Estudio Exploratorio de la Expresividad del Cálculo `ntcc` aplicado al Modelamiento de Sistemas Biológicos

Ángela Villota Gómez código: 0603069
Director: Juan Francisco Díaz Frias

Universidad del Valle
Facultad de Ingeniería
EISC

7 de octubre de 2009

There is nothing you can know, that isn't known,
nothing you can see that isn't shown,
Nowhere you can be that isn't where you're meant to be,
it's easy...

All you need is love,
all you need is love,
all you need is love, love...
love is all you need...

Lennon/Mcartney

Agradecimientos

La idea de realizar una maestría nació a finales del 2004, cuando aun no había terminado el pregrado y justo antes de que ocurriera un evento importante en mi vida. La ejecución de dicha idea o aspiración viene ocurriendo desde agosto del 2006, y desde ese momento hasta ahora, Julio de 2009, cuento con un grupo de personas quienes han sido mi grupo de apoyo en muchos sentidos y a quienes agradezco inmensamente porque sé que mis estudios de maestría, la propuesta y desarrollo de esta tesis no habrían sido realizados sin estas personas a mi alrededor.

Quiero agradecer principalmente a Juan Francisco quien me ha acompañado y guiado en este proyecto desde el momento en que surgió la idea. Conozco a Juan Francisco desde hace unos 9 años cuando aún era estudiante de Ingeniería de Sistemas, ha sido mi jefe en múltiples proyectos y director de tesis en pregrado y posgrado. Ha sido también mi amigo y maestro, alguien con quien puedo hablar y de quien constantemente estoy aprendiendo.
Agradezco también a mis compañeros del grupo AVISPA, especialmente a Jesús Aranda[1], quien con su asesoría me ayudó a encontrarle el lado a los problemas de modelamiento.
Gracias también a mis profesores de la maestría, en especial a Martha, Ángel e Irene, disfruté cada una de sus clases y fueron éstas un medio para desarrollar las habilidades que he utilizado en este proyecto.
A mi familia quiero darle gracias por su apoyo y presencia, sobretodo por cuidar a mi bebe mientras yo estaba en la universidad. En especial doy gracias a Gerardo quien le ha dado alas a cada una de mis locas ideas y a Juan Martín quien es un niño ahora y ha crecido feliz, te doy gracias nene porque has comprendido pacientemente mi ausencia.

Gracias a todos, GRACIAS de corazón!!

[1]Research partially supported by the INRIA Equipe Associee FORCES. The work of Jesús Aranda has been supported by COLCIENCIAS (Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnologia "Francisco Jose de Caldas"), INRIA and EGIDE (Centre francais pour l' accueil et les echanges internationaux)

Índice general

1. Introducción **1**
1.1. Preguntas de Investigación y Contribuciones 1

2. Cálculos de procesos **4**
2.1. Elementos de un cálculo de procesos 5
2.2. Cálculos de procesos en la Bioinformática 7
2.2.1. Problemas biológicos y cálculos de procesos 8
2.3. *C*oncurrent *C*onstraint *P*rogramming 9
2.3.1. Sintáxis 10
2.3.2. Semántica Operacional 11
2.3.3. Cálculo tcc 12
2.4. Cálculo `ntcc` 13
2.4.1. Semántica Operacional 13
2.4.2. Semántica Denotacional 16
2.4.3. Aplicaciones 16

3. Cálculos de procesos inspirados en la biología **19**
3.1. CHAM (CHemical Abstract Machine) 19
3.1.1. Sintaxis 19
3.1.2. Reglas de transformación 20
3.2. BioAmbients 21
3.2.1. Sintaxis 21
3.3. Brane 25
3.3.1. Sintaxis 26
3.3.2. Reglas de reducción 30
3.4. Sistemas de Membranas - *P-Systems* 31
3.4.1. Sintaxis 33
3.4.2. Reglas de evolución 34

4. Modelando sistemas de membranas en `ntcc` **37**
4.1. Modelando Membranas, Localización y Jerarquías 38
4.1.1. Sintaxis 39
4.2. Modelando Reglas de evolución como reacciones químicas 40
4.2.1. Reacciones usando escogencia no determinista 41

4.2.2. Reacciones usando composición paralela 42
4.2.3. Reglas de evolución para un sistema de membranas en `ntcc` 43
4.3. Simulando sistemas de membranas en `ntcc` 44
4.3.1. Declaración Variables 46
4.3.2. Definición de procesos derivados de *Reaction* 46
4.3.3. Definición de proceso de control 49

5. Modelo en `ntcc` de la ruta endocítica de la Lipoproteína baja densidad (LDL) del colesterol 50
5.1. Descripción del sistema desde la biología 50
5.1.1. Ruta Endocítica de la LDL del Colesterol 51
5.1.2. Modelo en `ntcc` 53
5.1.3. Modelo utilizando escogencia no determinista (+) 55
5.1.4. Modelo utilizando composición paralela (|) 56

6. Conclusiones y trabajo futuro 59

Bibliografía 62

Índice de figuras

2.1. Un *store* CCP con cuatro agentes. 10
2.2. Sistema Reactivo 13

3.1. Operaciones que modelan interacción entre compartimientos en BioAmbients, tomada de [15] 23
3.2. Ejemplo de las operaciones de ingreso y salida de membranas . . 25
3.3. Ejemplo de las operaciones de unión de membranas 25
3.4. Coloración bitonal, tomada de [18]. 25
3.5. Polaridad de las membranas, tomada de [18]. 26
3.6. Notación gráfica de un sistema, tomada de [17]. 26
3.7. Notación gráfica de las operaciones de Brane inspiradas en la endo y exocitosis. Tomada de [17]. 29
3.8. Notación gráfica de las operaciones de Brane inspiradas en la división celular. Tomada de [17]. 29
3.9. Estructura de una membrana 32

5.1. Endocitosis mediada por receptor 51
5.2. Ruta endocítica de la lipoproteína de baja densidad del colesterol, tomada de [24] 52

Índice de cuadros

2.1. Sintaxis de CCS 5
2.2. Reglas de transición de CCS 7
2.3. Sintaxis de *CCP* 11
2.4. Reglas de Transición de *CCP* 12
2.5. Sintaxis de `ntcc` 14
2.6. Reglas de Transición de `ntcc` 15
2.7. Semántica Denotacional de `ntcc` 17

3.1. Sintaxis de BioAmbients. 22
3.2. Reglas de reducción de BioAmbients. En la figura, *M/H* se refiere a Madre/Hija, y *H/M* se refiere a Hija/Madre 24
3.3. Sintáxis del cálculo Brane. 27
3.4. Reglas de reducción básicas. 30
3.5. Reglas de reducción de las acciones. 30
3.6. Vista gráfica del modelo de la ruta endocítica de la LDL del colesterol 31
3.7. Modelo de la ruta endocítica de la LDL del colésterol usando Brane 31
3.8. Modelo de la ruta endocítica de la LDL del colésterol usando *P-Systems* 36

4.1. Ejemplo: Sistema con dos niveles y cuatro objetos. 40
4.2. Sintaxis de *ntccSim* 45
4.3. Funciones auxiliares de *Reaction*1 47
4.4. Función auxiliares de *Reaction*2 48
4.5. Implementación del proceso Control en Mozart 49

5.1. Modelo de la ruta endocítica de la LDL del colésterol con el cálculo `ntcc` usando escogencia no determinista 56
5.2. Modelo con paralelismo y el proceso $Control_{gen}$ 58

Resumen

El cálculo de procesos `ntcc` ha sido propuesto como un mecanismo apropiado para modelar sistemas concurrentes que evolucionan en el tiempo, como los sistemas biológicos entre otros. En el estudio de la expresividad de `ntcc`, en particular su aplicación en la especificación, simulación y análisis de sistemas biológicos, surge un interrogante: ¿Se pueden modelar sistemas de membranas con ntcc? En este documento se presenta una contribución a la solución de dicho interrogante y se muestra un mecanismo general para lograr modelar sistemas de membranas en `ntcc`. La aplicación de dicho mecanismo se ilustra con un ejemplo: la propuesta de un modelo para la ruta endocítica de la LDL del colesterol en `ntcc` usando sistemas de membranas.
Palabras claves: *ntcc, sistemas de membranas, modelos de sistemas biológicos*

Capítulo 1

Introducción

Los sistemas biológicos pueden ser vistos como sistemas en donde existen procesos que interactuán de forma concurrente y evolucionan en el tiempo. El área que aborda la biológía desde esta perspectiva es llamada *biología de sistemas*[10, 11]. En el contexto de la biología de sistemas se ha evidenciado la utilidad de las herramientas computacionales que permiten modelar y simular procesos biológicos conocidos. Entre estas herramientas se encuentran los cálculos de procesos que han mostrado ser útiles para la modelación, simulación y verificación de propiedades de sistemas biológicos [1, 2, 14, 15, 17, 19]. De entre las distintas posibilidades, este trabajo explora la expresividad de `ntcc`, un cálculo basado en el modelo CCP [8] (*Concurrent Constraint Programming*), en particular su adecuación para modelar sistemas de membranas. En artículos como [1, 2, 19] se muestra la utilización de `ntcc` para modelar procesos biológicos destacando los beneficios que trae la representación de información parcial, la independencia entre los constructores del cálculo y la demostración de propiedades a través de una lógica temporal (LTL).

1.1. Preguntas de Investigación y Contribuciones

En este proyecto de investigación se desea contribuir a dar respuesta a las siguientes preguntas:

- ¿Son los constructores del cálculo `ntcc` suficientes para modelar sistemas biológicos?
- ¿Se pueden modelar con el cálculo ntcc todos los conceptos que modelan los cálculos bioinspirados, específicamente, modela el concepto de ambientes y operaciones entre membranas?
- ¿Se pueden codificar en el cálculo las operaciones de membranas? ¿Cómo hacerlo?

- Si el cálculo ntcc no modela todos los conceptos que modelan los cálculos de membranas (ambientes), entonces, ¿qué conceptos deben incluirse en el cálculo para poder llevar a cabo dichos modelos?

Desde el punto de vista de las ciencias de la computación, este trabajo de investigación contribuye al estudio de la expresividad del cálculo ntcc, porque explora la capacidad de modelar sistemas biológicos en constraste con los cálculos de procesos inspirados en la biología. Al mismo tiempo, el determinar que con `ntcc` se pueden modelar los conceptos que modelan cálculos bioinspirados, se hace contribución a la tarea de consolidar un único formalismo que sea útil en diferentes áreas. Aquí se debe anotar que este trabajo de investigación hace parte del proyecto REACT (*Robust theories for Emerging Applications in Concurrency Theory*), realizado por el grupo AVISPA. En este proyecto se exploró la utilización de `ntcc` en la biología, seguridad e interacción multimedial.

En concreto los pricipales propósitos y características de este trabajo pueden resumirse de la siguiente manera:

Un estudio de los formalismos inspirados en las membranas biológicas. Se revisaron algunos de los formalismos que modelan Membranas destacando los conceptos que hacen parte de los formalismos, su sintáxis y las reglas de evolución o de transición.

Un conjunto de técnicas que permiten modelar sistemas de membranas en `ntcc`. Se hizo una comparación de los conceptos que modela `ntcc` en contraste con los formalismos estudiados y se realizó la propuesta de un conjunto de técnicas que permiten modelar sistemas de membranas sin incluir nuevos conceptos a `ntcc` (extensión del cálculo).

Aplicaciones.

- Se implementaron unos procedimientos y funciones en lenguaje Mozart que permiten simular sistemas de membranas utilizando *ntccSim* y siguiendo las técnicas propuestas.
- Se modeló y simuló la ruta endocítica de la LDL del colesterol usando las técnicas propuestas y la implementación realizada. El simulador *ntccSim*, más la librería de membranas y los archivos con la simulación se pueden descargar de:
  `http://eisc.univalle.edu.co/~avillota`.

Publicaciones. Se presentó el artículo titulado *Modelando Sistemas de Membranas en* `ntcc` en conjunto con Jesús Aranda y Juan Francisco Díaz. Fué aprobado y será presentado en la Conferencia Latinoamericana de Informática (CLEI 2009) en septiembre de este año.

Organización

Este documento está organizado de la siguiente forma:

- En el primer capítulo se encuentra el marco teórico que abarca los conceptos importantes de cálculos de procesos, haciendo énfasis en el modelo *CCP* y en `ntcc`
- El segundo capítulo contiene el estudio de los formalismos y cálculos de procesos que modelan sistemas de membranas.
- En el tercer capítulo se describen las técnicas que permiten modelar sistemas de membranas en `ntcc` y también las que permiten simular modelos en *ntccSim*.
- El cuarto capítulo presenta el modelo de la ruta endocítica de la LDL del colesterol, incluyendo una breve descripción de los conceptos biológicos y el sistema que se desea modelar.
- En el quinto capítulo se presentan las conclusiones y trabajo futuro de este proyecto de investigación.

Capítulo 2

Cálculos de procesos

Los cálculos de procesos son una alternativa para el estudio de sistemas concurrentes. Estos surgieron a principios de los 70's cuando los investigadores en Ciencias de la Computación se dieron cuenta que los formalismos existentes eran insuficientes para hacer verificaciones y probar propiedades en sistemas concurrentes. Los sistemas concurrentes son aquellos que tienen procesos que interactúan de forma paralela, esto es, varios procesos que comparten datos y que los procesan al mismo tiempo. En un sistema concurrente los procesos pueden interactuar por medio de:

- Composición paralela; dos procesos están activos al mismo tiempo, por lo general denotada con el símbolo |.
- Composición alternativa (choice); dos procesos cumplen las condiciones para estar activos, pero solo uno de ellos se activa y el otro se descarta. Se usa el símbolo + entre los procesos que hacen parte de la alternativa.
- Composición secuencial; primero se activa uno y después el otro.

Las características de los sistemas concurrentes trajeron consigo cuestionamientos sobre los formalismos que ya existían tales como: ¿Cómo incluir la composición paralela dentro de la semántica del lenguaje? y ¿Cómo se va a representar el estado de una ejecución? [4]

En 1971 H. Bekic en [6] plantea por primera vez la necesidad de cambiar el enfoque cuando se trabaja con concurrencia y propone trabajar con cálculos con el fin de tener una perspectiva de alto nivel. Más adelante autores como Milner y Hoare hacen destacables aportes con los cálculos CCS y CSP respectivamente. En la actualidad existen cálculos de procesos que van más allá de modelar concurrencia sino que también modelan restricciones, tiempo, procesos estocásticos, inspirados en la biología, entre otros.

De forma más precisa un *cálculo de Procesos* permite especificar sistemas concurrentes, simular la evolución de un sistema y además provee herramientas

para probar propiedades. Los sistemas concurrentes se modelan por medio de abstracciones que emulan el comportamiento de un sistema. Para describir el comportamiento hay que tener en cuenta:

- El conjunto de eventos o acciones que un sistema puede realizar,
- el orden en el que se ejecutan los eventos o acciones, y
- el tiempo en que ocurren, incluso la probabilidad de esta ejecución.

En resumen los cálculos de procesos son formalismos para modelar sistemas concurrentes; manejan los conceptos de proceso e interacción; tienen definidas operaciones primitivas; y permiten la construcción de los modelos de abajo hacia arriba (son modulares). Todo lo anterior permite que sean vistos como herramientas para la abstracción de sistemas dinámicos complejos.

2.1. Elementos de un cálculo de procesos

Como mecanismo para el estudio de sistemas concurrentes, los cálculos de procesos ofecen un medio para especificar los sistemas (sintaxis), un mecanismo para hacer seguimiento a la evolución (semántica operacional) y otro para la verificación de propiedades (semántica axiomática). Estos elementos están acompañados además de una semántica denotacional que está relacionada con la descripción matemática de los procesos; por lo general no es utilizada para especificar sistemas, porque está orientada a la parte matemática del cálculo. Para describir los elementos de un cálculo de procesos se utiliza como ejemplo el cálculo CCS [23].

Sintaxis

Son el conjunto de reglas para escribir correctamente sentencias en el cálculo, por lo general están expresadas por medio de una gramática en BNF. A continuación se muestra la sintaxis de CCS, la gramática puede encontrarse en el cuadro 2.1.

$P, Q, \ldots$	$::=$	K	Proceso constante
	$\mid$	$\alpha.P$	Action
	$\mid$	$\sum_{i \in I} P_i$	Sumatoria
	$\mid$	$P \mid Q$	composición paralela
	$\mid$	$P[f]$	Renombramiento
	$\mid$	$P \setminus L$	Restricción

Cuadro 2.1: Sintaxis de CCS

La especificación de la gramática inicia con un conjunto de nombres para denotar procesos, es muy común que estos sean denotados por letras iniciando

en P, pero esto puede variar en los distintos cálculos. Luego se muestran las distintas producciones de la gramatica que permiten la especificación de un proceso; estas se conocen también como los constructores del cálculo. Un sistema se modela por medio de una expresión que sigue las reglas de sintáxis del cálculo. La especificación de un sistema se hace de forma modular, esta es construida como la composición (siguiendo la sintáxis) de procesos previamente definidos. Considere el siguiente ejemplo:

$$\begin{aligned} P &\stackrel{def}{=} R|Q \\ R &\stackrel{def}{=} \alpha.R_1 + \beta.R_2 + \gamma.R_3 \\ Q &\stackrel{def}{=} \alpha.Q_1|\beta.Q_2 \end{aligned}$$

Aquí se describe un proceso P que consta de dos procesos Q y R que están en paralelo; para especificar R y Q también se usan los constructores del cálculo.

Semántica Operacional

Son el conjunto de reglas de transición que especifican los pasos computacionales de un proceso, es decir, la forma en que debe ser ejecutado por la máquina; cada uno de los pasos es llamado una transición. En general la semántica operacional se define por medio de reglas de la forma:

$$\frac{\text{Condiciones}}{\text{Conclusión}}$$

Si las condiciones se obtienen por medio de algún conjunto de transiciones, entonces el estado siguiente está dado por la conclusión. En 2.2 se muestra la semántica operacional de CCS

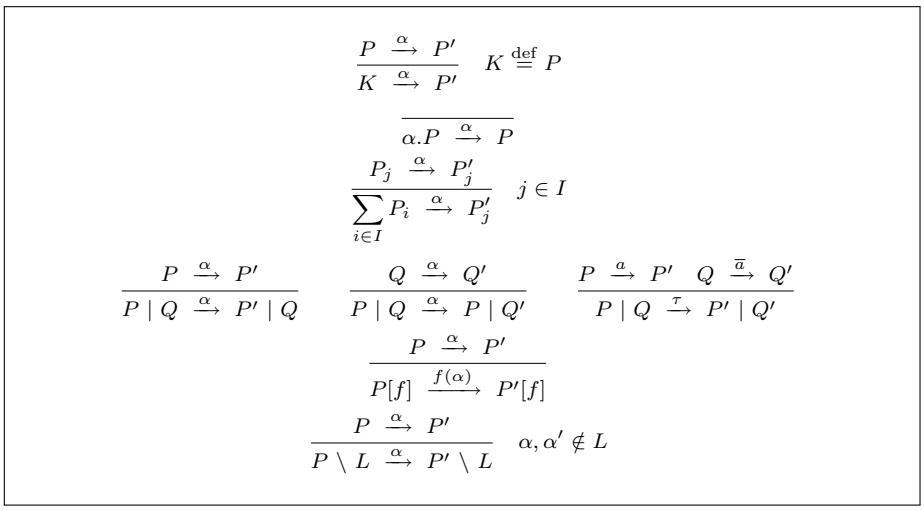
Semántica Denotacional

Son un conjunto de reglas que permiten interpretar a los procesos como objetos matemáticos (v.gr. funciones). Denotan la forma matemática en que debe ser interpretado un proceso. De esta forma, un proceso se trata como una objeto matemático $[\![\ ]\!]$ el cual se contruye de forma composicional, teniendo en cuenta la sintaxis de dicho proceso y su *traducción* con un objeto abstracto, llamado denotación (o significado).

Semántica Algebraica

Son un conjunto de axiomas que permiten hacer demostraciones; por lo general la semántica algebraica está asociada a una lógica dependiendo de los conceptos que modele el cálculo. Por ejemplo, en el caso en que el cálculo modele tiempo, es necesario asociarlo a una lógica temporal, como ocurre con `ntcc`

Algunos cálculos de procesos tienen asociado un lenguaje que es utilizado para escribir las abstracciones realizadas con dicho cálculo. Los lenguajes de programación son una implementación de subconjuntos de los lenguajes asociados



$$\frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'} \quad K \stackrel{\text{def}}{=} P$$

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\frac{P_j \xrightarrow{\alpha} P'_j}{\sum_{i \in I} P_i \xrightarrow{\alpha} P'_j} \quad j \in I$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$\frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad \alpha, \alpha' \notin L$$

Cuadro 2.2: Reglas de transición de CCS

a los cálculos. Por ejemplo $MozArt$[1] es la implementación de OZ, el lenguaje asociado al cálculo ρ [7].

Se puede afirmar que entre mayor sea la cantidad de conceptos que modela el cálculo, mayor será el conjunto de problemas que pueden ser resueltos usando el lenguaje de programación. $MozArt$ es un lenguaje de programación multiparadigma que permite hacer programación declarativa, concurrente, por restricciones, orientada a objetos, entre otras.

2.2. Cálculos de procesos en la Bioinformática

Los sistemas biológicos pueden ser vistos como sistemas que tienen procesos que interactúan de forma concurrente, que se comunican por medio de canales, que evolucionan en el tiempo y que son no deterministas. Así como también, se caracterizan por tener desarrollada una alta tolerancia a fallos. Si se lee con atención la anterior definición, se puede notar que también se ha utilizado (desde una perspectiva de ciencias de la computación) para definir los sistemas distribuidos [9].

Esta similaridad explica la tendencia a emplear cálculos de procesos para modelar sistemas biológicos.

Sería falso afirmar que la mayoría de proyectos de investigación que combinan cálculos de procesos con bioinformática, se inclinan hacia un solo cálculo, o cálculos con las mismas características. En realidad, es amplia la cantidad de cálculos que existen, así como también es amplia la aplicación que se le ha dado a cada uno de ellos.

[1] http://www.mozart-oz.org/

Es más preciso afirmar que existen tres tipos o grupos de cálculos muy utilizados [1], estos son:

- Cálculos de procesos existentes, por ejemplo π y CSP.
- Cálculos inspirados en la biología, como $BioAmbients$, $Brane$, κ.
- Extensión de cálculos existentes para agregar conceptos a los cálculos; estas extensiones se hacen pensando en un problema específico. Entre estos están $Stochastic\text{-}\pi$ y $CCS - R$ (Calculus of Communicating Systems, con retroalimentación).

2.2.1. Problemas biológicos y cálculos de procesos

En [1, 5] se dividen los problemas a resolver en la bioinformática en 3 grupos; relacionados con secuencias, relacionados con la estructura y relacionados con el funcionamiento. En los problemas relacionados con la secuencia se intenta analizar, u obtener información de las secuencias de proteínas o ADN; en esta área no se han utilizado cálculos de procesos como estrategia para proponer una solución porque en estos no hay procesos concurrentes.

Los problemas relacionados con la estructura se enfocan en las proteínas. Se pueden dividir entre predicción de la estructura secundaria o terciaria de una proteína, esto es, predecir la conformación en $3D$ de la proteína a partir de la secuencia; también están los problemas de predicción de interacción o acoplamiento de dos o más proteínas, basándose en la predicción de su estructura secundaria. Para este tipo de problemas se han propuesto soluciones usando el cálculo π y CCP o programación con restricciones [1].

El tercer grupo de problemas, los relacionados con el funcionamiento de las entidades biológicas, representa un conjunto de problemas interesantes desde el punto de vista biológico y computacional. La biología hasta ahora se ha caracterizado por ser una ciencia descriptiva, pero las descripciones de las secuencias de las moléculas de ADN y proteínas, así como de su función no son suficientes para entender, o explicar el porqué, el cuándo y el cómo se transmite la información genética, por qué algunos genes cancerígenos se expresan y otros no, qué condiciones externas sirven como interruptores de dichos genes, por mencionar algunos aspectos.

En la actualidad la comunidad científica se está acercando a la biología sistémica, tratando de observar los entes biológicos como sistemas, con el fin de entender el comportamiento y de observar propiedades emergentes.

En los seres vivos existen sistemas a diferentes niveles. Por ejemplo, una célula por sí sola representa un sistema, porque tiene procesos de alimentación, de respiración, de reproducción, entre otros. Un conjunto de células que conforman un órgano también puede ser visto como un sistema; ellas se comunican, por medio de señales o procesos de control, existe transporte de sustancias entre las células, deben ser vistas en conjunto, entre otras características.

Se espera que en un futuro próximo se pueda entender el funcionamiento de los sistemas biológicos a un nivel mucho mayor; por ejemplo: el sistema

respiratorio, circulatorio, etc. Pero para llegar a este nivel es necesario poder explicar los fenómenos a nivel celular y de conjuntos de células, con el fin poder construir el conocimiento de forma modular, aprovechando que los organismos vivos también son modulares.

En [1, 5] se menciona que existen 3 tipos de tareas que realiza una célula y que explicándolas podríamos entender, o explicar su funcionamiento: 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. La importancia de estas tareas está en que reunen grupos de problemas que son modelados por medio de cálculos de procesos.

Los procesos encargados del control en la célula, se llaman también *Redes de Regulación genética* y están relacionados con los mecanismos biológicos (enzimas) que inhiben o estimulan la expresión de un gen. Se ha utilizado programación por restricciones y el calculo *hcc* [2] para abordar este tipo de problemas [1].

El uso de cálculos de procesos también permite la validación de propiedades por medio de la lógica asociada al cálculo. Esto proporciona una gran ventaja sobre las otras estrategias de solución, puesto que en ocaciones la información biológica no es de libre acceso o es insuficiente para realizar validaciones.

2.3. *C*oncurrent *C*onstraint *P*rogramming

CCP es un enfoque para especificar sistemas concurrentes en términos de restricciones y fué propuesto en [8] por Viyay Saraswat en 1990 como resultado de su tesis doctoral.

A diferencia de los modelos que habían sido propuestos hasta el momento, este modelo hace uso de la lógica y está basado en la comunicación por medio de variables compartidas. *CCP* hace uso de un modelo de memoria distinto al propuesto por Von Neumann, en donde cada variable está ligada a un único valor. Este modelo permite que las variables estén ligadas a un domino, o conjunto de valores que son consistentes con un conjunto de restricciones. En *CCP* las restricciones son el mecanismo que permiten la especificación de modelos usando información parcial.

Una *restricción* es una fórmula de primer orden que representa información parcial sobre las variables compartidas del sistema. Ejemplos de restricciones son:

$$x \geq 42$$
$$x + y \leq 30$$

La información de las variables compartidas reside en un *store*, el cual es representado por la conjunción de todas las restricciones aplicadas a las variables (también es una restricción). El *store* puede ser accedido por *agentes* (procesos

[2] Hybrid Concurrent Constraint Programming

que interactúan con el *store*) con dos operaciones básicas: **ask** y **tell**. Contrario a otros cálculos de procesos, que definen una comunicación punto a punto, *CCP* es un modelo de comunicación con memoria compartida. Intuitivamente, esto implica que un proceso en una comunicación lanza mensajes a todos los otros agentes en el sistema. La figura 2.1 muestra el clásico ejemplo de cuatro agentes interactuando con el *store*. Los agentes de la izquierda le dicen al *store* que la variable x estará instanciada entre los valores 30 y 50. Los agentes de la derecha hacen preguntas, el agente que está más arriba pregunta si x es igual a 50; el agente de la parte inferior pregunta si x está entre 0 y 100.

Figura 2.1: Un *store* CCP con cuatro agentes.

Un agente o proceso es entonces un dispositivo de procesamiento de información que interactúa con su ambiente mediante una restricción compartida que representa la información común conocida [8].

2.3.1. Sintáxis

A continuación se muestra la gramática que permite construir procesos (y sistemas modularmente) en CCP. Tal como se mencionó en la sección anterior, partiremos de un conjunto de definiciones que nos permiten nombrar procesos y restricciones.

Sean $\mathcal{X}$ un conjunto de variables y $\mathcal{C}$ un conjunto de restricciones; el conjunto de términos *CCP*, denotado por $\mathcal{P}$, está definido inductivamente por la sintaxis en el cuadro 2.3.

Los operadores Restricción, Composición Paralela, Sumatoria y Llamado a Procedimiento juegan los mismos roles que sus contrapartes en otros cálculos de procesos. El constructor Tell impone una restricción c en el *store*. El constructor Ask indaga el *store* para saber si la condición c puede ser deducida; si puede, el proceso P es activado, de lo contrario el agente es bloqueado hasta que su

P, Q	$::=$	c	Tell
	$\mid$	$c \rightarrow P$	Ask
	$\mid$	$\exists X.P$	Restricción
	$\mid$	$P \wedge Q$	Composición Paralela
	$\mid$	$P \square Q$	Sumatoria
	$\mid$	$p(X)$	Llamado a Procedimiento

Cuadro 2.3: Sintaxis de *CCP*

condición pueda ser deducida de la información que contiene el *store*.

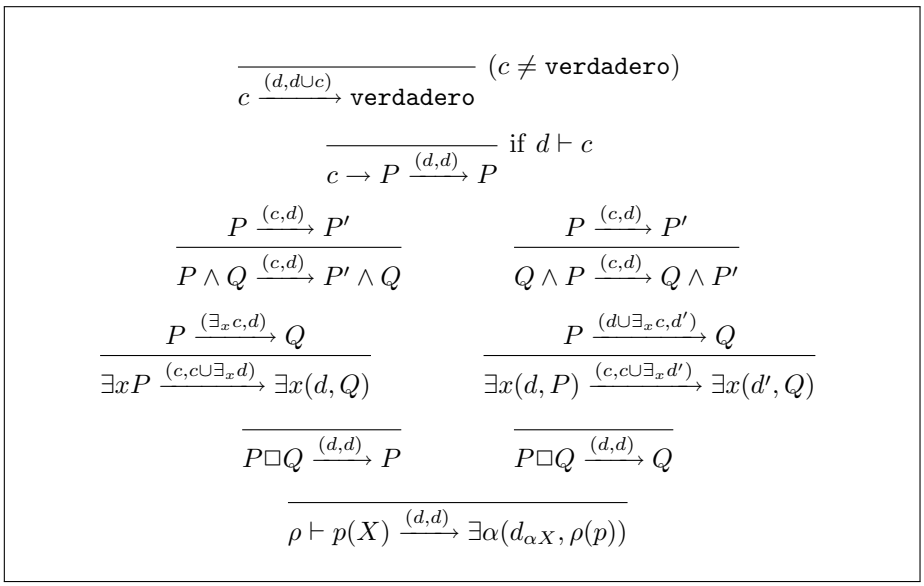
Este cálculo está parametrizado en un *sistema de restricciones* [8] el cual especifica qué tipo de restricciones puede manejar el *store*. Un sistema de restricciones está compuesto por un alfabeto y una teoría. El alfabeto Σ está compuesto por los símbolos de constantes, funciones y relaciones que se pueden usar para construir sintácticamente las restricciones. Las restricciones se construyen de la misma manera que se escriben sentencias en la lógica. La teoría Δ es un conjunto de sentencias escritas con el alfabeto válidas en el sistema de restricciones. Por ejemplo en el sistema de restricciones de dominios finitos:

- Las variables pueden tomar valores en conjuntos finitos de valores pertenecientes a los números naturales, estos son los dominios. Ejemplo X en [1..,10].
- Las operaciones aritmeticas $(+, -, *, \ldots$ etc), las relaciones de igualdad y desigualdad y funciones como potencia, forman parte del alfabeto.
- Los teoremas de la aritmética forman parte de la teoría.

2.3.2. Semántica Operacional

Para expresar el comportamiento operacional de los procesos concurrentes por restricciones, se utiliza la siguiente notación: $\rho \vdash P \xrightarrow{(c,d)} Q$ significa que el agente P con un *store* de entrada c puede, en un paso ininterrumpible, actualizar el *store* a d, y como consecuencia, comportarse como Q (si no es relevante, el "$\rho \vdash$" puede ser omitido). La semántica operacional del lenguaje de procesos está dada en el cuadro 2.4.

Estas reglas afirman lo siguiente: el proceso c adiciona la información de c al *store* compartido d en un solo paso; el proceso $c \rightarrow P$ espera hasta que el *store* contenga al menos tanta información como c, luego se comporta como P; las reglas de transición para $P \wedge Q$ muestran que en una composición paralela, en una sola transición, solo un proceso puede evolucionar, además que muestra que P y Q nunca se comunican de manera síncrona, es decir, todas las comunicaciones toman lugar de manera asíncrona con información agregada por un agente al *store* común para que otro agente la use; las reglas para $\exists x P$ afirman que toda la información sobre x en c está escondida de $\exists x P$, y toda la información de x



Cuadro 2.4: Reglas de Transición de *CCP*

que P produce está escondida del ambiente (similar para agentes con un *store* interno); el proceso $P \Box Q$ se comporta como P ó Q (el proceso escoge uno de ellos no determinísticamente); finalmente, los llamados a procedimientos $p(X)$ son manejados mirando los procedimientos en el ambiente ρ.

La semántica de este cálculo está basada en las observaciones hechas a los procesos. Estas observaciones son el conjunto de *resting points* (puntos de descanso). Un punto de descanso de un proceso es una restricción c tal que si el proceso es ejecutado con c como *store* de entrada, dicho proceso finalmente parará sin producir ninguna información adicional, i.e. el *store* se mantiene intacto(ver [8] para más detalles).

2.3.3. Cálculo tcc

El cálculo `tcc` [29] tiene como objetivo la programación y el modelado de sistemas reactivos (sistemas concurrentes que evolucionan en el tiempo). Este modelo extiende *CCP* con la noción de tiempo e incluye las operaciones de *retraso* (*delay*) y *finalización* (*time-out*). El paralelismo en `tcc` es por *interleaving* (intercalamiento).

En sistemas reactivos, el tiempo es conceptualmente dividido en *intervalos discretos* (o *unidades de tiempo*). En un intervalo de tiempo, un proceso recibe un estímulo del ambiente, computa (reacciona), y responde al ambiente. Gráficamente un sistema reactivo puede verse como en la figura 2.2.

Cada proceso P_i recibe un estímulo i_i y responde con o_i. El estímulo normal-

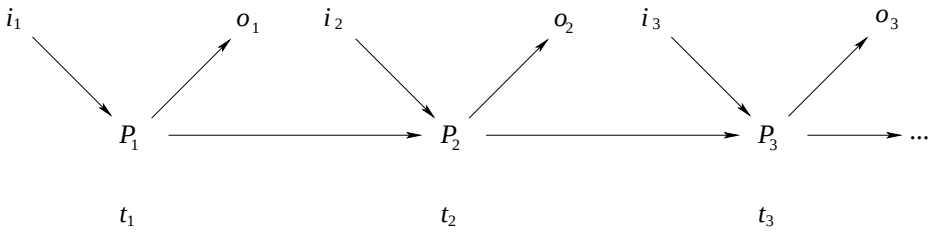


Figura 2.2: Sistema Reactivo

mente es alguna pieza de información usada por el proceso para ejecutar algo. Cuando el proceso está listo (llamado su *punto de descanso*) responde al ambiente con otra pieza de información y evoluciona a otro proceso a ser ejecutado en la siguiente unidad de tiempo. La duración de cada estímulo/respuesta determina la unidad de tiempo t_i. Para el caso de `tcc` el estímulo es una restricción que representa el *store* inicial, y la salida es otra restricción que representa el *store* final (con la información adicional que el proceso haya agregado). La siguiente secuencia ilustra la notación formal de las interacciones estímulo/respuesta entre un ambiente que entra $c_1, c_2, \ldots$ y un proceso que retorna $d_1, d_2, \ldots$ con dichas entradas:

$$P_1 \overset{(c_1,d_1)}{\Longrightarrow} P_2 \overset{(c_2,d_2)}{\Longrightarrow} \cdots P_i \overset{(c_i,d_i)}{\Longrightarrow} P_{i+1} \overset{(c_{i+1},d_{i+1})}{\Longrightarrow} \cdots$$

Aquí el tiempo es un conjunto de puntos que actúan como marcadores que distinguen un intervalo de tiempo del siguiente.

2.4. Cálculo `ntcc`

El cálculo de programación concurrente por restricciones temporal, `ntcc` [3], es un formalismo para sistemas reactivos. El cálculo `ntcc` extiende `tcc` con las nociones de asincronía y no determinismo. La noción de tiempo es introducida por medio de una secuencia de unidades de tiempo, en donde cada unidad tiene su propio almacén de restricciones (*store*) y dura mientras los procesos realizan sus computaciones y añaden (*tell*) o infieren (*ask*) información del *store*.

2.4.1. Semántica Operacional

Los procesos `ntcc` están parametrizados en un *sistema de restricciones*, el cual especifica qué tipo de restricciones puede manejar el *store*. Los procesos $P, Q, \ldots \in Proc$ son construidos como se muestra en el cuadro 2.5.

El proceso **tell**(c) adiciona la restricción c al *store*, dejando a c disponible para los otros procesos en la unidad de tiempo actual. La selección con guarda $\sum_{i \in I}$ **when** c_i **do** P_i, donde I es un conjunto finito de índices, representa un proceso que, en la unidad de tiempo actual, escoge de manera no determinista

$P, Q, \ldots$	::=	$\mathbf{tell}(c)$	Tell
	|	$\sum_{i \in I}$ **when** c_i **do** P_i	Selección con guarda
	|	$P \parallel Q$	Composición Paralela
	|	(**local** x) P	Comportamiento Local
	|	**next** P	Retraso de Unidad
	|	**unless** c **next** P	Vencimiento
	|	$!P$	Replicación
	|	$\star P$	Retraso acotado

Cuadro 2.5: Sintaxis de `ntcc`

un proceso P_j ($j \in I$) cuya guarda c_j es deducida del *store* (las alternativas no escogidas son descartadas). El proceso $P \parallel Q$ representa la composición paralela entre P y Q (en una unidad de tiempo P y Q operan concurrentemente, comunicándose por medio del *store*). El proceso (**local** x) P se comporta como P pero la información de la variable x es local a P, i.e. P no puede ver la información sobre una variable global x y los procesos que no hacen parte de P no pueden ver la información producida por P sobre x. El proceso **next** P ejecuta el proceso P en la siguiente unidad de tiempo. El proceso **unless** c **next** P representa la activación de P en la siguiente unidad de tiempo si y solo si c no puede ser deducida a partir del *store* en la unidad de tiempo actual. El proceso $!P$ ejecuta P en todas las unidades de tiempo a partir de la actual (esta es la versión de retraso del operador replicación del cálculo π). El proceso $\star P$ representa un retraso sin límite pero finito de P, i.e. P finalmente será ejecutado en alguna unidad de tiempo en el futuro.

La semántica operacional está dada por la consideración de las transiciones entre configuraciones γ de la forma $\langle P, d\rangle$, donde P es un proceso `ntcc` y d es un *store*. Las transiciones están divididas en dos: la transición interna $\langle P, c\rangle \rightarrow \langle Q, d\rangle$ denota el proceso P con un *store* c reduciéndose en un paso interno al proceso Q con *store* d; la transición observable $P \stackrel{(c,d)}{\Longrightarrow} Q$ denota el proceso P con entrada c del ambiente, reduciéndose en una unidad de tiempo, al proceso Q y retornando como salida d al ambiente. La transición observable está definida en términos de una secuencia terminal de transiciones internas $\langle P, c\rangle \rightarrow^* \langle Q, d\rangle$ comenzando en P con almacén c y terminando en Q con almacén d. Al igual que en en `tcc` el *store* no se transfiere automáticamente de una unidad de tiempo a la siguiente. El cuadro 2.6 describe las reglas de transición (ver [3] para más detalles).

Ahora explicaremos estas reglas. Un proceso **tell** adiciona información al *store* y termina. La regla SUM dice que un proceso P_j es escogido no determinísticamente para su ejecución entre aquellos cuya guarda c_i pueda ser deducida del *store* actual. Si un proceso P evoluciona a P' entonces la misma transición puede ocurrir si ejecutamos P en paralelo con algún proceso Q (la composición paralela es conmutativa). La regla UNL afirma que no se hace nada en un pro-

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

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

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

UNL: $$\frac{}{\langle \mathbf{unless}\ c\ \mathbf{next}\ P, d\rangle \rightarrow \langle \mathbf{skip}, d\rangle} \text{ si } d \models c$$

LOC: $$\frac{\langle P, c \wedge \exists_x d\rangle \rightarrow \langle P', c'\rangle}{\langle (\mathbf{local}\ x, c)P, d\rangle \rightarrow \langle (\mathbf{local}\ x, c')P', d \wedge \exists_x c'\rangle}$$

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

REP: $$\frac{}{\langle !P, d\rangle \rightarrow \langle P \parallel \mathbf{next}\ !P, d\rangle}$$

STR: $$\frac{\gamma_1 \rightarrow \gamma_2}{\gamma_1' \rightarrow \gamma_2'} \text{ si } \gamma_1 \equiv \gamma_2 \text{ y } \gamma_1' \equiv \gamma_2'$$

OBS: $$\frac{\langle P, c\rangle \rightarrow^* \langle Q, d\rangle \not\rightarrow}{P \xRightarrow{(c,d)} R \equiv F(Q) = \begin{cases} \mathbf{skip} & \text{si } Q = \sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ Q_i \\ F(Q_1) \parallel F(Q_2) & \text{si } Q = Q_1 \parallel Q_2 \\ (\mathbf{local}\ x)F(R) & \text{si } Q = (\mathbf{local}\ x, c)R \\ R & \text{si } Q = \mathbf{next}\ R \text{ or} \\ & \quad Q = \mathbf{unless}\ c\ \mathbf{next}\ R \end{cases}}$$

Cuadro 2.6: Reglas de Transición de `ntcc`

ceso **unless** cuando c es deducida por el *store*. Para un proceso **local**, un *store* "local" c es creado para esconder x del ambiente (toda información sobre x que P produce es escondida al ambiente, y viceversa) pero el *store* al final de la transición es aumentado con las restricciones del *store* local excepto las que incluyan x. La regla STAR modela el hecho de que el proceso P será ejecutado en un futuro (indeterminado). La replicación $!P$ es lograda ejecutando el proceso P en la unidad de tiempo actual y enviando el mismo proceso $!P$ a la siguiente unidad de tiempo. Finalmente, la regla STR afirma que configuraciones estructuralmente congruentes tienen las mismas reducciones.

La regla restante OBS es la que va de una unidad de tiempo a la siguiente. Cuando no hay más transiciones internas (todos los procesos han alcanzado

sus puntos de descanso), los procesos que contienen constructores **next** son programados para la siguiente unidad de tiempo (estos incluyen los definidos por procesos **unless** cuya guarda no puede ser vinculada por el *store* actual). Los otros procesos dependen de la función futuro $F(Q)$ (ver parte inferior del cuadro 2.6).

En `ntcc` lo que pasa dentro de una unidad de tiempo no puede ser observado directamente. Las observaciones de un proceso (lo que un observador puede ver del comportamiento de un objeto) ha sido abstraído de las transiciones internas. Si un proceso corre de la siguiente manera:

$$P = P_1 \xrightarrow{(c_1, c_1')} P_2 \xrightarrow{(c_2, c_2')} P_3 \xrightarrow{(c_3, c_3')} \cdots$$

este proceso puede ser representado como $P \xrightarrow{(\alpha, \alpha')} {}^{\omega}$ donde $\alpha = c_1 \cdot c_2 \cdot c_3 \cdots$ y $\alpha' = c_1' \cdot c_2' \cdot c_3' \cdots$

2.4.2. Semántica Denotacional

La semántica denotacional está dada mediante la consideración de puntos fijos de los procesos, es decir, aquellas secuencias de restricciones donde los procesos pueden ejecutarse sin adicionar más información. La semántica está descrita por las denotaciones en el cuadro 2.7.

Para el proceso **tell** las secuencias en las cuales el proceso puede ejecutarse sin adicionar información son aquellas cuyo primer elemento es más fuerte que c. La ecuación DSUM expresa que las secuencias en las que $\sum_{i \in I}$ **when** c_i **do** P_i puede correr sin agregar información son aquellas cuyo primer elemento implica uno de los c_i (y ellos también deben ser fijos para P_i) o ninguno de los c_i. La ecuación para la composición paralela dice que una secuencia es fija para $P \parallel Q$ si lo es para P y Q. La ecuación semántica para el operador local afirma que si un proceso (**local** x)P con entrada α adiciona nueva información a ella, entonces dicha información no es sobre x en α; pero si P puede ejecutarse con otra entrada α' sin agregar información y α' tiene la misma información que α excepto por aquella sobre x, entonces (**local** x)P puede ejecutarse sobre α sin adicionar ninguna información. El proceso **next**P tiene influencia solo en el sufijo de la secuencia de entrada. El caso del proceso **unless** es similar al de **next** excepto si el primer elemento implica a c. Para el proceso $!P$ una secuencia es fija si todo sufijo de ella es fijo para P. De manera análoga, una secuencia es fija para $\star P$ si existe un sufijo que sea fijo para P.

2.4.3. Aplicaciones

El grupo AVISPA ha utilizado el cálculo `ntcc` para modelar procesos biológicos destacando los beneficios que trae la representación de información parcial, la independencia entre los constructores del cálculo y la demostración de propiedades a través de una lógica temporal (LTL).

DTELL:	$[\![\mathbf{tell}(c)]\!] = \{d.\alpha \mid d \models c\}$
DSUM:	$[\![\sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i]\!] = \bigcup_{i\in I}\{d.\alpha \mid d \models c_i \text{ y } d.\alpha \in [\![P_i]\!]\} \cup \bigcap_{i\in I}\{d.\alpha \mid d \not\models c_i\}$
DPAR:	$[\![P \parallel Q]\!] = [\![P]\!] \cap [\![Q]\!]$
DLOC:	$[\![(\mathbf{local}\ x)P]\!] = \{\alpha \mid \text{existe } \alpha' \in [\![P]\!] \text{ t.q. } \exists_x\alpha' = \exists_x\alpha\}$
DNEXT:	$[\![\mathbf{next}\ P]\!] = \{d.\alpha \mid \alpha \in [\![P]\!]\}$
DUNL:	$[\![\mathbf{unless}\ c\ \mathbf{next}\ P]\!] = \{d.\alpha \mid d \models c\} \cup \{d.\alpha \mid d \not\models c \text{ y } \alpha \in [\![P]\!]\}$
DREP:	$[\![\mathbf{!}P]\!] = \{\alpha \mid \text{para todo } \beta, \alpha' \text{ t.q. } \alpha = \beta.\alpha', \text{ entonces } \alpha' \in [\![P]\!]\}$
DSTAR:	$[\![\star P]\!] = \{\beta.\alpha \mid \alpha \in [\![P]\!]\}$

Cuadro 2.7: Semántica Denotacional de `ntcc`

En [1] Camilo Rueda, Julián Gutierrez y Jorge Perez muestran los resultados de su estudio sobre la utilización de cálculos de procesos para modelar sistemas biológicos; en este documento se resalta la adecuación de un cálculo basado en restricciones para el modelamiento de sistemas biológicos e introducen la utilización de `ntcc` como alternativa para este tipo de modelos porque ofrece la noción de tiempo, no determinismo y asincronía, consideradas como útiles para modelar sistemas biológicos.

Más adelante Julián Gutierrez y Alejandro Arbelaez supervisados por Camilo Rueda realizan su tesis de pregrado [19] en la cual utilizan el cálculo `ntcc` para modelar el Operón de la Lactosa. En este trabajo se proponen un conjunto de procesos en *ntcc* que son útiles para el modelamiento de redes de regulación genética. También los resultados de la simulación de dicho proceso biológico utilizando *ntccSim*, una intérprete de `ntcc` en lenguaje Mozart construida por el grupo AVISPA.

En [2] Frank Valencia, Camilo Rueda, Julián Gutierrez y Jorge Perez utilizaron el cálculo `ntcc` para modelar procesos de transporte a través de la membrana célular. En este artículo se resalta la utilización de `ntcc` para el modelamiento

de reacciones químicas y se muestra un modelo en `ntcc` de la Bomba de Sódio Potasio. En este documento se utiliza también la escogencia no determinista para el modelamiento de etapas o fases en la evolución de un proceso y de replicación para hacer persistentes los procesos que corresponden al funcionamiento de la célula.

Actualmente existen otros trabajos en proceso que exporan la utilización de *ntcc* para el modelamiento de procesos biológicos, destaco el trabajo realizado actualmente por Jesús Aranda en conjunto con Jorge Perez quienes están trabajando en una extención a `ntcc` que permita modelar procesos estocásticos con el fin de dar paso a la inclusión de velocidades de reacción o cantidades a las sustancias presentes en el proceso biológico.

Capítulo 3

Cálculos de procesos inspirados en la biología

En esta sección se muestra una breve descripción de cuatro formalismos que están inspirados en la biología y que sirven como modelos para la especificación de sistemas concurrentes. Estos formalismos comparten una característica en común: el concepto de membrana y su sintáxis permite que un sistema sea especificado como una jerarquía de membranas. De cada uno de estos formalismos se describen: su sintáxis, sus reglas de transición y en especial el concepto de membrana.

3.1. CHAM (CHemical Abstract Machine)

La noción de máquina química abstracta fué propuesta en 1990 por Gérard Berry y Gérard Boudol en [20], con el fin de proponer una máquina abstracta para modelar sistemas concurrentes asíncronos. La intención de los autores es proponer un mecanismo para especificar y modelar sistemas concurrentes que sea tan expresivo como los cálculos de procesos, pero que sea menos rígido para establecer comunicación entre procesos. Los sistemas pueden ser modelados de tal forma que los componentes concurrentes se muevan libremente y se comunican unos con otros cuando hagan contacto. En [20] se resume el estado de un sistema como una *solución química* en la cual flotan *moléculas* las cuales interactúan conforme a un conjunto de *reglas de reacción*, esta idea está basada en la propuesta de Banatre y Le Metayer en [21].

3.1.1. Sintaxis

Una máquina química abstracta C está compuesta por los siguientes elementos:

- **Moléculas:** $m_1, m_2, \ldots, m_n$ en donde cáda molécula m_i es es un término en algún cálculo de procesos.

- **Soluciones:** $S^1, S^2, \ldots, S^n$, en donde S^i es un multiconjunto finito de moléculas.

- **Membranas:** son utilizadas para delimitar las soluciones y poder incluir *subsoluciones*. De esta forma es posible construir sistemas que están organizados como una jerarquía de membranas. Si se desea mostrar el contenido de una solución S entonces se utiliza el operador de membrana para delimitar el conjunto de moléculas/soluciones que la componen como se muestra a continuación:
  Sea S una solución y $m_1, m_2, \ldots m_n, S^1, S^2, \ldots, S^n$ las moléculas y subsoluciones que S contiene, si se desea mostrar el contenido de S de forma explícita entonces se escribe $\{|m_1, m_2, \ldots m_n, S^1, S^2, \ldots, S^n|\}$.

- **Burbujas (*airlock*[1]):** las burbujas están asociadas a las membranas, y son el mecanismo por medio del cual una molécula que se encuentra al interior de una membrana puede reaccionar con otras que se encuentran por fuera de dicha membrana. Existe un operador de burbuja $\lhd$ que se utiliza de la siguiente forma:
  Sean m una molécula y $S = \{|m, m_1, \ldots m_n|\}$ una solución que contiene a m, se escribe $m \lhd S$ para representar la burbuja que contiene a m y que hace parte de S.

3.1.2. Reglas de transformación

- **Reglas de calentamiento/enfriamiento:** estas reglas se utilizan para activar/desactivar las reacciones entre las moléculas que se encuentran en una solución.
  La regla de calentamiento establece que, dada una molécula m de la forma $p|q$ (los procesos p y q actuando en paralelo) si se aumenta la temperatura de m entonces ésta se descompone en p, q, formalmente:

  $$p|q \rightharpoonup p, q$$

  Por el contrario dados p, q si se disminuye la temperatura, entonces se reconstruye m a la forma $p|q$, formalmente.

  $$p, q \rightharpoonup p|q$$

  La regla de enfriamiento es el complemento de la regla de calentamiento

- **Regla de reacción:** las reglas de reacción tienen la siguiente forma:

  $$m_1, m_2, \ldots, m_n \rightarrow m'_1, m'_2, \ldots, m'_n$$

[1] Se utiliza la palabra airlock para nombrar un dispositivo que permite el paso de un objeto entre ambientes con características distintas, tales como presión, temperatura, o cantidad de gases, entre otros.

Las reglas de reacción deben ser las reglas del cálculo de procesos que se utiliza para expresar las moléculas.

Ejemplo: A continuación se muestra un ejemplo tomado de [20] en donde las moléculas son procesos en el cálculo CCS [23] y la regla de reacción es:

$$a.m, \bar{a}.n \rightarrow m, n$$

$$\begin{array}{lcll}
 & \{|\ a,0 \mid \{|\ \bar{a}.p|q\ |\}\ |\} & \\
\rightarrow * & \{|\ a,0,\ \{|\ \bar{a}.p, q\ |\}\ |\} & \text{(calentamiento, membrana)} \\
\rightarrow & \{|\ a,0,\ \{|\ (\bar{a}.p) \lhd \{|\ q\ |\}\ |\}\ |\} & \text{(burbuja)} \\
\rightarrow * & \{|\ a,0,\ \bar{a}.(p, \lhd \{|\ q\ |\}\ )\ |\} & \text{(ion pesado)} \\
\rightarrow * & \{|\ 0\ ,\ \ (p \lhd \{|\ q\ |\}\ )\ |\} & \text{(reacción)} \\
\rightarrow * & \{|\ \{|\ p|q\ |\}\ |\} & \text{(burbuja, enfriamiento)}
\end{array}$$

3.2. BioAmbients

BioAmbients [15] es una extensión al cálculo Ambients [16] en el cual se modela el concepto de *ambiente* como un espacio limitado donde ocurre la computación. El cálculo Ambients es utilizado para modelar compartimientos y movilidad de procesos. La motivación de la extensión es la utilización de BioAmbients para la especificación y simulación de sistemas biológicos. En Ambients se incorpora el concepto de *ambiente* que es el compartimiento en donde residen y evolucionan los procesos. En cambio en BioAmbients se usa el ambiente como una abtracción de las membranas biológicas. Dado que los ambientes (membranas) hacen parte de los conceptos del cálculo, la especificación de membranas hace parte de la sintáxis del cálculo (en ambos cálculos). Es por esto que existen procesos que tienen el constructor de membranas en su especificación. Si se considera que los ambientes alojan procesos, entonces tenemos que la sintaxis de BioAmbients permite la especificación de sistemas como una jerarquía de membranas.

Las membranas de BioAmbients son una abstracción de las membranas de la célula, vistas como un delimitador entre los componentes de la célula. Las membranas además de delimitar, cuentan con un espacio en donde se pueden alojar sustancias que reciben señales para activar procesos al interior de la célula, o para iniciar un proceso de intercambio de sustancias con el exterior, entre otros. En BioAmbients solo se toma en cuenta la función delimitadora de la membrana. Por tanto podríamos decir que las membranas de BioAmbients son unidimensionales, se pueden pintar con una línea.

3.2.1. Sintaxis

BioAmbients contiene al *stochastic π-calculus*, por tanto conserva muchos constructores de dicho cálculo. Los operadores adicionales son aquellos que permiten especificar procesos con membranas y las interacciones entre procesos que

n, m, p			Nombres
π	:==		Acciones
		$\$n!\{m\}$	Salida
		$\$n?\{m\}$	Entrada
$\$$	:==		Direcciones
		local	Dentro del ambiente
		$s2s$	Entre membranas hermanas
		$p2c$	De membrana madre a hija
		$c2p$	De membrana hija a madre
M, N	:==		*Capabilities*
		enter n	Solicitar entrada
		accept n	Aceptar entrada
		exit n	Solicitar salida
		expel n	Aceptar salida
		$merge+\ n$	Solicitar mezcla
		$merge-\ n$	Aceptar mezcla
$P, Q, \ldots$	:==		Procesos
	|	$(\nu\ x)\ P$	Restricción
	|	$P \mid Q$	Paralelo
	|	$!P$	Replicación
	|	$[P]$	Ambientes
	|	$\pi.P$	Prefijo de comunicación
	|	$M.P$	Prefijo de *Capability*
	|	$\Sigma_{i \in I} \pi_i.P_i$	Sumatoria de comunicación

Cuadro 3.1: Sintaxis de BioAmbients.

cuenten con membranas (ya sea que estén al interior o al exterior de la membrana). A continuación se describen los operadores de BioAmbients que tienen que ver con membranas:

n[P] Denota una membrana con nombre n que contiene el proceso P. P puede incluir otras membranas (ver el ejemplo A de la figura 3.1).

enter/accept Son acciones que asociadas a un par de procesos denotan la capacidad de una membrana m de ingresar a una membrana n (que está al mismo nivel que m, ni adentro, ni afuera). Estas acciones son complementarias, es decir para que una membrana m ingrese a una membrana n, esta debe contener al menos un proceso P con la acción *enter*. A su vez n debe contener al menos un proceso R con la acción *accept* (ver el ejemplo B de la figura 3.1).

exit/expel De forma similar a las acciones anteriores, estas acciones son complementarias y denotan la acción de expulsar una membrana m (hija) de otra (madre)(ver el ejemplo C de la figura 3.1).

merge+/merge- Este par de acciones permiten que dos membranas m y n se mezclen (unan) la una con la otra. Las membranas deben estar en el mismo nivel. Al igual que las acciones anteriores *merge+/merge-* son complementarias, por tanto es necesario que una de las membranas, contenga al menos un proceso con la acción *merge+* y que la otra contenga al menos un proceso con la acción *merge-* (ver el ejemplo C de la figura 3.1).

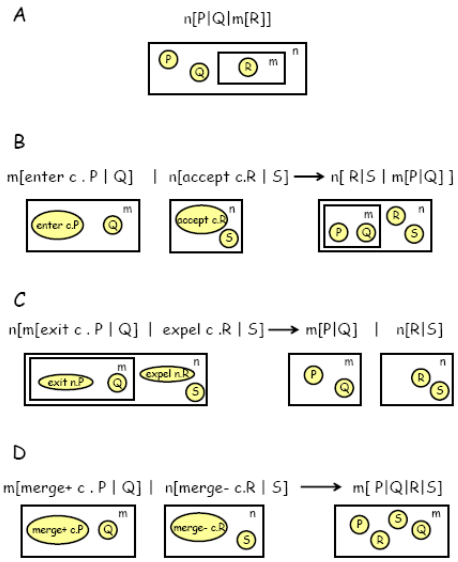


Figura 3.1: Operaciones que modelan interacción entre compartimientos en BioAmbients, tomada de [15]

La sintáxis completa de BioAmbients se encuentra en el cuadro 3.1.

La comunicación en este cálculo es similar a la de π. La comunicación es sincrónica y debe hacerse por medio de canales. Se utilizan nombres para los canales, y a diferencia de π se utilizan los símbolos ? y ! para denotar entrada y salida respectivamente . Otra diferencia se encuentra en la *dirección*. La dirección es una consecuencia de las membranas y permite establecer restricciones en cuanto a los niveles en los cuales es posible establecer la comunicación entre procesos que compartan canales; por tanto cada canal tiene una dirección. Existen tres direcciones para la comunicación, las cuales se distinguen por medio de etiquetas:

local La comunicación es local cuando ocurre entre dos procesos que están en la misma membrana.

s2s Denota la comunicación entre procesos que están en distintas membranas, pero estas membranas están en el mismo nivel (hermanas).

p2c/c2p Este par de etiquetas sirve para realizar comunicación entre dos procesos que se encuentran en membranas distintas, pero que una está dentro de la otra (se llama madre a la más externa e hija a la más interna). Para establecer la comunicación, la membrana madre debe tener un canal con la etiqueta *p2c* y la membrana hija la etiqueta *c2p*.

En el cuadro 3.2 se encuentran las reglas de reducción asociadas a las acciones de comunicación y a las operaciones entra membranas (*Capabilities*)

$[enter\ n.P\|Q]\|[accept\ n.R\|S]$	$\rightarrow$	$[[P\|Q]\|R\|S]$	Ingreso
$[[exit\ n.P\|Q]\|expel\ n.R\|S]$	$\rightarrow$	$[P\|Q]\|[R\|S]$	Salida
$[merge+\ r\ n.P\|Q]\|[T'+merge-\ \ n.R\|S]$	$\rightarrow$	$[P\|Q\|R\|S]$	Mezcla
$local\ n!\{m\}.P\|(local\ n?\{p\}.Q)$	$\rightarrow$	$P\|Q\{p \leftarrow m\}$	Com. Local
$p2c\ n!\{m\}.P\|[c2p\ n?\{p\}.Q\|R]$	$\rightarrow$	$P\|[Q\{p \leftarrow m\}\|R]$	Com. M/H
$[c2p\ n?\{m\}.P\|R]\|p2c\ n!\{p\}.Q$	$\rightarrow$	$[P\|R]\|Q\{p \leftarrow m\}$	Com. H/M
$[R\|T+s2s\ n!m.P]\|[s2s\ n?p.Q+vT'\|S]$	$\rightarrow$	$[R\|P]\|Qp \leftarrow m\|S$	Com. Hnas
$P \rightarrow P \ \Rightarrow\ (new\ n)P \rightarrow (new\ n)Q$			Restricción
$P \rightarrow Q \ \Rightarrow\ [P] \rightarrow [Q]$			Ambiente
$P \rightarrow Q \ \Rightarrow\ P\|R \rightarrow Q\|R$			Paralelo
$P \equiv P',\ P \rightarrow Q,\ Q \equiv Q' \ \Rightarrow\ P' \rightarrow Q'$			$\equiv$

Cuadro 3.2: Reglas de reducción de BioAmbients. En la figura, *M/H* se refiere a Madre/Hija, y *H/M* se refiere a Hija/Madre

Ejemplo

El siguiente ejemplo es tomado de [15]. En él hay tres procesos C_1, C_2 y M.

$$
\begin{array}{lcl}
C_1 & ::= & célula[\ molécula[\ enter\ nuc.D1|D2]\ |\ núcleo[\ accept\ nuc.C]\ ] \\
C_2 & ::= & célula[\ núcleo[\ expel\ nuc.C\ |\ molécula[\ exit\ nuc.D1|D2]\ ]\ ] \\
M & ::= & m1[\ (new\ bb)merge+\ \ D1|D2]\ |\ m2[\ (new\ bb)merge-\ \ D3|D4]
\end{array}
$$

C_1 y C_2 modelan una membrana con etiqueta *célula* que contiene dos membranas *molécula* y *núcleo*, además de un proceso $M1$. En C_1 las membranas *célula* y *núcleo* están al mismo nivel (en paralelo) y cuentan con las acciones *enter/accept* que al reaccionar causan que la membrana *molécula* ingrese a *núcleo*. Esta es la configuración inicial de C_2, en este caso las acciones son *expel/exit* y causan que las dos membranas vuelvan al mismo nivel (configuración de C_1). Este ejemplo se puede ver con claridad en la figura 3.2.

En el proceso M se modelan dos membranas llamadas $m1$ y $m2$ que están en paralelo y contienen las acciones $merge+/merge-$. La reacción de las acciones $merge+/merge-$ causa que los contenidos de $m2$ formen parte de $m1$. La figura 3.3 muestra gráficamente el anterior ejemplo.

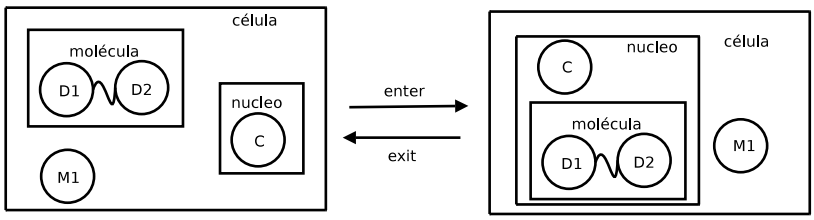


Figura 3.2: Ejemplo de las operaciones de ingreso y salida de membranas

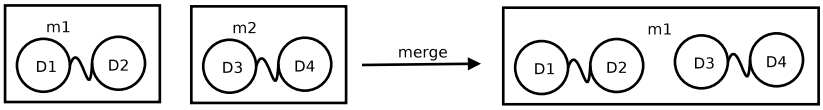


Figura 3.3: Ejemplo de las operaciones de unión de membranas

3.3. Brane

Brane [17] es un cálculo inspirado en las interacciones de la membrana celular. La principal diferencia entre una membrana en *BioAmbients* y una membrana en *Brane* está en que las membranas en Brane son delimitadores pero también contenedores de procesos. Es correcto afirmar que en BioAmbients las membranas son fronteras que encierran procesos sin acciones propias; en cambio una membrana en Brane, es un espacio en donde residen procesos que interactúan con el ambiente.

Brane es una propuesta de Luca Cardelli, quien procura que las membranas de Brane sean análogas a las membranas naturales. En la naturaleza las membranas son dinámicas: permiten el ingreso de partículas, expulsan partículas, pueden partirse o mezlar sus contenidos; estas reacciones siguen ciertas restricciones de polaridad entre los elementos que conforman la membrana; los que están al interior y los que están al exterior de ésta. Cardelli observa en [18] que si un conjunto de membranas anidadas es pintado alternamente con dos colores distintos, esta coloración se mantiene a pesar de las anteriores reacciones. Por esta razón Cardelli llama *bitonales* a estas reacciones. En las figuras 3.4 y 3.5 se puede observar un ejemplo de coloración bitonal y la relación entre la polaridad de las membranas y su contenido.

Figura 3.4: Coloración bitonal, tomada de [18].

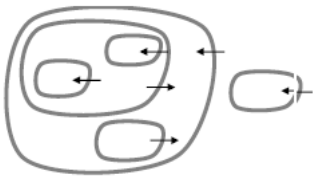


Figura 3.5: Polaridad de las membranas, tomada de [18].

Brane modela las interacciones entre membranas biológicas, por medio de seis reacciones bitonales que hacen parte del cálculo. En este cálculo los procesos se definen en términos de *sistemas* y *membranas*. Un sistema es un conjunto de membranas anidadas, y una membrana es una colección de *acciones*. Dado que las membranas son abstracciones de las membranas biológicas, éstas son espacios que además de contener y proteger el interior de una partícula, sirven principalmente como punto de comunicación. Las *acciones* que componen una membrana en Brane, pueden ser vistas como canales o puertos que activan la interacción entre los elementos del sistema. En la figura 3.6 se muestra la notación gráfica de un sistema compuesto por una membrana que contiene a P (otro sistema); σ es el conjunto de acciones asociado a la membrana.

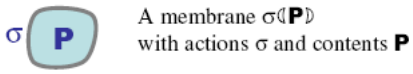


Figura 3.6: Notación gráfica de un sistema, tomada de [17].

Las operaciones entre membranas que provee el cálculo son seis; tres de ellas están inspiradas en la *endocitosis* y en la *exocitosis* que son los procesos por medio de los cuales una partícula ingresa y sale de una membrana respectivamente. Las siguientes tres operaciones están inspiradas en la división celular; la notación gráfica de las operaciones del cálculo se muestran en las figuras 3.7 y 3.8.

3.3.1. Sintaxis

A diferencia de los anteriores cálculos de procesos, para especificar un modelo en el cálculo Brane no se utiliza la palabra *proceso*. En Brane la especificaciónse hace en terminos de: **sistemas** y **membranas**. En el cuadro 3.3 se encuentra la sintaxis del cálculo Brane.

También en el cuadro 3.3 se puede observar que el autor del cálculo Brane utiliza operadores distintos en la sintaxis de los sistemas y membranas. En el caso de los sistemas se utiliza $\circ$ y $\diamond$ para la composición paralela y vacío respectivamente. En las membranas se utiliza ! para composición paralela y 0 para vacío.

P, Q	$::=$	$\diamond \mid P \circ Q \mid !P \mid \sigma(\!\lvert P \rvert\!) \mid m$	(Sistemas)
σ, τ	$::=$	$0 \mid \sigma \mid \tau \mid !\sigma \mid a.\sigma$	(Membranas)
$a ::=$	$::=$	$\leftthreetimes_n \mid \leftthreetimes^{\perp}_n(\sigma) \mid \rightthreetimes_n \mid \rightthreetimes^{\perp}_n \mid \circledcirc(\sigma) \mid$ $mate_n \mid mate^{\perp}_n \mid bud_n \mid bud^{\perp}_n(\sigma)$ $\mid drip(\sigma) \mid p(q) \rightrightarrows p'(q')$	(Acciones)

Cuadro 3.3: Sintáxis del cálculo Brane.

Sistemas

Un sistema en el cálculo Brane puede ser construido utilizando: vacío ($\diamond$), composición paralela ($\circ$), replicación (!) o una membrana σ que encierra un sistema ($\sigma(\!\lvert P \rvert\!)$). En el item *Sistemas* del cuadro 3.3 se puede observar que un sistema también puede ser una mólecula m. Las moléculas son nombres que se utilizan para modelar aquellas sustancias que ingresan o que son expulsadas de una membrana cuya estructura interna no es considerada. De manera formal se dice que m hace parte del conjunto de nombres de moléculas M ($m \in M$).

Membranas

Las membranas (también llamadas *branes*) son colecciones de *acciones* y encierran a los sistemas. Estas pueden ser construidas usando vacío (0), composición paralela de membranas ($\sigma \mid \tau$), replicación (!) y secuenciación de una acción y una membrana ($a.\sigma$).

Acciones

Las acciones están divididas en tres conjuntos: *phago/exo/pino*, *mate/bud/drip* y *Bind&Release*. El primer conjunto modela el ingreso y egreso de sistemas a una membrana. El segundo conjunto modela la unión de dos membranas y también el desprendimiento de una membrana a partir de otra. El tercer conjunto en realidad se refiere a una sola pareja de acción y coacción que es requerida para modelar el ingreso y egreso de moléculas a una membrana.

phago/exo/pino. El conjunto de acciones *phago/exo/pino* está compuesto por las acciones y co-acciones inspiradas en los procesos celulares que permiten el ingreso y egreso de sustancias a la célula a través de la membrana celular. Estos procesos son *endocitosis* y *exocitosis*. La *endocitosis* puede ser de dos tipos: *fagocitosis* cuando la sustancia que ingresa es un sólido y *pinocitosis* cuando la sustancia es líquido. El ingreso de cualquier sustancia a través de la membrana implica que ésta sustancia ingresa envuelta en una burbuja o vesícula construida a partir de una porción de la membrana celular con el fin de conservar las propiedades del interior de la célula y del medio en que se encontraba dicha sustancia.

El conjunto de acciones *phago/exo/pino.* está comprendido por una pareja de acción y coacción que modela fagocitosis, una pareja de acción y coacción que modela exocitosis y una acción que modela pinocitosis. La correcta utilización de estos términos de la biología en la propuesta de Luca Cardelli en [17] es una inquietud que surge cuando se estudia el cálculo Brane. Para resolver este interrogante se consultó [24] (un texto de biología molecular), en donde se encuentra que los términos y conceptos utilizados por Luca Cardelli son acordes con la biología. En la figura 3.7 se muestra la notación gráfica de este conjunto de acciones.

phago La pareja *phago* está compuesta por una acción y su complemento y dado que está inspirada en el proceso de fagocitosis, estas acciones representan el ingreso de una membrana envuelta en una burbuja(que también es una membrana) a otra membrana.
La acción se representa con el símbolo $\circlearrowleft_n$ que muestra una membrana y una flecha hacia adentro; la coacción se representa con el símbolo $\circlearrowleft^{\perp}_n$. En ambas acciones aparece n que debe ser un nombre, este nombre debe ser compartido por la acción y el complemento para que exista una reacción, similar a las operaciones de membrana de BioAmbients.

exo De forma similar al anterior, se compone de una pareja de acción y coacción. Estas acciones están inspiradas en la exocitosis, por tanto modelan la salida de un sistema (que está envuelto en una membrana) de una membrana. Se utilizan los símbolos $\circlearrowleft_n$ y $\circlearrowleft^{\perp}_n$ para la acción y coacción respectivamente. Al igual que en la pareja anterior, n es un nombre y para que una pareja acción-coacción puedan reaccionar deben compartir el mismo n.

pino Comprende una acción que no requiere complemento. Esta acción permite el ingreso de una membrana que contiene el sistema vacío. Esta acción se representa con el símbolo $\circledcirc(\rho)$ en donde σ (parámetro) es la membrana que ingresa.

mate/bud/drip. Este conjunto de acciones puede ser derivado del conjunto *phago/exo/pino* y está inspirado en los procesos de división celular. Está compuesto de dos parejas de acción-coacción y una acción que no requiere coacción. En general estas operaciones modelan la unión/desprendimiento de membranas. Todas estas acciones requieren de un n (nombre) que permite la sincronización de cada pareja acción-coacción. En la figura 3.8 se muestra la notación gráfica de este conjunto de acciones.

mate Es una pareja de acción-coacción que modela la unión de dos membranas. Cuando dos membranas se unen, forman una nueva membrana que contiene los sistemas de las anteriores y que tiene en paralelo las acciones restantes de ambas membranas. Se utiliza $mate_n$ y $mate^{\perp}_n$ para representar la acción y la coacción respectivamente.

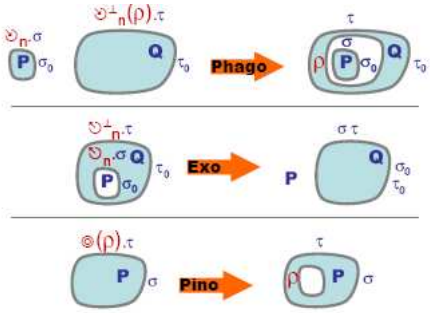


Figura 3.7: Notación gráfica de las operaciones de Brane inspiradas en la endo y exocitosis. Tomada de [17].

bud La pareja de acciones que comprende *bud* modela el desprendimiento de una vesícula de una membrana. Estas acciones se utilizan para expulsar un sistema que ya está envuelto en una membrana, al interior de una nueva membrana. Se utiliza bud_n y $bud_n^{\perp}(\sigma)$ para la acción y coacción. En el caso de la coacción, σ será la nueva membrana que envuelve el sistema que se desprende de la membrana acual.

drip La acción *drip* no necesita coacción y modela el desprendimiento de una membrana(nueva) con contenido vacío. Se usa $drip_n(\sigma)$ en donde σ contiene las acciones de la nueva membrana.

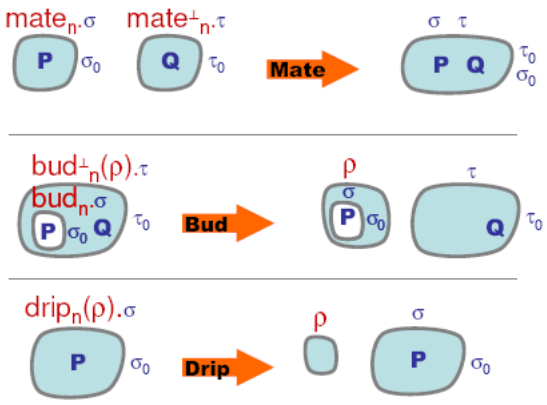


Figura 3.8: Notación gráfica de las operaciones de Brane inspiradas en la división celular. Tomada de [17].

Bind&Release. La acción *Bind&Release* es representada por la construcción de símbolos $p(q) \rightrightarrows p'(q')$ en donde p y q son las moléculas que se desean ligar

y p' y q' son las moléculas que se liberan.

3.3.2. Reglas de reducción

Las reglas de reducción del cálculo Brane son aquellas que están asociadas a cada pareja de acción-coacción y unas reacciones básicas entre sistemas y membranas al estilo del cálculo π. En el cuadro 3.4 se encuentran las reglas asociadas a las acciones del cálculo Brane y en el cuadro 3.5 están las reglas básicas.

$P \to Q \quad \Rightarrow \quad P \circ R \to Q \circ R$	Paralelo
$P \to Q \quad \Rightarrow \quad \sigma(\!\!\mid P \mid\!\!) \to \sigma(\!\!\mid Q \mid\!\!)$	Ambiente
$P \equiv P',\ P \to Q,\ Q \equiv Q' \quad \Rightarrow \quad P' \to Q'$	$\equiv$

Cuadro 3.4: Reglas de reducción básicas.

$\circlearrowleft_n.\sigma(\!\!\mid P \mid\!\!) \circ \circlearrowleft_n^{\perp}(\rho).\tau(\!\!\mid Q \mid\!\!)$	$\to$	$\tau(\!\!\mid \rho(\!\!\mid \sigma(\!\!\mid P \mid\!\!) \mid\!\!) \circ Q \mid\!\!)$	(phago)
$\circlearrowleft_n^{\perp}.\tau(\!\!\mid \circlearrowleft_n.\sigma(\!\!\mid P \mid\!\!) \circ Q$	$\to$	$P \circ \tau \mid \sigma(\!\!\mid Q \mid\!\!)$	(exo)
$\circledcirc(\rho).\sigma(\!\!\mid P \mid\!\!)$	$\to$	$\sigma(\!\!\mid \rho(\!\!\mid \ \mid\!\!) \circ P \mid\!\!)$	(pino)
$mate_n.\sigma(\!\!\mid P \mid\!\!) \circ mate_n^{\perp}.\tau(\!\!\mid Q \mid\!\!)$	$\to$	$\sigma \mid \tau(\!\!\mid P \circ Q \mid\!\!)$	(mate)
$bud_n^{\perp}(\rho).\tau(\!\!\mid bud_n.\sigma(\!\!\mid P \mid\!\!) \circ Q \mid\!\!)$	$\to$	$\rho(\!\!\mid \sigma(\!\!\mid P \mid\!\!) \mid\!\!) \circ \tau(\!\!\mid Q \mid\!\!)$	(bud)
$drip(\rho).\sigma(\!\!\mid P \mid\!\!)$	$\to$	$\rho(\!\!\mid \ \mid\!\!) \circ \sigma(\!\!\mid P \mid\!\!)$	(drip)
$p \circ p(q) \rightrightarrows p'(q').\sigma(\!\!\mid q \circ P \mid\!\!)$	$\to$	$p' \circ \sigma(\!\!\mid q' \circ P \mid\!\!)$	(B&R)

Cuadro 3.5: Reglas de reducción de las acciones.

Ejemplo

A continuación se muestra un ejemplo tomado de [14] en el cual se modela la ruta endocítica del colesterol. De forma resumida, el proceso que se desea modelar puede ser visto como el mecanismo del que dispone una célula para capturar partículas de colesterol de baja densidad, llevarlas a su interior y luego procesarlas al interior de sus organelas (endosoma y lisosoma). Para más información sobre el proceso biológico modelado ver el cápitulo 5. En el cuadro 3.6 se puede observar la configuración en dos dimensiones del modelo. En el cuadro 3.7 está la descripción detallada de cada sistema y de cada membrana.

El modelo está compuesto por un sistema que tiene una molécula ($LDLpart$) y una membrana que representa a una célula, ambas en paralelo. La membrana tiene tres acciones dispuestas de forma secuencial. La primera acción es de la forma *Bind&Release* y da a lugar la captura de la molécula ($LDLpart$). La segunda acción modela el ingreso de la molécula a la célula y está dada por una acción de la forma *pino* que permite el ingreso de una membrana vacía con

$LDLpart \circ \begin{array}{l} LDLpart(\diamond) \rightrightarrows \diamond(\diamond).\textcircled{o}(ldlRecPart).\ \circlearrowright^{\perp}_{rec} \\ \boxed{RlateEndo \circ RLysosome} \end{array}$

Cuadro 3.6: Vista gráfica del modelo de la ruta endocítica de la LDL del colesterol

una acción dada por *ldlRecPart* (ver cuadro 3.7). El proceso biológico modelado incluye la unión de la membrana que acaba de ingresar con un endosoma tardío, así como también el producto de esta unión con el lisosoma. Lo anterior es modelado por medio de las acciones de la membrana que se produce al ejecutar la acción *pino* en conjunto con las acciones de las membranas de los procesos *RLateEndo* y *RLysosome* del cuadro 3.7, los cuales modelan el endosoma tardío y el lisosoma de la célula.

$System$	$::=$	$LDLpart \circ ldlReceptor(\!\|RLateEndo \circ RLysosome\|\!)$
$RLateEndo$	$::=$	$mate^{\perp}_{le}(\!\|acid\|\!) \circ RLateEndo$
$RLysosome$	$::=$	$mate^{\perp}_{lyso}(\!\|DestroyEnz\|\!) \circ RLysosome$
$ldlReceptor$	$::=$	$LDLpart(\diamond) \rightrightarrows \diamond(\diamond).\textcircled{o}(ldlRecPart).\ \circlearrowright^{\perp}_{rec}$
$ldlRecPart$	$::=$	$mate_{le} \mid$
		$\diamond(acid) \rightrightarrows \diamond(LDLpart).drip(recVesicle).mate_{lyso}$
		$\mid \diamond(DestroyEnz \circ LDLpart) \rightrightarrows \diamond(LDLconstituents)$
$recVesicle$	$::=$	$\circlearrowright_{rec}.ldlReceptor$

Cuadro 3.7: Modelo de la ruta endocítica de la LDL del colésterol usando Brane

3.4. Sistemas de Membranas - *P-Systems*

Se llama sistema de membranas o *P-Systems* a un modelo computacional propuesto por G. Paun en [27] que está inspirado en la forma en que las células procesan compuestos químicos a través de su estructura de membranas o compartimientos. Se debe resaltar que al igual que *CHAM*, los *P-Systems* no son un cálculo de procesos y no fueron propuestos con el único propósito de modelar sistemas biológicos. En realidad la idea del autor es hacer uso de este modelo computacional para modelar distintos tipos de problemas en los que haya concurrencia. De forma similar a los algoritmos genéticos, o las redes neuronales, *P-Systems* es un modelo inspirado en la naturaleza que puede ser utilizado para dar solución a problemas considerados difíciles.

Tal como su nombre lo indica, este modelo está inspirado en la estructura de la célula que está dividida en compartimientos o membranas, pero vistas "desde los ojos de las matemáticas"[28]. Esto último quiere decir que se percibe la estructura celular como una configuración jerárquica de membranas, en donde

se encuentra una membrana más externa o principal, regiones o espacios al interior de las membranas, membranas anidadas y membranas elementales, que son aquellas que no contienen otras membranas. En la figura 3.9, tomada de [28], se encuentra la abstracción de la estructura de membranas de una célula que se toma como referencia en *P-Systems*.

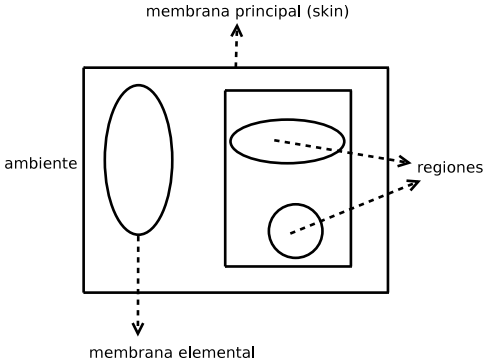


Figura 3.9: Estructura de una membrana

La idea principal de *P-Systems* es resolver un problema por medio de un mecanismo computacional concurrente que evoluciona dando lugar a la solución de dicho problema. El modelo propuesto para la solución de un problema debe incluir la definición de los siguientes componentes:

- Una configuración jerárquica de membranas, en donde se llama *región* al espacio encerrado por una membrana. Esta configuración puede cambiar durante la evolución del problema. En los *P-Systems* clásicos los cambios son producidos por la destrucción de una membrana, en ningún momento se crean nuevas membranas, tampoco ocurre unión de membranas. Existe una variante llamada *P Systems con membranas activas* que sí permite estos cambios en la estructura y modela disolución de una membrana, unión, endocitosis, exocitosis y desprendimiento de membranas, similar al cálculo Brane.

- Un conjunto de *objetos* que serán localizados en las regiones. Los objetos son nombres y representan objetos del problema. La utilización de objetos está inspirada en las moléculas o compuestos químicos que hacen parte de las reacciones y que se mueven a través de las membranas.

- Un conjunto de *reglas de evolución* que permiten que los objetos se transformen (se conviertan en otros objetos), o se muevan a través de las membranas. Las reglas de evolución son las que permiten que la configuración inicial de la membrana cambie.

Todas las reglas de evolución que puedan ser aplicadas deben hacerlo de

forma concurrente. Si hay competencia entre dos o más reglas la escogencia es no determinista.

3.4.1. Sintaxis

No existe una gramática que dictamine la construcción de un *P-System*, pero sí existen algunas reglas sencillas que deben seguirse para definir un *P-System*. Formalmente un *P-System* está definido así:

$$\pi = (V, \mu, M_0, \ldots, M_n, R_0, \ldots, R_n, i_0)$$

En donde:

- V es un alfabeto que contiene los símbolos que dan a lugar a los objetos. Los objetos son cadenas en el alfabeto V. Se puede también definir un conjunto de símbolos C que contiene a aquellos objetos que hacen el papel de catalizadores, o inducen una reacción química.

- μ es una configuración jerárquica de membranas, la configuración inicial consta de n membranas que están etiquetadas. Para describir esta configuración se utilizan corchetes [ ] para representar una membrana. Esta configuración no incluye la localización de objetos en cada una de las regiones. Todas las configuraciones deben tener una membrana principal llamada *skin* la cual es la membrana más externa de la configuración. Ejemplo de una configuración:

  $$[\,[\,[\,]_{core}]_{cell}]_{skin}$$

  En esta configuración se modela una membrana con la etiqueta *cell* que contiene otra membrana con la etiqueta *core*, todo está incluido en la membrana principal que tiene la etiqueta *skin*. Por lo general las membranas se etiquetan con números naturales reservando el cero para la membrana más externa o *skin*.

- M_i $0 \leq i \leq n$ son multiconjuntos finitos de cadenas en el alfabeto V (o $V \cup C$, según sea el caso) y que representan los objetos que se localizan en cada una de las regiones (interior de cada membrana). Cada M_i corresponde al multiconjunto de objetos de la región i. Ejemplo:

  $$\begin{gathered}\text{Dados } V = \{A, B, C\} \text{ y la configuración: } [\,[\,[\,]_2]_1]_0 \\ M_0 = \{A, AB\} \\ M_1 = \{B, BB\} \\ M_2 = \{C, BC\}\end{gathered}$$

  Aquí se define una configuración como la del ejemplo anterior usando números como etiquetas en donde 0 es la membrana más externa y 2 la más interna. De las definiciones del ejemplo se tiene que hay dos objetos en cada una de las regiones; A, AB en la región encerrada por la membrana 0, B, BB en la 1 y C, BC en la 2.

- R_i $0 \leq i \leq n$ son conjuntos de reglas de evolución asociadas a cada región $0, 1, 2, \ldots, n$ de la configuración inicial μ. Los diferentes tipos de reglas de evolución se explican más adelante.

- i_0 representa la *membrana de salida* de π. i_0 es un número como tal y especifica la membrana que debe observarse como resultado de la evolución de π. La evolución de un *P-System* π está dada por la aplicación de las reglas, cuando ninguna regla pueda ser aplicada se dice que la computación ha terminado. i_0 es un número entre 1 y n (cantidad de membranas), es posible también que i_0 tome el valor de ∞ que quiere decir que la membrana que contiene el resultado de la computación es la membrana principal o *skin*.

3.4.2. Reglas de evolución

La computación está dada en términos de las reglas de evolución. Estas reglas están también inspiradas en los mecanismos que permiten el paso de sustancias o membranas a través de las membranas celulares. Las reglas de evolución que se muestran a continuación son aquellas que son comunes a todas las variantes de *P-Systems* o *P-Systems* clásicos.

Regla de transformación (o re-escritura).
Son reglas de la forma:

$$u \rightarrow v \text{ en donde } u, v \in V^+.$$

Existen tres variantes de la regla de transformación:

- Cuando u y v son de la forma $u = xc$, $v = yc$ la regla se llama cooperativa, y c es un objeto *catalizador* que induce la reacción; si el conjunto C (objetos catalizadores) fué definido entonces $c \in C$.

- Cuando v es de la forma $v = v'\delta$ la regla se llama de *disolución* y la membrana que encierra la región en donde la regla es aplicada desaparece de la configuración y los objetos residentes en dicha membrana pasan a formar parte del conjunto de objetos de la membrana de nivel superior. La membrana principal no puede ser disuelta.

- En ocasiones se puede añadir a la reacción la localización del resultado. Cuando esto ocurre se puede añadir las palabras *here, in, out*, o especificar directamente la etiqueta de la membrana en que ocurre el resultado.

Reglas de unión y descomposición.
Las reglas de unión y descomposición son de la forma.

$$u, v \rightarrow u\&v \quad \text{(Unión)}$$
$$u\&v \rightarrow u, v \quad \text{(Descomposición)}$$

$$u, v \rightarrow u\&v$$
$$u\&v \rightarrow u, v$$

La regla de unión modela la unión de dos objetos. Se utiliza para modelar la conformación de un compuesto a partir de dos sustancias. La regla de descomposición es inversa y modela la separación de las sustancias que hacen parte de un compuesto.

Reglas de transporte.
Las reglas de transporte están divididas en dos, *Antiport* y *Symport*. La palabra *antiport* es utilizada para designar a una proteína de transporte que está ubicada en la membrana celular y que mueve simultáneamente, a través de la membrana, a dos moléculas diferentes en direcciones opuestas. Se llama *symport* una proteína como la anterior que realiza el movimiento de dos moléculas distintas en la misma dirección. Estas reglas son de la forma:

$$u[_i v]_i \rightarrow v[_i u]_i \quad \text{(Antiport)}$$
$$[_i u, v]_i \rightarrow u, v[_i \ \ ]_i \text{ ó } u, v[_i \ \ ]_i \rightarrow [_i u, v]_i \quad \text{(Symport)}$$

Ejemplo

El siguiente ejemplo fué tomado de [14] y modela la ruta endocítica del colesterol (el proceso biológico de la sección anterior). De forma resumida, el proceso que se desea modelar puede ser visto como el mecanismo del que dispone una célula para capturar partículas de colesterol de baja densidad, llevarlas a su interior y luego procesarlas al interior de sus organelas (endosoma y lisosoma).

El *P-System* que modela este proceso biológico es el siguiente:

$$\pi = (V, \mu, M_0, \ldots, M_3, R_0, \ldots, R_3, i_0)$$

En donde:

- El alfabeto es $V = \{S, LDLpart, LDLRcpt, Clth, LateEndo, Chol\}$
- La configuración de la estructura $\mu = [_0[_1[_2[_3 \ \ ]_3]_2]_1]_0$. Esta estructura puede verse de forma gráfica en el cuadro 3.8.
- Multiconjuntos de Objetos en las regiones:
  - $M_0 = \{LDLpart\}$
  - $M_1 = \{LDLRcpt\}$
  - $M_2 = \{Clht, LateEndo\}$
  - $M_3 = \emptyset$
- Reglas de evolución para cada region:

  $R_0 = \{LDLpart + LDLrcpt \rightarrow Ldlpart\&LDLRcpt\}$

  $R_1 = \{R_{1,1}, R_{1,2}, R1, 3\}$, en donde

$$
\begin{aligned}
R_{1,1} = \quad & [_1 LDLRcpt]_1 \rightarrow LDLRcpt[_1 \ \ ]_1] \\
R_{1,2} = \quad & LDLpart\&LDLRcpt[_1 \ \ ]_1 \rightarrow [_1 LDLpart\&LDLRcpt]_1 \\
R_{1,3} = \quad & LDLRcpt[_1 \ \ ]_1 \rightarrow [_1 LDLRcpt]_1\}
\end{aligned}
$$

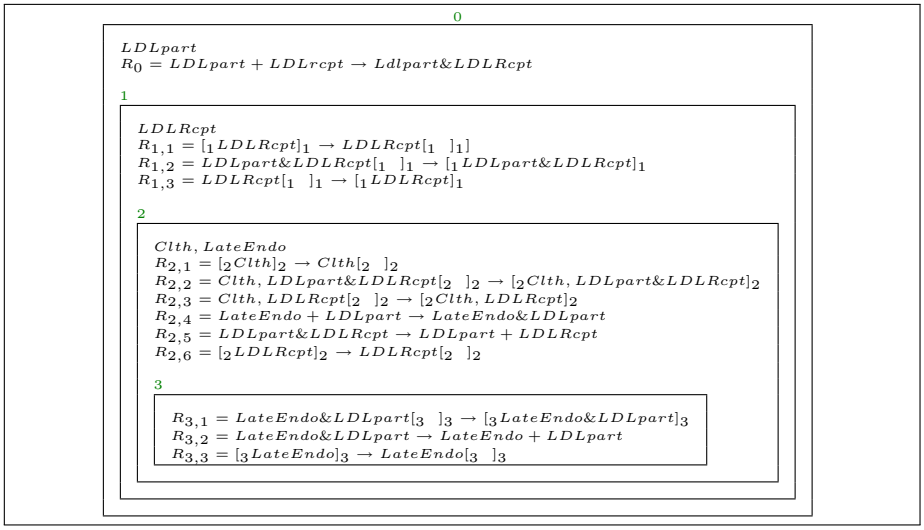
$R_2 = \{R_{2,1}, R_{2,2}, R_{2,3}, R_{2,4}, R_{2,5}, R_{2,6}\}$, en donde

$$
\begin{aligned}
R_{2,1} = \quad & [_2 Clth]_2 \rightarrow Clth[_2 \ \ ]_2 \\
R_{2,2} = \quad & Clth, LDLpart\&LDLRcpt[_2 \ \ ]_2 \rightarrow [_2 Clth, LDLpart\&LDLRcpt]_2 \\
R_{2,3} = \quad & Clth, LDLRcpt[_2 \ \ ]_2 \rightarrow [_2 Clth, LDLRcpt]_2 \\
R_{2,4} = \quad & LateEndo + LDLpart \rightarrow LateEndo\&LDLpart \\
R_{2,5} = \quad & LDLpart\&LDLRcpt \rightarrow LDLpart + LDLRcpt \\
R_{2,6} = \quad & [_2 LDLRcpt]_2 \rightarrow LDLRcpt[_2 \ \ ]_2
\end{aligned}
$$

$R_3 = \{R_{3,1}, R_{3,2}, R_{3,3}\}$, en donde

$$
\begin{aligned}
R_{3,1} = \quad & LateEndo\&LDLpart[_3 \ \ ]_3 \rightarrow [_3 LateEndo\&LDLpart]_3 \\
R_{3,2} = \quad & LateEndo\&LDLpart \rightarrow LateEndo + LDLpart \\
R_{3,3} = \quad & [_3 LateEndo]_3 \rightarrow LateEndo[_3 \ \ ]_3
\end{aligned}
$$

- Membrana de salida $i_0 = 3$, esto quiere decir que el resultado debe estar en M_3



Cuadro 3.8: Modelo de la ruta endocítica de la LDL del colésterol usando *P-Systems*

Capítulo 4

Modelando sistemas de membranas en `ntcc`

En este capítulo se propone un método que permite describir sistemas biológicos en `ntcc` de forma similar a los formalismos que modelan membranas (Sistemas de Membranas) descritos en el capítulo 3. Inicialmente se realizó una comparación de la forma en que se modelan sistemas en Brane y BioAmbients, con respecto a `ntcc`, y en general con cualquier derivado de CCP. Se encontraron tres diferencias importantes:

- **El mecanismo de comunicación entre procesos:** en CCP la comunicación es asíncrona y se hace por medio del *store* que es visible para todos los procesos. Los estímulos o entradas están dados por medio de restricciones que los procesos pueden imponer o consultar. En BioAmbients/Brane, la comunicación es síncrona y requiere de la presencia de acciones y coacciones, o de canales de comunicación en los procesos.
- **La forma en que se construyen los sistemas:** en BioAmbients/Brane las membranas son contenedores de procesos que pueden tener otras membranas al interior (anidamiento), por tanto los sistemas en estos cálculos pueden ser vistos como jerarquías de membranas. En CCP los sistemas pueden ser especificados de forma modular, pero no existen jerarquías entre ellos.
- **Movilidad:** los cálculos de membranas proveen un conjunto de operaciones que permiten modelar el ingreso/salida de membranas; esto puede realizarse gracias a que existe movilidad como concepto del cálculo. En CCP no hay movilidad como concepto del cálculo y aunque es posible modelar la movilidad como está propuesto en [26], no es posible modelar el mismo comportamiento que en BioAmbients y Brane.

Las anteriores diferencias explican las dificultades que se presentan para modelar las operaciones de membranas con los constructores de `ntcc`. Pero esto

no quiere decir que no se puedan modelar sistemas biológicos teniendo en cuenta la localización de sustancias y procesos (al estilo de membranas). Para ello es necesario definir un conjunto de procesos y variables que hagan posible modelar el comportamiento de las operaciones de membranas y especificar sistemas en los cuales exista una jerarquía de membranas en `ntcc`.

A continuación se presentan dos técnicas que son claves para la modelación de sistemas de membranas: El modelamiento de membranas, su localización y jerarquía (ver sección 5.1), y el modelamiento de reglas de evolución, las cuales son el mecanismo de interacción entre los objetos situados en las membranas (ver sección 5.2).

4.1. Modelando Membranas, Localización y Jerarquías

Los procesos y definiciones propuestos a continuación están inspirados en la forma en que se modelan las membranas con *CHAM* y *P-Systems clásicos* (sin membranas dinámicas) principalmente porque son alternativas en las que no hay concepto de movilidad y en donde la evolución está dictaminada por reglas inspiradas en las reacciones químicas.

Se debe resaltar que en los acercamientos a la utilización de `ntcc` para la modelación, simulación y verificación de sistemas biológicos de la sección anterior se encuentra una constante y es la utilidad de `ntcc` para modelar reacciones químicas, que representan la transformación de una sustancia en otra y la precencia de catalizadores para inducir una reacción. `ntcc` agrega además la posibilidad de incluir al modelo la noción de tiempo y establecer una secuencia en la dinámica del modelo.

Para la especificación de sistemas al estilo de membranas se utilizará un sistema de restricciones sobre dominios finitos. La ubicación de los elementos del sistema se realizará por medio de variables. La estructura del sistema es estática, los contenidos de las membranas son modificados debido al paso de sustancias a través de las membranas (entrada y salida de sustancias), pero no hay desplazamiento de membranas como tal. Una membrana en este enfoque, es una frontera que encierra una región llamada **nivel** y que tiene asociado un proceso `ntcc` que modela las acciones de dicho nivel. Los niveles que conforman un modelo pueden variar dependiendo de la descripción del proceso biológico que se desea modelar.

Al interior de cada nivel se encuentra ubicado un conjunto de objetos que representan las sustancias presentes en la descripción biológica del sistema. La ubicación de cada uno de estos objetos (y de los compuestos que conforman), está dada por una variable declarativa que puede tomar valores en los naturales, y denota la cantidad de cierto objeto (y compuesto) en un nivel dado.

4.1.1. Sintaxis

Para especificar un modelo de un sistema biológico usando sistemas de membranas en `ntcc` es necesario definir los elementos que se describen a continuación.

- Un conjunto de Nombres de variables que representan todos los objetos del sistema, tanto los simples como los compuestos. Los dominios de estas variables son por defecto $D_i = [0...sup]$ en donde sup es un entero muy grande, asumiendo que no hay límite en la cantidad de unidades que pueden residir en un nivel dado. Los límites en las cantidades de unidades deben ser explícitamente especificados.

- Una configuración de las membranas o niveles, esta configuración no tiene una notación explícita pero es recomendable usar los corchetes[ ] para representar membranas como en CHAM, BioAmbients y P-Systems. Cada nivel debe tener una etiqueta única. Es necesario declarar un nivel más externo o principal.

- Un conjunto de declaraciones que determinan la configuración inicial del sistema y que representan la presencia o ausencia de objetos en cada uno de los niveles. Debe existir una declaración por cada nivel de la configuración y esta declaración debe contener la cantidad ($c \geq 0$) de unidades de cada objeto en un nivel. Una declaración se escribe así:

  $$Decl_i = \{n1_i = x_1, n2_i = x_2, \ldots, nm_i = x_m\}$$

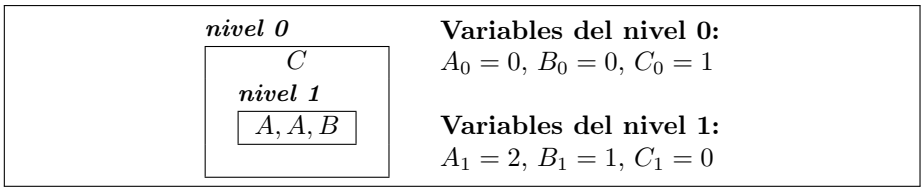
  En donde $Decl_i$ es la declaración del nivel i ($0 \leq i \leq k$) en una configuración con k niveles; $n1_i, n2_i, \ldots, nm_i$ son las variables y los x_j son las cantidades de cada uno de los objetos en el nivel i

- Un conjunto de procesos `ntcc`, uno por cada nivel. Estos procesos modelan las acciones de cada uno de los niveles por medio de unas reglas de evolución. En la siguiente sección se describen las caracteríscticas de estos procesos.

Los nombres de las variables, en conjunto con la configuración inicial del sistema determinan las variables del sistema, por cada $n \in Nombres$ deben existir k variables (una por cada nivel); se pueden omitir variables según el sistema que se está modelando. Por ejemplo en un proceso en donde la interacción este dada por un receptor que está en la membrana, en este caso se puede omitir la variable que representa al receptor en los niveles que dicho receptor no puede alcanzar. Las declaraciones serán las restricciones de entrada.

Ejemplo
En el cuadro 4.1 se muestra un ejemplo sencillo de un sistema con dos niveles, en donde se han ubicado tres tipos de objetos A, B y C. Los elementos de un modelo en `ntcc` para este ejemplo son:

- **Nombres de Variables:** $Vars = \{A, B, C\}$
- **Configuración:** $[\ [\ \ ]_{nivel1}\ ]_{nivel0}$
- **Declaraciones:**
  $Decl_0 = \{A_0 = 0, B_0 = 0, C_0 = 1\}$
  $Decl_1 = \{A_1 = 2, B_1 = 1, C_1 = 1\}$
- **Procesos:** $Procs = \{P_1, P_2\}$. Estos procesos no se muestran en el diagrama, en la siguiente sección se explica cómo deben ser construidos estos procesos.



Cuadro 4.1: Ejemplo: Sistema con dos niveles y cuatro objetos.

4.2. Modelando Reglas de evolución como reacciones químicas

Al modelar la interacción de cada nivel por medio de un proceso `ntcc` se logra que las acciones de un nivel sean agentes que interactúan con las acciones de los demás niveles. La interacción de cada nivel será modelada como un conjunto de reglas de evolución vistas como reacciones químicas.

Las reacciones químicas son procesos en los cuales una o más sustancias (reactivos) sufren transformaciones químicas para convertirse en otras (productos). Las reacciones químicas que se modelan a continuación son reacciones de *transformación*, *unión/descomposición* y *transporte*, o combinaciones de ellas.

$$\begin{array}{ll} A \rightarrow B & \text{(transformación)} \\ AB \leftrightarrow A, B & \text{(unión/descomposición)} \\ [A]_1 \rightarrow [\ \ ]_1, A & \text{(transporte)} \end{array}$$

Para modelar reglas de evolución como reacciones químicas en `ntcc` se define un proceso genérico llamado *Reaction*. Cada regla es una reacción química de la forma:

$$R_1, R_2, \ldots, R_n \rightarrow P_1, P_2, \ldots, P_m$$

vista como *condiciones* $\rightarrow$ *consecuencia*. Existen dos enfoques o formas para definir el proceso *Reaction*, dependiendo de la forma en que se compongan

los conjuntos de reglas en cada uno de los niveles. La primera alternativa es hacer uso de la escogencia no determinista (+) entre las reglas de un nivel; la segunda alternativa es utilizar la composición paralela (|).

El tipo de alternativa a seguir depende directamente del problema biológico que se desea modelar. Puede parecer que lo más acertado es hacer uso de la composición paralela, pero ocurre que la evolución de dos o más reglas en una misma unidad de tiempo pueden causar incosistencias en el sistema, por tanto usar escogencia no determinista sería una mejor alternativa. Estas consideraciones aplican también para la composición de los procesos de cada nivel. Es necesario tener en cuenta también que los procesos, o interacciones en cada nivel deben ser persistentes, es decir deben estar disponibles en todas las unidades de tiempo, para esto se usa replicación (!) de las reglas.

Ejemplo

Con base en el ejemplo de la sección anterior y que puede verse en el cuadro 4.1, suponga que existen tres reglas $S_{1,1}, S_{1,2}, S_{1,3}$, en el nivel 1. Siguiendo la primera alternativa, el proceso P_2 que modela la interacción del nivel 1 sería:

$$P_2 \stackrel{def}{=} S_{1,1} \ + \ S_{1,2} \ + \ S_{1,3}$$

Siguiendo la segunda alternativa, el proceso P_2 que modela la interacción del nivel 1 sería:

$$P_2 \stackrel{def}{=} S_{1,1} \mid S_{1,2} \mid S_{1,3}$$

A continuación se muestra el proceso *Actividad* que modela la actividad del sistema de membranas del cuadro 4.1. Siguiendo la primera alternativa, el proceso *Actividad* es de la forma:

$$Actividad \stackrel{def}{=} !(P_1 \ + \ P_2)$$

Siguiendo la segunda alternativa, el proceso *Actividad* es el siguiente:

$$Actividad \stackrel{def}{=} !(P_1 \mid P_2)$$

4.2.1. Reacciones usando escogencia no determinista

Las reglas de evolución que se componen usando escogencia no determinista son procesos `ntcc` de la forma:

$$\begin{array}{lll} Reaction & \stackrel{def}{=} & \textbf{when } (R_1 \geq 1 \wedge \ldots \wedge R_n \geq 1) \textbf{ do next} \\ & & (\ \textbf{tell}(\ R_1 = R_1 - 1 \wedge \ldots \wedge R_n = R_n - 1 \wedge \\ & & P_1 = P_1 + 1 \wedge \ldots \wedge P_m = P_m + 1 \wedge Rest)\ ) \end{array}$$

en donde el antecedente de una reacción (condiciones) es un conjunto de restricciones que pueden ser deducidas (o no) del *store*. En el caso en que estas restricciones puedan ser deducidas, se imponen otras en la siguiente unidad de tiempo (esto sucede a causa del constructor **next**).
Con el fin de dar continuidad a la evolución del proceso, los valores de las variables que están por fuera de una reacción son pasados a la siguiente unidad de tiempo sin alteración; esto es representado por *Rest*. Cabe anotar que siguiendo este enfoque, solo una regla puede ser activada en una unidad de tiempo. En el caso en que dos o más reglas cumplan con las restricciones necesarias para evolucionar, solo una de ellas lo hará a causa de la escogencia no determinista.

4.2.2. Reacciones usando composición paralela

Cuando se modela un sistema de membranas usando reglas en paralelo pueden ocurrir inconsistencias, porque el paralelismo permite la activación de varias de ellas en la misma unidad de tiempo; por tanto también es necesario modificar el proceso *Reaction* de la sección anterior reemplazando *Rest* por el proceso $Control_i$ que evita que los valores de las variables que hacen parte de la reacción sean modificados por otros procesos. En este enfoque también debe darse continuidad a la evolución del proceso, para ello se incluye el proceso $Control_{gen}$ a la actividad del sistema de membranas. $Control_{gen}$ se encarga de actualizar los valores de las variables que no hacen parte de una reacción y preservar la continuidad de la evolución del sistema. El proceso $Control_{gen}$ se compone de varios procesos en paralelo, uno por cada variable. Dichos procesos son de la forma:

$$\textbf{unless } Control\_var = 1 \textbf{ do next } var \ \forall \ var \in Vars,$$

donde $Vars$ es el conjunto de variables del sistema. Para el funcionamiento de los procesos de control es necesario declarar una variable de control por cada una de las variables del sistema. El nombre de cada variable de control está dado por el prefijo *Control_* y el nombre de la variable con la que está relacionada.

En cada reacción el proceso $Control_i$ asigna uno a las variables de control asociadas a cada una de las variables de la reacción, ya sea como antecedente o consecuencia de una reacción. El proceso $Control_{gen}$ consulta al *store*, por cada variable de control, si se puede deducir que dicha variable es igual a 1 (por medio de **unless**). En el caso en que no se pueda deducir que una variable es igual a 1, entonces quiere decir que la variable asociada a la variable de control no hace parte de ninguna reacción y su valor debe pasarse a la siguiente unidad de tiempo. De otra forma, dicha variable hace parte de una reacción y su valor en la siguiente unidad de tiempo es establecido por dicha reacción. Los procesos *Reaction*, $Control_i$ y $Control_{gen}$ se muestran a continuación:

$$
\begin{array}{lll}
Reaction & \stackrel{def}{=} & \textbf{when } (R_1 \geq 1 \wedge \ldots \wedge R_n \geq 1) \textbf{ do next} \\
 & & (\ \textbf{tell}(\ R_1 = R_1 - 1 \wedge \ldots \wedge R_n = R_n - 1 \wedge \\
 & & P_1 = P_1 + 1 \wedge \ldots \wedge P_m = P_m + 1) \parallel Control\ ) \\
Control & \stackrel{def}{=} & Control\_R_i = 1, \quad Control\_P_j = 1 \text{ para } 1 \leq i \leq n \text{ y } 1 \leq j \leq m \\
Control_{gen} & \stackrel{def}{=} & \Pi\ \textbf{\textit{unless}}\ Control\_var := 1\ \textbf{\textit{do next}}\ var, \forall\ var \in Vars
\end{array}
$$

4.2.3. Reglas de evolución para un sistema de membranas en `ntcc`

Con base en el proceso *Reaction* (cualquiera de los dos), a continuación se muestran los tipos de reglas de evolución que pueden ser utilizados y su representación en `ntcc`. Estas reglas están inspiradas en las reglas de *CHAM* y *P-Systems*.

Reglas de transformación
Se utilizan para modelar aquellos procesos en los que una sustancia se transforma en otra. En ocasiones, estos procesos son inducidos por una tercera sustancia (catalizador) que puede quedar intacta o ser consumida. Estas reglas se dividen en reglas de transformación con o sin catalizador. A continuación se muestran las reglas y el proceso `ntcc` que las modela:

$$
\begin{array}{lll}
(1) & A \rightarrow B & \textbf{when } (A \geq 1) \textbf{ do next} \\
 & & (\textbf{tell}(B = B + 1 \wedge A = A - 1)) \\
(2) & A, C \rightarrow B, C & \textbf{when } (A \geq 1 \wedge C \geq 1) \textbf{ do next} \\
 & & (\textbf{tell } (B = B + 1 \wedge A = A - 1 \wedge C = C)) \\
(3) & A, C \rightarrow B, & \textbf{when } (A \geq 1 \wedge C \geq 1) \textbf{ do next} \\
 & & (\textbf{tell } (B = B + 1 \wedge A = A - 1 \wedge C = C - 1))
\end{array}
$$

La regla (1) modela una regla de transformación sin catalizador, en donde una sustancia A se transforma en B. (2) y (3) modelan reglas de transformación con catalizador en donde una sustancia A se transforma en B por la presencia de una sustancia C. En (2) el catalizador (objeto C) queda intacto, en cambio en 3 este catalizador es consumido.

Reglas de síntesis/descomposición
Estas reglas modelan los procesos de unión/separación de sustancias (objetos en el modelo). A continuación se muestran las reglas y el proceso `ntcc` que las modela:

(1) $A, B \rightarrow AB$ **when** $(A \geq 1 \wedge B \geq 1)$ **do next** (**tell**$(A\&B = A\&B + 1 \wedge A = A - 1 \wedge B = B - 1)$)

(21) $AB \rightarrow A, B$ **when** $(A\&B \geq 1)$ **do next** (**tell**$(A = A + 1 \wedge B = B + 1 \wedge A\&B = A\&B - 1)$)

en (1) se modela una regla de síntesis en donde las sustancias A, B se unen para formar el compuesto $A\&B$. (2) es una regla de descomposición, aquí el compuesto $A\&B$ se separa en las sustancias A, B

Reglas de transporte
Son aquellas que modelan el paso de sustancias a través de las membranas. Por medio de los procesos *ntcc* que se muestran a continuación, es posible modelar *symport* y *antiport* las reglas de transporte de *P-Systems*.

(1) $[\ [A]_i\ ]_j \rightarrow [\ [\ \ ]_i\ A]_j$ **when** $(A_i \geq 1)$ **do next** (**tell**$(A_j = A_j + 1 \wedge A_i = A_i - 1)$)

(2) $[A\ [\ B\ ]_i\ ]_j \rightarrow [B[A]_i\ ]_j$ **when** $(A_j \geq 1 \wedge B_i \geq 1)$ **do next** (**tell**$(A_j = A_j - 1 \wedge B_i = B_i - 1 \wedge A_i = A_i + 1 \wedge B_j = B_j + 1)$)

(3) $[\ [A, B]_i\ ]_j \rightarrow [A, B[\ \ ]_i\ ]_j$ **when** $(A_i \geq 1 \wedge B_i \geq 1)$ **do next** (**tell**$(A_i = A_i - 1 \wedge B_i = B_i - 1 \wedge A_j = A_j + 1 \wedge B_j = B_j + 1)$)

(1) modela el paso de una sustancia A que se encuentra dentro de un nivel i a otro nivel j. En (2) se modela la regla de antiport en donde dos sustancias A y B se mueven a través de la membrana en direcciones opuestas. (3) modela symport: el paso de dos sustancias A, B a través de la membrana en la misma dirección. (1), (2) y (3) muestran el paso de sustancias en una sola dirección, pero este puede hacerse también en dirección opuesta.

En adelante los procesos que modelen reglas de evolución serán representados de la forma *condiciones* $\rightarrow$ *consecuencia*.

4.3. Simulando sistemas de membranas en `ntcc`

La simulación de un modelo especificado con las anteriores técnicas debe realizarse utilizando un intérprete del cálculo `ntcc`. En este trabajo se utiliza *ntccSim* que es una herramienta desarrollada por el grupo AVISPA[1] en lenguaje Mozart. Con *ntccSim* se pueden escribir procesos siguiendo la sintaxis de `ntcc`

[1]Descargas de la versión utilizada en esta tesis en http://eisc.univalle.edu.co/~avillota

para simularlos siguiendo las reglas de la semantica operacional del cálculo [19]. Gracias a la implementación en Mozart, es posible que se utilicen los sistemas de restricciones que existen para la simulación de procesos. La simulación de un proceso `ntcc` en *ntccSim* se hace por medio de un procedimiento llamado *Simulate*. Esta función toma 4 entradas:

- una lista de procesos `ntcc` escritos con la sintáxis del simulador. Los procesos de esta lista son considerados en paralelo. En el cuadro 4.2 encuentra la sintáxis del simulador.
- un registro que almacena las variables del sistema. Cada campo de este registro es una variable de dominio finito y que tiene el nombre de una variable del problema.
- un número que representa el total de unidades de tiempo que se van a simular.
- un procedimiento escrito en Mozart que será ejecutado al final de cada unidad de tiempo y se utiliza por lo general para mostrar el valor de las variables, guardar en archivos o imprimir mensajes.

$\langle restricciones \rangle$	:==	proc{\$ Var}...end	Restricciones
$\langle tell \rangle$	:==	tell($\langle restricciones \rangle$)	Tell
$\langle ask \rangle$	:==	when($\langle restricciones \rangle \langle ntcc - proc \rangle$)	Ask
$\langle sum \rangle$	:==	sum($\{\langle ask \rangle\}^+$)	Escogencia con guarda
$\langle par \rangle$	:==	par($\{\langle ntcc - proc \rangle\}^+$)	Composición Paralela
$\langle next \rangle$	:==	next($\langle ntcc - proc \rangle$)	Retraso de Unidad
$\langle unless \rangle$	:==	unless($\langle restricciones \rangle \langle ntcc - proc \rangle$)	Vencimiento
$\langle bang \rangle$	:==	rep($\langle ntcc - proc \rangle$)	Replicación
$\langle star \rangle$	:==	star($\langle ntcc - proc \rangle Limit$)	Retraso acotado
$\langle ntcc - proc \rangle$	:==	$\langle tell \rangle \mid \langle ask \rangle \mid \langle sum \rangle \mid \langle bang \rangle$	Procesos `ntcc`
	|	$\langle par \rangle \mid \langle next \rangle \mid \langle unless \rangle \mid \langle star \rangle$	En *ntccSim*

Cuadro 4.2: Sintaxis de *ntccSim*

Para utilizar las técnicas expuestas en la sección anterior fueron implementados un conjunto de funciones y procedimientos que facilitan la simulación de un modelo biológico en sistemas de membranas. Estos procedimientos y funciones tienen en cuenta la sintáxis para modelar sistemas de membranas de la sección anterior, en donde para definir el modelo es necesario establecer: un conjunto de nombres, una configuración, unas declaraciones y un conjunto de procesos (en forma de reacciones). La implementación de los procedimientos y funciones están en el functor `membranas.oz` que se incluye en los archivos del simulador, estos procedimientos y funciones facilitan al usuario la declaración de las variables y del registro que las contiene, definir procesos derivados de *Reaction* y definir procesos de control.

Debido a que el simulador trabaja con un registro con las variables del sistema, cada variable será un campo en dicho registro. En Mozart los nombres de los campos de los registros son átomos, es decir combinaciones de caractéres imprimibles que empiecen por minúscula. De esta manera, los nombres de las variables deben también ser átomos.

4.3.1. Declaración Variables

Para la declaración de las variables se utiliza el conjunto de nombres de la sintáxis y la cantidad de niveles del sistema a partir de la configuración. Existen dos procedimientos: `declararVars1` y `declararVars2` que son utilizados para declarar las variables.

`declararVars1` se utiliza cuando se planea utilizar escogencia no determinista entre las reglas de evolución de cada nivel y entre niveles. `declararVars2` debe ser utilizado en el caso en que se decida utilizar composición paralela entre los procesos que modelan la interacción, porque declara también las variables de control.

Los procedimientos `declararVars1` y `declararVars2` reciben 4 entradas:

- Una lista de átomos que representa la lista de nombres de objetos del sistema.
- Un entero que representa la cantidad de niveles del sistema.
- Un entero que determina la cantidad máxima de objetos de cada tipo que puede localizarse en cada nivel.
- Una aariable sin instanciar que se ligará al registro de variables que requiere *ntccSim* para iniciar la simulación.

4.3.2. Definición de procesos derivados de *Reaction*

Las reglas que modelan la interacción de los niveles en forma de reacciones químicas son procesos `ntcc` que deben ser escritos utilizando la sintáxis del simulador. Para hacer más sencilla la especificación de estos procesos se implementaron dos funciones: `Reaction1` y `Reaction2`. Estas funciones implementan el proceso *Reaction* para las dos alternativas de composición de las reglas, `Reaction1` para las reglas que se componen usando escogencia no determinista y `Reaction2` para composición paralela.

En la sección anterior se mostró que todas las reglas derivadas del proceso genérico *Reaction* tienen la forma *condiciones* $\rightarrow$ *consecuencias*. Esta idea se conservó para la implementación de las funciones *Reaction*1 y *Reaction*2 y en consecuencia, ambas funciones tienen dos entradas:

- una lista de átomos (nombres de los objetos) que representa las condiciones para activar una reacción.
- una lista de átomos (nombres de los objetos) que representan las concecuencias de que una reacción haya sido activada.

La función *Reaction*1 se muestra a continuación, aquí *Elem* son las condiciones y *Comp* son las consecuencias.

```
fun{Reaction1 Elem Comp} Proc C Cons in
   C={Constraint Elem}
   Cons={Consecuence1 Elem Comp}
   Proc= when(C next(tell(Cons)))
end
```

Esta función retorna un proceso `ntcc` (escrito con la sintaxis de *ntccSim*) con estructura **when**($\langle restricciones \rangle \langle ntcc - proc \rangle$), en donde $\langle restricciones \rangle$ es un procedimiento en lenguaje Mozart que impone las restricciones de las condiciones por medio del llamado a la función auxiliar `Constraint`.

$\langle ntcc - proc \rangle$ impone las restricciones de las consecuencias con un proceso **tell** y el llamado a la función `Consecuence1`. En el cuadro 4.3 se muestran las funciones `Constraint` y `Consecuence1`.

```
fun{Constraint Elem}
   proc{$ Root}
      {ForAll Elem
       proc {$ X}
          Root.current.X >:0
       end
      }
   end
end

fun{Consecuence1 Elem Comp}
   proc{$ Root}
      {ForAll Lvars
       proc {$ X}
          if {Member X Elem} then
             Root.current.X =: Root.residual.X - 1
          elseif {Member X Comp}then
             Root.current.X =: Root.residual.X + 1
          else
             Root.current.X =: Root.residual.X
          end
       end
      }
   end
end
```

Cuadro 4.3: Funciones auxiliares de *Reaction*1

Los cambios que ocurren en la implementación de *Reaction*2 son debido a la inclusión del proceso de control interno de la reacción que evita que el proceso de control general ($Control_{gen}$) actualice los valores de las variables involucradas en la reacción. La función *Reaction*2 se muestra a continuación:

```
fun{Reaction2 Elem Comp} Proc C Cons in
   C={Constraint Elem}
   Cons={Consecuence2 Elem Comp}
   Proc= when(C
              par(
                 next(tell(Cons))
                 tell({Control_in {Append Elem Comp}})
                 )
             )
end
```

De forma similar a la función *Reaction*1, en esta función se construye el proceso *ntcc* que corresponde a una regla de evolución y las restricciones de las condiciones por medio del llamado a la función auxiliar `Constraint`. En este caso las restricciones de las consecuencias de una reacción las impone la función *Consecuence*2 que modifica únicamente los valores de las variables involucradas en la reacción. Se hace un llamado a la función $Control\_in$ que asigna un 1 a las variables de control asociadas a las variables involucradas en la reacción.

```
fun{Consecuence2 Elem Comp}
   proc{$ Root}
      {ForAll Elem
       proc{$ X}
          Root.current.X =: Root.residual.X - 1
       end
      }
      {ForAll Comp
       proc{$ X}
          Root.current.X =: Root.residual.X + 1
       end
      }
   end
end
```

Cuadro 4.4: Función auxiliares de *Reaction*2

4.3.3. Definición de proceso de control

El proceso de control es la implementación de *Controlgen* de la sección anterior. Se utiliza cuando los procesos del sistema están en composición paralela. El proceso no tiene entradas y utiliza una función auxiliar que crea un proceso de control por cada variable, las variables están en una lista, ligada a la variable global `Lvars`. En el cuadro 4.5 se puede ver el código en Mozart de la función `Control`.

```
fun{ControlProcess Vars}
   {Map Vars
    fun{$ X}Proc in
       Proc=unless(
               proc{$ Root}
                  Root.current.{StringToAtom
                                {Append "c_"{AtomToString
X}}}=:1
               end
               tell(
                  proc{$ Root}
                     Root.current.X =: Root.residual.X
                  end))
    end
   }
end

fun{Control}
   {List.toTuple par {ControlProcess Lvars}}
end
```

Cuadro 4.5: Implementación del proceso Control en Mozart

Capítulo 5

Modelo en `ntcc` de la ruta endocítica de la Lipoproteína baja densidad (LDL) del colesterol

En este capítulo se aplican las técnicas de la sección 4 para modelar la ruta endocítica de la LDL del colesterol. Aquí se presentan dos modelos, uno usando escogencia no determinista entre las reglas y el segundo utilizando composición paralela. En la primera parte del capítulo se encuentra la descripción biológica del proceso que se desea modelar.

5.1. Descripción del sistema desde la biología

A continuación se hará una breve descripción de algunos conceptos que serán utilizados en la descripción de la ruta endocítica de la ldl del colesterol.

Endocitosis/Exocitosis

Es el proceso mediante el cual la célula internaliza algún objeto que se encuentra fuera de ella. Durante este proceso, una pequeña región de la membrana celular se invagina para formar una nueva vesícula interna que está limitada por una membrana. Exocitosis es el proceso que permite a la célula expulsar material (desechos, proteínas, entre otros), aquello que debe ser expulsado, es transportado al interior de una vesícula cuya membrana es luego absorbida por la membrana celular.

Receptor

Un receptor es una proteína que reside en la membrana de la célula, en el citoplasma, o en la membrana del núcleo y que es la encargada de atrapar o atraer una molécula específica, llamada ligando.

Endocitosis mediada por receptor

Es el mecanismo por medio del cual un receptor localizado en la membrana celular se enlaza con un ligando formando un solo compuesto que es llevado a dentro de la célula por medio de endocitosis para luego ser procesado por la célula.

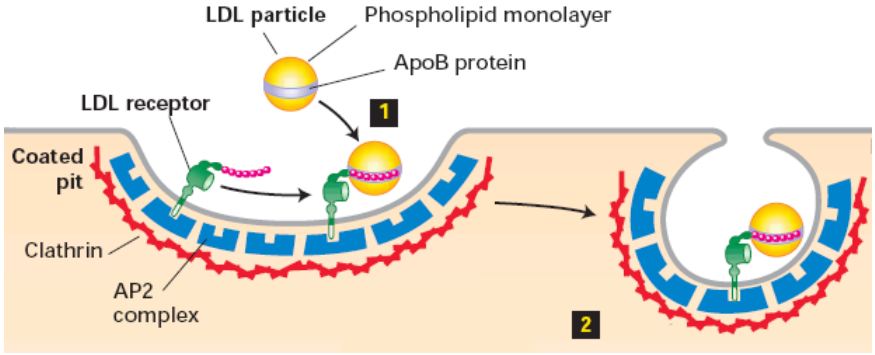


Figura 5.1: Endocitosis mediada por receptor

Lipoproteína de baja densidad *(LDL)*

Es un complejo que transporta colesterol a través del torrente sanguíneo. Está compuesta por una cubierta formada por fosfolípidos y una proteína llamada *apoB100*, el núcleo de la partícula contiene colesterol. En los mamíferos existen células que producen receptores que enlazan la proteína *apoB100* con el fin de ingresar las partículas de LDL para luego procesarlas y obtener el colesterol que albergan.

Clatrina

Es una de las proteínas que conforman la membrana celular, cuando ocurre la endocitosis mediada por receptor, la vesícula formada está rodeada por clatrina.

5.1.1. Ruta Endocítica de la LDL del Colesterol

El conjunto de pasos por medio de los cuales una célula lleva a su interior una partícula, para luego ser procesada por sus organelas, es conocido como *ruta endocítica*. Las células llevan a su interior partículas de colesterol siguiendo una

ruta endocítica. El colesterol puede ser de dos clases: de alta densidad (high density) o de baja densidad (low density). El colesterol de baja densidad se transporta por el torrente sangineo al interior de una lipo-proteína conocida como la LDL (*low density lipoprotein*). En la Figura 5.2 (tomada de [24]) se muestra el proceso de la ruta endocítica de la LDL del colesterol.

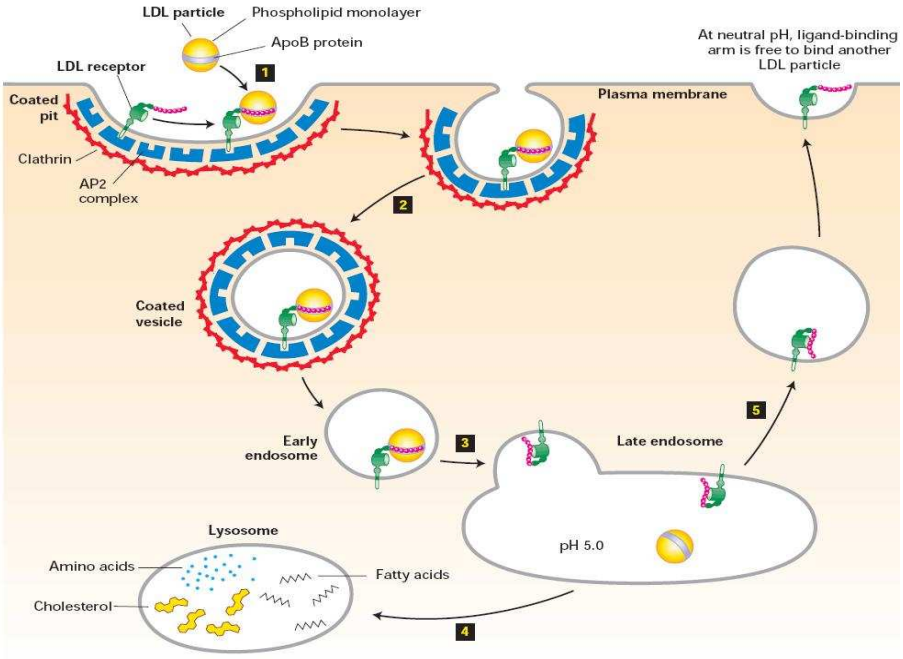


Figura 5.2: Ruta endocítica de la lipoproteína de baja densidad del colesterol, tomada de [24]

El primero de los pasos en la ruta endocítica se llama *endocitosis mediada por receptor*. El proceso se inicia cuando un receptor (ubicado en la membrana celular) enlaza a la partícula de LDL (ligando), a través de la proteína *apoB100* que está ubicada en la superficie de la partícula (paso 1 de la figura 5.2).

Una vez ocurrido el enlace, el complejo receptor-partícula ingresa a la célula en una vesícula de transporte formada por endocitosis; ésta vésicula está rodeada por clatrina (proteína que recubre la membrana). En el momento en que la clatrina se desprende de la vesícula, ésta se convierte en un endosoma temprano (pasos 2 y 3 de la figura 5.2).

A continuación, el endosoma temprano se fusiona con un endosoma tardío (una de las organelas de la célula). El ambiente al interior del endosoma tardío es ácido en comparación con el exterior (ph entre 5 y 6), lo que causa que se rompa el enlace entre el receptor y el ligando. Después de la separación, el receptor debe ser devuelto a la superficie de la membrana celular con el fin de

servir como mecanismo para la captura de otras partículas de LDL al exterior de la célula. El receptor es envíado en el interior de una vesícula que se desprende del endosoma tardío (paso 5 de la figura 5.2).

El endosoma tardío que contiene únicamente la partícula de LDL se fusiona con un lisosoma. El lisosoma es una organela que contiene las enzimas necesarias para degradar la cubierta de la partícula y extraer el colesterol de su interior para luego ser aprovechado por la célula (paso 6 de la figura 5.2).

5.1.2. Modelo en `ntcc`

Con base en la descripción biológica de [24] (descrita anteriormente) y las técnicas propuestas en la sección anterior, se proponen dos modelos, uno en el cual se utilice escogencia no determinista y un segundo modelo en el que se utilice composición paralela entre las reglas. Esto con el fin de mostrar la utilización de las dos estrategias propuestas en el capítulo anterior y porque las reglas propuestas a continuación permiten la aplicación de ambas alternativas a las mismas reglas sin causar incosistencias cuando se utiliza composición paralela.

Ambos modelos tienen en común la definición de objetos, configuración de la jerarquía de membranas, declaraciones y las reglas que modelan el comportamiento de los niveles. A continuación se describen los elementos que son comunes a ambos modelos.

Objetos

El conjunto de objetos se obtiene de la unión entre los objetos básicos y los compuestos ($Vars = B \cup C$). Los objetos básicos son:

$$B = \{LDLpart, LDLRcpt, Clth, LateEndo\}$$

en donde $LDLpart$, representa la lipoproteína, $LDLRcpt$ representa el receptor, $Clth$ representa a la proteína que cubre la vesícula que ingresa a la célula (clatrina) y $LateEndo$ es el endosoma tardío. Los objetos anteriormente mencionados pueden formar compuestos. Estos compuestos se representan por la unión de los nombres de los objetos con un "&" intermedio; por ejemplo $LDLpart\&LDLRcpt$ representa al compuesto que resulta al unirse un ligando con el receptor. Los objetos compuestos son:

$$C = \{LDLpart\&LDLRcpt,\ \ Clth\&LDLpart\&LDLRcpt,\\ LateEndo\&LDLpart\&LDLRcpt,\ \ LateEndo\&LDLpart\}$$

Configuración

Para este modelo se utiliza una configuración con cuatro niveles anidados:

$$[\ [\ [\ [\ \ ]_{nivel_3}\ ]_{nivel2}\ ]_{nivel1}\ ]_{nivel0}$$

más adelante se explica qué parte del sistema biológico representan cada uno de los niveles.

Declaraciones

Los valores iniciales del modelo están dados por:

$$Decl = \{LDLrcpt_1 = 1, Clth_2 = 1, LateEndo_2 = 1\}$$

Las demás variables son cero inicialmente.

Estos valores denotan que se ubican un lisosoma, y un endosoma temprano en el nivel 2 (interior de la célula) y un receptor sobre la membrana (nivel 1).

Reglas

Las reglas, o conjuntos de reglas deben ser propuestos para cada nivel. En esta definición no se toma en cuenta el proceso que se utilizará para componer las reglas (paralelismo, escogencia no determinista), por lo tanto se describen cuales son las reglas que existen en cada nivel, teniendo en cuenta que en conjuto conforman un proceso.

- **Nivel 0:** Modela el exterior de la célula, en este nivel está localizada la lipoporteína de LDL. No hay reacciones en este nivel.
- **Nivel 1:** Modela la membrana celular, no como una división, sino como el espacio en donde está localizado el receptor. En esta región se localiza una reacción que permite la unión entre el ligando ($LDLpart$) y el receptor ($LDLRcpt$), así como también la localización del compuesto formado en el nivel 1 (membrana de la célula).

$$R_{1,1} = LDLpart_0, LDLRcpt_1 \rightarrow LDLpart\&LDLRcpt_1$$

- **Nivel 2:** Modela el interior de la célula, aquí están localizados el endosoma tardío y el lisosoma. Comprende seis reacciones que modelan el comportamiento al interior de la célula. El proceso $R_{2,1}$ modela el paso 2 de la figura 5.2, en donde una vesícula envuelta en clatrina que contiene el receptor y el ligando unidos, ingresa a la célula. Por medio de los procesos $R_{2,2}$ y $R_{2,3}$ se modela el momento en que la clatrina se desprende de la vesícula, convirtiéndose en endosoma temprano y la fusión de éste con el endosoma tardío. En $R_{2,4}$ se modela la ruptura del enlace entre el receptor y el ligando, posteriormente en $R_{2,5}$ se modela el paso del receptor del nivel 2, al nivel 1 para que sea reutilizado, como se puede ver en el paso 5 de la figura 5.2. Por medio del proceso $R_{2,6}$ se modela la fusión entre el endosoma tardío y el lisosoma.

$$\begin{aligned}
R_{2,1} = &\quad Clth_2,\ LDLpart\&LDLRcpt_1 \rightarrow Clth\&LDLpart\&LDLRcpt_2 \\
R_{2,2} = &\quad Clth\&LDLpart\&LDLRcpt_2 \rightarrow Clth_2,\ LDLpart\&LDLRcpt_2 \\
R_{2,3} = &\quad LateEndo_2,\ LDLpart\&LDLRcpt_2 \rightarrow \\
&\quad LateEndo\&LDLpart\&LDLRcpt_2 \\
R_{2,4} = &\quad LateEndo\&LDLpart\&LDLRcpt_2 \rightarrow \\
&\quad LateEndo\&LDLpart_2,\ LDLrcpt_2 \\
R_{2,5} = &\quad LDLrcpt_2 \rightarrow LDLrcpt_1 \\
R_{2,6} = &\quad LateEndo\&LDLpart_2 \rightarrow LateEndo\&LDLpart_3
\end{aligned}$$

- **Nivel 3:** Modela el interior del lisosoma; este es el nivel a donde debe llegar la partícula de LDL para ser degradada por la célula. Este nivel alberga tres reacciones, los procesos $R_{3,1}$ y $R_{3,3}$ modelan el procesamiento de la LDL del colesterol y el proceso $R_{3,2}$ permite retornar el objeto *LateEndo* al nivel 2, para que pueda ser reutilizado en un nuevo proceso.

$$\begin{array}{lcl} R_{3,1} & = & LateEndo\&LDLpart_3 \rightarrow LateEndo_3,\ LDLpart_3 \\ R_{3,2} & = & LateEndo_3 \rightarrow LateEndo_2 \\ R_{3,3} & = & LDLpart_3 \rightarrow Chol_3 \end{array}$$

5.1.3. Modelo utilizando escogencia no determinista (+)

Retomando algunos conceptos del capítulo anterior, en el caso de un modelo en el cual se utiliza la escogencia no determinista para componer las reglas, solo una regla puede reaccionar en cada unidad de tiempo. El proceso *Reaction* que se usa para derivar las reglas es el siguiente:

$$\begin{array}{lcl} Reaction & \stackrel{def}{=} & \textbf{when } (R_1 \geq 1 \wedge \ldots \wedge R_n \geq 1) \textbf{ do next} \\ & & (\ \textbf{tell}(\ R_1 = R_1 - 1 \wedge \ldots \wedge R_n = R_n - 1 \wedge \\ & & P_1 = P_1 + 1 \wedge \ldots \wedge P_m = P_m + 1 \wedge Rest)\ ) \end{array}$$

Se recuerda al lector que *Rest* denota que los valores de las variables que no interfieren en la reacción se pasan sin modificaciones a la siguiente unidad de tiempo y esto debe hacerse para dar continuidad a la evolución del modelo debido a que de acuerdo a la semántica operacional de `ntcc` el *store* inicial en la siguiente unidad de tiempo solo contiene la información que está acompañada por un **next** (esto incluye a **!** y **unless**).

En el modelo del proceso biológico que se está describiendo, los procesos `ntcc` que modelan el comportamiento en cada uno de los niveles son:

$$\begin{array}{lcl} Nivel\ \ 1 & \stackrel{def}{=} & R_{1,1} \\ Nivel\ \ 2 & \stackrel{def}{=} & R_{2,1} + R_{2,2} + R_{2,3} + R_{2,4} + R_{2,5} + R_{2,6} \\ Nivel\ \ 3 & \stackrel{def}{=} & R_{3,1} + R_{3,1} + R_{3,1} \end{array}$$

Procesos adicionales

Los anteriores niveles componen el proceso *Actividad* por medio de + (escogencia no determinista) para evitar que dos o más reacciones de distintos niveles, sean activadas y causen inconsistencias. Este proceso modela el comportamiento de la célula. La característica principal de este proceso es la persistencia. Por medio del operador **!** se garantiza que durante la evolución del sistema, los procesos que simulan la célula están disponibles en todas las unidades de tiempo. A continuación se muestra el proceso *Actividad*:

$$CellActivity \stackrel{def}{=} !(\ Nivel\ \ 1 + Nivel\ \ 2 + Nivel\ \ 3\ )$$

$$
\begin{array}{lll}
Nivel1 & \stackrel{def}{=} & LDLpart_0, LDLRcpt_1 \rightarrow LDLpart\&LDLRcpt_1 \\
Nivel2 & \stackrel{def}{=} & R_{2,1} + R_{2,2} + R_{2,3} + R_{2,4} + R_{2,5} + R_{2,6} \\
R_{2,1} = & & Clth_2, LDLpart\&LDLRcpt_1 \rightarrow Clth\&LDLpart\&LDLRcpt_2 \\
R_{2,2} = & & Clth\&LDLpart\&LDLRcpt_2 \rightarrow Clth_2, LDLpart\&LDLRcpt_2 \\
R_{2,3} = & & LateEndo_2, LDLpart\&LDLRcpt_2 \rightarrow LateEndo\&LDLpart\&LDLRcpt_2 \\
R_{2,4} = & & LateEndo\&LDLpart\&LDLRcpt_2 \rightarrow LateEndo\&LDLpart_2, LDLrcpt_2 \\
R_{2,5} = & & LDLrcpt_2 \rightarrow LDLrcpt_1 \\
R_{2,6} = & & LateEndo\&LDLpart_2 \rightarrow LateEndo\&LDLpart_3 \\
Nivel3 & \stackrel{def}{=} & R_{3,1} + R_{3,1} + R_{3,1} \\
R_{3,1} = & & LateEndo\&LDLpart_3 \rightarrow LateEndo_3, LDLpart_3 \\
R_{3,2} = & & LateEndo_3 \rightarrow LateEndo_2 \\
R_{3,3} = & & LDLpart_3 \rightarrow Chol_3 \\
 & & \\
Actividad & \stackrel{def}{=} & !(\ Nivel\ \ 1 + Nivel\ \ 2 + Nivel\ \ 3\ ) \\
Init & \stackrel{def}{=} & \textbf{\textit{tell}}\ (LDLrcpt_1 = 1 \wedge Clth_2 = 1 \wedge LateEndo_2 = 1) \\
System & \stackrel{def}{=} & \textbf{\textit{tell}}\ (LDLpart_0 = n_0) \parallel Init \parallel Actividad
\end{array}
$$

Cuadro 5.1: Modelo de la ruta endocítica de la LDL del colésterol con el cálculo `ntcc` usando escogencia no determinista

Se incluye un proceso llamado *Init* que asigna valores iniciales a las variables otorgando una configuración inicial para la simulación; en este caso en particular se utilizan los valores de las *declaraciones* realizadas en la sección anterior, en donde se ubican un lisosoma, y un endosoma temprano en el nivel 2 (interior de la célula) y un receptor sobre la membrana (nivel 1).

$$Init \stackrel{def}{=} \textbf{\textit{tell}}\ (LDLrcpt_1 = 1 \wedge Clth_2 = 1 \wedge LateEndo_2 = 1)$$

El proceso principal es *System*, el cual contiene la actividad de la célula en paralelo con las condiciones iniciales y el proceso **tell**$(LDLpart_0 = n_0)$ que inicia la evolución del proceso, con n_0 partículas de LDL del colesterol al exterior de la célula. Todos los procesos aquí descritos se encuentran en el cuadro 5.1.

La simulación de este modelo utilizando las funciones propuestas en el capítulo anterior e implementadas en Mozart, puede ser descargado de:
`http://eisc.univalle.edu.co/~avillota`.

5.1.4. Modelo utilizando composición paralela (|)

En este caso también se retomarán algunos conceptos del capítulo anterior. En un modelo en el cual las reglas y los niveles están en paralelo es necesario tener en cuenta ciertas consideraciones. Principalmente porque es posible que una o más reglas de un mismo nivel, o de distintos niveles puedan reaccionar en la misma unidad de tiempo y esto puede traer consigo fallos o incosinstencias. Las inconsistencias ocurren cuando hay dos procesos que imponen restricciones

inconsistentes en el *store*, o que asignan valores distintos a una variable, esto ocurre porque se utilizan variables declarativas, que no pueden cambiar su valor, así que $x = 4$ y $x = 5$ es una inconsistencia porque x no puede ser igual a 4 y a 5 a la vez.

El proceso *Reaction* cambia porque ahora una reacción no debe preservar la continuidad de la evolución pasando los valores de las variables no involucradas en la reacción. Se añaden dos procesos de control, uno local que impone una restricción a una variable de control que indica que su valor no debe ser modificado por el proceso de control general. El proceso de control general se encarga de pasar a la siguiente unidad de tiempo los valores de las variables que no participaron en alguna reacción. Los procesos *Reaction*, $Control_i$ son los siguientes:

$$\begin{array}{lll} Reaction & \stackrel{def}{=} & \textbf{when } (R_1 \geq 1 \wedge \ldots \wedge R_n \geq 1) \textbf{ do next } (\ \textbf{tell}(\ R_1 = R_1 - 1 \wedge \ldots \\ & & \wedge R_n = R_n - 1\ P_1 = P_1 + 1 \wedge \ldots \wedge P_m = P_m + 1)\ )\ \|\ Control_i \\ Control_i & \stackrel{def}{=} & Control\_Ri = 1,\ \ Control\_Pj = 1 \text{ para } 1 \leq i \leq n \text{ y } 1 \leq j \leq m \end{array}$$

Teniendo en cuenta las consideraciones anteriores, los procesos `ntcc` que modelan el comportamiento en cada uno de los niveles usando composición paralela de reglas son:

$$\begin{array}{lll} Nivel\ 1 & \stackrel{def}{=} & R_{1,1} \\ Nivel\ 2 & \stackrel{def}{=} & R_{2,1} \parallel R_{2,2} \parallel R_{2,3} \parallel R_{2,4} \parallel R_{2,5} \parallel R_{2,6} \\ Nivel\ 3 & \stackrel{def}{=} & R_{3,1} \parallel R_{3,1} \parallel R_{3,1} \end{array}$$

Procesos adicionales

Los procesos *Actividad* e $Init$ son necesarios en este modelo también. La diferencia está en que en *Actividad* se incluye el proceso $Control_{gen}$ que se encarga de actualizar los valores de las variables que no hacen parte de una reacción y preservar la continuidad de la evolución del sistema. El proceso $Control_{gen}$ se compone de varios procesos en paralelo, uno por cada variable. Como aparece en el capítulo anterior, los procesos que componen a $Control_{gen}$ son de la forma:

$$\textbf{unless } Control\_var = 1 \textbf{ do next } var\ \forall\ var \in Vars,$$

donde $Vars$ es el conjunto de variables del sistema. El proceso $System$ es igual al de la sección anterior. En el cuadro 5.2 se encuentran los procesos que cambian para el segundo modelo (las reglas son las mismas).

La simulación de este modelo utilizando las funciones propuestas en el capítulo 4 e implementads en Mozart, se puede descargar de:
`http://eisc.univalle.edu.co/~avillota`.

$Nivel1$	$\stackrel{def}{=}$	$LDLpart_0, LDLRcpt_1 \rightarrow LDLpart\&LDLRcpt_1$
$Nivel2$	$\stackrel{def}{=}$	$R_{2,1} \parallel R_{2,2} \parallel R_{2,3} \parallel R_{2,4} \parallel R_{2,5} \parallel R_{2,6}$
$Nivel3$	$\stackrel{def}{=}$	$R_{3,1} \parallel R_{3,1} \parallel R_{3,1}$
$Actividad$	$\stackrel{def}{=}$	$!(\ Nivel\ \ 1 \parallel Nivel\ \ 2 \parallel Nivel\ \ 3 \parallel Control_{gen})$
$Control_{gen}$	$\stackrel{def}{=}$	Π ***unless*** $Control\_var = 1$ ***do next*** $var, \forall\ var \in Vars$

Cuadro 5.2: Modelo con paralelismo y el proceso $Control_{gen}$

Capítulo 6

Conclusiones y trabajo futuro

En este trabajo de investigación se estudió la expresividad del cálculo `ntcc` en cuanto a su adecuación para el modelamiento de sistemas biológicos. Este trabajo cumplió con los objetivos propuestos y tiene como principal contribución la propuesta de un conjunto de técnicas para modelar sistemas biológicos por medio de sistemas de membranas. En cuanto a las preguntas de investigación que se plantearon en la propuesta de investigación, es correcto afirmar que este trabajo de investigación es también una contribución a la respuesta de dichas preguntas.

Esta tesis hace parte de un conjunto de proyectos [30, 31, 32, 33] los cuales han mostrado que el cálculo `ntcc` es apropiado para el modelamiento de sistemas biológicos. Ante la pregunta: ¿Son los constructores del cálculo `ntcc` suficientes para modelar sistemas biológicos? no existe aún una respuesta completa. Pero sí es posible responder que los constructores del cálculo son suficientes para modelar redes de regulación genética [19], procesos de intercambio de sustancias a través de la membrana celular como en [2, 31] y sistemas de membranas en donde la configuración (jerarquía de membranas) no cambia durante la evolución del proceso [35].

En este documento se muestra también un estudio de algunos cálculos inspirados en la biología y que comparten el concepto de membranas. En este estudio se encontró que `ntcc` es útil para modelar aquellas operaciones que simulan una reacción química; pero tambiém que dicho cálculo tiene limitaciones para modelar aquellas operaciones que causan cambios en la jerarquía de membranas. A pesar de las limitaciones existentes se propuso un conjunto de técnicas que permiten el modelamiento de membranas, representación jerárquica de membranas y reglas de evolución.

Para la especificación de sistemas al estilo de membranas se utilizó un sistema de restricciones sobre dominios finitos. La ubicación de los elementos del sistema se realizó por medio de variables. La estructura del sistema es estática, los contenidos de las membranas son modificados debido al paso de sustancias

a través de las membranas (entrada y salida de sustancias), pero no hay desplazamiento de membranas como tal. Una membrana en este enfoque, es una frontera que encierra una región llamada *nivel*. Los niveles que conforman un modelo pueden variar dependiendo de la descripción del proceso biológico que se desea modelar.
Al interior de cada nivel se encuentra ubicado un conjunto de objetos que representan las sustancias presentes en la descripción biológica del sistema. La ubicación de cada uno de estos objetos (y de los compuestos que conforman), está dada por una variable declarativa que puede tomar valores en los naturales, y denota la cantidad de cierto objeto (y compuesto) en un nivel dado. Se mostró también la aplicación de las técnicas propuestas para modelar la ruta endocítica de la lipoproteína de baja densidad del colesterol. Sin embargo, en el momento de la simulación se encontró que existen aún inconvenientes con la selección no determinista y la composición paralela entre las reacciones y los niveles.

En el caso de la selección no determinista entre las reacciones y los niveles, se encontró que es útil porque permite garantizar que solo una reacción pueda ser activada en cada unidad de tiempo; esto es necesario ya que las variables declarativas no cambian su valor. No obstante el modelo no garantiza la ejecución de cada proceso (activación de una reacción) en un momento dado, e.g con el proceso $!(r_1 + r_2)$ no se puede garantizar que en alguna unidad de tiempo r_1 sea ejecutado. Este problema interfiere con la automatización de la simulación del proceso biológico, ya que no se puede garantizar que la máquina alcance algunos estados del proceso.

Este problema es resuelto parcialmente con la composición paralela, pero requiere de un cuidadoso manejo de las variables declarativas. Una variable declarativa no puede cambiar su valor, por tanto no se puede permitir que dos reacciones que involucran una misma sustancia sean activadas en la misma unidad de tiempo. Por ejemplo, en el caso de estudio de este trabajo, si existen más de una partícula de LDL al exterior de la célula y más de un receptor en la membrana, se genera una inconsistencia cuando se imponen las restricciones $LDL\_part\&LDL\_Rcpt = LDL\_part\&LDL\_Rcpt + 1$, y $LDL\_part\&LDL\_Rcpt = LDL\_part\&LDL\_Rcpt - 1$ en la misma unidad de tiempo. La inconsistencia no se presenta cuando hay más de una partícula de LDL al exterior y un solo receptor, así que en este caso la competencia por un recurso no es un problema. La utilización de composición paralela hace que el modelo sea más cercano al proceso biológico.

Las técnicas propuestas en este trabajo modelan membranas como fronteras de los niveles. También es posible modelar interacción al interior de una membrana, como ocurre en la ruta endocítica de la LDL del colesterol. Para lograr esto, se incluyó en el modelo un nivel (nivel 1) y se ubicó en dicho nivel los objetos y reacciones que representan el comportamiento de la membrana celular en dicho proceso biológico.

Todo apunta a que la mejor alternativa es preservar la composición paralela entre las reacciones y los niveles, pues así los modelos son más fieles al

sistema biológico (evolucionan concurrentemente), y es posible automatizar la simulación y verificar propiedades. Las inconsistencias son un problema inherente al tipo de variable y los dominios. Una propuesta interesante es examinar la utilización de otros sistemas de restricciones, como por ejemplo flujos, en los cuales se puedan actualizar los valores de las variables; así desaparecerían las inconsistencias.

Trabajo Futuro

Como trabajo futuro se propone el estudio de los sistemas de restricciones basados en flujos [26] para resolver completamente el problema de las incosistencias, dado que este inconveniente no surge sino hasta que se activen dos reacciones que comparten una variable en la misma unidad de tiempo. En algunos casos, como en la ruta endocítica de la LDL del colesterol, no hay inconsistencias en parte porque el modelamiento del proceso biológico se realizó evitando las inconsistencias. Para ello nos valimos de los operadores **next** y **when** para incluir cierto orden en los pasos de la evolución de los procesos que evitara la aparición de inconsistencias. Adicionalmente, se espera aplicar la lógica temporal (LTL) para verificar propiedades en el modelo.

Para modelar sistemas de membranas dinámicos John Vargas, integrante del grupo AVISPA, propone utilizar un sistema de restricciones basado en lógicas espaciales que permitan al usuario utilizar reglas de ingreso/egreso de una membrana a otra en un sistema de membranas. Esto último en conjunto con las técnicas propuestas en este proyecto dotarían a `ntcc` conlas características necesarias para el modelamiento de sistemas de membranas.

En cuanto a la movilidad en el cálculo `ntcc`, no existe un como un constructor del cálculo, pero existen propuestas como en [26] en el cual se modela mobilidad en CCP. De forma más concreta Carlos Olarte en [30] incluyó el concepto de movilidad a un cálculo con restricciones (`tcc`) proponiendo así el cálculo `utcc`. Actualmente existe un proyecto que estudia la codificación de los operadores de Brane en `utcc`.

Bibliografía

[1] Gutierrez J., Perez J.A., Rueda C.: Modelamiento de Sistemas Biologicos usando Calculos de Procesos Concurrentes. Epiciclos, 4(1) pp. 79–101 (2005)

[2] Gutierrez J., Perez J.A., Rueda C., Valencia F.D.: Timed Concurrent Constraint Programming for Analysing Biological Systems. Electronic Notes in Theoretical Computer Science (ENTCS) Volume 171(2), pp. 117–137 (2007)

[3] Valencia F.: Temporal Concurrent Constraint Programming. Tesis doctoral, Universidad de Aarhus (2002)

[4] Baeten J.M.: A Brief History of Process algebra. Report CS-R 04-02, Department of mathematics and computer science, Technische Universiteit Eindhoven (2003)

[5] Backofen R., Gilbert D.: Bioinformatics and Constraints. Journal of Constraints. Vol 6. 141-156 (2001)

[6] Bekic H.: Towards a mathematical theory of process. Technical Report TR 25.125, IBM Laboratory Viena (1971)

[7] Smolka G.: A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards. Research Report. Deutsches Forschungszentrum für Künstliche Intelligenz (1994)

[8] Saraswat V., Rinard M.C., Panangaden P.: The Semantic Foundations of Concurrent constraint Programming. POPL 1990, pp. 232–245 (1990)

[9] Tanenbaum A., van Steen M.: Distributed systems: principles and paradigms. Prentice Hall, capítulo 1 (2002)

[10] Kitano H.: Systems Biology: A Brief Overview. Science Vol 295 (2002)

[11] Kitano H.: Systems Biology: Toward System-level Understanding of Biological Systems. Foundations of Systems Biology, pp. 1–29. MIT Press, Cambridge/MA (2001)

[12] Saraswat V., Rinard M., Panangaden P.: The semantic foundations of concurrent constraint programming. POPL 91, pags: 333-352 (1991)

[13] Cardelli L.: Languages for Systems Biology. Position Paper for Grand Challenges UK, GC1 InVivo<=>InSilico (2004)

[14] Busi N., Zandron C.: Modeling and Analysis of Biological Processes by Mem(Brane) Calculi and Systems. En: Simulation Conference, 2006. WSC 06. Volumen de invierno diciembre 3-6, pp. 1646–1655 (2006)

[15] Regev A., Panina E.M., Silverman W., Cardelli L., Shapiro E.: BioAmbients: an abstraction for biological compartments. Theoretical Computer Science Volume 325 (1), pp. 141–167 (2004)

[16] Cardelli L., Gordon A.D.: Mobile Ambients. En Foundations of Software Science and Computation Structures: First International Conference, FOSSACS 98. Springer-Verlag (1998)

[17] Cardelli L.: Brane Calculi Interactions of Biological Membranes. En: Computational Methods in Systems Biology, International Conference CMSB (2004)

[18] Cardelli L. Bitonal Membrane Systems. MeCBIC2006 Workshop on Membrane Computing and Biologically Inspired Process Calculi (2006)

[19] Arbelaez A., Gutierrez J., Olarte C., Rueda C.: A Generic Framework to Model, Simulate and Verify Genetic Regulatory Networks (Poster). En: CLEI 2006 (32nd Latinamerican Conference on Informatics) (2006)

[20] Berry G., Boudol G.: The Chemical Abstract Machine. POPL '90: Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 81–94 (1990)

[21] Boudol G.: A Generic Membrane Model. Lecture Notes in Computer Science, Springer, pp. 208–222 (2004)

[22] Milner R., Parrow J., Walker D.: A calculus of mobile processes, partes i and ii. Information and Computation, 100(1):1–40, September (1992)

[23] Milner R.: A Calculus of Communicating Systems. Lecture Notes in Computer Science. Springer-Verlag (1980)

[24] Lodish H., Berk A., Scott M.: Molecular cell biology. - 5 ed. W. H. Freeman (2003)

[25] J. Gutiérrez, J. A. Pérez, C. Rueda and F. Valencia. Timed Concurrent Constraint Programming for Analyzing Biological Systems. En: Proceedings of MeCBIC 2006, Vol 171/2 pp 117-137 of Electronic Notes in Theoretical Computer Science, Elsevier.

[26] Laneve C., Montanari U.: Mobility in the cc-paradigm. Mathematical foundatios of computer science. LNCS 629, pp. 336–345 (1992)

[27] Paun G.: Computing with Membranes. Journal of Computer and System Sciences, 61(1) pp. 108–143 (2000)

[28] Paun G.: Introduction to Membrane Computing. Applications of Membrane Computing, Springer, Natural Computing Series, pp. 1–42 (2006)

[29] Saraswat V., Jagadeesan R., Gupta V.: Foundations of timed concurrent constraint programming. En Ninth Annual IEEE Symposium on Logic in Computer Science, Paris, France, pp. 71–80, (1994)

[30] Olarte C., Valencia F.: Universal Concurrent Constraint Programming: Symbolic Semantics and Applications to Security. En: Proc. of SAC'08., ACM Press (2008)

[31] Aranda J.: Modeling Rho-GTP proteins in an extension of NTCC. Charla en el PUJ Colloquium in Concurrency (2008)

[32] Aranda J. Perez J., Rueda C. Valencia F.: Stochastic Behaviour and Explicit Discrete Time in Concurrent Constraint Programming. En ICLP'08 Springer-Verlag. (2008)

[33] Perez H., Muñoz J.A. Aplicación de la programación concurrente por restricciones en el modelado de procesos en la membrana celular. Tesis de pregrado. Universidad Javeriana (2009)

[34] Toro-Bermudez M, Rueda C., Assayag G., Agón C.. Ntccrt: A Concurrent Constraint Framework for Real-time Interaction with examples of integration into OpenMusic and Pure Data (2009)

[35] Villota A., Aranda J., Díaz J.: Modelando Sistemas de Membranas en ntcc. A ser publicado en las memorias del CLEI 2009, (2009)