundamentales a distintos *niveles*.

2.3. Trabajo relacionado

En los últimos años se han realizado varios trabajos enfocados en el uso de cálculos de procesos y técnicas basadas en restricciones para el estudio de sistemas biológicos (ver [GPR05, PPQ05, BG01]). Muchos de estos trabajos han sido desarrollados con el uso de cálculos tradicionalmente pensados para la especificación de sistemas de comunicación, otros adaptando cálculos existentes, y otros más con el uso de cálculos diseñados específicamente para la descripción de funciones biológicas. Un estudio de estos trabajos permite hacer comparaciones que conduzcan a la identificación de ventajas y desventajas del uso de un formalismo basado en el modelo CCP sobre otras alternativas semejantes.

2.3.1. Cálculo π

Quizás el formalismo más representativo dentro de los cálculos de procesos es el cálculo π [Mil99], propuesto por Robin Milner a principios de los 90. Este cálculo extiende CCS [Mil89], su antecesor inmediato, con la idea de *nombres*, entidades que pueden ser tanto datos como canales de comunicación. Este principio, simple a primera vista, resulta poderoso para representar esquemas de comunicación complejos donde la topología del sistema puede cambiar.

La sintaxis del cálculo π es la siguiente:

donde P y Q son procesos, x es un nombre y π es un *prefijo*. El prefijo denota una acción atómica previa a un proceso. Esta acción puede ser el envío o recepción de un dato por un canal. Por ejemplo, el envío de un dato z por el canal x se denota por $\overline{x}\langle z\rangle$; por su parte, la recepción de un

$$P, Q \ldots \quad ::= \quad P \parallel Q \mid !P \mid (\nu x)P \mid \sum_{i \in I} \pi_i.P_i$$

Cuadro 2.5: Sintaxis del cálculo π

dato b por el canal c se representa por $c(b)$. El proceso $P \parallel Q$ denota la composición paralela de los subprocesos P y Q. Además, permite que P y Q se comuniquen. El proceso $!P$ representa la *replicación* de P. Es decir, define la ejecución *infinita* de P. Un nombre puede estar restringido al contexto de un proceso particular. Este es el propósito de $(\nu x)P$, en donde el nombre x es local a P y solamente visible dentro de él. Finalmente, la escogencia no determinista entre una serie de procesos $\pi_i.P_i$, donde $i \in I$, se representa como su *sumatoria* $\sum_{i \in I} \pi_i.P_i$. Esta escogencia depende de la comunicación de dichos procesos. Dos procesos importantes pueden derivarse de la sumatoria: $\mathbf{0}$ y $\pi.P$. El primero representa la acción nula y es la abreviación de la sumatoria cuando $I = \emptyset$, mientras que el segundo se da cuando solo hay un proceso involucrado en la escogencia.

El cálculo π ha sido utilizado (tanto en su versión original como en una versión que lo extiende con elementos estocásticos [Pri95]) por diversos autores para el modelado de sistemas en biología (ver por ejemplo [KJ04, CCT02, PRSS01]). Sin embargo, las ideas más precisas al respecto provienen del grupo de trabajo liderado por Ehud Shapiro, quienes no sólo han usado el cálculo para *representar* sistemas biológicos reales [RS02], sino que han trabajado intensamente en el desarrollo y uso de simuladores derivados de la extensión estocástica del cálculo para el estudio de dichos sistemas. Su principal contribución consiste en una abstracción que asocia los elementos biológicos fundamentales con componentes del cálculo [RS04]. Dicha abstracción, denominada "*molécula como computación*" [Sha02], asocia de manera sencilla entidades y eventos presentes en el lenguaje con entidades y eventos biológicos. Por ejemplo, los motivos son asociados con los prefijos de entrada y salida del cálculo, aprovechando la clara complementariedad que poseen ambas entidades en sus respectivos contextos.

En el campo aplicativo, la mayor contribución es BioPSI [bio06]. Se trata de una aplicación diseñada para la simulación de sistemas bioquímicos especificados en el cálculo π [RSS01]. Está construida sobre Flat Concurrent Prolog (FCP), lo que permite la implementación de las principales características del cálculo π: movilidad y comunicación sincronizada. Existen dos compiladores disponibles para hacer simulaciones. El primero, denominado PI, permite especificar sistemas usando el cálculo original y ejecutar simulaciones semi-cuantitativas, mientras que SPI (i.e., el segundo compilador) utiliza una modificación de la versión estocástica del cálculo, por lo que es apropiada para practicar simulaciones cuantitativas exactas. Tanto PI como SPI incluyen herramientas de depuración y de soporte para la simulación.

2.3.2. Brane: un cálculo de interacción entre membranas

Uno de los principales problemas en bioinformática es el relacionado con los mecanismos por los cuales la célula se comunica con su exterior a través de la membrana. Tales mecanismos comúnmente involucran el traspaso de elementos (e.g., aminoácidos, oxígeno, agua, etc.) nece-

sarios para el correcto desarrollo de los procesos celulares. Por lo tanto, alteraciones en este tipo de sistemas pueden descontrolar el funcionamiento de la célula, incluso al punto de llevarla a la muerte o a la generación de productos nocivos.

El cálculo Brane [Car04] hace parte de los cálculos centrados en el modelado de las interacciones a través de membranas biológicas. La motivación es la abundancia de interacciones de este tipo en los procesos celulares. En Brane la comunicación no se restringe a la realizada a través de la membrana citoplasmática sino que también incluye la realizada a través de la membrana nuclear y de otras como las membranas de los organelos (e.g., mitocondrias, cloroplastos, etc.) las cuales tienen alta actividad dentro de la célula.

Dos aspectos fundamentales en el cálculo Brane son el tipo de *interacción* entre las membranas y el tipo de *transferencia* de información a través de ellas. Con respecto a lo primero, se pueden identificar tres tipos de interacción, a saber, la fagocitosis, la pinocitosis y la exocitosis. En ellos, la inspiración biológica esta dada por el proceso de división celular y los procesos en los que se acepta o se expulsa un agente biológico del interior de la célula. Por otro lado, la transferencia de la información a través de las membranas se puede clasificar en: paso de información por una membrana (e.g., comunicación a través de la membrana nuclear) y paso de información por dos membranas (e.g., paso de información del citoplasma de una célula a otra).

Por otro lado, en el campo aplicativo es poco lo que se ha logrado. Pese a que con Brane es posible verificar propiedades de los sistemas modelados, no existe aún una herramienta que implemente las ideas formales del cálculo. Mientras no se avance en esa dirección no es posible verificar la utilidad real que pueda tener el cálculo a nivel biológico.

2.3.3. BioAmbients: espacios de computación biológica

Este cálculo también está enfocado en los problemas relacionados con las interacciones a través de membranas. BioAmbients [RPS$^+$04] proviene del cálculo Ambient [CG98], por lo que conserva sus principales características como la creación de ambientes independientes y móviles en los que se ejecutan procesos del cálculo π.

BioAmbients define tres tipos de comunicación y seis primitivas para la manipulación de los ambientes. Los tipos de comunicación son: uno local entre procesos de un mismo ambiente (i.e., dos procesos en paralelo), uno entre procesos localizados en ambientes que se encuentren a un mismo nivel jerárquico (i.e., dos ambientes en paralelo) y uno entre procesos localizados en distintos ambientes en los que uno de ellos tiene un nivel jerárquico mayor que el otro (i.e., un ambiente se encuentra dentro de otro). Por otro lado, se tienen primitivas para *entrar*, *aceptar*, *expulsar*, *salir* y *mezclar* ambientes.

Una ventaja de este cálculo es que permite pensar en un sistema como un conjunto de partes que interactúan, pero que a su vez cada parte (o componente biológico en este caso) tiene características que lo definen y que se encuentran dentro de un ambiente independiente. Otra ventaja es la facilidad de incluir un modelo dentro de otro simplemente encerrándolo dentro de un ambiente y definiéndole una interfaz de interacción con el exterior. Una desventaja es la

necesidad de complementariedad entre los componentes biológicos para su interacción, algo que realmente no es siempre necesario a menos que se trate de una interacción a través de motivos moleculares.

Desde lo práctico BioAmbients cuenta con un herramienta de simulación llamada BioPSI, descrita anteriormente en la sección 2.3.1.

2.3.4. hcc

hcc [GJSB94] es una extensión de cálculo cc [Sar93] que permite modelar tiempo continuo y discreto. Para entender sus características, es preciso describir antes su evolución así como la relación con otros formalismos (i.e., cálculos de procesos) pertenecientes al modelo CCP [GJS96].

Una de las principales características del cálculo cc es que permite detectar la *presencia* de información, por medio de la operación primitiva **if** c **then** P, que ejecuta el proceso P si c se deduce de la información presente en el almacén de restricciones. Sin embargo, ciertos sistemas pueden describirse de forma más adecuada por medio de condicionales que evalúen la *ausencia* de información. En dichos casos, la operación descrita es inapropiada para el modelado, porque en el caso en que c no se deduzca del almacén, la ejecución del sistema se bloquea indefinidamente.

El problema consiste entonces en detectar que una restricción no se deduce del almacén. La primera solución consistió en la inclusión del concepto de *fases de ejecución*, induciendo así una noción de temporalidad. De esta forma, la ausencia de información se evalúa al *final* de una fase de ejecución. Bajo este contexto, se adicionaron dos operaciones al cálculo cc. La primera, **if** c **else next** P, ejecuta el proceso P en el *siguiente* instante de tiempo si no se deduce c en el actual. La segunda, **hence** P, ejecuta P en todos los instantes de tiempo después del actual. Este modelo extendido se conoce como tcc (Timed Concurrent Constraint) [SJG94].

A pesar de esta extensión, en ciertos sistemas es indispensable reaccionar *inmediatamente* cuando cierta información no está disponible. Para superar esta falencia se propuso Default cc, en donde se *estima* un almacén final antes de iniciar la computación. Más específicamente, Default cc agrega la operación **if** c **else** P, que ejecuta P *por defecto* y de manera *inmediata* si no de deduce c del almacén estimado. Al final del cómputo, el almacén obtenido y el estimado inicialmente son comparados. Si son iguales, la estimación fue correcta y se acepta el computo. En caso contrario se produce un fallo.

Finalmente, hcc extiende Default cc adicionándole la noción de fases de ejecución sobre *tiempo continuo* [GJS98]. En otras palabras, un programa hcc ejecuta múltiples programas Default cc, uno en cada fase de ejecución. Cuando en alguna de las fases de ejecución un programa Default cc falla (i.e., almacén estimado y obtenido distintos) se elimina esa rama de los cómputos o se hace backtracking. La sintaxis precisa del cálculo hcc se presenta en el cuadro 2.6.

Primitiva	Proposición
c	Impone la restricción c
if c **then** P	Ejecuta el proceso P si c se deduce del almacén
new X **in** P	Crea la variable X local en P
P , Q	Ejecuta los procesos P y Q en paralelo
if c **else** P	Ejecuta el proceso P si c no se deduce del almacén
hence P	Ejecuta P siempre al principio de cada fase

Cuadro 2.6: Sintaxis del cálculo hcc

2.3.5. CSPs

En el contexto de la Programación por Restricciones (CP) [MS98] es posible identificar un conjunto de problemas denominados *problemas de satisfacción de restricciones* (o CSPs por sus siglas en ingles). Su formulación se basa en la definición precisa de dos componentes que caracterizan los problemas que ellos modelan: un conjunto de variables con dominios asociados (i.e., rangos de los posibles valores que las variables pueden tomar) y un conjunto de restricciones que relacionan tales variables (posiblemente) limitando sus dominios.

En la actualidad se conocen algunos problemas en bioinformática que han sido modelados con CSPs o con extensiones de ellos tal como se hace en [CB03]. Algunos de los trabajos más relevantes son, por ejemplo, la predicción de la estructura espacial de una proteína [PDF04, Bac97], la búsqueda de caminos en rutas biológicas [DDD05, DDD04b], el modelado restringido de ecuaciones diferenciales [CB03] o la definición de CSPs sobre cadenas de ADN [BS05] entre otros. A continuación se presenta una breve descripción de algunos de estos trabajos para tener una visión más clara de la aplicabilidad o uso de los CSPs y sus extensiones en la solución de problemas en bioinformática.

El problema de la predicción de la estructura espacial de una proteína o *protein folding*, consiste en determinar la estructura tridimensional de una proteína a partir de la secuencia de aminoácidos que la conforman. Este problema ha sido abordado en [Bac97] haciendo uso de modelos simplificados (*lattice models*), donde cada aminoácido es clasificado dentro de uno de dos grupos según sus características químicas. La solución de este problema consiste en buscar el valor mínimo de una función de energía que relaciona todos los elementos del sistema dada su posición en el espacio.

El problema de la búsqueda de caminos en rutas bioquímicas también ha sido modelado en [DDD04b] haciendo uso de CSPs, donde el principal objetivo es estudiar el comportamiento de los componentes biológicos que conforman estas rutas (e.g., proteínas, azucares, etc.) bajo unas condiciones particulares (i.e., restricciones sobre el sistema).

Otro trabajo interesante es el desarrollado en [CB03], donde se propone una nueva categoría de problemas estrechamente relacionados con los CSP denominados CSDP (*Constraint Satisfaction Differential Problems*). Los CSDPs intentan capturar las principales características de los sistemas continuos por medio de la declaración de ecuaciones diferencias como restricciones del sistema. Este nuevo formalismo ha permitido abordar problemas complejos relacionados con el

diseño de drogas o el estudio de epidemias por nombrar algunos casos.

Pese a los interesantes resultados *prácticos* relacionados con el uso de CSPs en bioinformática, los resultados teóricos aun exigen un mayor esfuerzo en el campo de verificación de propiedades biológicas. Hasta la fecha los CSPs han sido usados más como un formalismo para solucionar problemas en bioinformática que para razonar sobre los sistemas que ellos modelan. Por tal motivo, técnicas para verificar propiedades sobre tales sistemas no han sido explotadas como en el caso de los cálculos de procesos.

2.4. Resumen

En este capítulo se presentó un estudio de los principales temas tratados en este trabajo de grado. Inicialmente se explican las principales características del modelo CCP y como éstas son entendidas en el contexto del cálculo de procesos `ntcc`. Para ello, se proporcionan detalles sobre la sintaxis, semántica y lógica asociada del cálculo. Posteriormente, se muestra la forma en que los conceptos teóricos presentados hasta esa instancia son llevados a la práctica mediante el uso de la herramienta de simulación **ntccSim**.

Luego, a través de la bioinformática, se expone el rol de las ciencias de la computación en el contexto biológico. Tal proceso de contextualización inicia con la presentación de algunos conceptos básicos de la biología molecular, y partiendo de ellos se plantea una clasificación de problemas biológicos a nivel molecular. La sección concluye con la descripción de los aspectos más relevantes relacionados con la biología sistémica, una área de estudio estrechamente relacionada con la biología molecular y las ciencias de la computación.

Al final del capítulo se presenta un estudio del estado del arte sobre el modelado de sistemas biológicos usando cálculos de procesos. Esto permite comparar los formalismos basados en el modelo CCP contra enfoques provenientes de otros modelos de concurrencia. Este estudio claramente enriquece los análisis que se derivan de los resultados obtenidos en este trabajo de grado.

3 El Operón de la Lactosa

En este capítulo se presenta una descripción de las principales características de la estructura y el funcionamiento del operón de la lactosa. Siguiendo un enfoque de estudio basado en la biología sistémica, tal descripción es hecha desde distintos *niveles de detalle*, permitiendo reconocer en cada nivel de descripción los aspectos más importantes.

Se proponen tres niveles de estudio según los posibles modelos para este sistema biológico: poblacional, celular y molecular. En cada uno de estos niveles se especifican los elementos *observables* del sistema y como ellos interactúan entre si. De esta forma, es posible proponer diferentes modelos que representen el sistema según (i) la información que se desee apreciar o (ii) la información que se tenga disponible. Un especial énfasis se realiza en la descripción del operón de la lactosa a nivel molecular.

Al final del capítulo se presenta, además, una descripción de algunos tipos de modelos para la representación de sistemas biológicos. Cada uno de estos tipos de modelos pretende describir la estructura y el funcionamiento de los sistemas tomando en cuenta diferentes características identificables en ellos, como por ejemplo, la *presencia* de una proteína, el *valor* de su concentración en un momento determinado o la *variación* de tal concentración en el tiempo.

3.1. Descripción General

Un operón [Lew00] es un *conjunto de genes* que regulan una o más actividades celulares. Por tal motivo, los operones se encuentran dentro de un tipo de sistemas presentes a nivel celular llamados las *redes de regulación genética* o redes de regulación de la expresión génica, como también son conocidos. Su funcionamiento a nivel celular puede entenderse como el de un "switch" o interruptor que se enciende y se apaga según las necesidades celulares. De ésta forma los operones activan o inhiben los procesos que ellos regulan. En particular, el *operón de la lactosa* es una red de regulación encargada de controlar las fuentes de energía dentro de la célula.

El operón de la lactosa está conformado por una región que controla su expresión o activación y por tres genes estructurales: LacZ, LacY y LacA. Ellos codifican las proteínas β-galactosidasa, permeasa y tiogalactósido transacetilasa respectivamente. Además, en la red de regulación existe un gen, llamado LacI, que interactúa de forma constante con el operón. Su función es codificar una proteína represora. Por tal motivo este gen es conocido también como gen represor.

Una característica importante de los operones es que la expresión (i.e., transcripción y traducción) del conjunto de genes que los conforman está regulada de forma coordinada, es decir, o todos se expresan o ninguno de ellos lo hace. Comúnmente, esta regulación puede ser de dos tipos: por control positivo o por control negativo. Los genes que se encuentran bajo *control positivo* se pueden transcribir solamente si una proteína activadora está presente en el ambiente de la red de regulación. De forma contraria, los genes que se encuentran bajo *control negativo* se expresan siempre a menos que una proteína inductora este presente. También existen genes que no se encuentran bajo ninguno de estos dos tipos de regulación; a ellos se les conoce como genes constitutivos.

A continuación se presentan las descripciones a nivel poblacional, celular y molecular del operón de la lactosa [VGL03]. En cada una de ellas la descripción del sistema se centrara en la presentación precisa de los elementos observables a cada nivel de detalle y las interacciones entre tales elementos, dejando de lado otros aspectos que no sean relevantes.

3.2. Descripción a nivel poblacional

A nivel poblacional es posible analizar el operón de la lactosa en su forma más primitiva, es decir, como un interruptor que activa o inhibe los procesos celulares que permiten obtener glucosa a partir de la lactosa en el medio. De esta forma, en una población de células es posible identificar u *observar* aquellas que se encuentran con el operón activo y aquellas que lo tienen inactivo. Como consecuencia, en este nivel de descripción no es relevante, por ejemplo, información relacionada con el nivel de expresión del operón, la concentración de las proteínas que el produce, la forma en que éstas interactúan entre si con el resto del sistema, etc.

Sin embargo, tener un modelo que permita conocer el número de células con el operón de la lactosa activo, permite estudiar los efectos que puede tener un medio con unas condiciones específicas (e.g., acidez, temperatura, humedad, etc.) sobre este sistema de regulación en una población de células. La figura 3.1 representa de forma gráfica una población de células en la que se estudia el comportamiento del operón de la lactosa en un medio determinado. El color de cada una de ellas indica el estado (i.e., negro para activo y blanco para inactivo) del operón. La jerarquía significa que las células que tienen activo el operón de la lactosa, generan células hijas con la misma característica.

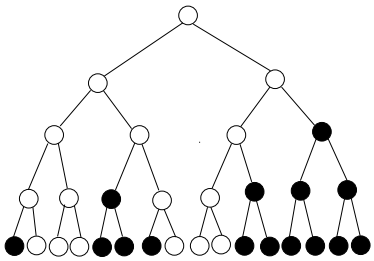


Figura 3.1: Comportamiento del operón de la lactosa en una población de células

3.3. Descripción a nivel celular

A nivel celular cobran importancia datos que a nivel poblacional no son relevantes. Por ejemplo, en este nivel es necesario conocer las concentraciones de los elementos que conforman el sistema (ver figura 3.2) y la forma en que éstas cambian en el tiempo, determinando así su comportamiento. En este nivel de descripción, más detallado que el anterior, el interruptor visto a nivel poblacional es estudiado ahora como un sistema de control que regula los procesos biológicos que permiten obtener glucosa a partir de la lactosa en el medio.

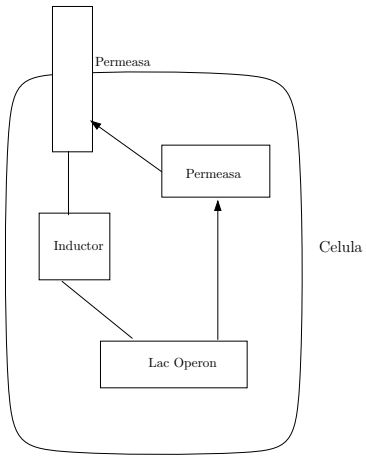


Figura 3.2: Estructura del operón de la lactosa a nivel celular

Dado que la célula *prefiere la glucosa sobre la lactosa* como fuente de energía, la descripción del funcionamiento de la red de regulación se presentara tomando en cuenta las cuatro situaciones posibles según la ausencia o presencia de la glucosa y la lactosa en el medio celular. Cada una de estas cuatro situaciones determina la activación o inhibición del operón. Estas condiciones se resumen en el cuadro 3.1.

Medio con glucosa y sin lactosa En este caso la presencia de glucosa en el medio celular hace que el operón de la lactosa este *inactivo*. Los procesos de regulación determinando este tipo de comportamiento son los siguientes: dado que existe glucosa en el medio celular, se inhibe la producción del complejo activador del operón, por lo que su expresión (i.e., transcripción y traducción) no es estimulada. Además, la falta de lactosa en el medio hace que no exista un inductor que impida que el complejo represor del operón de la lactosa inhiba su activación.

Medio con glucosa y con lactosa En este caso la presencia de glucosa en el medio nuevamente *inactiva* o inhibe al operón de la lactosa, aunque los procesos de regulación son distintos. Al igual que en el caso anterior, la presencia de glucosa en el medio hace que no exista un activador del operón por lo que es difícil iniciar su transcripción. Sin embargo, la presencia de la lactosa hace que exista un complejo inductor que intenta bloquear la acción del complejo represor.

Glucosa	Lactosa	Activación del operón
No	No	No
No	Si	Si
Si	No	No
Si	Si	No

Cuadro 3.1: Descripción funcional del operón de la lactosa a nivel celular

Medio sin glucosa y con lactosa En este caso el operón se *activa*. Por un lado, la falta de glucosa en el medio favorece la producción del complejo activador del operón, es decir, el operón se encuentra bajo control positivo. Por otro lado, la presencia de lactosa en el medio incrementa la concentración del complejo inductor del operón de la lactosa, encargado de bloquear al complejo represor.

Medio sin glucosa y sin lactosa En este caso las condiciones del medio hacen que el operón se encuentre *inactivo*. Pese a que la falta de glucosa incrementa la concentración del complejo activador del operón, la falta de lactosa hace que no exista un complejo inductor que bloquee la acción del complejo represor.

Por lo tanto, la activación del operón de la lactosa sólo ocurre cuando existe un complejo activador que facilite la expresión del operón y un complejo inductor que bloquee la acción del complejo represor. En cualquier otro caso el operón se encontrara inactivo. Este comportamiento se mostrara más detalladamente en la siguiente sección donde se explicará el comportamiento a nivel molecular de la región de control del operón de la lactosa.

3.4. Descripción a nivel molecular

En este nivel de descripción se incluye explícitamente en el análisis de la red de regulación del operón de la lactosa la interacción entre moléculas. Mientras algunas de estas interacciones pueden producir modificaciones en la estructura espacial de las moléculas, definiendo cambios "inmediatos" en su funcionamiento, en otras ocasiones tales interacciones culminan en la formación de nuevos complejos moleculares. En el operón de la lactosa existen muchas interacciones de este tipo, las cuales serán explicadas a continuación.

Primero en la sección 3.4.1 se describen los elementos y procesos moleculares que determinan la activación del operón de la lactosa. Luego, en la sección 3.4.2, se presenta una descripción de la estructura de los genes que conforman el operón de la lactosa y de los productos (i.e., proteínas) que se sintetizan a partir de la información que ellos codifican. Las secciones 3.4.3 y 3.4.4 son dedicadas a la explicación de los procesos de regulación presentes en el operón, basándose en las interacciones entre las moléculas que intervienen en cada proceso de control biológico. Al final, en la sección 3.4.5 se exponen algunos tipos de comportamiento irregular en la red de regulación según modificaciones moleculares en el genoma (i.e., ADN) del operón. Estos comportamientos

anormales no serian posibles de explicar si no se tiene una visión molecular de este sistema biológico.

3.4.1. Región de control

La región de control del operón de la lactosa, constituida por 82 pares de bases [1], es el lugar donde se realiza la regulación de la transcripción de los genes del operón. En ella pueden identificarse dos secuencias específicas según el tipo de control o regulación que ejercen sobre el operón: la *promotora* y la *operadora* (ver figura 3.3). Mientras la secuencia promotora está estrechamente relacionada con los procesos biológicos que inducen una regulación positiva en el operón, la secuencia operadora lo está con aquellos procesos conducentes a una regulación negativa.

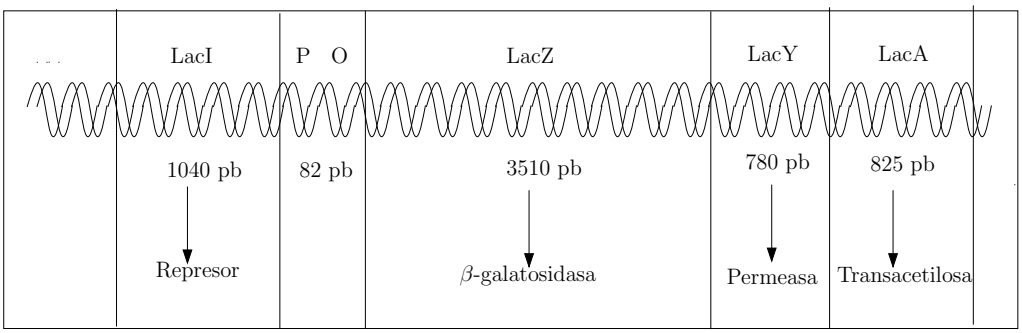


Figura 3.3: Estructura del operón de la lactosa a nivel molecular

La secuencia promotora (llamada también factor *cis*) es el lugar de la región de control donde se realizan los principales procesos que permiten *iniciar la transcripción* del operón. En esta secuencia, la cual ocupa más del 50 % de la región de control, se encuentran los códigos genéticos que le permiten a la ARN polimerasa identificar los genes del operón de la lactosa y de esta forma poderlos transcribir. Sin embargo, reconocer tales códigos del ADN del operón no es sencillo, por lo que este proceso de reconocimiento es facilitado por el complejo molecular CAP-cAMP (el cual está formado por una molécula de CAP [2] y otra de cAMP [3]) cuando esta molécula se enlaza a una subregión de la secuencia promotora (*CAPsite*).

Por otro lado, la secuencia operadora, constituida por 26 pares de bases, es el lugar de la región de control donde el complejo represor se ubica para impedir que la ARN polimerasa realice la transcripción de los genes del operón. Dado que la secuencia operadora se encuentra entre el promotor y los genes del operón, la *represión* se da cuando el complejo represor bloquea el paso de la ARN polimerasa impidiendo que esta última avance para iniciar el proceso de transcripción.

[1] unidad de medida del tamaño de las cadenas de ADN y ARN relacionada con el número de nucleótidos que las componen

[2] Proteina activadora de catabolito

[3] Adenosin monofosfato cíclico

Tetrámero represor	Complejo molecular CAP-cAMP	Activación del operón
No	No	No
No	Si	Si
Si	No	No
Si	Si	No

Cuadro 3.2: Descripción funcional de la región de control del operón de la lactosa

El cuadro 3.2 resume el funcionamiento lógico de la región de control según la presencia o ausencia del complejo o tetrámero represor y del complejo molecular CAP-cAMP. Ver sección 3.3 para verificar las condiciones del medio que determinan la presencia o ausencia de tales complejos moleculares.

3.4.2. Región estructural

La región estructural está constituida por los genes (i.e., genes estructurales) que conforman la región codificante del operón. Por tal motivo a esta región también se le conoce como región codificante. Por ser aquella que contiene la información genética del operón es la de mayor tamaño con una longitud de 5115 pares de bases (contra 82 en la región de control como se muestra en la Figura 3.3). Como se ha mencionado anteriormente, el operón de la lactosa está compuesto por tres genes (i.e., LacZ, LacY y LacA) cuyo funcionamiento se describe a continuación.

El gen LacZ codifica la proteína β-galactosidasa. Esta proteína es la encargada de romper o dividir las moléculas de lactosa en dos más pequeñas, una de glucosa y otra de galactosa, mediante un proceso conocido como hidrólisis.

El gen LacY codifica la proteína permeasa. Esta molécula es la encargada de facilitar el transporte o paso, a través de la membrana celular, de la lactosa que se encuentra en el exterior de la célula hacia el interior de ella.

El gen LacA codifica la proteína tiogalactósido transacetilasa. La función de esta proteína aún no está bien definida, pero se sabe que no interfiere con los procesos regulatorios relacionados con el operón de la lactosa. Por tal motivo, el comportamiento de esta proteína normalmente no es modelado cuando se estudian los procesos de regulación relacionados con el operón.

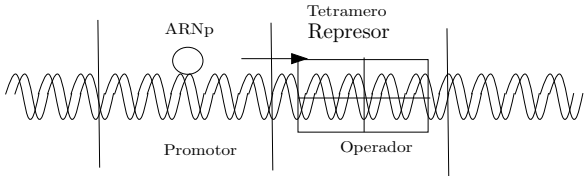
3.4.3. Control negativo

Pese a que el control negativo es realizado por una proteína represora, este tipo de regulación no implica la represión del operón de la lactosa. Sin embargo, para comprender este proceso regulatorio es fundamental entender el comportamiento de la proteína represora, la cual es codificada por el gen LacI. Este gen, que pertenece a la red de regulación del operón de la lactosa, se encuentra ubicado relativamente cerca de la región de control. En el estudio del control negativo del operón de la lactosa se pueden identificar dos casos particulares determinados por

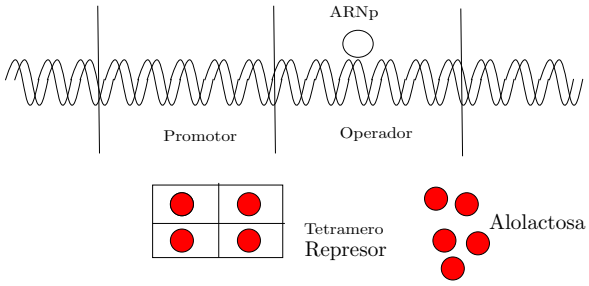
la presencia o en ausencia de la alolactosa (ver figura 3.4). Cada uno de estos casos se detalla a continuación.

Dado que el gen represor es constitutivo (i.e., un gen no regulado) se encuentra constantemente sintetizando la proteína represora. Esta molécula, inactiva para procesos de represión en su estado natural, se une en grupos de cuatro moléculas para conformar un tetrámero represor que si es activo en cuanto a procesos de represión se refiere. De esta forma, el tetrámero represor es el complejo molecular que realmente se adhiere a la región operadora del operón para evitar la transcripción de los genes estructurales. En este caso, en el cual el operón se encuentra inactivo o reprimido, se habla de una regulación negativa en ausencia de alolactosa (ver figura 3.4(a)).

Sin embargo, el proceso de represión ejercido por el tetrámero represor es evitado en presencia de la alolactosa (i.e., una molécula inductora del operón) ya que esta molécula tiene la capacidad de unirse al tetrámero y hacerle cambiar su *conformación espacial*, disminuyendo de esta forma la afinidad o facilidad que el represor tiene de enlazarse a la región operadora. Este proceso de control tiene como consecuencia que la ARN polimerasa no encuentre obstáculos que le impidan iniciar el proceso de transcripción. En este caso, contrario al mencionado anteriormente, se habla de un control negativo en presencia de alolactosa en el cual el operón no sufre represión (ver figura 3.4(b)).



(a) Control negativo en ausencia de alolactosa



(b) Control negativo en presencia de alolactosa

Figura 3.4: Control negativo del operón de la lactosa

3.4.4. Control positivo

Pese a que el control positivo sobre el operón de la lactosa es realizado por el complejo activador CAP-cAMP (ver figura 3.5), el análisis de este tipo de regulación se hace en base a la concentración celular de glucosa. Esto se debe a que la concentración celular de la molécula cAMP tiene una dependencia inversa de la concentración de glucosa. De esta forma, se pueden identificar

dos casos de estudio en el control positivo: cuando la concentración de glucosa es alta y cuando es baja.

En el primer caso, es decir, en presencia de glucosa como principal fuente de energía, los niveles de concentración de cAMP son bajos haciendo difícil la formación del complejo inductor CAP-cAMP. Esto tiene como consecuencia que la ARN polimerasa no pueda identificar fácilmente la región promotora y por tanto sea más complicado el inicio de la transcripción.

Por otro lado, en ausencia de glucosa se elevan los niveles de cAMP permitiendo que la concentración celular del complejo activador CAP-cAMP se incremente sustancialmente. Este incremento en la concentración celular del complejo activador CAP-cAMP facilita que la ARN polimerasa reconozca más rápidamente los patrones en la región de control que le permiten iniciar la transcripción del operón.

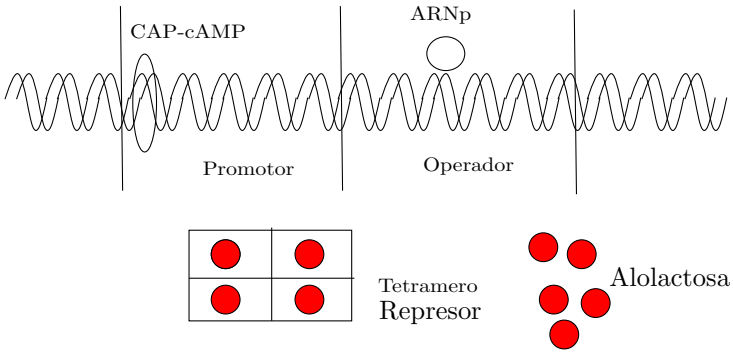


Figura 3.5: Control positivo del operón de la lactosa

3.4.5. Mutaciones del operón de la lactosa

Las mutaciones en las redes de regulación genética pueden cambiar el mecanismo de expresión de los genes o causar que un sistema no tenga ningún tipo de regulación. En el caso del operón de la lactosa puede causar, por ejemplo, que siempre se expresen los genes estructurales aún cuando el represor este presente o de forma contraria, que no se expresen incluso cuando el represor se encuentra inhibido. Las principales mutaciones que se pueden encontrar en el operón de la lactosa se resumen a continuación:

- Represor inactivo o ausente $LacI^{-}$: el gen LacI sufre una mutación que codifica un tetrámero represor siempre inactivo haciendo que este no se pueda unir al operador. Por tanto siempre es posible iniciar el proceso de transcripción sin importar si el inductor está presente o no.
- Represor siempre activo $LacI^{s}$: el represor sufre una mutación que evita que el inductor pueda reconocer su sitio de unión. Por tanto el represor siempre puede unirse a la región operadora y evitar la transcripción sin importar que el inductor este o no presente.

- El represor no se une al operador $LacI^{-d}$: tiene el mismo efecto que la mutación Lac^{-} pero en esta ocasión es debido a que el represor no puede reconocer la región operadora, haciendo que el proceso de transcripción siempre se puede llevar a cabo sin importar si el inductor se encuentra presente o no.
- El operador no se puede reconocer O^{c}: es una mutación en la región operadora que le impide al represor identificar el sitio de enlace permitiendo que la ARN polimerasa siempre pueda iniciar el proceso de transcripción.
- LacZ no funcional $LacZ^{-}$: es una mutación en el gen LacZ que afecta la producción de β-galactosidasa. Por tal motivo la lactosa dentro de la célula no puede ser degradada.
- LacY no funcional $LacY^{-}$: es una mutación en el gen LacY que afecta la producción de permeasa. Esto genera que el operón no pueda tomar la lactosa que hay en el medio externo.
- LacA no funcional $LacA^{-}$: es una mutación en el gen LacA que afecta la producción de tiogalactósido transacetilasa. Esta mutación no afecta el funcionamiento del operón ya que esta proteína no participa en los procesos de regulación relacionados con el operón de la lactosa.

3.5. Modelos biológicos

En esta sección se presentan tres tipos de modelos o formas de representar sistemas biológicos. El primero de ellos se fundamenta en una interpretación *lógica* de los sistemas. El segundo, desde una perspectiva continua del cambio de los sistemas, los describe mediante un conjunto de *ecuaciones diferenciales*. Finalmente, la tercera forma de modelarlos (la cual tiene en cuenta la posible falta de información y la presencia de eventos discretos) se basa en el uso de *parámetros* para caracterizar los elementos que lo componen (e.g., genes, proteínas, etc.) y las interacciones entre ellos (e.g., reacciones bioquímicas, conformación de complejos moleculares, procesos de degradación natural, etc.). Además, con el objetivo de independizar el comportamiento de algunos elementos que componen el sistema, en algunas ocasiones se omiten ciertas relaciones o interacciones que no sean determinantes en el funcionamiento total del sistema.

3.5.1. Modelos lógicos

En los modelos lógicos la descripción de los sistemas se enfoca en la representación de cuatro características fundamentales: por un lado, la *presencia* o *ausencia* de los elementos que conforman el sistema y por otro, la *activación* o *bloqueo* de su funcionamiento. En [MPLD04] puede encontrarse un modelo de este tipo que representa el comportamiento lógico del operón de la lactosa usando redes booleanas.

3.5.2. Modelos continuos basados en ecuaciones diferenciales

Probablemente las ecuaciones diferenciales han sido el formalismo matemático más utilizado para representar sistemas continuos. Esto es debido a que estas estructuras matemáticas capturan de forma natural las características más relevantes de un sistema mediante la definición del *cambio temporal* de las variables que lo componen o representan. En otras palabras, estos modelos se enfocan en la descripción explicita de la dinámica de los sistemas.

Sin embargo, pese a que este tipo de modelos representan de forma detallada y completa el comportamiento de los sistemas biológicos, en la actualidad muchos biólogos no trabajan con este tipo de formalismos debido a la gran complejidad matemática, y en ocasiones computacional, que pueden llegar a tener. Por otro lado, no es fácil extender este tipo de modelos cuando se desean adicionar variables al sistema que se está estudiando ya que la adición de una variable puede implicar la modificación total del sistema de ecuaciones diferenciales que se tenía previamente. Una característica importante de los modelos basados en ecuaciones diferenciales es que su complejidad hace que normalmente sean definidos para sistemas pequeños, es decir, de pocas variables y en los que sea posible medir en laboratorio las interacciones entre ellas.

Algunos modelos que representan el *comportamiento a nivel celular* del operón de la lactosa pueden encontrarse en [YSM04, YM03, WGK97]. En particular, el modelo biológico presentado en [YSM04] será utilizado en el capítulo 4 como caso de estudio.

3.5.3. Modelos híbridos basados en parámetros

Los modelos híbridos asumen la presencia de características continuas y discretas (posiblemente lógicas) en el comportamiento de los sistemas. En este tipo de modelos, a diferencia de aquellos basados en ecuaciones diferenciales, no se cuenta con modelos matemáticos que describan el comportamiento de *todo* el sistema. Sin embargo, se tiene un conjunto de parámetros que caracterizan los elementos del sistema (e.g., genes, proteínas, etc.) y los procesos o interacciones que los relacionan, determinando la dinámica del sistema. Por tal motivo, la mayoría de los parámetros relacionados con las interacciones entre los elementos de los sistemas a modelar son, desde una perspectiva continua, las *velocidades* que determinan la rapidez con que se producen las transformaciones entre los elementos del sistema, y desde una perspectiva discreta, las *condiciones de limite* que producen cambios inmediatos en el comportamiento del sistema.

Una característica importante de este tipo de modelos es que son más fácilmente extensibles que aquellos expresados en términos de ecuaciones diferenciales. Esto se debe a que al adicionar una nueva variable al sistema sólo es necesario redefinir el conjunto de parámetros sobre los cuales la nueva variable pueda tener efecto. Este enfoque es claramente más simple que uno en el que la adición de una nueva variable al sistema implique la modificación total del modelo matemático anterior.

En [MM04] pueden encontrarse los parámetros que caracterizan las interacciones a nivel molecular entre los elementos que componen la red de regulación del operón de la lactosa. Las

variables y parámetros allí presentados son usados como base del modelo biológico presentado en el capítulo 5.

3.6. Resumen

En este capítulo se presento la estructura y el funcionamiento del operón de la lactosa desde tres niveles de descripción o detalle. El primero, un nivel de descripción *poblacional*, estudia los efectos del medio sobre la red de regulación del operón de la lactosa. De esta forma, es posible pensar en esta red de regulación como un interruptor que activa o inhibe los procesos celulares que regulan la cantidad de lactosa disponible. El segundo, un nivel de descripción *celular*, permite observar las concentraciones de los principales elementos que componen el operón. El estudio del funcionamiento de la red de regulación a este nivel de detalle se basa en el análisis de las respuestas regulatorias del operón según las concentraciones de glucosa y lactosa en el medio. El tercer nivel de detalle, un nivel de descripción *molecular*, tiene en cuenta los mismos aspectos del nivel celular pero le adiciona a los eventos observables del sistema la presencia de características discretas en el modelo debido a las interacciones moléculas presentes en los procesos regulatorios.

Posteriormente, al final del capítulo, se presentaron tres posibles formas de modelar sistemas biológicos según las características que se deseen destacar o tener en cuenta en ellos. En primero se centra en el comportamiento (de control) *lógico* de las redes de regulación genética. El segundo se enfoca sólo en las características *continuas* de este tipo de sistemas. Y el tercero, integrando características tanto discretas como continuas, propone un acercamiento *híbrido* al estudio del comportamiento de la redes de regulación. En cada uno de los posibles modelos biológicos descritos se referencian algunos trabajos en los que estas formas de modelar los sistemas son aplicadas al caso de la red de regulación genética del operón de la lactosa. Dos de ellos son usados más adelante en los capítulos 4 y 5.

4 Representación de Sistemas Continuos en ntcc usando Ecuaciones Diferenciales

Las ecuaciones diferenciales han sido desde hace mucho tiempo la forma más natural de modelar situaciones en las ciencias naturales, ingeniería, y otras disciplinas. Por tal motivo, existen muchos sistemas presentes en la naturaleza que ya cuentan con una formulación matemática en términos de ecuaciones diferenciales describiendo su comportamiento. Este hecho hace que sea muy útil poder describir de forma precisa este tipo de sistemas al momento de modelar sistemas físicos.

En este capítulo se presenta un modelo genérico en `ntcc` para la representación de sistemas continuos descritos por medio de sistemas de ecuaciones diferenciales. Para tal efecto se propone un esquema de ejecución de programas `ntcc` que permita describir las características de un sistema continuo mediante los operados temporales, de naturaleza discreta, que provee el cálculo. Se presentan ejemplos que muestran la forma de usar el modelo dada una especificación matemática (i.e., un sistema de ecuaciones diferenciales).

El modelo inicialmente propuesto se extiende para permitir el modelado de algunos sistemas con características temporales más complejas. Estos últimos tienen la característica de no tener una solución analítica, así es que solo métodos iterativos son posibles para su solución. Finalmente el modelo extendido se pone a prueba mediante el modelado del operón de la lactosa. Se muestran algunos aspectos de tipo práctico como generalidades sobre su implementación en Mozart-Oz y resultados de simulación en **ntccSim**.

Según nuestro mejor saber, capacidades de representación precisa de este tipo de sistemas, diferentes a las aquí propuestas, solo son disponibles en hcc en cuanto a cálculos de procesos se refiere. Esto seguramente es debido a la posibilidad de incluir fácilmente información cuantitativa en modelos de computación (i.e., cálculos de procesos) basados en restricciones.

4.1. Definición del modelo

El modelo para la representación de ecuaciones diferenciales usando el cálculo de procesos `ntcc` se propone en forma de un conjunto de procesos genéricos y paramétricos (i.e., "*encodings*") que representan las características necesarias para la descripción y simulación (i.e., búsqueda de solución) de sistemas de ecuaciones diferenciales.

4.1.1. Continuidad

Para la representación de sistemas continuos es necesario poder modelar dos características de este tipo de sistemas: continuidad en el *tiempo* y persistencia de los valores de las *variables* del sistema.

Para modelar el primer tipo de continuidad, relacionado con la naturaleza continua del tiempo, se propone un esquema de ejecución en el que en una unidad de tiempo continuo se ejecutan múltiples unidades de tiempo en `ntcc`. Para llevar un control explícito del tiempo, se define un proceso ($Time$) que permita conocer en todo momento el valor del tiempo. La definición formal del proceso $Time$ es la siguiente:

$$Time(t) \quad \stackrel{\text{def}}{=} \quad \mathbf{tell}(ts = t) \parallel \mathbf{next}\, Time(t + dt)$$

El parámetro t es el *valor* del tiempo continuo (representado por la variable ts) en la unidad de tiempo de `ntcc` inmediatamente anterior. Es importante recordar que el llamado a procedimientos usado está definido como paso de parámetros por valor tal como se describe en el capítulo 6 de [Val02]. Por otro lado, dt es una constante que representa la *granularidad temporal* o *resolución temporal* del sistema modelado. Por ejemplo, si el valor de la constante dt es 0.01 entonces se ejecutan 100 unidades de tiempo en `ntcc` para la representación de una unidad de tiempo continuo.

La constante dt, dada su definición, debe cumplir con la condición: $dt \times n = 1$ tal que $n \in N$. De esta forma, el valor máximo de esta constante es $dt = 1$, indicando que una unidad de tiempo continuo es representada por una unidad de tiempo en `ntcc`. Es claro que esto no es lo que se busca, puesto que en ese caso el sistema continuo sería simulado como un sistema discreto. Por tal motivo, se requiere tomar un valor lo suficientemente pequeño para dt de tal forma que se defina una resolución temporal adecuada para el sistema a modelar. Sin embargo, es necesario tener en cuenta que una consecuencia práctica importante de la escongencia del valor de la constante dt es el rendimiento o eficiencia de las simulaciones de este tipo de modelos. Entre menor sea el valor de dt, mayor será el número de unidades de tiempo en `ntcc` (i.e., n en la condición) que se requerirán ser ejecutadas para simular una unidad de tiempo continuo.

Para modelar el segundo tipo de continuidad, relacionado con los valores de la variables, es necesario contar con un mecanismo que permita transferir el estado del sistema (representado con el valor de sus variables) a la siguiente unidad de tiempo. Como las variables en `ntcc` son lógicas no es posible que tengan dos valores distintos en una misma unidad de tiempo. Por tal motivo, se usará una variable adicional x' por cada variable x del sistema, representando su valor en la unidad de tiempo anterior. Se define entonces, un proceso ($State_x$) que transfiera el valor de una variable del sistema (i.e., x en este caso) a la siguiente unidad de tiempo. La especificación formal de éste proceso es la siguiente:

$$State_x(v_x) \quad \stackrel{\text{def}}{=} \quad \mathbf{tell}(x' = v_x) \parallel \mathbf{next}\, State_x(x)$$

De la definición del proceso $State_x$, se deduce que el parámetro v_x es el *valor* de la variable x en la unidad de tiempo anterior que debe ser asignado a x'. Además, el valor actual de x es transferido a la siguiente unidad de tiempo. Ahora, se puede utilizar el proceso $State_x$ para definir un proceso más complejo ($State$) que permita mantener el estado del sistema completo en toda unidad de tiempo. La definición de éste proceso es la siguiente:

$$State(\rho_1, ..., \rho_n) \stackrel{\text{def}}{=} \prod_{x \in X} (\textbf{tell}(x = \rho_j) \parallel \textbf{next}\, State_x(\rho_x))$$

donde X es el conjunto de variables del sistema, y ρ_j (para $1 \leq j \leq n$ y $n = |X|$) el valor de la variable x en la primera unidad de tiempo.

Siguiendo el estilo de diseño composicional inherente a los cálculos de procesos, se procede a relacionar los procesos $Time$ y $State$ por medio del operador de composición paralela de siguiente forma:

$$Continuity \stackrel{\text{def}}{=} Time(0.0) \parallel State(\rho_1, ..., \rho_n)$$

4.1.2. Ecuaciones diferenciales de primer orden en ntcc

En esta sección se propone el uso de un sistema de restricciones sobre dominios continuos [Gro04] para la construcción de variables que representen ecuaciones diferenciales en `ntcc`. El sistema de restricciones, como es explicado en el capítulo 2, es usado como parámetro de los modelos definidos usando los operadores del cálculo.

Inicialmente se define un modelo general para la representación de ecuaciones diferenciales ordinarias (ODEs) de primer orden [Zil02a]. Suponga la existencia de un sistema de n ecuaciones diferenciales. Por cada una de estas ecuaciones se define una variable x. Se cuenta, además, con n variables x' según las definiciones presentadas en la sección 4.1.1. Cada variable x del sistema guarda el valor de la solución de la ecuación diferencial que ella representa en la unidad de tiempo actual.

Por lo anterior, la forma de interpretar la solución de una ecuación diferencial de primer orden representada por la variable x (la cual es necesariamente una ecuación no diferencial) es con la función definida por todos los valores solución de la variable x a lo largo del tiempo. Como consecuencia, la interpolación de todos estos valores define realmente una aproximación a la solución exacta de la ecuación diferencial. La precisión de esta aproximación depende de forma directa del número de valores que se tengan de x por cada unidad de tiempo continuo (i.e., número de unidades de tiempo en `ntcc` usadas para la representación de una unidad de tiempo

continuo). más precisamente, en el caso de este modelo depende del valor de la constante dt definida en la sección anterior.

Una consideración importante al encontrar la solución de una ecuación diferencial es el tipo de problema que ésta resuelve. El modelo propuesto en este capítulo se centra en la solución de problemas de valor inicial. Esta determinación permite utilizar los parámetros del proceso *State* para definir condiciones iniciales de las ecuaciones diferenciales de forma natural en el modelo.

4.1.3. Representación de cambios diferenciales

Dado el esquema de procesamiento de modelos en `ntcc` propuesto en la sección 4.1.1 es viable pensar que con cada ejecución de una unidad de tiempo en `ntcc` se va calculando de forma *diferencial* la solución completa del sistema de ecuaciones diferenciales. Se propone entonces, la utilización del *Método de Euler* [Zil02b] para calcular la solución del sistema de ecuaciones diferenciales.

Suponga la siguiente ecuación diferencial:

$$\frac{dx}{dt} = f(x,t)$$

Según el *Método de Euler* se puede definir la siguiente ecuación, conocida como *formula de Euler* [Zil02b], para calcular de forma iterativa la solución de una ecuación diferencial:

$$x_{n+1} = x_n + h \times f(x,t)$$

donde x_{n+1} es la solución de la ecuación diferencial después de $n+1$ iteraciones, x_n la solución en la iteración anterior y h el cambio diferencial de la variable independiente de la ecuación diferencial. Para sistemas continuos que cambian o evolucionan con respecto al tiempo se tiene que $h = t_{n+1} - t_n$. En el caso de la representación de este tipo de sistemas usando `ntcc` se cumple que $h = dt$.

Con base en lo anterior, se define el proceso DEq_x para la representación en `ntcc` de la ecuación diferencial $\frac{dx}{dt} = f(x,t)$:

$$DEq_x \stackrel{\text{def}}{=} \mathbf{next\,tell}(x = x' + dt \times f(x,t))$$

donde x y x' son las soluciones de la ecuación diferencial en los tiempos t y $t - dt$ respectivamente. Cabe anotar que solo es necesario definir un proceso para representar ecuaciones diferenciales de

primer orden ya que toda ecuación diferencial de orden superior [Zil02a] puede ser representada por un conjunto de ecuaciones diferenciales de primer orden.

Existe una variación al modelo planteado anteriormente para la representación de una ecuación diferencial cuando ésta se encuentra definida a trozos. En ese caso, se modifica el proceso DEq_x de la siguiente forma:

$$DEq_x \stackrel{\text{def}}{=} \sum_{i \in I} \textbf{when } i \textbf{ do next tell}(x = x' + dt \times f_i(x,t))$$

donde I es el conjunto de restricciones definiendo las condiciones que determinan los intervalos en los cuales esta definida la ecuación diferencial. En otras palabras, I define los trozos de la ecuación diferencial. Se puede observar que la primera definición de DEq_x es realmente un caso particular de la versión generalizada para aquellas definidas a trozos (i.e., cuando $I = \{\textbf{true}\}$).

A partir de la definición anterior para la representación en `ntcc` de una ecuación diferencial se construye de forma inmediata la definición de un sistema de ecuaciones diferenciales por medio del operador de composición paralela. Además, se *replican* los procesos que determinan la solución del sistema de ecuaciones diferenciales para obtener la solución del sistema completo en toda unidad de tiempo. La definición en `ntcc` es la siguiente:

$$DEqs \stackrel{\text{def}}{=} \; ! \prod_{x \in X} DEq_x$$

donde X es el conjunto de variables que representan las ecuaciones diferenciales del sistema. A continuación se integran los procesos presentados hasta el momento y se muestran algunos ejemplos sencillos que muestran la forma de usar el modelo completo.

4.1.4. Formalización del modelo integrado y aplicaciones

Las definiciones anteriormente planteadas permiten proponer un modelo general en `ntcc` para la representación de sistemas de ecuaciones diferenciales ordinarias de primer orden. Su especificación formal en `ntcc` es la siguiente:

$$DESystem \stackrel{\text{def}}{=} \textbf{local } ts, x_1, ..., x_n, x'_1, ..., x'_n \textbf{ in } (\ DEqs \parallel Continuity\ )$$

Con el proceso $DESystem$ es posible modelar una gran variedad de sistemas dada la generalidad de su especificación. A continuación se presentan dos ejemplos que muestran de forma concreta la forma de utilizar el modelo aquí propuesto.

Ejemplo 4.1 *Movimiento de una partícula.*

Suponga que desea modelar en `ntcc` *el comportamiento de una partícula cuyo movimiento esta determinado por la siguiente ecuación diferencial:*

$$\frac{dx}{dt} = cos(t)$$

Se sabe que la partícula inició su movimiento en la posición $x = 0$*. Usando un sistema de restricciones sobre dominios continuos, tal como se indicó en la sección 4.1.2, se plantea el siguiente modelo en* `ntcc`*:*

$$\begin{array}{lcl}
Time(t) & \stackrel{\text{def}}{=} & \mathbf{tell}(ts = t) \parallel \mathbf{next}\, Time(t + dt) \\
State_x(v_x) & \stackrel{\text{def}}{=} & \mathbf{tell}(x' = v_x) \parallel \mathbf{next}\, State_x(x)\ ) \\
State(0.0) & \stackrel{\text{def}}{=} & \mathbf{tell}(x = 0.0) \parallel \mathbf{next}\, State_x(0.0) \\
Continuity & \stackrel{\text{def}}{=} & Time(0.0) \parallel State(0.0) \\
DEq_x & \stackrel{\text{def}}{=} & \mathbf{next}\, \mathbf{tell}(x = x' + dt \times (cos(ts))) \\
DEqs & \stackrel{\text{def}}{=} & !\ DEq_x \\
DESystem & \stackrel{\text{def}}{=} & \mathbf{local}\ ts, x, x'\ \mathbf{in}\ (\ DEqs \parallel Continuity\ )
\end{array}$$

El comportamiento del modelo anterior, determinado por la semántica operacional de `ntcc`*, es simulado usando* ***ntccSim****. Los resultados de la simulación son los siguientes:*

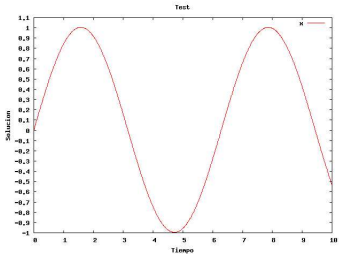


Figura 4.1: Movimiento de una partícula

Ejemplo 4.2 *Interacción entre dos genes.*

El siguiente ejemplo es extraído de [BC02] donde es modelado usando el cálculo hcc. Se busca describir la interacción entre dos genes. Este sistema genético presenta características discretas y continuas en su comportamiento, determinadas por el siguiente sistema de ecuaciones diferenciales:

$$\begin{aligned}
\frac{dy}{dt} &= 0.01x \\
\frac{dx}{dt} &= \begin{cases} -0.02x + 0.01 & para \quad y < 0.8 \\ -0.02x & para \quad y \geq 0.8 \end{cases}
\end{aligned}$$

La formalización del anterior sistema de ecuaciones diferenciales es la siguiente:

$$\begin{array}{lcl}
Time(t) & \stackrel{\text{def}}{=} & \mathbf{tell}(ts = t) \parallel \mathbf{next}\, Time(t + dt) \\
State_y(v_y) & \stackrel{\text{def}}{=} & \mathbf{tell}(y' = v_y) \parallel \mathbf{next}\, State_y(y) \\
State_x(v_x) & \stackrel{\text{def}}{=} & \mathbf{tell}(x' = v_x) \parallel \mathbf{next}\, State_x(x) \\
State(0.0, 0.0) & \stackrel{\text{def}}{=} & (\ \mathbf{tell}(y = 0.0) \parallel \mathbf{next}\, State_y(0.0)\ ) \\
 & & \parallel \\
 & & (\ \mathbf{tell}(x = 0.0) \parallel \mathbf{next}\, State_x(0.0)\ ) \\
Continuity & \stackrel{\text{def}}{=} & Time(0.0) \parallel State(0.0, 0.0) \\
DEq_y & \stackrel{\text{def}}{=} & \mathbf{next}\, \mathbf{tell}(y = y' + dt \times (0.01x')) \\
DEq_x & \stackrel{\text{def}}{=} & \mathbf{when}\ y < 0.8\ \mathbf{do}\ \mathbf{next}\, \mathbf{tell}(x = x' + dt \times (-0.02x' + 0.01)) \\
 & & + \\
 & & \mathbf{when}\ y \geq 0.8\ \mathbf{do}\ \mathbf{next}\, \mathbf{tell}(x = x' + dt \times (-0.02x')) \\
DEqs & \stackrel{\text{def}}{=} & !\ \ (\ DEq_y \parallel DEq_x\ ) \\
DESystem & \stackrel{\text{def}}{=} & \mathbf{local}\ ts, x, y, x', y'\ \mathbf{in}\ (\ DEqs \parallel Continuity\ )
\end{array}$$

Los resultados de la simulación del modelo anterior son presentados en la figura 4.2. Se puede apreciar que la gráfica concuerda con la solución presentada en [BC02].

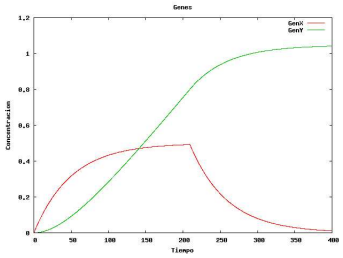


Figura 4.2: Interacción entre dos genes

4.2. Sistemas dinámicos con dependencias temporales complejas

En algunas ocasiones las ecuaciones diferenciales ordinarias no son lo suficientemente expresivas para describir de forma precisa ciertos sistemas continuos. Este es el caso de sistemas en los que alguna de las variables usadas para modelar su comportamiento depende en el tiempo t de una función de la forma $f(t_p)$, donde $t_p < t$. Es decir, existe un *retraso* en el sistema. Por ejemplo un sistema de ecuaciones diferenciales como el que se muestra a continuación:

$$\begin{aligned} \frac{df(t)}{dt} &= cos(t) \\ \frac{dg(t)}{dt} &= 5f(t-2) \end{aligned}$$

Aquí puede observarse que la ecuación diferencial $\frac{dg(t)}{dt}$ depende en el tiempo t del valor de la función f en el tiempo $t = t - 2$.

Otro ejemplo concreto de este tipo de sistemas son las redes de regulación genética. En ellas existen genes expresándose de tal forma que la cantidad de proteína producida por un gen depende de forma directa de la cantidad de ARN mensajero que había en el sistema en un momento en el pasado. Este tipo de comportamiento es comúnmente descrito por *ecuaciones diferenciales retrasadas* [HNW05]. Un sistema bastante complejo modelado con este tipo de ecuaciones diferenciales es el operón de la lactosa.

A continuación se presenta una extensión al modelo planteado en la sección anterior para que sea posible especificar formalmente en `ntcc` sistemas descritos con ecuaciones diferenciales retrasadas.

4.2.1. Extensión del modelo en ntcc

Para modelar este tipo de ecuaciones diferenciales es necesario superar dos limitantes que tiene el modelo anteriormente planteado. Primero, poder calcular el valor solución de una ecuación diferencial que depende de un valor pasado. Segundo, determinar el valor inicial de la ecuación diferencial entre el tiempo inicial (i.e., $t = 0.0$) y el tiempo de retraso de la ecuación diferencial (i.e., $t = \tau$).

Se define un conjunto de variables y, tal que $y \in Y$, donde Y es el conjunto de variables que tienen un retraso en el sistema, es decir, variables que dependen de un valor del pasado de una variable x (donde $x \in X$; X es el conjunto de variables del sistema definidas anteriormente). Para superar el primer obstáculo mencionado se define un proceso en `ntcc` de la siguiente forma:

$$\begin{aligned} Q_y(v_x, tf) &\stackrel{\text{def}}{=} \mathbf{when}\ ts < tf\ \mathbf{do\ next}\, Q_y(v_x, tf) \\ &\quad + \\ &\quad \mathbf{when}\ ts \geq tf\ \mathbf{do\ tell}(y = v_x) \\ R_{x,y}(td) &\stackrel{\text{def}}{=} Q_y(x, ts + td) \parallel \mathbf{next}\, R_{x,y}(td) \end{aligned}$$

donde td es el tiempo de retraso de y con respecto a x, y ts el tiempo continuo del sistema. De esta forma, la variable y en un tiempo $t \geq (ts + td) = tf$ tiene el valor de la variable x (i.e.,

v_x) en el tiempo $t = ts$. La desigualdad anterior (i.e., $t \geq tf$) es debida a que no siempre tf es múltiplo exacto de dt (la resolución temporal del sistema).

Como puede deducirse de la especificación del proceso anterior, la variable y queda determinada para todo instante de tiempo superior o igual a td. Sin embargo, es necesario definir un proceso más para determinar el valor de esta variable para todo instante de tiempo entre cero y td. La definición de este proceso permite que se supere el segundo obstáculo mencionado al principio de esta sección. La especificación formal en `ntcc` para este proceso es la siguiente:

$$P_y(v_x, td) \stackrel{\text{def}}{=} \textbf{when } ts < td \textbf{ do } (\ \textbf{tell}(y = v_x) \parallel \textbf{next } P_y(v_x, td)\ )$$

donde v_x es el valor inicial de la solución de la ecuación diferencial representada por x.

Con los procesos definidos anteriormente (i.e., P_y, Q_y y $R_{x,y}$) se procede a formular un proceso unificado que de cuenta del retraso de la variable y con respecto a la variable x:

$$Delay_{x,y} \stackrel{\text{def}}{=} P_y(\rho_x, \tau) \parallel R_{x,y}(\tau)$$

donde ρ_x es el valor inicial de x y τ el retraso de y con respecto a x.

Ahora se define un proceso que integre los retrasos temporales indicados por las variables que están en el conjunto Y.

$$Delay \stackrel{\text{def}}{=} \prod_{x \in X, y \in Y} Delay_{x,y}$$

Finalmente se integra el proceso $Delay$ con aquellos usados para definir $DESystem$:

$$DDESystem \stackrel{\text{def}}{=} \textbf{local } ts, x_i, ..., x_n, x_i', ..., x_n', xd_j, ..., xd_k \textbf{ in } (\ DEqs \parallel Continuity \parallel Delay\ )$$

4.3. Modelo celular del operón de la lactosa en ntcc

En esta sección se muestra de forma concreta el uso del proceso $DDESystem$ para la representación de sistemas de ecuaciones diferenciales retrasadas. Para tal efecto, se presenta su uso sobre un modelo biológico (i.e., un sistema de ecuaciones diferenciales) del operón de la lactosa [YSM04, YM03].

El modelo en [YSM04, YM03] asume un ambiente o medio rico en lactosa y pobre en glucosa. Su formulación se produjo basándose en la experimentación sobre una población de células. Por tal motivo, los resultados encontrados corresponden al comportamiento generalizado de un grupo de células y no al comportamiento particular (a veces con características discretas) de una sola célula, donde un modelo molecular es el más apropiado. El modelo biológico a continuación (i.e., un sistema de ecuaciones diferenciales retrasadas) no tiene solución analítica, por lo que en cualquier caso es necesario de un método iterativo para encontrar una aproximación a su solución. Para detalles biológicos sobre el funcionamiento del operón de la lactosa el lector puede referirse al capítulo 3.

4.3.1. Modelo biológico

El siguiente modelo matemático fue tomado de [YSM04, YM03]. Detalles biológicos sobre los parámetros usados en el modelo pueden encontrarse en el apéndice B de [YM03].

El sistema de ecuaciones diferenciales es el siguiente:

$$\begin{aligned}
\frac{dm}{dt} &= \alpha_M \frac{1 + K_1(e^{-\mu\tau_M} a_{\tau_M})^n}{K + K_1(e^{-\mu\tau_M} a_{\tau_M})^n} - (\gamma_M + \mu)m \\
\frac{db}{dt} &= \alpha_B e^{-\mu\tau_B} m_{\tau_B} - (\gamma_B + \mu)b \\
\frac{dp}{dt} &= \alpha_P e^{-\mu(\tau_B+\tau_P)} m_{\tau_B+\tau_P} - (\gamma_P + \mu)p \\
\frac{dl}{dt} &= \alpha_L p \frac{L_e}{K_{L_e} + L_e} - \beta_{L_1} p \frac{l}{K_{L_1} + l} - \alpha_A b \frac{l}{K_L + l} - (\gamma_L + \mu)l \\
\frac{da}{dt} &= \alpha_A b \frac{l}{K_L + l} - \beta_A b \frac{a}{K_A + a} - (\gamma_A + \mu)a
\end{aligned}$$

Parámetro	**Valor**	**Parámetro**	**Valor**
n	2	μ_{max}	$0{,}0347 min^{-1}$
γ_M	$0{,}411 min^{-1}$	γ_B	$0{,}000833 min^{-1}$
γ_A	$0{,}52 min^{-1}$	K	1000,0
α_M	$0{,}000997 mM\ min^{-1}$	τ_B	$2{,}0 min$
α_A	$17600{,}0 min^{-1}$	K_{L_1}	$1{,}81 mM$
α_B	$0{,}0166 min^{-1}$	K_A	$1{,}85 mM$
β_A	$21500{,}0 min^{-1}$	τ_M	$0{,}1 min$
K_L	$0{,}97 mM$	γ_L	$0{,}0 min^{-1}$
γ_P	$0{,}65 min^{-1}$	α_L	$2880{,}0 min^{-1}$
α_P	$10{,}0 min^{-1}$	τ_P	$0{,}83 min$
β_{L_1}	$2650 min^{-1}$	K_{L_e}	$0{,}26 mM$
K_1	$25200{,}0 mM^{-2}$		

Cuadro 4.1: Parámetros del modelo celular del operón de la lactosa

donde m, b, p, l y a son las concentraciones celulares de ARN mensajero, β-galactosidasa,

permeasa, lactosa y alolactosa respectivamente. Además, m_{τ_B}, $m_{\tau_B+\tau_P}$ y a_{τ_M} son los valores de las concentraciones de ARN mensajero y alolactosa un tiempo τ en el pasado. La razón de ser de estas dependencias temporales es que la producción (i.e., síntesis) de β-galactosidasa y permeasa depende en el tiempo $t = ts$ de la cantidad de ARN mensajero que había en los tiempos $t = ts - \tau_B$ y $t = ts - (\tau_B + \tau_P)$. Así, estos retrasos en el sistema (i.e., τ_B y τ_P) tienen en cuenta el tiempo que demora la ARN polimerasa en transcribir los genes $LacZ$ y $LacY$, codificantes de las proteínas β-galactosidasa y permeasa. Un análisis similar puede hacerse para el caso de la alolactosa.

Otros datos usados en el modelo son las constantes $L_e = 0.08$ y $\mu = 0.0226$, representando la concentración promedio de lactosa en el medio y la rata de crecimiento de la población de células respectivamente.

4.3.2. Modelo computacional en ntcc

El modelo matemático del operón de la lactosa mostrado en la sección anterior es representado en `ntcc` por medio del proceso $DDESystem$ presentado en la sección 4.2.1. Las condiciones iniciales del sistema de ecuaciones diferenciales aquí descrito fueron tomadas del modelo matemático presentado en [YM03]:

Variable	**Valor inicial**	**Variable**	**Valor inicial**	**Variable**	**Valor inicial**
m_o	$0.000626mM$	l_o	$0.372mM$	p_o	$0.0149mM$
b_o	$0.0mM$	a_o	$0.038mM$		

Cuadro 4.2: Condiciones iniciales del modelo celular del operón de la lactosa

Estas condiciones iniciales (presentadas en el cuadro 4.2) son usadas para la definición del proceso $Continuity$.

$$
\begin{array}{lcl}
Time(t) & \stackrel{\text{def}}{=} & \mathbf{tell}(ts = t) \parallel \mathbf{next}\, Time(t + dt) \\
State_m(v_m) & \stackrel{\text{def}}{=} & \mathbf{tell}(m' = v_m) \parallel \mathbf{next}\, State_m(m) \\
State_b(v_b) & \stackrel{\text{def}}{=} & \mathbf{tell}(b' = v_b) \parallel \mathbf{next}\, State_b(b) \\
State_p(v_p) & \stackrel{\text{def}}{=} & \mathbf{tell}(p' = v_p) \parallel \mathbf{next}\, State_p(p) \\
State_l(v_l) & \stackrel{\text{def}}{=} & \mathbf{tell}(l' = v_l) \parallel \mathbf{next}\, State_l(l) \\
State_a(v_a) & \stackrel{\text{def}}{=} & \mathbf{tell}(a' = v_a) \parallel \mathbf{next}\, State_a(a) \\
State(m_o, b_o, p_o, l_o, a_o) & \stackrel{\text{def}}{=} & (\ \mathbf{tell}(m = m_o) \parallel \mathbf{next}\, State_m(m_o)\ ) \ \parallel \\
 & & (\ \mathbf{tell}(b = b_o) \parallel \mathbf{next}\, State_b(b_o)\ ) \ \parallel \\
 & & (\ \mathbf{tell}(p = p_o) \parallel \mathbf{next}\, State_p(p_o)\ ) \ \parallel \\
 & & (\ \mathbf{tell}(l = l_o) \parallel \mathbf{next}\, State_l(l_o)\ ) \ \parallel \\
 & & (\ \mathbf{tell}(a = a_o) \parallel \mathbf{next}\, State_a(a_o)\ ) \\
Continuity & \stackrel{\text{def}}{=} & Time(0.0) \parallel State(m_o, b_o, p_o, l_o, a_o)
\end{array}
$$

Ahora se utilizan las constantes τ_M, τ_B y τ_P y las variables a_m, m_b y m_p para definir las dependencias temporales *retrasadas* en las ecuaciones diferenciales, utilizando el proceso $Delay$, que

determinan el comportamiento del ARN mensajero, la β-galactosidasa y la permeasa respectivamente. La definición del proceso *Delay* es la siguiente:

$$
\begin{array}{lcl}
P_{m_b}(v_m, td) & \stackrel{\text{def}}{=} & \textbf{when } ts < td \textbf{ do } (\ \textbf{tell}(m_b = v_m) \parallel \textbf{next } P_{m_b}(v_m, td)\ ) \\
Q_{m_b}(v_m, tf) & \stackrel{\text{def}}{=} & \textbf{when } ts < tf \textbf{ do next } Q_{m_b}(v_m, tf) \ + \ \textbf{when } ts \geq tf \textbf{ do tell}(m_b = v_m) \\
R_{m,m_b}(td) & \stackrel{\text{def}}{=} & Q_{m_b}(m, ts + td) \parallel \textbf{next } R_{m,m_b}(td) \\
Delay_{m,m_b} & \stackrel{\text{def}}{=} & P_{m_b}(m_o, \tau_B) \parallel R_{m,m_b}(\tau_B) \\
P_{m_p}(v_m, td) & \stackrel{\text{def}}{=} & \textbf{when } ts < td \textbf{ do } (\ \textbf{tell}(m_p = v_m) \parallel \textbf{next } P_{m_p}(v_m, td)\ ) \\
Q_{m_p}(v_m, tf) & \stackrel{\text{def}}{=} & \textbf{when } ts < tf \textbf{ do next } Q_{m_p}(v_m, tf) \ + \ \textbf{when } ts \geq tf \textbf{ do tell}(m_p = v_m) \\
R_{m,m_p}(td) & \stackrel{\text{def}}{=} & Q_{m_p}(m, ts + td) \parallel \textbf{next } R_{m,m_p}(td) \\
Delay_{m,m_p} & \stackrel{\text{def}}{=} & P_{m_p}(m_o, \tau_B + \tau_P) \parallel R_{m,m_p}(\tau_B + \tau_P) \\
P_{a_m}(v_a, td) & \stackrel{\text{def}}{=} & \textbf{when } ts < td \textbf{ do } (\ \textbf{tell}(a_m = v_a) \parallel \textbf{next } P_{a_m}(v_a, td)\ ) \\
Q_{a_m}(v_a, tf) & \stackrel{\text{def}}{=} & \textbf{when } ts < tf \textbf{ do next } Q_{a_m}(v_a, tf) \ + \ \textbf{when } ts \geq tf \textbf{ do tell}(a_m = v_a) \\
R_{a,a_m}(td) & \stackrel{\text{def}}{=} & Q_{a_m}(a, ts + td) \parallel \textbf{next } R_{a,a_m}(td) \\
Delay_{a,a_m} & \stackrel{\text{def}}{=} & P_{a_m}(a_o, \tau_M) \parallel R_{a,a_m}(\tau_M) \\
Delay & \stackrel{\text{def}}{=} & Delay_{m,m_b} \parallel Delay_{m,m_p} \parallel Delay_{a,a_m}
\end{array}
$$

Finalmente se definen los procesos `ntcc`, inspirados en la formulación de Euler, para representar las ecuaciones diferenciales y se integra el modelo completo en el proceso $DDESystem$:

$$
\begin{array}{lcl}
DEq_m & \stackrel{\text{def}}{=} & \textbf{next tell}(m = m' + dt \times (\alpha_M \frac{1+K_1(e^{-\mu\tau_M} a_m)^n}{K+K_1(e^{-\mu\tau_M} a_m)^n} - (\gamma_M + \mu)m')) \\
DEq_b & \stackrel{\text{def}}{=} & \textbf{next tell}(b = b' + dt \times (\alpha_B e^{-\mu\tau_B} m_b - (\gamma_B + \mu)b')) \\
DEq_p & \stackrel{\text{def}}{=} & \textbf{next tell}(p = p' + dt \times (\alpha_P e^{-\mu(\tau_B+\tau_P)} m_p - (\gamma_P + \mu)p')) \\
DEq_l & \stackrel{\text{def}}{=} & \textbf{next tell}(l = l' + dt \times (\alpha_L p' \frac{L_e}{K_{L_e}+L_e} - \beta_{L_1} p' \frac{l'}{K_{L_1}+l'} - \alpha_A b' \frac{l'}{K_L+l'} - (\gamma_L + \mu)l')) \\
DEq_a & \stackrel{\text{def}}{=} & \textbf{next tell}(a = a' + dt \times (\alpha_A b' \frac{l'}{K_L+l'} - \beta_A b' \frac{a'}{K_A+a'} - (\gamma_A + \mu)a')) \\
DEqs & \stackrel{\text{def}}{=} & !\ (\ DEq_m \parallel DEq_b \parallel DEq_p \parallel DEq_l \parallel DEq_a\ ) \\
DDESystem & \stackrel{\text{def}}{=} & \textbf{local } ts, m, b, p, l, a, m', b', p', l', a', m_b, m_p, a_m \textbf{ in} \\
 & & \quad (\ DEqs \parallel Continuity \parallel Delay\ )
\end{array}
$$

4.4. Implementación en Mozart-Oz

La implementación del modelo anterior se hizo usando el simulador de procesos `ntcc` descrito en el capítulo 2. A continuación se muestra la estructura general del programa en Mozart-Oz. Las variables usadas son construidas a partir de un sistema de restricciones sobre dominios reales (XRI) [Gro04].

Inicialmente se definen los paquetes que se van a importar para ser usados en el programa. También, se declaran las variables y constantes usadas en el modelo. Posteriormente se definen sus dominios y valores respectivamente. En cada caso, los puntos suspensivos indican definiciones del mismo tipo a la indicada encima de ellos.

```
functor
import
   %Packages and files used in the program
define
   %Declaration of variables and global constants
   %Variables
   DEVars = vars(m:_ b:_ p:_ a:_ l:_ t:_)   %Differential Equations Vars
   DEDels = vars(mb:_ mp:_ am:_)   %Delay vars
   T = vars(tntcc:_)   %ntcc time
   SVars = {Record.adjoin T {Record.adjoin DEVars DEDels}}  %System Vars
   %Constants
   Resolution = 10.0   %System Resolution
   Dt = 1.0 / Resolution   %Differential of time
   MaxTime = 200   %Time of simulation (system units)
   %Definition of biological parameters (constants)
   Leo = 0.08
   ...
   %Definition of initial conditions of the equations (constants)
   Mo = 0.000626
   ...
   %Definition of domains (variables)
   {Record.forAll DEVars proc{$ Xi} Xi = {RI.var.exp "[0.0, 100.0]"} end}
   ...
```

Para la implementación de los procesos *Continuity* y *Delay* se declaran un conjunto de funciones en Mozart-Oz que retornan código simulable (ver detalles de la herramienta de simulación en el capítulo 2). La implementación de los procesos $State_i$ no es necesaria dado que el simulador cuenta con los campos *current* y *residual* en la variable `Root` tal como se describe en el capítulo 2. Por otro lado, en la implementación del proceso *Delay* los valores de las variables con retraso se calculan inmediatamente en el instante de tiempo en que se producen, y son enviadas por medio de operadores **next** al instante de tiempo en que se requieren. Esto es debido a que en la herramienta de simulación no es posible tener el valor de una variable en *cualquier* instante de tiempo anterior.

```
   %CONTINUITY PROCESS
   %General process definitions
   %Calling functions
   %DELAY PROCESS
   %General process definitions
   %Calling functions
   DelayM = {DelayXY a am Ao Tm}
   DelayB = {DelayXY m mb Mo Tb}
   DelayP = {DelayXY m mp Mo Tp}
   Delay  = par(DelayM DelayB DelayP)
```

En la implementación de cada proceso DEq_x se utiliza una plantilla (i.e., proceso *genérico* en `ntcc`) para su construcción. Así, todos los procesos DEq_x tienen la misma forma en su implementación, pero usan diferentes variables y restricciones para su representación en Mozart-

Oz según su definición en `ntcc`. En el ejemplo que se muestra a continuación (i.e., implementación del proceso DEq_m) tienen lugar las siguientes correspondencias:

Especificación en `ntcc`	**Implementación en Mozart-Oz**
m	`Mc`
m'	`Mr`
dt	`Dt`
$\alpha_M \frac{1+K_1(e^{-\mu\tau_M} a_m)^n}{K+K_1(e^{-\mu\tau_M} a_m)^n} - (\gamma_M + \mu)m'$	`Mode`
$m = m' + dt \times (\alpha_M \frac{1+K_1(e^{-\mu\tau_M} a_m)^n}{K+K_1(e^{-\mu\tau_M} a_m)^n} - (\gamma_M + \mu)m')$	`DeM`
next tell$(m = m' + dt \times (\alpha_M \frac{1+K_1(e^{-\mu\tau_M} a_m)^n}{K+K_1(e^{-\mu\tau_M} a_m)^n} - (\gamma_M + \mu)m'))$	`DEqm`

```
%DEqs PROCESS
%mRNA
DEqm = next(
            tell(
                 proc{$ Root}
                     Mc Mr Mode DeM in
                     Mc = Root.current.m
                     Mr = Root.residual.m
                     Mode = ...
                     DeM = eq(Mc plus(Mr times(Mode Dt)))
                    {RI.hc4 DeM}
                 end
                )
           )
%Differential equation system
DEqs = rep(
           par(
               DEqm DEqb DEqp DEql DEqa
              )
          )
%System
DDESystem = par(
                DEqs Continuity Delay
               )
```

En el programa anterior, la línea {`RI.hc4 DeM`} lanza o impone un propagador sobre la restricción `DeM` usando la técnica de consistencia hc4 [Gro04] implementada sobre los XRI. Se omiten los procedimientos que implementan las demás ecuaciones diferenciales del sistema (i.e., `DEqb`, `DEqp`, `DEql` y `DEqa`).

Finalmente, el programa termina con la declaración de un procedimiento que se desee ejecutar entre dos unidades de tiempo en `ntcc`. En nuestro caso, se declara el procedimiento `SaveAllFiles`, para guardar los resultados de la simulación de una unidad de tiempo en `ntcc` (i.e., el estado de las variables en el almacén de restricciones actual). Además, se llama al simulador de procesos `ntcc`, se indican cuales son las variables del sistema, las unidades de tiempo en `ntcc` a ejecutar y el procedimiento para guardar los resultados de la simulación. La linea {`Application.exit 0`} finaliza el programa.

```
   SaveAllFiles = ...
in
   {
    NTCC.simulate [DDESystem]        %Simulation Process
    SVars                            %System Vars
    {FloatToInt Resolution}*MaxTime  %Simulation Time (ntcc time units)
    SaveAllFiles                     %Saving Simulation Results
   }
   {Application.exit 0}
end
```

Como puede apreciarse, el programa anterior calcula la solución del sistema de ecuaciones diferenciales que representa el comportamiento del operón de la lactosa a nivel celular. Se omiten detalles del programa completo.

4.4.1. Resultados de la simulación en ntccSim

Los resultados de la simulación del programa presentado en la sección anterior muestran el comportamiento dinámico de las concentraciones de las proteínas tenidas en cuenta para determinar el comportamiento regulatorio del operón de la lactosa. Las gráficas de la simulación se presentan en la figura 4.3.

La gráfica 4.3(a) muestra el comportamiento del ARN mensajero cuando el operón de la lactosa inicia el proceso de regulación. Como se indico anteriormente, el medio en el cual se encuentra la población de células presenta alta concentración de lactosa y baja concentración de glucosa. Por tal motivo, la población de células es inducida por la lactosa del medio resultando en una activación del operón. El primer resultado verificable de la inducción del operón es el incremento en la producción de ARN mensajero producto del proceso de *transcripción*.

En las gráficas 4.3(b) y 4.3(c) se aprecia que después de iniciado el proceso de transcripción (i.e., gráfica del ARN mensajero), se inicia la *traducción* del operón. Esto resulta en la síntesis de las proteínas β-galactosidasa y permeasa, las cuales son codificadas por los genes $LacZ$ y $LacY$ respectivamente. Hasta este punto las gráficas 4.3(a), 4.3(b) y 4.3(c), muestran el comportamiento del operón de la lactosa siguiendo el dogma central de la biología molecular.

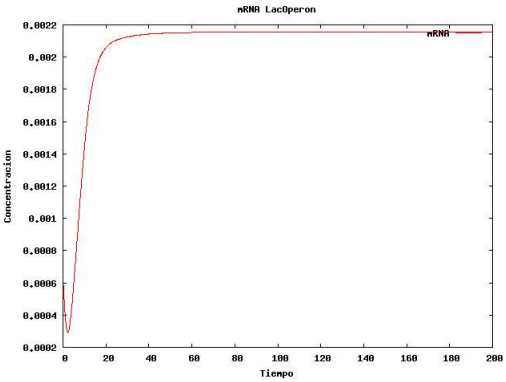
Producto del proceso de *regulación* genética anterior, en las gráficas 4.3(d) y 4.3(e) puede notarse que la población de células pese a que no desperdicia fuentes de energía, tampoco las consume desmesuradamente. Por tal motivo, después de tener la cantidad suficiente para cumplir con sus funciones celulares tanto las concentraciones de lactosa como de alolactosa alcanzan un estado de estabilidad.

Finalmente, aunque en el operón de la lactosa es posible identificar más elementos biológicos si la red de regulación se analiza con más detalle, el sistema aquí presentado muestra los aspectos más relevantes que determinan su comportamiento a *nivel celular*. Un enfoque de estudio de sistemas biológicos de este tipo (i.e., por niveles) es claramente promovido por los resultados encontrados en la biología sistémica.

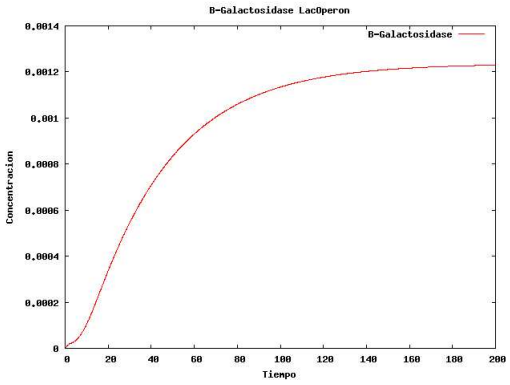
4.5. Resumen

En este capítulo se presentó un esquema de ejecución de modelos en `ntcc` y un conjunto de procesos básicos que permiten la representación de sistemas continuos descritos con ecuaciones diferenciales ordinarias (ODEs). En particular, se definieron procesos para modelar continuidad temporal y persistencia en las variables del sistema. Además, se definió un proceso para la solución iterativa de ecuaciones diferenciales basado en el Método de Euler. Dos ejemplos se presentan para mostrar de forma concreta la manera de usar los "encodings" de ecuaciones diferenciales en `ntcc` aquí propuestos.

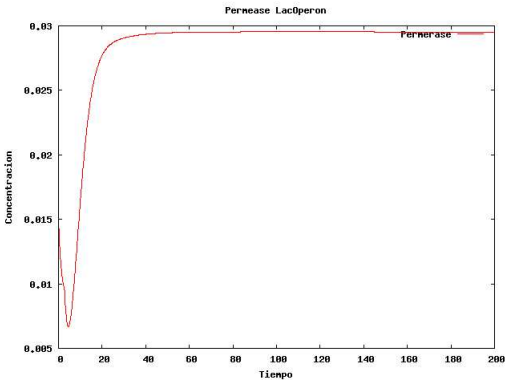
Posteriormente, el modelo en `ntcc` se extendió para permitir la representación de sistemas con dependencias temporales más complejas (i.e., sistemas de ecuaciones diferenciales retrasadas). El modelo extendido fue usado para describir el comportamiento a nivel celular del operón de la lactosa. Siguiendo un enfoque de programación declarativa, inherente a los modelos de computación basados en restricciones, se implementó y simuló el modelo en `ntcc` del operón de la lactosa. Un aspecto importante de la herramienta de simulación es que ésta ejecuta procesos del cálculo `ntcc` siguiendo su semántica operacional. Finalmente se presentan las gráficas con los resultados de la simulación. El funcionamiento del sistema es explicado a la luz del dogma central de la biología molecular y la biología sistémica.



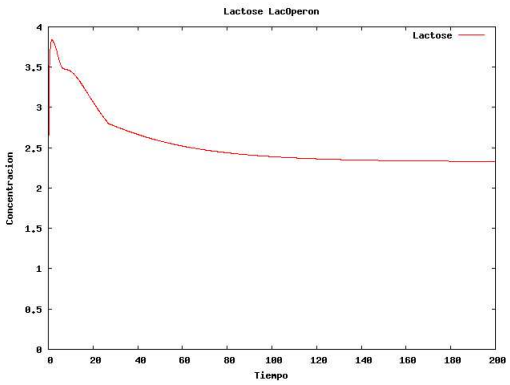
(a) ARN mensajero



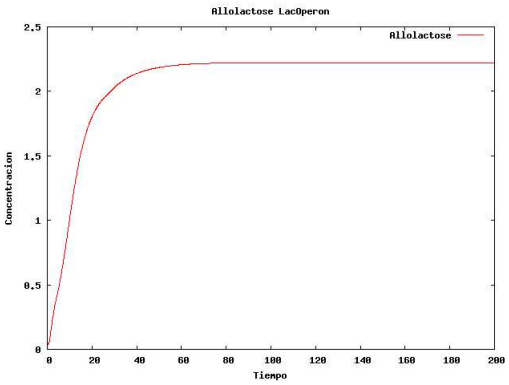
(b) β-galactosidasa



(c) Permeasa



(d) Lactosa



(e) Alolactosa

Figura 4.3: Comportamiento dinámico del operón de la lactosa a nivel celular

5 Modelado, simulación y verificación de redes de regulación genética

En este capítulo se presentan un conjunto de herramientas que permiten (i) *describir* formalmente el comportamiento de redes de regulación genética usando un cálculo de procesos temporal basado en restricciones, (ii) *simular* los modelos derivados de tal proceso de descripción y (iii) *verificar* propiedades sobre estos modelos. Además, se presenta un modelo basado en parámetros del operón de la lactosa, el cual es descrito usando los procesos definidos en este capítulo.

En la primera parte de este capítulo se presenta un conjunto de procesos *genéricos* y *paramétricos* en `ntcc` que permiten modelar las principales características de las redes de regulación genética. Posteriormente en la sección 5.2, tales procesos son usados para proponer un modelo que describe la estructura y el comportamiento a nivel molecular del operón de la lactosa. Este modelo es simulado en Mozart-Oz usando **ntccSim** con el fin de observar el comportamiento del sistema. Finalmente en la sección 5.4 se muestra como es posible usar el sistema de inferencia asociado con `ntcc` para *verificar* propiedades sobre el modelo.

5.1. Redes de regulación genética en ntcc

Desde el nacimiento de la biología sistémica, las redes de regulación genética se han constituido como uno de los sistemas más estudiados por su importancia a nivel celular. Estas controlan o regulan los procesos celulares según la *información* presente en el ADN de cada organismo. Sin embargo, existen características a nivel molecular en este tipo de redes que las vuelven especialmente complejas. Por ejemplo, la presencia combinada de elementos *discretos* y *continuos* tanto en su constitución como en su funcionamiento, haciendo compleja la tarea de formular un modelo matemático compacto que describa su comportamiento tal como se mostró en el capítulo anterior. Por tal motivo, los procesos que se presentan a continuación intentan modelar comportamientos tanto discretos como continuos para representar de la mejor forma posible los eventos moleculares presentes en estas redes.

5.1.1. Definición de variables

En la definición de los procesos genéricos y paramétricos presentados más adelante en este capítulo se usarán de forma combinada variables reales y discretas en el modelado según sea el

caso. Al igual que en el capítulo anterior, se usará el proceso *Continuity* para tener una visión continua de las variables de los sistemas a modelar. Por la definición del proceso *State*, éste puede ser usado también para hacer persistente el valor de las variables discretas del sistema. De esta forma, en este modelo se usarán de forma combinada *variables discretas* definidas a partir de un sistema de restricciones sobre enteros de dominio finito [Con06] y *variables reales* definidas a partir de un sistema de restricciones sobre dominios continuos [Gro04].

La notación de las variables usadas en el proceso *Continuity* será m_i y m_i', donde $i \in N$, para indicar su valor en la unidad de tiempo actual y anterior respectivamente, según la codificación usada para la representación de sistemas continuos descrita anteriormente. En adelante, por cuestiones de simplicidad, se hará mención a la entidad biológica representada por la variable m_i (e.g., concentración de moléculas, niveles de expresión de un gen, estado de una región genómica, etc.) simplemente haciendo referencia al subíndice de la variable (i.e., i para indicar la variable m_i). Así, el modelo estará compuesto por variables de la forma $m_1, m_2, ..., m_n$, donde n es el número de entidades biológicas identificadas en el sistema a nivel molecular.

5.1.2. Procesos de señalización

En la descripción de sistemas a nivel molecular existen muchos eventos que tienen que ser considerados para realizar un modelado adecuado. Por ejemplo, cuando un grupo de moléculas interactúa con otro, realiza una tarea o acción específica o produce una señal de control biológico. Dada la naturaleza de este tipo de eventos, se usarán variables *discretas* para indicar su presencia o ausencia en el sistema a modelar. Estas tipo de variables serán llamadas *variables de señalización* en el resto del documento. Se define, entonces, el proceso genérico *Signal* para modelar este tipo de comportamiento:

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

donde E es el conjunto de restricciones expresando eventos moleculares en el sistema, y S el conjunto de variables de señalización. Es importante notar que la anterior especificación no es igual a una estructura del tipo *if-then-else* ya que a diferencia de ésta el proceso *Signal* puede razonar además sobre la *ausencia* de información. De esta forma, es siempre posible determinar el valor de *svar* sin importar si la restricción e puede ser inferida o no a partir de la información en el almacén de restricciones. Por ejemplo, cuando $store \models e$ se impondrá la restricción $svar = 1$ en la siguiente unidad de tiempo. De lo contrario, si e no puede ser inferida a partir de la información (parcial) en el almacén de restricciones, entonces el proceso **unless** impondrá $svar = 0$ en la siguiente unidad de tiempo.

5.1.3. Procesos de control

Mientras que los procesos de señalización se encargan monitorear las variables que indican la presencia o ausencia de eventos moleculares (e.g., *cambio* de estado de un gen, *superación* de un nivel de concentración, *ocurrencia* de una reacción química, etc.), los procesos de control se encargan de determinar el valor de las variables que representan entidades biológicas en el sistema (e.g., *estado* de un gen, *valor* de la concentración de una proteína, etc.). Para lograr tal control se definen dos procesos en `ntcc`: $Regulate_i$ y $Status_i$.

Por un lado, el proceso paramétrico $Regulate_i$ se define para controlar o regular procesos biológicos en los que siempre es posible determinar un valor de regulación dada una variable de señalización. La formalización en `ntcc` del proceso paramétrico $Regulate_i$ es la siguiente:

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

Dada la especificación anterior se tiene que el proceso P_i es ejecutado cuando el evento biológico marcado por la variable de señalización $svar$ ocurre. De otra forma, el proceso N_i es ejecutado. Es importante notar que un solo tipo de regulación es hecha sobre i ya que el operador "+" al escoger entre los dos tipos de regulación hace que el otro se elimine definitivamente en la unidad de tiempo en que se hace la escogencia. El proceso $Regulate_i$ es especialmente útil cuando se modelan sistemas en los que es común regulación positiva o negativa según un conjunto de condiciones.

Por otro lado, el proceso genérico $Status_i$ se define para controlar procesos en los que existe alguna de las siguientes dos características: (i) dado un conjunto finito de variables de señalización existen varias acciones válidas posibles de control o (ii) no es posible determinar tal conjunto finito de variables de señalización por lo que es necesario introducir un estado por defecto para su control. La especificación formal del proceso $Status_i$ en `ntcc` es la siguiente:

$$\begin{aligned} Status_i \stackrel{\text{def}}{=} \ !\ (\ (\ \sum_{c \in C} & \textbf{when } condition_c \textbf{ do next tell}(m_i = fc_i(m_i'))\ )\ \| \\ & \textbf{unless } \ knownConditions \ \textbf{ next } \ \textbf{tell}(m_i = m_i')\ ) \end{aligned}$$

donde C es el conjunto de índices de condiciones o restricciones conocidas para determinar un cambio en el estado de i. El nuevo valor o estado de i es definido por la función fc_i relacionada con la restricción $condition_c$ escogida por $\sum$. Si ninguna de estas condiciones puede ser inferida a partir de la información en el almacén de restricciones actual, se conserva el estado actual en la siguiente unidad de tiempo gracias al proceso **unless**. Lo anterior sugiere que la restricción $knownConditions$ es de la forma: $knownConditions = \bigvee_{c \in C} condition_c$.

5.1.4. Genes

En esta sección se propone una especificación en `ntcc` para el modelado de la entidad biológica más importante en una red de regulación genética: un gen. El modelo involucra una visión completa del comportamiento del gen según el dogma central de la biología molecular. Por lo tanto, en el modelo se describen los procesos de transcripción (i.e., producción de ARN mensajero) y traducción (i.e., síntesis de proteínas). La especificación, además, modela el nivel de expresión del gen. Su formulación se basa en el modelo propuesto para la descripción de la estructura y el funcionamiento de un gen presentado en [MM04]. El proceso en `ntcc` que representa este tipo de comportamiento es el siguiente:

$$
\begin{array}{lcl}
GenStatus_i & \stackrel{\text{def}}{=} & !\ (\ (\ \textbf{when}\ tbegin_i = 1 \wedge tend_i = 0\ \textbf{do next tell}(m_i = m_i' + 1)\ + \\
 & & \quad \textbf{when}\ tbegin_i = 0 \wedge tend_i = 1\ \textbf{do next tell}(m_i = m_i' - 1)\ )\ \parallel \\
 & & \quad \textbf{unless}\ \ tbegin_i \neq tend_i\ \ \textbf{next}\ \ \ \textbf{tell}(m_i = m_i')\ ) \\
MRNA_j(p_j, d_j) & \stackrel{\text{def}}{=} & Regulate_j(tbegin_i, \textbf{next tell}(m_j = m_j' + p_j - dt \times (d_j \times m_j')), \\
 & & \quad \textbf{next tell}(m_j = m_j' - dt \times (d_j \times m_j'))) \\
PROTEIN_k(p_k, d_k) & \stackrel{\text{def}}{=} & Regulate_k(mrnah_j, \textbf{next tell}(m_k = m_k' + dt \times (p_k \times m_j' - d_k \times m_k')), \\
 & & \quad \textbf{next tell}(m_k = m_k' - dt \times (d_k \times m_k'))) \\
Gen_{i,j,k}(p_j, d_j, p_k, d_k) & \stackrel{\text{def}}{=} & GenStatus_i \parallel\ !\ MRNA_j(p_j, d_j) \parallel\ !\ PROTEIN_k(p_k, d_k)
\end{array}
$$

El proceso $GenStatus_i$ es construido a partir de la especificación genérica de $Status_i$. Las variables de señalización $tbegin_i$ y $tend_i$ se usan para indicar que una molécula de ARN polimerasa inicia o termina la transcripción del gen. Así, la variable discreta m_i mide el *nivel de expresión del gen* representado por el número de moléculas de ARN polimerasa que se encuentran transcribiéndolo. Mientras no existan moléculas iniciando o finalizando la transcripción del gen, éste permanece con su mismo nivel de expresión, es decir, m_i conserva su valor en el tiempo. Nótese además que la condición en el proceso **unless** (i.e., $knownConditions$ en el proceso genérico) es como se indicó anteriormente (i.e., $tbegin_i \neq tend_i \equiv ((tbegin_i = 1 \wedge tend_i = 0) \vee (tbegin_i = 0 \wedge tend_i = 1))$).

El proceso $MRNA_j$ utiliza la especificación paramétrica $Regulate_i$ para su definición. Básicamente cuando la variable de señalización $tbegin_i$ se activa (i.e., $tbegin_i = 1$), m_j (midiendo la concentración de ARN) modifica su valor según el parámetro p_j (i.e., el factor de producción). En cualquier otro caso, la concentración de ARN disminuye según el parámetro de degradación natural d_j. Finalmente, el operador de replicación "!" permite que el proceso $Regulate_j$ controle la concentración de ARN en todo instante de tiempo.

Siguiendo el mismo estilo de descripción usado para $MRNA_j$, se utiliza el proceso $Regulate_k$ para describir el comportamiento dinámico de la síntesis de proteínas en el proceso de traducción. El proceso ahora es regulado por la variable de señalización $mrnah_j$, la cual indica cuando el nivel de concentración de ARN ha superado un límite y por lo tanto es posible que se inicie la traducción. Nuevamente se cuenta con dos constantes (i.e., p_k y d_k) parametrizando las expresiones (i.e., restricciones) que regulan la síntesis y degradación de la proteína que codifica el gen. Nótese que el proceso de producción o síntesis de la proteína, a diferencia del de degradación,

depende de la concentración de ARN en el sistema. Esto se evidencia en la presencia de la expresión $p_k \times m'_j$ en la restricción.

5.2. Modelo molecular del operón de la lactosa en ntcc

Esta sección muestra el uso de los procesos genéricos y paramétricos presentados anteriormente para el modelado de redes de regulación genética en la descripción del comportamiento a nivel molecular del operón de la lactosa. Tal proceso de modelado se basa en la descripción de los elementos y comportamientos observables del sistema. Tanto los unos como los otros son parametrizados por constantes que determinan un comportamiento biológico particular en ellos.

5.2.1. Modelo biológico

En este nivel de descripción (i.e., nivel molecular) el modelo biológico esta constituido por las *entidades biológicas* identificadas (cuadro 5.1), los *parámetros* que las caracterizan (cuadros 5.2, 5.3 y 5.4), y la *descripción del funcionamiento* del sistema (la cual es presentada en detalle en el capítulo 3). La integración coherente de estos tres elementos permite tener una visión completa del sistema a modelar.

Con respecto a los parámetros del sistema es importante clarificar lo siguiente. En el cuadro 5.2 se muestran los parámetros relacionados con el dogma central de la biología molecular, lo cuales caracterizan los procesos de *transcripción* y *traducción* de los genes que constituyen la red de regulación genética del operón de la lactosa. El cuadro 5.3 presenta los parámetros que determinan la velocidad de los procesos de degradación natural de algunas de las entidades biológicas identificadas. Y en el cuadro 5.4 se presentan parámetros relacionados con procesos diversos como el movimiento de moléculas, el enlace a sitios del ADN del operón, la conformación de complejos moleculares y la ocurrencia de reacciones bioquímicas, entre otras.

A continuación se presentan las variables y los parámetros usados en este modelo. Una descripción detallada de cada uno de ellos puede ser encontrada en [MM04], documento del cual fueron tomados.

5.2.2. Modelo computacional en ntcc

En el modelo en `ntcc` que se presenta en esta sección se usan la variables $m_1, ..., m_{30}$ y el proceso *Continuity* como se indica en la sección 5.1.1. De igual forma, el proceso *Signal* para la creación de todas las variables de señalización. La presentación del modelo se divide en cuatro partes por motivos de claridad. Primero, se presenta un modelo discreto de la región de control (positivo y negativo) del operón de la lactosa. Posteriormente, se expone la forma en que se especifica la estructura y el funcionamiento de los genes que componen el operón. Al final, en las dos últimas partes, son modelados los procesos de regulación por inducción y represión, la producción de alolactosa, el ingreso de lactosa al medio celular y su posterior hidrólisis en glucosa y galactosa.

Variable	Tipo	Descripción
m_1	discreta	Región promotora del operón
m_2	discreta	Región de enlace de la molécula CAP-cAMP (CAPsite)
m_3	discreta	Región operadora del operón
m_4	real	Proteína activadora de catabolito (CAP)
m_5	real	Adenosin monofosfato cíclico (cAMP)
m_6	real	Glucosa
m_7	real	Complejo represor unido a la región operadora del ADN del operón
m_8	real	Alolactosa
m_9	real	Lactosa dentro de la célula
m_{10}	real	Complejo alolactosa-represor
m_{11}	real	Adenosin monofosfato (AMP)
m_{12}	real	Adenosin difosfato (ADP)
m_{13}	discreta	Región promotora del gen represor LacI
m_{14}	real	ARN mensajero del gen represor LacI
m_{15}	real	Proteína del gen represor LacI
m_{16}	real	Complejo represor
m_{17}	real	Complejo represor unido al ADN del operón
m_{18}	real	Complejo represor libre en la célula
m_{19}	real	ARN mensajero del gen LacZ
m_{20}	real	Proteína del gen LacZ (β-galactosidasa)
m_{21}	discreta	Región del ADN del operón entre los genes LacZ y LacY
m_{22}	discreta	Estado de expresión del gen LacY
m_{23}	real	ARN mensajero del gen LacY
m_{24}	real	Proteína del gen LacY (permeasa)
m_{25}	discreta	Región del ADN del operón entre los genes LacY y LacA
m_{26}	discreta	Estado de expresión del gen LacA
m_{27}	real	ARN mensajero del gen LacA
m_{28}	real	Proteína del gen LacA (Tiogalactosido transacetilasa)
m_{29}	real	Lactosa fuera de la célula
m_{30}	real	Galactosa

Cuadro 5.1: Variables del modelo molecular del operón de la lactosa

Región de control del operón

El modelado de la región de control del operón de la lactosa se fundamenta en la especificación del comportamiento *lógico* de los procesos de control que determinan si el operón se encuentra activo (i.e., bajo inducción) o inactivo (i.e., bajo represión). Por tal motivo, se modelaran dos sitios específicos del ADN del operón de la lactosa: el sitio de enlace del complejo CAP-cAMP (donde se facilita la inducción) en el proceso $CAPsite_2$, y el sitio de enlace del complejo represor o región operadora (donde tiene lugar la represión) en el proceso $Operator_3$. En ambos casos para la construcción de los procesos $CAPsite_2$ y $Operator_3$ se usará el proceso genérico $Status_i$ como una *plantilla* para su modelado.

Se usarán cuatro variables de señalización: *capcamph*, *repressorh*, *releasecap* y *releaserep*. Las dos primeras son usadas para indicar cuando las concentraciones de los complejos sean lo suficientemente altas para permitir la inducción (i.e., $capcamp = 1$) o represión (i.e., $repressorh = 1$) del operón según sea el caso. Por otro lado, las dos últimas son usadas para indicar que los

Velocidad de transcripción	Valor	Velocidad de traducción	Valor
Gen represor LacI	1.082	Gen represor	m_{14}
Gen LacZ	3.075	Gen LacZ	m_{19}
Gen LacY	1.254	Gen LacY	$m_{23}/2$
Gen LacA	0.682	Gen LacA	$m_{27}/5$

Cuadro 5.2: Velocidades de transcripción y traducción en el operón de la lactosa

Parámetro	Velocidad de degradación natural	Valor
d_4	Proteína activadora de catabolito (CAP)	1/10
d_{14}	ARN mensajero del gen represor	1/10
d_{15}	Proteína del gen represor	1/10000
d_{17}	Complejo represor unido al ADN del operón	1/10
d_{18}	Complejo represor libre en la célula	1/10
d_7	Complejo represor unido a la región operadora	1/10
d_{19}	ARN mensajero del gen LacZ	1/10
d_{20}	Proteína del gen LacZ	1/10000
d_{23}	ARN mensajero del gen LacY	1/10
d_{24}	Proteína del gen LacY	1/10000
d_{27}	ARN mensajero del gen LacA	1/10
d_{28}	Proteína del gen LacA	1/10000
d_{29}	Lactosa fuera de la célula	1/10000
d_9	Lactosa dentro de la célula	1/10000
d_8	Alolactosa	1/2
d_{30}	Galactosa	1/10000
d_6	Glucosa	1/10
d_{10}	Complejo alolactosa-represor	1/10000
d_5	Adenosin monofosfato cíclico (cAMP)	1/10000
d_{11}	Adenosin monofosfato (AMP)	1/10000
d_{12}	Adenosin difosfato (ADP)	1/10000

Cuadro 5.3: Velocidades de degradación natural en el operón de la lactosa

complejos CAP-cAMP y represor se liberan de sus sitios de enlace, es decir, $releasecap = 1$ y $releaserep = 1$ respectivamente. La especificación en `ntcc` que formaliza lo anterior es la siguiente:

$$
\begin{aligned}
CAPsite_2 &\stackrel{\text{def}}{=} \; !\,(\,(\textbf{when}\ capcamph = 1 \wedge releasecap = 0\ \textbf{do next tell}(m_2 = m_2' + 1)\ + \\
&\qquad \textbf{when}\ capcamph = 0 \wedge releasecap = 1\ \textbf{do next tell}(m_2 = m_2' - 1))\ \parallel \\
&\qquad \textbf{unless}\ \ capcamph \neq releasecap\ \ \textbf{next}\quad \textbf{tell}(m_2 = m_2')\,) \\
Operator_3 &\stackrel{\text{def}}{=} \; !\,(\,(\textbf{when}\ repressorh = 1 \wedge releaserep = 0\ \textbf{do next tell}(m_3 = m_3' + 1)\ + \\
&\qquad \textbf{when}\ repressorh = 0 \wedge releaserep = 1\ \textbf{do next tell}(m_3 = m_3' - 1))\ \parallel \\
&\qquad \textbf{unless}\ \ repressorh \neq releaserep\ \ \textbf{next}\quad \textbf{tell}(m_3 = m_3')\,) \\
ControlRegion &\stackrel{\text{def}}{=} \; CAPsite_2 \parallel Operator_3
\end{aligned}
$$

donde $m_2 \geq 1$ indica que el complejo molecular CAP-cAMP esta presente y por tanto se ejerce un control positivo sobre el operón. Esta condición (i.e., $m_2 \geq 1$) favorece la *inducción* de la expresión del operón. Por otro lado, $m_3 \geq 1$ representa la presencia del complejo o tetrámero

Velocidad de producción o conformación de macromoléculas	**Valor**
Conformación del complejo represor	$4m_{15}/5$
Conformación del complejo alolactosa-represor	$m_8 \times m_{16}$
Producción de alolactosa a partir de la lactosa interna	$m_9/5$
Producción de alolactosa a partir de la lactosa externa	$m_{29}/10$
Producción de glucosa y galactosa a partir de la lactosa interna (vg)	$(m_{20} \times m_9)/(m_9 \times 10m_{20})$
Velocidad de movimiento molecular	**Valor**
ARN polimerasa entre LacZ y LacY	0.051
ARN polimerasa entre LacY y LacA	0.065
Lactosa a través de la membrana celular (vp)	$(m_{24} \times m_{29})/(m_{29} + 10m_{24})$
Velocidad de reacciones químicas	**Valor**
Transformación de cAMP en AMP	$m_5/10$
Transformación de AMP en cAMP	$m_{11}/10$
Transformación de AMP en ADP	$m_{11}/10$
Transformación de ADP en AMP	$m_{12}/10$
Velocidad de enlace a regiones del ADN	**Valor**
Enlace del complejo represor a la región operadora	$96m_{16}/100$
Enlace del complejo represor al ADN codificante del operón	$399m16/10000$

Cuadro 5.4: Otros parámetros del modelo molecular del operón de la lactosa

represor en el sistema. Esta condición (i.e., $m_3 \geq 1$) es entonces reflejo de que se esta efectuando un control negativo sobre el operón y por lo tanto favorece la *represión* del funcionamiento del operón. Por tal motivo, la transcripción óptima del operón se inicia realmente cuando existe inducción (i.e., $m_2 \geq 1$) y no existe represión (i.e., $m_3 = 0$). Se usará la variable de señalización $tbegin_1$ para indicar tal evento. La especificación en `ntcc` de esta condición, formalizada como un subproceso en *Signal*, es la siguiente:

$$\begin{array}{c} \textbf{when } m_2 \geq 1 \wedge m_3 = 0 \textbf{ do next tell}(tbegin_1 = 1) \\ \| \\ \textbf{unless } \ m_2 \geq 1 \wedge m_3 = 0 \ \ \textbf{next} \quad \textbf{tell}(tbegin_1 = 0) \end{array}$$

Región codificante o de genes estructurales

El modelado de los genes del operón de la lactosa (i.e., LacZ, LacY y LacA) es fácilmente descrito usando el proceso $Gen_{i,j,k}$ presentado en la sección 5.1.4. Además, en esta sección se modelan las regiones de ADN que separan cada uno de los genes.

$$\begin{array}{lcl} Promoter_1 & \stackrel{\text{def}}{=} & !\ (\ (\ \textbf{when } tbegin_1 = 1 \wedge tend_1 = 0 \textbf{ do next tell}(m_1 = m_1' + 1)\ + \\ & & \quad \textbf{when } tbegin_1 = 0 \wedge tend_1 = 1 \textbf{ do next tell}(m_1 = m_1' - 1)\ )\ \| \\ & & \quad \textbf{unless } \ tbegin_1 \neq tend_1 \ \ \textbf{next} \quad \textbf{tell}(m_1 = m_1')\ ) \\ MRNA_{19}(p_{19}, d_{19}) & \stackrel{\text{def}}{=} & Regulate_{19}(tbegin_1, \textbf{next tell}(m_{19} = m_{19}' + p_{19} - dt \times (d_{19} \times m_{19}')), \\ & & \quad \textbf{next tell}(m_{19} = m_{19}' - dt \times (d_{19} \times m_{19}'))) \\ PROTEIN_{20}(p_{20}, d_{20}) & \stackrel{\text{def}}{=} & Regulate_{20}(mrnah_{19}, \textbf{next tell}(m_{20} = m_{20}' + dt \times (p_{20} \times m_{19}' - d_{20} \times m_{20}')), \\ & & \quad \textbf{next tell}(m_{20} = m_{20}' - dt \times (d_{20} \times m_{20}'))) \\ Gen_{1,19,20}(1.0, d_{19}, 1.0, d_{20}) & \stackrel{\text{def}}{=} & Promoter_1 \ \| \ !\ MRNA_{19}(1.0, d_{19}) \ \| \ !\ PROTEIN_{20}(1.0, d_{20}) \end{array}$$

En el proceso anterior se modela la región promotora (i.e., proceso $Promoter_1$) del operón de la lactosa, el comportamiento dinámico del ARN mensajero transcrito a partir del gen LacZ (i.e., proceso $MRNA_{19}$) y la síntesis de la β-galactosidasa (i.e., proceso $PROTEIN_{20}$). Las variables de señalización $tbegin_1$, $tend_1$ y $mrnah_{19}$ tienen los mismos significados dados en la sección 5.1.4. En particular, para la definición de la variable $tend_1$ (al igual que como se hace para las variables $tend_{22}$ y $tend_{26}$ presentadas más adelante) se usan los datos de la velocidad de transcripción de cada gen mostradas en el cuadro 5.2.

Los procesos para la descripción de los genes LacY y LacA se construyen de forma análoga al proceso anterior, por lo que sólo se presentan los procesos paramétricos $Gen_{22,23,24}$ y $Gen_{26,27,28}$ que los describen. De esta forma, el modelo en `ntcc` de los tres genes que componen el operón de la lactosa es el siguiente:

$$
\begin{array}{lcl}
Gen_{1,19,20}(1.0, d_{19}, 1.0, d_{20}) & \stackrel{\text{def}}{=} & Promoter_1 \parallel\ !\, MRNA_{19}(1.0, d_{19}) \parallel\ !\, PROTEIN_{20}(1.0, d_{20}) \\
Gen_{22,23,24}(1.0, d_{23}, 0.5, d_{24}) & \stackrel{\text{def}}{=} & GenStatus_{22} \parallel\ !\, MRNA_{23}(1.0, d_{23}) \parallel\ !\, PROTEIN_{24}(0.5, d_{24}) \\
Gen_{26,27,28}(1.0, d_{27}, 0.2, d_{28}) & \stackrel{\text{def}}{=} & GenStatus_{26} \parallel\ !\, MRNA_{27}(1.0, d_{27}) \parallel\ !\, PROTEIN_{28}(0.2, d_{28})
\end{array}
$$

donde los procesos $PROTEIN_{24}$ y $PROTEIN_{28}$ modelan la permeasa y la tiogalactosido transacetilasa respectivamente.

Haciendo uso nuevamente del proceso genérico $Status_i$ se modelan las regiones de ADN entre los genes del operón de la lactosa. Estos modelos dan cuenta del número de moléculas de ARN polimerasa que se encuentran moviéndose entre cada gen. Aquí se usan los parámetros mostrados en el cuadro 5.4 para definir las variables de señalización usadas en el modelado de estas regiones. Los procesos en `ntcc` son los siguientes:

$$
\begin{array}{lcl}
DelayZY_{21} & \stackrel{\text{def}}{=} & !\,(\,(\ \textbf{when}\ tbegin_{21} = 1 \wedge tend_{21} = 0\ \textbf{do next tell}(m_{21} = m'_{21} + 1)\ + \\
 & & \quad\ \textbf{when}\ tbegin_{21} = 0 \wedge tend_{21} = 1\ \textbf{do next tell}(m_{21} = m'_{21} - 1)\ )\ \parallel \\
 & & \quad\ \textbf{unless}\ \ tbegin_{21} \neq tend_{21}\ \ \textbf{next}\ \ \ \textbf{tell}(m_{21} = m'_{21})\ ) \\
DelayYA_{25} & \stackrel{\text{def}}{=} & !\,(\,(\ \textbf{when}\ tbegin_{25} = 1 \wedge tend_{25} = 0\ \textbf{do next tell}(m_{25} = m'_{25} + 1)\ + \\
 & & \quad\ \textbf{when}\ tbegin_{25} = 0 \wedge tend_{25} = 1\ \textbf{do next tell}(m_{25} = m'_{25} - 1)\ )\ \parallel \\
 & & \quad\ \textbf{unless}\ \ tbegin_{25} \neq tend_{25}\ \ \textbf{next}\ \ \ \textbf{tell}(m_{25} = m'_{25})\ )
\end{array}
$$

Finalmente, los modelos anteriores se integran en el proceso $StructuralGenes$ haciendo uso del operador de composición paralela:

$$
\begin{array}{lcl}
StructuralGenes & \stackrel{\text{def}}{=} & Gen_{1,19,20}(1.0, d_{19}, 1.0, d_{20}) \parallel DelayZY_{21} \parallel \\
 & & Gen_{22,23,24}(1.0, d_{23}, 0.5, d_{24}) \parallel DelayZY_{25} \parallel \\
 & & Gen_{26,27,28}(1.0, d_{27}, 0.2, d_{28})
\end{array}
$$

Inducción y represión

Los procesos de inducción y represión, tal como se explicó en el capítulo 3, son los que determinan que el operón de la lactosa se encuentre activo o inactivo. La especificación de estos procesos biológicos hace uso frecuente del proceso paramétrico $Regulate_i$. A continuación se presentan los modelos en `ntcc` para la descripción del proceso de inducción debido al comportamiento del cAMP, el AMP, el ADP y el CAP.

$$
\begin{array}{lcl}
CAMP & \stackrel{\text{def}}{=} & Regulate_5(glucl, \\
 & & \qquad \textbf{next tell}(m_5 = m'_5 + dt \times (0.1m'_{11} - 0.1001m'_5)), \\
 & & \qquad \textbf{next tell}(m_5 = m'_5 - dt \times 0.1001m'_5)) \\
AMP & \stackrel{\text{def}}{=} & Regulate_{11}(glucl, \\
 & & \qquad \textbf{next tell}(m_{11} = m'_{11} + dt \times (0.1m'_5 + 0.1m'_{12} - 0.2001m'_{11})), \\
 & & \qquad \textbf{next tell}(m_{11} = m'_{11} + dt \times (0.1m'_5 + 0.1m'_{12} - 0.1001m'_{11}))) \\
ADP & \stackrel{\text{def}}{=} & !\ \textbf{next tell}(m_{12} = m'_{12} + dt \times (0.1m'_{11} - 0.2m'_{12})) \\
CAP & \stackrel{\text{def}}{=} & !\ \textbf{next tell}(m_4 = m'_4 + dt \times (1.0 - 0.1m'_4)) \\
Induction & \stackrel{\text{def}}{=} & !\ CAMP \parallel\ !\ AMP \parallel ADP \parallel CAP
\end{array}
$$

La variable de señalización *glucl* indica cuando el nivel de concentración celular de glucosa es lo suficientemente *bajo* para que se "dispare" un incremento en la producción de cAMP al igual que en la formación del complejo CAP-cAMP. Nótese que los procesos ADP y CAP no usan el proceso paramétrico $Regulate_i$ ya que ellos no presentan regulación.

De forma similar a como se procedió para la descripción del proceso de inducción, se modelan los procesos relacionados con la represión. El proceso *Repression* presentado más adelante modela el gen represor LacI, la formación del complejo represor y la forma en que este se comporta dentro de la célula, i.e., mediante unión a la región operadora, a la región codificante del operón o permaneciendo libre dentro de la célula hasta su degradación. Además, se modela la formación del complejo alolactosa-represor.

Para la descripción del gen represor LacI se procede de igual forma que con aquellos modelados anteriormente. Por otro lado, la formación del complejo represor (modelado en el proceso *Repressor*) así como su comportamiento (modelado con los procesos $DNABinding$, $NotBinding$ y $OperatorBinding$) es determinada por la concentración de alolactosa dentro de la célula. El comportamiento de este sistema es el siguiente: si se *supera* cierto nivel de concentración celular de alolactosa, indicado por la variable de señalización *allolach*, se inicia la formación del complejo alolactosa-represor el cual favorece la activación del operón de la lactosa. Sin embargo, si tal nivel de concentración no se supera (i.e., $allolach = 0$), el complejo represor cumple la función para la cual es sintetizado. El modelo en `ntcc` para la descripción de la represión del operón de la lactosa es el siguiente:

$$
\begin{array}{lcl}
GenI_{13,14,15}(1.0, d_{14}, 1.0, d_{15}) & \stackrel{\text{def}}{=} & GenStatus_{13} \parallel\ !\ MRNA_{14}(1.0, d_{14}) \parallel\ !\ PROTEIN_{15}(1.0, d_{15}) \\
Repressor & \stackrel{\text{def}}{=} & Regulate_{16}(allolach, \\
 & & \qquad \textbf{next tell}(m_{16} = m'_{16} + dt \times (0.2m'_{15} - m'_8 \times m'_{16})), \\
 & & \qquad \textbf{next tell}(m_{16} = m'_{16} + dt \times (0.2m'_{15} - m'_{16})) \\
DNABinding & \stackrel{\text{def}}{=} & Regulate_{17}(allolach, \\
 & & \qquad \textbf{next tell}(m_{17} = m'_{17} - dt \times 0.1m'_{17}), \\
 & & \qquad \textbf{next tell}(m_{17} = m'_{17} + dt \times (0.0399m'_{16} - 0.1m'_{17})) \\
NotBinding & \stackrel{\text{def}}{=} & Regulate_{18}(allolach, \\
 & & \qquad \textbf{next tell}(m_{18} = m'_{18} - dt \times 0.1m'_{18}), \\
 & & \qquad \textbf{next tell}(m_{18} = m'_{18} + dt \times (0.001m'_{16} - 0.1m'_{18})) \\
OperatorBinding & \stackrel{\text{def}}{=} & Regulate_{7}(allolach, \\
 & & \qquad \textbf{next tell}(m_7 = m'_7 - dt \times 0.1m'_7), \\
 & & \qquad \textbf{next tell}(m_7 = m'_7 + dt \times (0.96m'_{16} - 0.1m'_7)) \\
Binding & \stackrel{\text{def}}{=} & DNABinding \parallel NotBinding \parallel OperatorBinding \\
ComplexAR & \stackrel{\text{def}}{=} & Regulate_{10}(allolach, \\
 & & \qquad \textbf{next tell}(m_{10} = m'_{10} + dt \times ((m_8 \times m_{16}) - 0.0001m'_{10})), \\
 & & \qquad \textbf{next tell}(m_{10} = m'_{10} - dt \times 0.0001m'_{10})) \\
Repression & \stackrel{\text{def}}{=} & GenI_{13,14,15}(1.0, d_{14}, 1.0, d_{15}) \parallel \\
 & & !\ Repressor \parallel\ !\ Binding \parallel\ !\ ComplexAR
\end{array}
$$

Degradación de la lactosa

Generar fuentes de energía a partir de la lactosa en el medio de la célula representa el fin verdadero de esta red de regulación genética. Los principales procesos biológicos aquí modelados son: el paso de la lactosa externa a través de la membrana hacia el interior de la célula, la división o hidrólisis de la lactosa, y la producción de alolactosa inducida por la presencia de la lactosa.

El primer proceso biológico es *regulado* por la variable de señalización *permh* y por el parámetro *vp* (ver cuadro 5.4). Por un lado, *permh* indica cuando la permeasa ha superado un límite mínimo para facilitar que la lactosa externa atraviese la membrana celular. Por otro, *vp* permite calcular el grado de regulación ejercido por la permeasa sobre la membrana celular. Por lo tanto, *permh* afecta las concentraciones de lactosa dentro de la célula (modelada en el proceso $LacIn$) y fuera de ella (modelada en el proceso $LacOut$).

El segundo proceso biológico (i.e., la hidrólisis de lactosa en glucosa y galactosa) es modelado haciendo uso de la variable de señalización *bgalh* y del parámetro *vg*. La variable de señalización *bgalh* indica cuando la concentración celular de β-galactosidasa es suficientemente alta para iniciar la hidrólisis de lactosa interna. Por otro lado, el grado de regulación del proceso de hidrólisis es calculado con el parámetro *vg*. De esta forma, tanto *bgalh* como *vp* afectan los procesos en `ntcc` que modelan la lactosa dentro de la célula (i.e., proceso $LacIn$), la glucosa (i.e., proceso *Glucose*) y la galactosa (i.e., proceso *Galactose*).

Finalmente, la concentración de alolactosa es *regulada* por las variables de señalización *lacinh*

y *allolach*. La primera indica cuando la concentración de lactosa dentro de la célula es lo suficientemente alta para inducir la producción de alolactosa y la segunda "señala" el mismo evento molecular indicado en el proceso de represión (i.e., formación del complejo alolactosa-represor).

Los procesos en `ntcc` que formalizan los tres procesos biológicos explicados son los siguientes:

$$
\begin{array}{lcl}
LacOut & \stackrel{\text{def}}{=} & Regulate_{29}(permh, \\
 & & \quad \textbf{next tell}(m_{29} = m'_{29} - dt \times (vp + d_{29}m'_{29})), \\
 & & \quad \textbf{next tell}(m_{29} = m'_{29} - dt \times (d_{29}m'_{29}))) \\
LacIn & \stackrel{\text{def}}{=} & Regulate_{9}(bgalh, \\
 & & \quad Regulate_{9}(permh, \\
 & & \quad \quad \textbf{next tell}(m_{9} = m'_{9} + dt \times (vp - vg - d_{9}m'_{9})), \\
 & & \quad \quad \textbf{next tell}(m_{9} = m'_{9} - dt \times (vg + d_{9}m'_{9}))), \\
 & & \quad Regulate_{9}(permh, \\
 & & \quad \quad \textbf{next tell}(m_{9} = m'_{9} + dt \times (vp - d_{9}m'_{9})), \\
 & & \quad \quad \textbf{next tell}(m_{9} = m'_{9} - dt \times (d_{9}m'_{9})))) \\
Glucose & \stackrel{\text{def}}{=} & Regulate_{6}(bgalh, \\
 & & \quad \textbf{next tell}(m_{6} = m'_{6} + dt \times (vg - d_{6}m'_{6})), \\
 & & \quad \textbf{next tell}(m_{6} = m'_{6} - dt \times (d_{6}m'_{6}))) \\
Galactose & \stackrel{\text{def}}{=} & Regulate_{30}(bgalh, \\
 & & \quad \textbf{next tell}(m_{30} = m'_{30} + dt \times (vg - d_{30}m'_{30})), \\
 & & \quad \textbf{next tell}(m_{30} = m'_{30} - dt \times (d_{30}m'_{30}))) \\
Allolactose & \stackrel{\text{def}}{=} & Regulate_{8}(allolach, \\
 & & \quad Regulate_{8}(lacinh, \\
 & & \quad \quad \textbf{next tell}(m_{8} = m'_{8} + dt \times ((d_{29}m'_{29} + 0.2m'_{9}) - (d_{8}m'_{8} + m_{8} \times m'_{16}))), \\
 & & \quad \quad \textbf{next tell}(m_{8} = m'_{8} + dt \times (d_{29}m'_{29} - (d_{8}m'_{8} + m'_{8} \times m'_{16})))), \\
 & & \quad Regulate_{8}(lacinh, \\
 & & \quad \quad \textbf{next tell}(m_{8} = m'_{8} + dt \times ((d_{29}m'_{29} + 0.2m'_{9}) - d_{8}m'_{8})), \\
 & & \quad \quad \textbf{next tell}(m_{8} = m'_{8} + dt \times (d_{29}m'_{29} - d_{8}m'_{8})))) \\
Hydrolysis & \stackrel{\text{def}}{=} & !\ LacOut \parallel\ !\ LacIn \parallel\ !\ Glucose \parallel\ !\ Galactose \parallel\ !\ Allolactose
\end{array}
$$

Integración de procesos

Todos los procesos presentados en esta sección se integran en GRN de la siguiente forma:

$$
\begin{array}{lcl}
GRN & \stackrel{\text{def}}{=} & \textbf{local}\ ts, svar_1, ..., svar_k, m_1, ..., m_{30}, m'_1, ..., m'_{30}\ \textbf{in} \\
 & & Continuity \parallel Signal \parallel \\
 & & ControlRegion \parallel StructuralGenes \parallel \\
 & & Induction \parallel Repression \parallel \\
 & & Hydrolysis
\end{array}
$$

donde $svar_1, ..., svar_k$ son todas las variables de señalización usadas en este modelo.

Procesos de Mutación

El proceso GRN puede extenderse para modelar mutaciones en la red de regulación, por ejemplo $LacI^{-}$ (ver el capítulo 3 para mayores detalles sobre los procesos de mutación). Esta mutación hace que el gen LacI produzca un tetrámero represor que no reconoce la región operadora, y por tanto siempre que exista la cantidad suficiente de moléculas activadoras se expresara el operón. Este proceso de mutación puede ocurrir en *cualquier instante de tiempo* haciendo que no se puedan unir más tetrámeros represores al operador. Obviamente, también es posible que tal mutación nunca suceda. Lo anterior se modela de la siguiente forma:

$$\begin{aligned} Mut \stackrel{\text{def}}{=} \; & \mathbf{skip} \\ & + \\ & \star\,(!\mathbf{tell}(mut = 1) \parallel \; !\mathbf{next}\,\mathbf{tell}(m_7 = m_7' - dt \times 0.1m_7')) \end{aligned}$$

donde el operador "+" escoge de forma *no determinista* si ocurre o no la mutación en el gen LacI. Mientras el proceso **skip** modela la posibilidad de que la mutación nunca ocurra, la restricción $mut = 1$ representa el hecho de que el gen esta codificando incorrectamente el tetrámero represor. Si la mutación ocurre, en un tiempo indeterminado, el gen seguirá funcionando de forma incorrecta por *siempre*. Adicionalmente se formaliza el hecho de que a partir del momento en que ocurre la mutación, la variable m_7 sólo decrece por el proceso de degradación natural. Como este evento (i.e., la mutación) afecta directamente la unión o enlace del complejo represor a la región operadora es necesario modificar el proceso $OperatorBinding$ de la siguiente forma:

$$\begin{aligned} OperatorBinding \stackrel{\text{def}}{=} \; & Regulate_7(allolach, \\ & \quad \mathbf{unless} \;\; mut = 1 \;\; \mathbf{next} \;\; \mathbf{tell}(m_7 = m_7' - dt \times 0.1m_7'), \\ & \quad \mathbf{unless} \;\; mut = 1 \;\; \mathbf{next} \;\; \mathbf{tell}(m_7 = m_7' + dt \times (0.96m_{16}' - 0.1m_7')) \end{aligned}$$

Integrando los nuevos procesos (i.e., Mut y $OperatorBinding$) a GRN se obtiene el siguiente sistema:

$$\begin{aligned} GRN \stackrel{\text{def}}{=} \; & \mathbf{local}\; ts, svar_1, ..., svar_k, m_1, ..., m_{30}, m_1', ..., m_{30}', mut \;\mathbf{in} \\ & Continuity \parallel Signal \parallel ControlRegion \parallel StructuralGenes \parallel \\ & Induction \parallel Repression^{*} \parallel Hydrolysis \parallel Mut \end{aligned}$$

donde $Repression^{*}$ incluye el proceso $OperatorBinding$ modificado. El comportamiento de este nuevo sistema no puede ser determinado sólo conociendo su especificación y condiciones iniciales, ya que una vez dada la mutación el operador "$\star$" induce un *no determinismo temporal* o *asincronía* en el sistema. Por otro lado, nunca es posible saber a priori si tal mutación tendrá lugar dado que la escogencia con "+" es *no determinista*. Los resultados de la simulación del proceso GRN (con y sin mutación) se presentan en la siguiente sección.

5.3. Implementación en Mozart-Oz y simulación en ntccSim

La implementación en Mozart-Oz del modelo en `ntcc` presentado en este capítulo es simulada usando **ntccSim**. El lector puede referirse al capítulo 2 para detalles sobre la forma de implementar procesos `ntcc` con la herramienta de simulación.

Al igual que en el capítulo anterior, a partir de la simulación del proceso GRN se obtienen un conjunto de gráficas que sirven para observar el comportamiento de los elementos modelados. Este comportamiento, a diferencia del capítulo anterior, es en ocasiones de tipo discreto por las características moleculares de estos sistemas mencionadas al inicio del capítulo. Además existe comportamiento *no determinista* inducido por la mutación en el sistema. Los valores iniciales (distintos de cero) de las variables usadas en la simulación del modelo se presentan en el cuadro 5.5 (estos valores fueron tomados de [MM04]).

Nombre	**Variable**	**Valor inicial**
Proteína activadora de catabolito (CAP)	m_4	5.0
Adenosin monofosfato cíclico (cAMP)	m_5	100.0
Adenosin monofosfato (AMP)	m_{11}	200.0
Adenosin difosfato (ADP)	m_{12}	200.0
β-galactosidasa	m_{20}	5.0
Permeasa	m_{24}	2.5
Tiogalactosido transacetilasa	m_{28}	1.0
Glucosa	m_6	50.0
Lactosa externa	m_{29}	50.0

Cuadro 5.5: Valores iniciales de las variables del modelo molecular del operón de la lactosa

En la figura 5.1, se presentan las gráficas que muestran el comportamiento del gen LacZ, la lactosa, la glucosa y la región de control del operón (con y sin mutación). La gráfica 5.1(a) puede ser analizada a la luz del dogma central de la biología molecular. En ella se nota como la producción de ARN mensajero (gracias al proceso de *transcripción*) permite el incremento de β-galactosidasa dentro de la célula (producto del proceso de *traducción*). La gráfica 5.1(b) muestra como la lactosa externa al atravesar la membrana celular incrementa la concentración de lactosa interna y a su vez como esta última se consume para producir glucosa. La gráfica 5.1(c) permite observar como después de que se consume prácticamente la totalidad de la glucosa, su concentración vuelve a incrementarse gracias al proceso de hidrólisis de la lactosa. Finalmente, en las gráficas 5.1(d) y 5.1(e) se aprecian los procesos de regulación positiva y negativa sobre la región de control del operón en condiciones normales y mutadas respectivamente. Como se puede apreciar, el comportamiento mostrado en estas gráficas se ajusta a la descripción funcional del sistema, así como a lo descrito en la literatura (e.g., ver [MM04]).

5.4. Verificación de propiedades en ntcc

En esta sección se presenta una forma de verificar propiedades biológicas usando el sistema de inferencia asociado con `ntcc`. Este enfoque o acercamiento basado en la *lógica* permite hacer

consultas sobre el modelo para verificar condiciones sobre el comportamiento de los sistemas modelados. La lógica, tal como se indico en el capítulo 2, es una lógica temporal lineal (LTL).

Como ejemplo del uso del sistema de inferencia para probar o verificar propiedades, se presenta una prueba de *estabilidad* sobre la proteína activadora de catabolito (CAP). La consulta es entonces, el valor de estabilidad de la concentración de esta proteína.

Primero, por motivos de claridad, se reescribe la definición en `ntcc` del proceso CAP:

$$CAP \stackrel{\text{def}}{=} \ !\,\mathbf{next\,tell}(m_4 = m_4' + dt \times (1.0 - 0.1m_4')) \ \equiv \ !\,\mathbf{next\,tell}(m_4 = m_4' + \Delta m_4)$$

La fórmula en la lógica LTL para el proceso CAP es: $CAP \vdash \Box\circ\ (\ m_4 = m_4' + \Delta m_4\ )$. De la definición de CAP se tiene que cuando CAP es estable, la sentencia $\Delta m_4 = 0.0$ tiene que ser cierta. Así, la condición para la estabilidad en CAP puede ser expresada con la siguiente definición en `ntcc`:

$$StableProperty \stackrel{\text{def}}{=} \ \star_n\mathbf{tell}(\Delta m_4 = 0.0)$$

donde n es un tiempo lo suficientemente grande para que CAP alcance la estabilidad. Nótese como aquí se hace uso del poder de expresividad de `ntcc` al permitir un modelo formal sobre un evento que se sabe va a suceder en un futuro, pero se desconoce el momento *exacto* de su ocurrencia. Esto es de alguna forma *información parcial* en el modelo, pero no sobre los valores de las variables sino sobre el *comportamiento temporal* del sistema.

La fórmula para el proceso $StableProperty$ es la siguiente: $StableProperty \vdash \Diamond\Delta m_4 = 0.0$. Por lo tanto, la siguiente aserción representa un estado de estabilidad para la proteína activadora de catabolito:

$$CAP \parallel StableProperty \vdash \Diamond\Box m_4 = Vs$$

donde Vs es el valor de estabilidad de CAP. Este valor es *consultado* de forma precisa mediante el uso del sistema de inferencia asociado con `ntcc`. La prueba es formalmente desarrollada como se muestra a continuación:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{CAP \vdash \Box\circ\ (\ m_4 = m_4' + \Delta m_4\ )} \quad \dfrac{\dfrac{\dfrac{}{StableProperty \vdash \Diamond\Delta m_4 = 0.0}}{StableProperty \vdash \Diamond\ (\ \Delta m_4 = 0.0 \,\dot\wedge\, dt \times (1.0 - 0.1m_4') = 0.0\ )}\ \text{LCONS}}{StableProperty \vdash \Diamond\ (\ \Delta m_4 = 0.0 \,\dot\wedge\, m_4' = 10.0\ )}\ \text{LCONS}}{CAP \parallel StableProperty \vdash\ (\ \Box\circ\ (\ m_4 = m_4' + \Delta m_4\ )\ )\ \dot\wedge\ (\ \Diamond\ (\ \Delta m_4 = 0.0 \,\dot\wedge\, m_4' = 10.0\ )\ )}\ \text{LPAR}}{CAP \parallel StableProperty \vdash\ (\ \Box\circ\ (\ m_4 = m_4' + \Delta m_4\ )\ )\ \dot\wedge\ (\ \Diamond m_4 = 10.0\ )\ )}\ \text{LCONS}}{CAP \parallel StableProperty \vdash \Diamond\ (\ \Box\circ\ (\ m_4 = m_4' + \Delta m_4\ )\ \dot\wedge\ (\ m_4 = 10.0\ )\ )}\ \text{LCONS}$$

Dado que el valor de m_4 en el tiempo t es igual al valor de m'_4 en la siguiente unidad de tiempo, es posible hacer la siguiente deducción:

$$\dfrac{\dfrac{\dfrac{}{CAP \parallel StableProperty \vdash \diamondsuit\ (\ \Box\circ\ (\ m_4 = m'_4 + \Delta m_4\ )\ \dot{\wedge}\ (\ m_4 = 10.0\ )\ )}}{CAP \parallel StableProperty \vdash \diamondsuit\ (\ (\ \Box\circ\ (\ m_4 = m'_4 + \Delta m_4\ )\ )\ \dot{\wedge}\ (\ m_4 = 10.0\ )\ \dot{\wedge}\ (\ \circ\ (\ m'_4 = 10.0\ )\ )\ )}\ \text{LCONS}}{CAP \parallel StableProperty \vdash \diamondsuit\Box m_4 = 10.0}\ \text{LCONS}$$

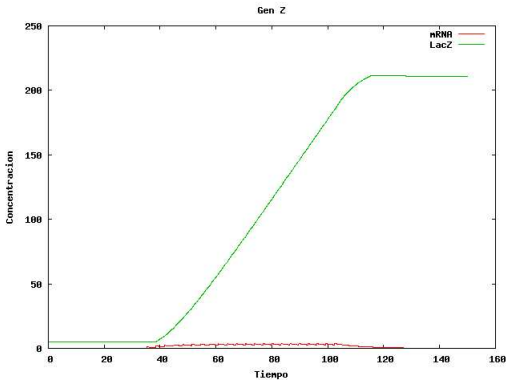
La anterior expresión lógica prueba estabilidad en la proteína CAP cuando $m_4 = Vs = 10.0$ en un tiempo *indeterminado* en el futuro. Este valor, por pertenecer a un estado estable de CAP, se mantiene *siempre* en el futuro. Finalmente, por el *Lema* 1.3 presentado en el capítulo 2, se puede asegurar que la prueba es válida tomando en cuenta el resto del sistema.

Pese a que el resultado de la prueba formal usando la lógica LTL corresponde al comportamiento obtenido a partir del simulador (ver Figura 5.2), estos dos resultados presentan una diferencia significativa: la prueba formal *asegura* que en algún momento en el futuro se alcanza la estabilidad de CAP, mientras que la simulación sólo puede garantizar un comportamiento hasta el *tiempo máximo* de simulación. Esto implica que es *imposible* garantizar una propiedad de este tipo (i.e., estabilidad) sólo con los resultados de una simulación, dado que ésta debería hacerse por siempre.

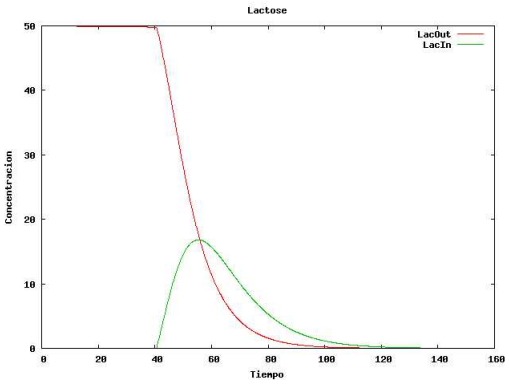
5.5. Resumen

En este capítulo se presentó una forma genérica de modelar, simular y verificar propiedades sobre redes de regulación genética. Un modelo a nivel molecular del operón de la lactosa fue usado como caso de estudio. En la parte de modelado se presentaron un conjunto de procesos genéricos y paramétricos en `ntcc` que permiten describir el comportamiento discreto y continuo presente en este tipo de sistemas. Una característica de la especificación en `ntcc` obtenida es que puede ser extendida mediante el uso del operador de composición paralela para adicionar más información sobre el comportamiento del sistema. De esta forma, es posible el modelado de procesos degenerativos en el sistema (e.g., una mutación). Tal modelo fue posteriormente simulado.

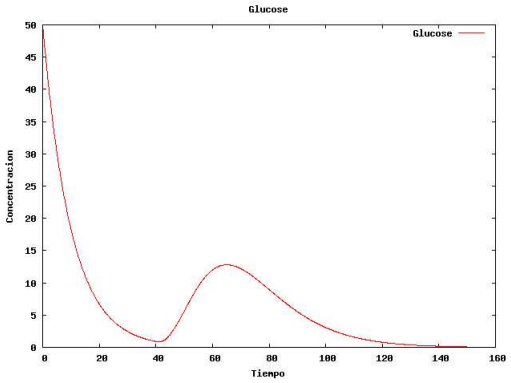
Después se presentó una prueba de estabilidad usando el sistema de pruebas asociado con `ntcc`. De esta forma, se utilizo un enfoque basado en una lógica temporal lineal para la verificación de propiedades sobre el sistema. Los resultados de la prueba fueron comparados contra aquellos obtenidos con la simulación del sistema. Lo más importante de esta prueba es que con ella se obtiene formalmente un resultado *imposible* de obtener por medios de simulación: el *aseguramiento* de una propiedad del sistema (i.e., estabilidad), ya que con la simulación sólo se tiene conocimiento de su comportamiento hasta el tiempo máximo de simulación.



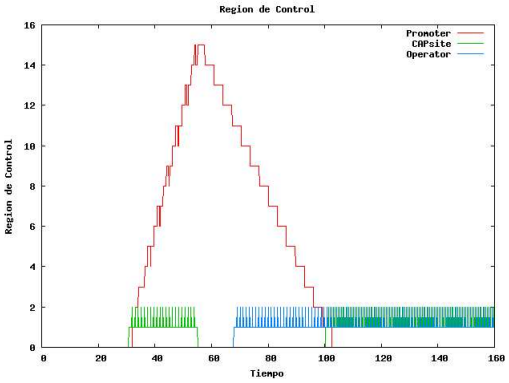
(a) Gen LacZ



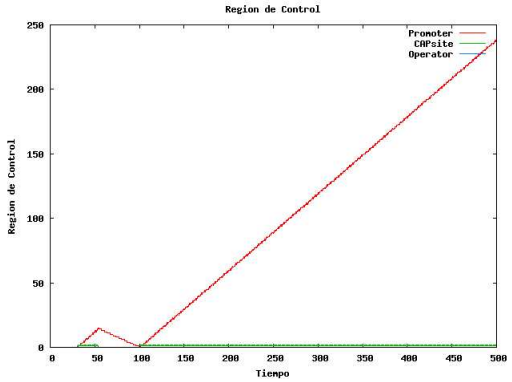
(b) Lactosa dentro y fuera de la célula



(c) Glucosa



(d) Región de control



(e) Región de control en presencia de la mutación $LacI^{-}$

Figura 5.1: Comportamiento dinámico del operón de la lactosa a nivel molecular

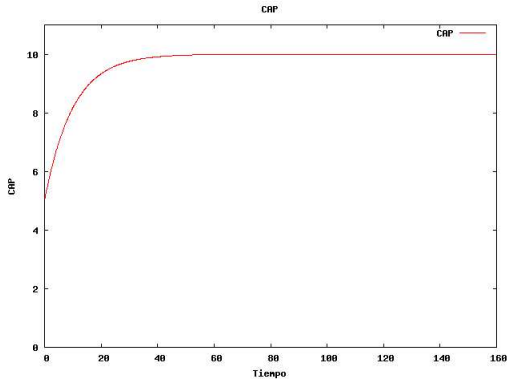


Figura 5.2: Comportamiento temporal de la proteína activadora de catabolito (CAP)

6 Análisis Biológico y Computacional

Este capítulo está dedicado a resaltar los aspectos más relevantes relacionados con el uso del modelo CCP para el estudio de sistemas biológicos. Inicialmente se resumen algunas ideas que reflejan las características más importantes del uso de cálculos de procesos en biología. Posteriormente se describen las conclusiones de este trabajo de grado, presentadas como ventajas y desventajas identificadas en el uso de `ntcc` para el modelado, simulación y verificación de propiedades en sistemas biológicos. Al final del capítulo se proponen unas actividades como trabajo futuro. Tales actividades están planteadas de tal forma que algunas de ellas intentan reforzar las ventajas del uso de CCP, mientras que otras pretenden superar, tanto desde lo teórico como desde lo práctico, las desventajas o limitaciones identificadas.

6.1. Modelado y verificación de sistemas biológicos usando cálculos de procesos

En el contexto biológico los *cálculos de procesos* se han perfilado como herramientas de especificación formal que permiten establecer *abstracciones* entre los elementos reales de los sistemas biológicos y los operadores básicos de los cálculos. Esto significa que el modelado de sistemas biológicos usando cálculos de procesos, se concentra en ciertos aspectos esenciales que determinan la estructura y comportamiento de los sistemas modelados, mientras que otras características menos importantes de acuerdo a algún criterio particular no son consideradas. De esta forma, la omisión de elementos no determinantes en el comportamiento de un sistema permite proponer especificaciones genéricas, válidas en un conjunto amplio de escenarios biológicos.

Sin embargo, pese a las potencialidades en el modelado de sistemas, lo más importante de los cálculos de procesos es que, gracias a su fundamentación matemática, es posible *verificar propiedades* de los sistemas que ellos modelan. Por ejemplo, en el caso de `ntcc` tales propiedades pueden ser probadas siguiendo un enfoque de verificación basado en la *lógica*, mediante el uso de su sistema de inferencia asociado. Estos procesos de chequeo del modelo, además de servir para probar la correctitud de las especificaciones, también permiten obtener resultados que pueden ser muy complicados o incluso imposibles de conseguir por medios prácticos.

6.2. Tiempo en el modelado de sistemas dinámicos complejos

En algunos escenarios biológicos, así como en muchos sistemas de control, lo más importante es conocer los estados inicial y final del sistema. Sin embargo, en otras situaciones se requiere poder observar y controlar la evolución del sistema a través del tiempo. Este es el caso cuando se necesita modelar y simular sistemas biológicos dinámicos como las redes de regulación genética. En ese sentido, tener un lenguaje de descripción en el que exista una noción explícita del tiempo es fundamental para (i) conseguir el control de los procesos modelados y (ii) supervisar u observar la evolución del sistema paso a paso. El cálculo de procesos `ntcc` posee operadores con los que es posible manipular de forma explícita los cambios temporales de los sistemas modelados. Dado que tales operadores están definidos sobre tiempo discreto, es necesario el uso de *encodings* que faciliten la representación de cambios continuos.

6.3. Información parcial

La *información parcial* aparece de forma natural en el modelado de sistemas biológicos, ya que la estructura y comportamiento de muchos de ellos aún es materia de estudio. Es posible distinguir dos clases principales de información parcial en estos sistemas: *cuantitativa* y *comportamental*.

Por un lado, la *información parcial cuantitativa* en un sistema biológico usualmente refleja la incertidumbre relacionada con el estado de las variables que lo representan. Este tipo de información parcial ha sido la más comúnmente usada en el contexto de los CSPs, donde las *restricciones* son el elemento formal usado para limitar los posibles valores de las variables (e.g., la concentración de un proteína es menor que x). Por otro lado, la *información parcial comportamental* está relacionada con la incertidumbre asociada al comportamiento de las interacciones presentes entre las entidades que componen un sistema biológico. En `ntcc` este tipo de información parcial puede ser descrita usando los operadores de *asincronía y no determinismo*. De esta forma, es posible modelar eventos que se sabe que ocurrirán en un tiempo futuro, pero se desconoce el momento exacto en que van a suceder (e.g., eventualmente ocurrirá una mutación en un gen).

El uso apropiado de operadores de los cálculos para la descripción de estos dos tipos de información parcial puede ayudar a la construcción de modelos en los que se representen, con cierto nivel de abstracción, patrones de comportamiento biológico bastante complejos.

6.4. Consideraciones prácticas

En esta sección se presentan dos ejemplos concretos en los que los resultados en la práctica reflejan dos inconvenientes cuando se propone una especificación en `ntcc`: por un lado, la imposibilidad de representar en la máquina cualquier número real de forma exacta, haciendo que en ciertas simulaciones los resultados esperados se alejen demasiado de los obtenidos. Por otro lado, la posibilidad de generar un número muy grande de agentes computacionales (i.e., procesos) en

una unidad de tiempo, produciendo que el sistema eventualmente colapse. En ambos casos, los problemas se presentan al realizar simulaciones por *mucho tiempo*. A continuación se detalla el problema expuesto en cada uno de estos dos ejemplos.

Ejemplo 6.1 *Resultados prácticos muy alejados de la teoría.*

Las variables en el modelo CCP están definidas con un intervalo de los posibles valores que ellas pueden tomar (i.e., dominios representando información parcial cuantitativa). Cuando se imponen restricciones sobre una variable, su dominio puede reducirse parcial (i.e., un dominio con menos valores que el anterior) o totalmente (i.e., el dominio de la variable contiene un solo valor). Si tal dominio no se reduce totalmente puede ser por una de dos razones: porque no es posible determinar un solo valor para la variable según el conjunto de restricciones que tiene asociadas, o porque no se puede representar en la máquina el único valor en el dominio de la variable (e.g., algunos números reales).

Este último problema, fruto de una limitante de la máquina, es solucionado (parcialmente) asociando a la variable el mínimo dominio que contenga el (único) valor que debería tener. Sin embargo, en algunas ocasiones, esta solución no es completamente adecuada. Por ejemplo, suponga la siguiente especificación en `ntcc`*:*

$$Num(x) \stackrel{\text{def}}{=} \mathbf{tell}(r = (x \times x) \div x) \parallel \mathbf{next}\, Num(r)$$

Sin importar el valor con que se llame $Num(x)$ el valor de r siempre debe ser el valor del parámetro x. Sin embargo, los siguientes dos llamados producen resultados totalmente diferentes con el paso del tiempo: $Num(1.0)$ y $Num(1.1)$. Mientras el primero hace que r sea siempre igual a 1.0, el segundo presenta resultados sumamente alejados de lo esperado teóricamente. Esto se debe a que el número 1.1 no puede ser representado de forma exacta *en la máquina mientras que el número 1.0 sí. Por tal motivo x realmente nunca vale 1.1 sino que esta variable tiene asociado un dominio distinto de $dom = [1.1, 1.1]$ en el que se encuentra 1.1. La simulación del proceso $Num(x)$ muestra que el valor de r (i.e., dominio de r) después de 42 unidades de tiempo es $dom = [0.0, 1000.0]$, donde dom es el dominio con que se inicia la variable r.*

Ejemplo 6.2 *Generación infinita de procesos.*

Algunas especificaciones válidas en la teoría presentan graves problemas en la práctica ya que exigen demasiados recursos del sistema de cómputo. Esto hace que cuando se modela es necesario pensar en especificaciones "seguras" que eviten este tipo de problemas. El problema que se presenta a continuación es la posibilidad de generar, a medida que avanza el tiempo, una gran cantidad de procesos que finalmente hacen que la máquina colapse (e.g., consumo total de la memoria o del procesador). Suponga las siguientes tres especificaciones en `ntcc`*:*

$$
\begin{aligned}
P_1 &\stackrel{\text{def}}{=} !\mathbf{tell}(x = 1) \\
P_2 &\stackrel{\text{def}}{=} !!\mathbf{tell}(x = 1) \\
P_3 &\stackrel{\text{def}}{=} !!!\mathbf{tell}(x = 1)
\end{aligned}
$$

Las tres especificaciones hacen que al final de todo instante de tiempo se observe en el store *únicamente la restricción* $x = 1$. *Sin embargo, el proceso* P_1 *sólo genera un proceso por unidad de tiempo, mientras que los procesos* P_2 *y* P_3 *generan múltiples procesos imponiendo la misma restricción. Tal número de procesos aumenta con el paso del tiempo afectando el rendimiento de la simulación. Por ejemplo, los tiempos de simulación de 500 unidades de tiempo de los procesos* P_1, P_2 *y* P_3 *son* $60ms$, $0.92s$ *y* $2min38s$ *respectivamente. Estas pruebas fueron hechas en un computador Intel Pentium 3 con 512Mb de RAM.*

6.5. Conclusiones

Dado el estudio realizado sobre la aplicación del modelo CCP en bioinformática, y en particular del uso del cálculo de procesos `ntcc` para el modelado de redes de regulación genética, tomando el operón de la lactosa como caso de estudio, se presentan las siguientes conclusiones en términos de las *ventajas* y *desventajas* identificadas en el modelado, simulación y verificación de propiedades en este tipo de sistemas usando `ntcc`.

Ventajas del uso de CCP en el modelado, simulación y verificación de propiedades en sistemas biológicos

Modelado

1. Los cálculos de procesos basados en restricciones son adecuados para modelar, con distintos *niveles de detalle*, *abstracciones* de los sistemas biológicos que ellos describen. Esto se debe a que el uso combinado de restricciones y operadores para razonar sobre información parcial permite hacer descripciones en las que es necesario especificar *sólo* la información conocida del sistema según el nivel de detalle deseado. Un aspecto interesante en este punto es la posibilidad de razonar incluso sobre la ausencia de información gracias a la adición de unidades de tiempo o fases de ejecución. Es importante destacar que en el contexto biológico existen muchos sistemas cuya estructura y comportamiento aún son materia de estudio, y por lo tanto la información que se tiene de ellos es escasa y en algunos casos nula.

2. El uso de *procesos* en el modelado de sistemas biológicos facilita la descripción de sistemas en los que se tienen identificados algunos de los subsistemas que los conforman. De esta forma, es posible modelar cada subsistema de forma *independiente* para después relacionarlos, es decir, definir la manera en que ellos interactúan con los otros y con el medio. Pese a que esta ventaja, relacionada con la *concurrencia* de los procesos, puede ser conseguida también en otros paradigmas de programación (e.g., concurrencia de programas imperativos en C), en CCP este concepto se tiene desde la misma concepción del modelo facilitando la labor de quien realiza la especificación de los sistemas. El uso de este concepto facilitó la especificación del operón de la lactosa una vez se habían identificado sus partes (e.g., región de control, región de genes estructurales) y tipos de comportamiento (e.g., inducción, represión, hidrólisis, etc.).

3. El uso de *diferentes sistemas de restricciones* como *parámetros* de los modelos hace que no sea necesario preocuparse por el tipo de datos de las variables que representan los sistemas. Así, el modelado de las interacciones o relaciones entre las entidades que conforman un sistema biológico puede ser hecho sin importar el tipo de variables utilizadas para representar los sistemas. En otras palabras, la restricción $x > y$ es igualmente válida *independientemente* del sistema de restricciones usado para definir las variables x y y. Esta característica permite además, darle un trato más preciso a las variables del sistema, ya que es posible usar el sistema de restricciones más adecuado según sea el caso. En el desarrollo del modelo a nivel molecular del operón de la lactosa, se aprovechó esta ventaja al imponer de forma combinada restricciones sobre enteros y sobre reales en la misma especificación.

4. La disponibilidad de operadores en `ntcc` para modelar comportamiento *asíncrono* y *no determinista* permite describir eventos biológicos que suceden de forma *impredecible*. Por lo tanto, con el uso adecuado de estos operadores se puede formalizar el hecho de que en los sistemas biológicos, al igual que en todo sistema presente en la naturaleza, existen muchas situaciones en las que no es posible asegurar el momento en que ellas ocurren (e.g., el momento en que alguien puede contraer una enfermedad) dadas unas condiciones ambientales. En nuestro caso de estudio el uso de estos operadores sirvió enormemente para modelar de una manera sencilla la posible mutación de un gen de la red de regulación.

5. El posible construir componentes reutilizables que representen el comportamiento de entidades biológicas presentes en una red de regulación genética. Esto se evidencia en la definición del componente para modelar el funcionamiento de un gen. Tal especificación puede considerarse un modelo de alto nivel, en el que se introduce al diseñador de la red de regulación en razonamientos más biológicos que computacionales.

Implementación o simulación

1. Dado que en CCP sólo interesan las *restricciones* que se imponen y no la forma o secuencia en que esto se hace para calcular un resultado, los modelos derivados se implementan usando el paradigma de *programación declarativa*, haciendo que no sea necesario definir estructuras que controlen la ejecución de los procesos. La sincronización de ellos se realiza usando la información que puede ser inferida del almacén de restricciones en cada unidad de tiempo. De esta forma, una arquitectura de cómputo paralela con un esquema de memoria compartida podria mejorar el rendimiento de la ejecución de un programa. Esto estaría sujeto a la paralelización de las acciones a realizar en cada unidad de tiempo. Por ejemplo, la red de regulación del operón de la lactosa descrita en el capítulo 4 es un sistema susceptible de tal paralelización.

2. Los modelos propuestos en la teoría usando el modelo CCP pueden ser llevados a la práctica mediante el uso de librerías, lenguajes y simuladores, que permitan observar el comportamiento de los sistemas modelados. Esto es definitivamente una ventaja cuando se compara contra otros cálculos de procesos en los que no existen herramientas para simular los modelos hechos con el cálculo (por ejemplo Brane según el mejor saber de los autores), y por tanto se vuelven lenguajes de especificación formal poco usados. En el caso del modelo CCP se cuenta por ejemplo, con librerías como GECODE [Sch06], lenguajes de programación como Mozart-Oz y herramientas de software como **ntccSim** o jcc [SJG03]. Es importante recordar que ya que en bioinformática se usan técnicas de las ciencias de la computación para la solución de problemas biológicos, contar con herramientas de este tipo es sumamente importante.

3. Dado que las implementaciones usando **ntccSim** son *multiplataforma*, los modelos hechos en `ntcc` pueden ser simulados en varios sistemas operativos sin tener que modificar su implementación. Esta característica de las implementaciones en **ntccSim** incrementa las posibilidades de usar el cálculo `ntcc` como lenguaje de descripción de sistemas biológicos ya que no se introducen mayores limitaciones computacionales, que normalmente generan divisiones en investigadores en las ciencias de la computación dedicados a la bioinformática.

4. El hecho de poder usar *múltiples sistemas de restricciones* en **ntccSim** permite considerar la posibilidad de implementar en Mozart-Oz sistemas de restricciones diseñados para modelar variables que representen características propias de la biología. De esta forma, por ejemplo, pueden abstraerse las características fundamentales de las redes de regulación genética como propiedades de grafos, y así poder modelar de forma intuitiva (posiblemente gráfica) procesos biológicos típicos en este tipo de sistemas.

Verificación de propiedades

1. Es posible *probar propiedades* de sistemas biológicos usando técnicas de razonamiento o inferencia basadas en la *lógica*. Esto se debe a que los procesos en `ntcc` pueden ser vistos como fórmulas de primer orden en la lógica LTL que tiene asociada. De esta forma fue posible verificar, usando el sistema de pruebas de `ntcc`, una propiedad de estabilidad en el operón de la lactosa *imposible de asegurar* por medios prácticos con **ntccSim**, ya que tal prueba implicaría una simulación infinita.

2. Las propiedades biológicas verificadas usando el sistema de pruebas de `ntcc` sobre uno o más agentes concurrentes de un modelo, son igualmente válidas tomando en cuenta los demás procesos especificados del sistema. Esto quiere decir que mientras no se modifique la descripción en `ntcc` del sistema biológico modelado, las pruebas que se realicen son correctas en todo momento y bajo cualquier ambiente. Nuevamente se encuentran diferencias contra lo que se puede obtener por medios de simulación, ya que en ese caso los datos resultantes pueden sugerir como generales casos particulares de comportamiento. Con las pruebas formales tal incertidumbre no tiene lugar.

Desventajas del uso de CCP en el modelado, simulación y verificación de propiedades en sistemas biológicos

Modelado

1. Las características *temporales discretas* de `ntcc` hacen necesario el uso de *encodings* para la representación sencilla de *sistemas continuos*. Esto claramente representa una desventaja de `ntcc` contra otros cálculos de procesos basados en restricciones como hcc, donde el tiempo está definido sobre tiempo continuo permitiendo el uso de sistemas de restricciones sobre *ecuaciones diferenciales*, la forma más natural y usada de modelar sistemas continuos. En biología, poder modelar este tipo de sistemas es definitivamente una necesidad, ya que el comportamiento de muchos procesos a nivel celular está descrito en términos de ecuaciones diferenciales.

2. Es necesaria una contraparte formal *gráfica* que soporte los modelos en `ntcc` para que puedan ser usados por expertos en biología. En ese sentido, la falta de *interfaces* amigables que le permitan a un usuario experto en biología pero poco relacionado con la computación, hace que la idea de modelar usando sólo las primitivas del cálculo sea de dominio exclusivo de personas del campo de las ciencias de la computación. En particular, las redes de regulación genética normalmente presentan representaciones gráficas (informales) en las que se especifican las reacciones mediante las cuales se producen, degradan y transforman los componentes biológicos que desarrollan los procesos de control celular.

3. No es posible sacar un provecho total de la capacidad que tiene `ntcc` de *razonar sobre la ausencia de información*, ya que toda decisión que se pueda tomar con base en esto sólo puede hacerse efectiva al menos una unidad de tiempo después. Por tal motivo, no es posible reaccionar de forma *inmediata* cuando se produce un evento molecular que cambia el comportamiento de los sistemas modelados (e.g., cambio de una variable de señalización), generando un retraso en los procesos de regulación. Sin embargo, este problema es resuelto parcialmente definiendo una resolución temporal pequeña en el esquema propuesto para el modelado de sistemas continuos con `ntcc`.

Implementación o simulación

1. La ejecución *concurrente* de los procesos hace que sea muy complicado localizar errores en el programa cuando se producen fallos en su simulación. Esto hace que el tiempo que se gana en el paso del modelo teórico en `ntcc` a su implementación en Mozart-Oz usando **ntccSim**, en ocasiones se pierda cuando se está ejecutando el programa. Nuevamente un ambiente gráfico en el que se modele y simule al mismo tiempo podría ayudar a usuarios biólogos, poco relacionados con la programación, a usar este tipo de herramientas.

2. Es preciso encontrar una forma *eficiente* de solucionar el problema de la propagación del error, mostrado en el ejemplo 6.1, cuando se hacen simulaciones usando variables sobre *reales*. Este problema puede resolverse inicialmente definiendo una precisión en los dominios de las variables reales según algún criterio biológico, por ejemplo, que concentraciones menores a una precisión particular puedan considerarse despreciables, y de esta forma poder escoger o buscar en poco tiempo cualquier valor de ese dominio como una solución de la variable en cuestión. La eficiencia en la ejecución de estos modelos es un tema importante, ya que normalmente los modelos biológicos son bastante complejos.

3. La ventaja de poder usar *varios sistemas de restricciones* en los modelos en `ntcc` es limitada con el hecho de que puede ser muy complicada su implementación *eficiente*. Por ejemplo, si se contara con un sistema de restricciones sobre ecuaciones diferenciales en Mozart-Oz, no hubiera sido necesario desarrollar *encodings* para representar este tipo de información. Sin embargo, en algunas ocasiones, pese a contar con el sistema de restricciones deseado aún persiste el problema relacionado con el rendimiento de las simulaciones.

Verificación

1. Los buenos resultados obtenidos con el uso del sistema de inferencia de `ntcc` sugieren la necesidad de desarrollar un *probador de teoremas* para automatizar el desarrollo de pruebas formales. Esta herramienta para realizar verificación de propiedades y chequeo de correctitud de los modelos puede ayudar a que el trabajo se concentre en el diseño de pruebas formales que sean válidas en un conjunto amplio de situaciones.

6.6. Trabajo Futuro

Algunas de las actividades propuestas como trabajo futuro son para potenciar las ventajas explicadas en la sección 6.5, mientras que otras nacen con el ánimo de superar las desventajas identificadas. Por tanto, estas sugerencias de trabajo futuro se construyen sobre bases muy firmes, ya que tales ventajas y desventajas tratadas en la sección anterior son realmente el fruto de las conclusiones de este trabajo de grado.

1. Estudiar la posibilidad de desarrollar extensiones de `ntcc` que faciliten el modelado de sistemas biológicos. Tales extensiones deben estar relacionadas con la capacidad de modelar tiempo continuo, razonamiento inmediato sobre la ausencia de información y la inclusión de elementos probabilísticos en los operadores de escogencia del cálculo. Estas extensiones deben incluir necesariamente la extensión del sistema de pruebas de `ntcc`. Algunos trabajos ya han sido iniciados, por ejemplo, en [OR05] se propone una extensión estocástica para `ntcc`.

2. Con el ánimo de mejorar la forma de representar información parcial cuantitativa, es recomendable el desarrollo de un sistema de restricciones sobre ecuaciones diferenciales para poder simular modelos en `ntcc` en los que no sea necesario el uso de *encodings* para la representación de este tipo de información. Actualmente existen trabajos adelantados, tales como los expuestos en [Cru03], donde se sientan las bases para la implementación de algoritmos de propagación sobre ecuaciones diferenciales.

3. Mejorar o modificar la implementación actual de **ntccSim** de la siguiente forma: (i) definir una sintaxis que sea aún más similar a la de `ntcc`, (ii) diseñar un depurador para los programas, (iii) desarrollar un módulo **bio-ntccSim** que permita modelar y simular de forma gráfica componentes básicos de las redes de regulación genética, y (iv) estudiar la posibilidad de usar GECODE como motor de restricciones en lugar del lenguaje de programación Mozart-Oz.

4. Desarrollar un probador de teoremas para el sistema de inferencia de `ntcc`, con el objetivo de poder verificar de forma automática propiedades de los modelos hechos con el cálculo.

Bibliografía

[AG99] M. Abadi and A. Gordon. A Calculus for Cryptographic Protocols: The spi Calculus. *Information Computing*, 148(1):1–70, 1999.

[AGOR06] A. Arbeláez, J. Gutiérrez, C. Olarte, and C. Rueda. A Generic Framework to Model, Simulate and Verify Genetic Regulatory Networks. In *CLEI'06.*, 2006.

[Bac97] R. Backofen. Using Constraint Programming for Lattice Protein Folding. In *German Conference on Bioinformatics*, page 123. Oxford University Press, 1997.

[BC01] A. Bockmayr and A. Courtois. Modeling Biological Systems in Hybrid Concurrent Constraint Programming. In *In Proceedings of the 2nd International Conference on System Biology*, 2001.

[BC02] A. Bockmayr and A. Courtois. Using Hybrid Concurrent Constraint Programming to Model Dynamic Biological Systems. In P. Stuckey, editor, *ICLP*, volume 2401 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 2002.

[BG01] R. Backofen and D. Gilbert. Bioinformatics and Constraints. *Constraints*, 6(2/3):141–156, 2001.

[bio06] The BIOPSI project, 2006. Available at www.wisdom.weizmann.ac.il/ biopsi.

[BS05] L. Bortolussi and A. Sgarro. Constraint Satisfaction Problems over DNA Strings. In *Proceedings of WCB05 Workshop on Constrint Based Methods for Bioinformatics*, 2005.

[Car04] L. Cardelli. Brane Calculi. In V. Danos and V. Schachter, editors, *CMSB*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–278. Springer, 2004.

[CB03] J. Cruz and P. Barahona. Constraint Satisfaction Differential Problems. In *CP 2003*, pages 259–273, 2003.

[CCT02] G. Ciobanu, V. Ciubotariu, and B. Tanasa. A pi-calculus Model of the Na Pump. *Genome Informatics*, pages 469–472, 2002.

[CG98] L. Cardelli and A. Gordon. Mobile Ambients. In M.Ñivat, editor, *FoSSaCS*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 1998.

[Con06] Mozart Consortium. The Mozart Programming Language, 2006. Available at www.mozart-oz.org.

[Cru03] J. Cruz. *Constraint Reasoning for Differential Equations*. PhD thesis, Universidade Nova de Lisboa, 2003.

[DDD04a] G. Dooms, Y. Deville, and P. Dupont. A Mozart Implementation of CP(BioNet). In P. Van Roy, editor, *MOZ*, volume 3389 of *Lecture Notes in Computer Science*, pages 237–250. Springer, 2004.

[DDD04b] G. Dooms, Y. Deville, and P. Dupont. Constrained Path Finding in Biochemical Networks. In *In Proceedings of JOBIM 2004*, June 2004.

[DDD05] G. Dooms, Y. Deville, and P. Dupont. Constrained Metabolic Network Analysis: Discovering Pathways using CP(Graph). In *Proceedings of WCB05 Workshop on Constraint Based Methods for Bioinformatics*, October 2005.

[DL04] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[DRV+01] J. Díaz, C. Rueda, F. Valencia, G. Alvarez, L. Quesada, and G. Tamura. Integrating Constraints and Concurrent Objects in Musical Applications: A Calculus and its Visual Language. *Constraints*, 6(1):21–52, 2001.

[ERdJ+04] D. Eveillard, D. Ropers, H. de Jong, C. Branlant, and A. Bockmayr. A multi-scale Constraint Programming Model of Alternative Splicing Regulation. *Theoretical Computer Sciense*, 325(1):3–24, 2004.

[GJP99] V. Gupta, R. Jagadeesan, and P. Panangaden. Stochastic Processes as Concurrent Constraint Programs. In *POPL '99: Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 189–202. ACM Press, 1999.

[GJS96] V. Gupta, R. Jagadeesan, and V. Saraswat. Models for Concurrent Constraint Programming. In U. Montanari and V. Sassone, editors, *CONCUR*, volume 1119 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 1996.

[GJS98] V. Gupta, R. Jagadeesan, and V. Saraswat. Computing with Continuous Change. *Science Computer Program*, 30(1-2):3–49, 1998.

[GJSB94] V. Gupta, R. Jagadeesan, V. Saraswat, and D. Bobrow. Programming in Hybrid Constraint Languages. In *Hybrid Systems*, pages 226–251, 1994.

[GPR05] J. Gutiérrez, J. A. Pérez, and C. Rueda. Modelamiento de Sistemas Biológicos usando Cálculos de Procesos Concurrentes. *Epiciclos*, 4, 2005.

[GPRV06] J. Gutiérrez, J. A. Pérez, C. Rueda, and F. D. Valencia. Using a Timed Concurrent Constraint Process Calculus for Modeling and Verifying Biological Systems. In *MecBIC'06*. Electronic Notes in Theoretical Computer Science (ENTCS) series, 2006.

[Gro04] AVISPA Research Group. *XRI: Extended Real Interval*, 2004. Available at http://home.gna.org/xrilpoz/.

[Gro06] AVISPA Research Group, 2006. Available at http://avispa.puj.edu.co.

[HNW05] E. Hairer, S.Ñorsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, chapter Delay Differential Equations, pages 339–355. Springer, 2005.

[HP00] O. Herescu and C. Palamidessi. Probabilistic Asynchronous pi-Calculus. In *Foundations of Software Science and Computation Structure*, pages 146–160, 2000.

[KD03] J. Krivine and V. Danos. Formal Molecular Biology done in CCS-R. In *BioConcur 2003, Workshop on Concurrent Models in Molecular Biology*, 2003.

[Kit01a] H. Kitano, editor. *Foundations of Systems Biology*. MIT Press, 2001.

[Kit01b] H. Kitano. *Foundations of Systems Biology*, chapter Systems Biology: Toward System-level Understanding of Biological Systems. MIT Press, 2001.

[KJ04] C. Kuttler and J.Niehren. Gene Regulation in the π Calculus: Simulating Cooperativity at the Lambda Switch. In *BioConcur: Workshop on Concurrent Models in Molecular Biology*, volume Extended Version, August 2004. Extended Version.

[Les02] A. Lesk. *Introduction to Bioinformatics*. Oxford University Press, 2002.

[Lew00] B. Lewin. *Genes VII*. Oxford University Press, 2000.

[Mil89] R. Milner. *Comunication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[Mil99] R. Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press, 1999.

[MM04] S. Miyano and H. Matsuno. How to Model and Simulate Biological Pathways with Petri Nets - A New Challenge for Systems Biology -. *25th International Conference on Application and Theory of Petri Nets*, June 2004.

[MPLD04] J. Mendel, N. Palfreyman, J. Lopez, and W. Dubitzky. Representing Bioinformatics Causality. *Briefings in Bioinformatics*, 5(3):270–283, 2004.

[MPV02] M.Nielsen, C. Palamidessi, and F. Valencia. Temporal Concurrent Constraint Programming: Denotation, Logic and Applications. *Nordic Journal of Computing*, 2002.

[MS98] K. Marriot and P. Stuckey. *Programming with Constraints: An Introduction*. The MIT Press, 1998.

[OR05] C. Olarte and C. Rueda. A Stochastic Non-deterministic Temporal Concurrent Constraint Calculus. In IEEE Computer Society, editor, *Proceedings of XXV International conference of the chilean computer science society*, 2005.

[PDF04] A. Dal Palù, A. Dovier, and F. Fogolari. Protein Folding Simulation in CCP. In B. Demoen and V. Lifschitz, editors, *ICLP*, volume 3132 of *Lecture Notes in Computer Science*, pages 452–453. Springer, 2004.

[Pev00] P. Pevzner. *Computational Molecular Biology: An Algorithmic Approach.* MIT Press, 2000.

[PPQ05] D. Prandi, C. Priami, and P. Quaglia. Process Calculi in a Biological Context. *Bulletin of the EATCS*, February 2005.

[Pri95] C. Priami. Stochastic π-Calculus. *The Computer Journal*, 38(6):578–589, 1995.

[PRSS01] C. Priami, A. Regev, W. Silverman, and E. Shapiro. Application of stochastic process algebras to bioinformatics of molecular processes. *Information Processing Letters*, 80:25–31, 2001.

[RPS+04] A. Regev, E. Panina, W. Silverman, L. Cardelli, and E. Shapiro. BioAmbients: An Abstraction for Biological Compartments. *Theoretical Computer Science*, 325(1):141–167, 2004.

[RS02] A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419:343, September 2002.

[RS04] A. Regev and E. Shapiro. *Modelling in Molecular Biology*, chapter The π-calculus as an abstraction for biomolecular systems, pages 219–266. Natural Computing Series. Springer, 2004.

[RSS01] A. Regev, W. Silverman, and E. Shapiro. Representation and Simulation of Biochemical Processes Using the π Calculus Process Algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[RV02] C. Rueda and F. Valencia. Proving Musical Properties Using a Temporal Concurrent Constraint Calculus. In *ICMC'02*, 2002.

[RV05] C. Rueda and F. Valencia. A Temporal CCP Calculus as an Audio Processing Framework. In *SMC'05*, 2005.

[Sar93] V. Saraswat. *Concurrent Constraint Programming.* MIT Press, 1993.

[Sch06] C. Schulte. GECODE. Generic Constraint Development Environment, 2006. Available at www.gecode.org.

[SGJ97] V. Saraswat, V. Gupta, and R. Jagadeesan. Probabilistic Concurrent Constraint Programming. In *CONCUR*, pages 243–257, 1997.

[Sha02] E. Shapiro. Molecule as computation: Towards an abstraction of biomolecular systems. In R. Guigó and D. Gusfield, editors, *WABI*, volume 2452 of *Lecture Notes in Computer Science*, page 418. Springer, 2002.

[SJG94] V. Saraswat, R. Jagadeesan, and V. Gupta. Foundations of Timed Concurrent Constraint Programming. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science, Paris, France, 4–7 July, 1994*, pages 71–80, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1994. IEEE.

[SJG03] V. Saraswat, R. Jagadeesan, and V. Gupta. jcc: Integrating Timed Default Concurrent Constraint Programming into Java. In *Proceedings of EPIA 2003*, volume 2902 of *LNCS*, 2003.

[Smo95] G. Smolka. The Oz Programming Model. In J. Van Leeuwen, editor, *Computer Science Today*, volume 1000 of *LNCS*, pages 324–343. Springer - Verlag, 1995.

[SRP91] V. Saraswat, M. Rinard, and P. Panangaden. The Semantic Foundations of Concurrent Constraint Programming. In *POPL '91*, pages 333–352, Jan 1991.

[Val02] F. Valencia. *Temporal Concurrent Constraint Programming*. PhD thesis, University of Aarhus, November 2002.

[Val03] F. Valencia. Concurrency, Time and Constraints. In C. Palamidessi, editor, *ICLP*, volume 2916 of *Lecture Notes in Computer Science*, pages 72–101. Springer, 2003.

[Val05] F. Valencia. Decidability of Infinite-State Timed CCP Process and First-Order LTL. *Theoretical Computer Science*, 330(3):577–607, 2005.

[VGL03] J. Vilar, C. Guet, and S. Leibler. Modeling network dynamics: the lac operon, a case study. *The journal of the Cell Biology*, 161(3):471–476, 2003.

[WGK97] P. Wong, S. Gladney, and J. Keasling. Mathematical Model of then Lac Operon: Inducer Exclusion, Catabolite Repression, and Diauxic Growth on Glucose and Lactose. *Biotechnol*, 13:132–143, 1997.

[YM03] N. Yildirim and M. Mackey. FeedBack Regulation in the Lactose Operon: A Mathematical Modeling Study and Comparison with Experimental Data. *Biophysical journal*, 84:2841–2851, 2003.

[YSM04] N. Yildirim, M. Santill'an, and M. Mackey. Modeling operon dynamics: the tryptophan and lactose operons as paradigms. *C. R. Biologies*, 327:211–224, 2004.

[Zil02a] D. Zill. *Ecuaciones diferenciales con aplicaciones de modelado*. International Thomson Editores, 2002. Séptima Edición.

[Zil02b] D. Zill. *Ecuaciones diferenciales con aplicaciones de modelado*, chapter Métodos Numéricos para Resolver Ecuaciones Diferenciales Ordinarias. International Thomson Editores, 2002.