Tesis defendida por

Fernando Guadalupe Razo Mendivil

y aprobada por el siguiente comité

Dr. Israel Marck Martínez Pérez
Director del Comité

Dra. Clara Elizabeth Galindo Sánchez
Miembro del Comité

Dr. Carlos Alberto Brizuela Rodríguez
Miembro del Comité

Dr. Hugo Homero Hidalgo Silva
Miembro del Comité

Dra. Ana Isabel Martínez García
Coordinador programa de
Posgrado en Ciencias de la Computación

Dr. Jesús Favela Vara
Director
Dirección de Estudios de Posgrado

abril de 2014

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA, BAJA CALIFORNIA

Programa de Posgrado en Ciencias

en Ciencias de la Computación

Inferencia lógica utilizando moléculas de ADN y desplazamiento de cadenas

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Fernando Guadalupe Razo Mendivil

Ensenada, Baja California, México

2014

Resumen de la tesis de Fernando Guadalupe Razo Mendivil, presentada como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Inferencia lógica utilizando moléculas de ADN y desplazamiento de cadenas

Resumen aprobado por:

Dr. Israel Marck Martínez Pérez

Director de Tesis

En esta tesis se presenta la creación de un modelo autónomo para la generación de inferencia lógica utilizando moléculas de ADN y desplazamiento de cadenas que permite utilizar las reglas básicas de la lógica proposicional. Los mecanismos diseñados incluyen las reglas *modus ponens*, *modus tollens*, silogismo hipotético, simplificación, adición, dilema constructivo y *modus tollendo ponens*; además de los principales conectivos lógicos: conjunción, disyunción y proposición condicional. Estas estructuras fueron diseñadas utilizando el lenguaje DSD y simuladas utilizando el simulador estocástico incluído en la herramienta Visual DSD, y posteriormente se analizaron sus propiedades a través de la herramienta PRISM. La generación de inferencia lógica posee un papel muy importante en el cómputo biomolecular, ya que permite brindar la capacidad de reacción a estímulos del ambiente a los dispositivos moleculares, con aplicaciones en áreas como la detección y tratamiento de enfermedades, terapia génica, ingeniería genética y formación de patrones, entre otras.

Palabras Clave: **Biocómputo, desplazamiento de cadenas, Lógica proposicional.**

Abstract of the thesis presented by Fernando Guadalupe Razo Mendivil, in partial fulfillment of the requirements of the degree of Master in Sciences in Computer Sciences.

Logical inference using DNA molecules and strand displacement

Abstract approved by:

Dr. Israel Marck Martínez Pérez

Thesis director

In this thesis we present an autonomous biomolecular computing model based on strand displacement which is capable of performing logical operations with DNA molecules. The model implements most of the inference rules from propositional logic including modus ponens, modus tollens, hypothetic syllogism, simplification, addition, constructive dilemma and modus tollendo ponens; it also includes the most important logical connectors: conjunction, disjunction and conditional proposition. All structures were designed using DSD language, simulated using the Gillespie algorithm, and analyzed using PRISM Model Checker. The ability to implement logical inference is central to molecular programming as it grants molecular devices the advantage of reacting to the environmental stimuli. Potential applications of these devices could be found in detection and treatment of diseases, gene therapy, genetic engineering and pattern formation.

Keywords: **Biocomputing,Strand displacement, Propositional Logic**

Dedicatoria

A mis padres, Eva y Elpidio. A mis hermanos, Julissa, Erika y José.

Agradecimientos

A mis padres, mi orgullo y ejemplo a seguir, gracias por su apoyo y cariño.

A mi director de tesis, el Dr. Israel Marck Martínez Pérez, por ser mi paciente guía durante el desarrollo de este trabajo.

A mi comité de tesis, por sus observaciones y comentarios para mejorar mi investigación.

A mis amigos, por su apoyo incondicional.

A mis compañeros de generación, por su compañía y apoyo durante mi estancia.

A los investigadores del posgrado en ciencias de la computación por su gran enseñanza académica.

Al personal del departamento de ciencias de la computación por hacer amena mi estancia en la institución.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar esta maestría.

Contenido

Lista de figuras

Lista de tablas

El cómputo biomolecular es un área interdisciplinaria de la computación en la cual se desarrollan sistemas derivados de moléculas biológicas como el Ácido Desoxiribonucleico (ADN), el Ácido Ribunocleico (ARN) y proteínas para realizar el procesamiento o almacenamiento de información.

Una de las metas de la nanotecnología y el cómputo biomolecular es la creación de dispositivos biomoleculares capaces de percibir el ambiente intracelular y reaccionar a él, debido a que la visión de esta área incluye la inserción y manipulación de dichos dispositivos en células humanas (Rodríguez-Patón *et al.*, 2011). Algunas aplicaciones de estas moléculas incluyen entre otros a la biomedicina (Benenson *et al.*, 2004; Martínez-Pérez *et al.*, 2007; Labean *et al.*, 2009), en donde se podrían utilizar para detectar enfermedades causadas por mutaciones en el genoma humano (como cáncer y enfermedades hereditarias), para su tratamiento como en la terapia génica, así como en la dosificación de medicamentos a nivel celular. Otra contribución significativa consiste en la aplicación de razonamiento lógico a nivel molecular con fines educativos como muestran Rodríguez-Patón *et al.* (2011) al utilizar un modelo de inferencia para resolver el problema de satisfacibilidad booleana (SAT), el cual es el caso tipo de los problemas NP-Completo. Además, se puede utilizar en áreas como la inteligencia artificial para crear sistemas expertos moleculares (Wasiewicz *et al.*, 1999, 2000; Lee *et al.*, 2003).

La implementación de operaciones lógicas constituye la parte central del control de los dispositivos biomolecular, por lo que un área del cómputo biomolecular se ha dedicado a investigar la implementación de inferencias lógicas a nivel molecular para su uso en nano máquinas (Wasiewicz *et al.*, 1999, 2000; Lee *et al.*, 2003; Ran *et al.*, 2009; Rodríguez-Patón *et al.*, 2011). Sin embargo, los trabajos de investigación actuales se han centrado en la creación de métodos para realizar inferencias de reglas de producción, y se encuentra limitados ya sea por el uso de enzimas, por escalabilidad, por dificultad para interpretar los resultados generados o pertenecen a un modelo no autónomo y requieren que un humano realice las

operaciones de laboratorio necesarias para implementar el cómputo. Actualmente no existe un modelo autónomo para generar inferencias con moléculas de ADN que cuente con la mayoría de las reglas de la lógica proposicional.

1.1. Planteamiento del problema

El problema de investigación consiste en el desarrollo de un modelo de cómputo biomolecular que permita la implementación de la mayoría de las reglas de inferencia de la lógica proposicional, utilizando para esto la técnica de desplazamiento de cadenas para que el método creado funcione de forma autónoma, es decir, sin necesidad de que un usuario realice operaciones de laboratorio sobre ellas.

1.2. Objetivos.

1.2.1. Objetivo General

El objetivo de esta tesis consiste en la creación de un modelo autónomo para la generación de inferencia lógica utilizando moléculas de ADN y desplazamiento de cadenas que permita utilizar los métodos de inferencia básicos de la lógica proposicional.

1.2.2. Objetivos Específicos

- Analizar la teoría de lógica proposicional.
- Investigar los modelos autónomos y herramientas de desplazamiento de cadenas que puedan ser útiles en el diseño del problema.
- Diseñar modelos de inferencias lógicas con moléculas de ADN utilizando los modelos y herramientas seleccionadas.
- Simular los modelos generados.
- Evaluar los resultados de la simulación de los modelos generados.

1.3. Metodología

Para el cumplimiento de los objetivos, primero se planteó el problema de investigación como un problema de diseño, por lo cual fue necesario seleccionar las reglas de inferencia y conectivos lógicos a utilizar, así como los métodos de biocómputo y herramientas computacionales que pudieran permitir llevar a cabo esta tarea.

Posteriormente, utilizando la representación utilizada en el software Visual DSD (Phillips y Cardelli, 2009), se llevó a cabo la propuesta de un diseño de los elementos básicos de la lógica proposicional: proposiciones y conectivos lógicos (conjunción, disyunción y proposición condicional), que es lo suficientemente genérico para ser utilizado en la implementación de las reglas de inferencia lógicas seleccionadas: *modus tollens*, *modus ponens*, simplificación, adición, dilema constructivo, silogismo hipotético y *modus tollendo ponens*.

Con el fin de verificar el desempeño de las estructuras diseñadas, se llevaron a cabo experimentos *in silico* utilizando el simulador incluído en la herramienta Visual DSD, el cual implementa al algoritmo de Gillespie (1977). Además de realizar las simulaciones se llevó a cabo el análisis de las propiedades del modelo utilizando la herramienta PRISM, lo cual permitió verificar que cada una de las estructuras posee un funcionamiento correcto.

Por último, se tomaron ejemplos de razonamientos lógicos de la literatura, los cuales fueron modelados utilizando las estructuras propuestas y posteriormente se realizaron simulaciones correspondientes con Visual DSD y el análisis de propiedades con PRISM.

1.4. Organización de la tesis

El documento se encuentra organizado de la siguiente manera: en el Capítulo 2 se presenta una revisión de los conceptos de lógica proposicional básicos, se analizan las diferentes reglas de inferencia y conectivos lógicos y se incluyen algunos ejemplos para facilitar su comprensión. El Capítulo 3 introduce al lector en conceptos básicos de biología molecular, conceptos que son fundamentales en el cómputo biomolecular, incluyendo el desplazamiento de cadenas.

En el Capítulo 4 se muestran los origenes y conceptos del biocómputo, así como sus principales modelos y áreas de investigación. Se presenta también al lenguaje DSD (*DNA Strand Displacement*) y a la herramienta visual que lo implementa, la cual se utilizó en esta investigación para realizar el diseño de las estructuras. También incluye un segmento sobre PRISM, un analizador de modelos probabilísticos utilizado para probar que los modelos de las estructuras creadas funcionen de forma correcta.

El Capítulo 5 describe el estado del arte del área, trabajos orientados a sistemas expertos moleculares, resolución por refutación y dos trabajos de investigación que implementan de manera autónoma las reglas *modus ponens* y *tollens*.

El Capítulo 6 presenta el modelo de inferencia lógico propuesto, las convenciones de las herramientas utilizadas para su diseño y análisis, y los resultados generados con la simulación estocástica y el análisis de propiedades. En el Capítulo 7 se utilizan las estructuras propuestas para modelar razonamientos tomados de la literatura, y probar que las compuertas poseen la capacidad de ser utilizadas en inferencias lógicas en cascada. Por último, se concluye el trabajo de tesis con una discusión sobre los resultados obtenidos y el trabajo a futuro.

Capítulo 2. Fundamentos de lógica proposicional

La lógica proposicional define un lenguaje formal para representar el conocimiento y realizar inferencias lógicas. Es un sistema formal que posee una sintaxis y reglas de inferencia, sus elementos más simples son las proposiciones y a través de conectivos lógicos se pueden generar proposiciones complejas.

2.1. Proposición

Es la unidad mínima de la lógica proposicional. Es una afirmación que puede ser verdadera o falsa (pero no ambas) y que representa un enunciado simple. Por ejemplo, los enunciados P=*'Sócrates es hombre'* y Q=*'Está lloviendo'* son dos proposiciones independientemente de su valor de verdad (si son verdaderas o falsas). Las proposiciones se pueden unir utilizando conectivos lógicos para crear proposiciones compuestas, en donde el valor de verdad de una proposición compuesta depende de los valores de verdad de las proposiciones simples que la componen y la definición lógica del operador que se utiliza para unirlas.

2.2. Tabla de verdad

Una tabla de verdad es un arreglo el cual contiene el valor de verdad de una proposición compuesta para cada una de las posibles combinaciones de sus componentes más simples. En cada línea de la tabla se muestra una posible combinación de valores de verdad. Por ejemplo, la Tabla 1 muestra la tabla de verdad del operador de negación ($\neg$), el cual invierte el valor de verdad de la proposición a la cual es aplicado.

Tabla 1: Tabla de verdad para el operador de negación

P	$\neg P$
V	F
F	V

Para construir una tabla de verdad primero es necesario determinar el número de renglones, el cual es igual al número de diferentes combinaciones de valores de verdad de las proposiciones simples que conforman a la proposición compuesta. El número de renglones (R) de una tabla de verdad está dado por:

$$R = 2^n,$$

donde n es el número de proposiciones simples en la proposicón compuesta. La Tabla 2 muestra el número de renglones necesarios para una tabla de verdad con 1, 2, 3, 4, 5 y 6 proposiciones simples.

Tabla 2: Cantidad de renglones necesarios para generar tablas de verdad con 1, 2, 3, 4, 5 y 6 proposiciones

Número de proposiciones simples	Líneas necesarias
1	2
2	4
3	8
4	16
5	32
6	64

2.3. Conectivos lógicos

Los conectivos lógicos son operaciones que unen proposiciones para generar proposiciones más complejas. Entre ellos podemos encontrar la conjunción, disyunción y la implicación lógica.

2.3.1. Conjunción

Dadas dos proposiciones P y Q, la proposición compuesta creada al utilizar el operador de conjunción, se denota como $P \wedge Q$, y se lee 'P *y* Q'. Esta proposición sólo será verdadera cuando ambas proposiciones (P y Q) lo sean (Tabla 3).

Tabla 3: Tabla de verdad para el operador de conjunción

P	Q	$P \wedge Q$
V	V	V
V	F	F
F	V	F
F	F	F

2.3.2. Disyunción

Sean P y Q dos proposiciones, la proposición compuesta 'P o Q' creada al utilizar el operador de disyunción, se denota como $P \vee Q$ y es verdadera cuando al menos una de las proposiciones es verdadera (Tabla 4).

Tabla 4: Tabla de verdad para el operador de disyunción

P	Q	$P \vee Q$
V	V	V
V	F	V
F	V	V
F	F	F

2.3.3. Proposición condicional

Sean P y Q dos proposiciones, la proposición condicional '*si* P *entonces* Q' se denota como $P \rightarrow Q$, en este caso la proposición P es llamada *antecedente* y Q es llamada *consecuente*. Esta proposición compuesta solamente será falsa cuando el antecedente es verdadero y el consecuente falso (Tabla 5).

Tabla 5: Tabla de verdad para la proposición condicional

P	Q	$P \rightarrow Q$
V	V	V
V	F	F
F	V	V
F	F	V

2.4. Tautologías y contradicciones

Una tautología es un enunciado el cual es obligatoriamente verdadero, sin importar el valor de verdad de las proposiciones que lo componen. La Tabla 6 muestra los valores de verdad para la proposición compuesta $P \vee \neg P$. Como se puede observar, el valor de verdad del enunciado es siempre verdadero, lo cual significa que es una tautología.

Tabla 6: Tabla de verdad para la proposición compuesta $P \vee \neg P$, la cual demuestra que es una tautología

P	$\neg P$	$P \vee \neg P$
V	F	V
F	V	V

Una contradicción es el caso contrario a una tautología, es decir, un enunciado que siempre es falso independientemente de los valores de verdad de las proposiciones que lo componen. La Tabla 7 muestra los valores de verdad para $P \wedge \neg P$. En este caso, el valor de verdad de la

proposición compuesta siempre es falso, lo cual comprueba que es una contradicción.

Tabla 7: Tabla de verdad para la proposición compuesta $P \wedge \neg P$, la cual demuestra que es una contradicción

P	$\neg P$	$P \wedge \neg P$
V	F	F
F	V	F

2.5. Implicación lógica

Dadas dos proposiciones P y Q, se dice que P implica lógicamente a Q (denotado como $P \implies Q$), si Q es verdadera cuando P es verdadera. A diferencia del enunciado '$P \rightarrow Q$' compuesto por las proposiciones P y Q, la proposición $P \implies Q$ es un enunciado que no depende de sus elementos simples ya que expresa que P implica lógicamente a Q, lo cual significa que '$P \rightarrow Q$' no solamente es verdadero para algunos casos, sino que es una tautología.

2.6. Reglas de inferencia

Las reglas de inferencia son esquemas que establecen una relación entre las premisas (proposiciones iniciales) y la conclusión (proposición final). Entre las más comunes se encuentran el *modus ponens*, *modus tollens*, simplificación, *modus tollendo ponens*, adición, silogismo hipotético y dilema constructivo.

2.6.1. *Modus ponens*

El *modus ponens* consiste de una implicación lógica, en donde el antecedente se tiene como premisa, lo cual permite inferir lógicamente al consecuente. Sean P y Q dos proposiciones. La regla *modus ponens* se denota como $(P \rightarrow Q) \wedge P \implies Q$.

Ejemplo:

'Si está lloviendo, entonces está nublado'

'Está lloviendo'

$\therefore$ 'Está nublado'.

Se extraen las proposiciones:

P='Está lloviendo'.

Q='Está nublado'.

y se escribe $(P \to Q) \wedge P \implies Q$.

$P \to Q$

P

$\therefore Q$

2.6.2. *Modus tollens*

Al igual que el *modus ponens*, el *modus tollens* consiste de una implicación lógica, pero en este caso se tiene la negación del consecuente como premisa, lo cual permite inferir lógicamente la negación del antecedente. Sean P y Q dos proposiciones. La regla *modus tollens* se denota como $(P \to Q) \wedge \neg Q \implies \neg P$.

Ejemplo:

'Si está lloviendo, entonces está nublado'

'No está nublado'

$\therefore$ 'No está lloviendo'.

Se extraen las proposiciones:

P='Está lloviendo'.

Q='Está nublado'.

y se escribe $(P \to Q) \wedge \neg Q \implies \neg P$.

$P \to Q$

$\neg Q$

$\therefore \neg P$.

2.6.3. Simplificación

La simplificación consiste en dos proposiciones unidas a través de un operador de conjunción. Dado que al utilizar este operador sólo se obtiene un valor verdadero cuando ambas proposiciones son verdaderas, es posible inferir lógicamente que ambas proposiciones son verdaderas.

Sean P y Q dos proposiciones, la simplificación puede encontrarse en dos casos:

- $P \wedge Q \implies P$.
- $P \wedge Q \implies Q$.

Ejemplo:

Caso 1:

'Llueve y está nublado'

$\therefore$ 'Llueve'.

Caso 2:

'Llueve y está nublado'

$\therefore$ 'Está nublado'.

de ambos casos pueden identificarse las siguientes proposiciones:

P='Está lloviendo'.

Q='Está nublado'.

El ejemplo se puede reescribir como

Caso 1:

$P \wedge Q$

$\therefore P$.

Caso 2:

$P \wedge Q$

$\therefore Q$.

2.6.4. Adición

En la adición lógica se introducen proposiciones que pueden o no existir en el contexto actual, es decir, dada un proposición P es posible agregar una proposición nueva Q y expresarlas como una disyunción $P \vee Q$.

Ejemplo:

‘ Viajaré en avión’

$\therefore$ ‘Viajaré en avión o en autobús’.

donde se identifican las proposiciones:

A=‘Viajaré en avión’.

B=‘Viajaré en autobús’.

El ejemplo puede escribirse de la siguiente manera:

A

$\therefore A \vee B$.

La regla de adición es una ley muy utilizada ya que permite introducir proposiciones faltantes en un sistema de enunciados, sin embargo, posee la desventaja de que puede utilizarse para agregar proposiciones contradictorias.

2.6.5. *Modus tollendo ponens*

El *modus tollendo ponens* consiste de dos proposiciones unidas con un operador de disyunción, y se tiene como premisa que una de las dos proposiciones es falsa. Dado que la disyunción requiere que al menos una de las proposiciones que la conforman sea verdadera, podemos inferir lógicamente que la proposición restante es verdadera.

Sean P y Q dos proposiciones, el *modus tollendo ponens* presenta los siguientes casos:

- $(P \vee Q) \wedge \neg P \implies Q$
- $(P \vee Q) \wedge \neg Q \implies P$

Ejemplo:

Caso 1:

'El cielo está nublado o despejado.'

'El cielo no está nublado.'

$\therefore$ 'El cielo está despejado.'.

Caso 2:

'El cielo está nublado o despejado.'

'El cielo no está despejado.'

$\therefore$ 'El cielo está nublado.'.

De estos casos se pueden extraer las siguientes proposiciones:

P='El cielo está nublado'.

Q='El cielo está despejado'.

y escribir nuevamente los casos de la siguiente manera:

Caso 1:

$P \vee Q$

$\neg Q$

$\therefore P.$

Caso 2:

$P \vee Q$

$\neg P$

$\therefore Q.$

2.6.6. Silogismo hipotético

Un silogismo es generalmente un argumento conformado por dos premisas y una conclusión. Se le llama silogismo hipotético cuando al menos una de las premisas está estructurada en forma de una condicional (si-entonces). Si ambas premisas son enunciados condicionales se trata de un silogismo hipotético puro y cuando sólo la primera de las premisas es una oración condicional se considera que es un silogismo hipotético mixto. Como se puede observar, el *modus ponens* es un silogismo hipotético mixto.

Sean P, Q y S proposiciones, el silogismo hipotético se escribe de la siguiente manera:

- $(P \to Q) \wedge (Q \to S) \implies P \to S$

Ejemplo:

'Si Sócrates es griego entonces Sócrates es hombre.'

'Si Sócrates es hombre entonces Sócrates es mortal.'

$\therefore$ 'Si Sócrates es griego entonces Sócrates es mortal.'.

De este ejemplo se pueden obtener las siguientes premisas:

G='Sócrates es griego'.

H=‘Sócrates es hombre’.

M=‘Sócrates es mortal’.

y escribir formalmente el ejemplo como:

$G \to H$

$H \to M$

$\therefore G \to M$.

2.6.7. Dilema constructivo

El dilema constructivo consiste de una premisa que es la conjunción de dos premisas condicionales, y una proposicion compuesta que es la disyunción de los antecedentes de las premisas condicionales de forma similar a un *modus ponens*. La conclusión consiste en una disyunción de los consecuentes de las premisas condicionales.

Sean P, Q, R y S proposiciones, el dilema constructivo se representa de la siguiente manera:

- $(P \to Q) \land (R \to S) \land (P \lor R) \implies Q \lor S$

Ejemplo:

‘Si optamos por energía nuclear corremos el riesgo de un accidente nuclear’.

‘pero si optamos por energía convencional, empeoramos el efecto invernadero’.

‘Debemos elegir energía nuclear o energía convencional’.

$\therefore$ 'corremos el riego de un accidente nuclear o empeoramos el efecto invernadero'.

Se separan los enunciados en premisas:

N='Optamos por energía nuclear'.

A='Corremos el riesgo de un accidente nuclear'.

C='Optamos por energía convencional'.

I='Empeoramos el efecto invernadero'.

Con estas premisas se escribe nuevamente el ejemplo de la siguiente manera:

$N \to A$

$C \to I$

$N \vee C$

$\therefore A \vee I.$

2.7. Razonamientos

Un razonamiento es un conjunto de enunciados, donde el último de ellos representa la conclusión y los anteriores son las premisas; es decir, posee la siguiente estructura:

$$P_1 \wedge P_2 \wedge ...P_n \to Q,$$

donde n es un número entero positivo. Las proposiciones P_i, $i \, \epsilon \, N$, son llamadas premisas y Q es la conclusión del razonamiento.

Un razonamiento es válido si la conclusión es implicada lógicamente por las premisas; es decir, si su conclusión es verdadera cuando todas sus premisas lo son:

$$P_1 \wedge P_2 \wedge ... \wedge P_n \implies Q.$$

Para validar que un razonamiento es válido primero es necesario expresarlo formalmente. Por ejemplo, si se desea validar el siguiente razonamiento

> 'Si la lavadora es barata o consume mucha energía, entonces no será negocio para el fabricante. Si la lavadora está pintada de rojo, entonces será negocio para el fabricante. La lavadora es barata. Por lo tanto, la lavadora no está pintada de rojo'.

Primero se separan sus premisas en proposiciones, y a cada una de ellas se les asigna una literal para identificarla:

B= 'La lavadora es barata'.

E= 'La lavadora consume mucha energía'.

N= 'La lavadora es negocio para el fabricante'.

R= 'la lavadora es roja',

Así el razonamiento puede escribirse como $((B \vee E) \rightarrow \neg N) \wedge (R \rightarrow N) \wedge B \implies \neg R$, el cual se puede utilizar en tablas de verdad para demostrar que es una tautología. Sin embargo, puesto que el ejemplo posee 4 premisas, la tabla de verdad tendría 16 renglones y, en problemas más complejos, la tabla de verdad sería tan grande que resultaría imposible demostrarlas utilizado este método.

Otro método utilizado para validar razonamientos lógicos consiste en separarlo en implicaciones más pequeñas, en donde algunas de ellas pueden corresponder a las reglas de inferencias explicadas en la sección anterior (adición, simplificación, *modus ponens* y *modus tollens*, etc). De esta manera, el ejemplo anterior podría representarse como:

$(B \vee E) \rightarrow \neg N$

$R \rightarrow N$

B

$\therefore \neg R.$

Nótese que la línea separa las premisas de la conclusión. Utilizando las reglas de inferencia es posible construir una justificación para el razonamiento. Para esto, se agrega un número al lado izquierdo de cada implicación y al lado derecho se agrega la justificación de por qué es verdadero en función de las líneas anteriores (exceptuando las premisas). A este método se le llama *derivación*, debido a que es una cadena de enunciados conectados a través de meta-enunciados (la justificación utilizada en cada línea). Por ejemplo, el caso anterior puede derivarse como:

1. $(B \vee E) \rightarrow \neg N$
2. $R \rightarrow N$
3. B

---

4. $B \vee E$ (3), Adición
5. $\neg N$ (1),(4), *Modus Ponens*
6. $\neg R$ (2),(5), *Modus Tollens*

A un razonamiento que tiene derivación se le llama *derivable*. Un razonamiento puede tener más de una derivación, por ejemplo, la siguiente es otra derivación posible del ejemplo anterior

1. $(B \vee E) \rightarrow \neg N$
2. $R \rightarrow N$
3. B

---

4. $B \vee E$ (3), Adición

5. $\neg N \rightarrow \neg R$ (1),(4), *Modus Ponens*

6. $(B \neg E) \rightarrow \neg R$ (1),(5), Silogismo hipotético

7. $\neg R$ (4),(6), *Modus Ponens*

Para demostrar que este ejemplo es válido se puede utilizar cualquiera de sus dos derivaciones, no importa que la segunda derivación sea más larga que la primera, lo importante es que existe, ya que un razonamiento es válido si y sólo si es derivable (Bloch, 2000).

Capítulo 3. Fundamentos biológicos

3.1. Nucleótidos

Los nucleótidos son moléculas formadas por un grupo fosfato, una pentosa o azúcar(monosacárido de cinco carbonos) y una base nitrogenada ya sea pirimidina (Citosina, Timina o Uracilo) o purina (Adenina o Guanina), capaces de unirse a través de un enlace fosfodiester (Amos, 2005).

3.2. Ácidos nucleicos

Los nucleótidos son las unidades básicas (monómeros) en la construcción de polímeros llamados ácidos nucleicos, entre ellos se puede encontrar el ácido desoxirribonucleico (ADN) y el ácido ribonucleico (ARN). El enlace fosfodiester creado entre las bases forma la columna vertebral de las cadenas. La forma en que se alternan la pentosa y el fosfato en las cadenas les provee de dirección, el extremo con la pentosa expuesta es conocido como 3', mientras que el extremo con el fosfato expuesto es conocido como 5' (Ignatova *et al.*, 2008).

3.2.1. ADN

En los organismos vivos, la información genética es heredada a través del ADN, el cual es un ácido nucléico que posee una desoxirribosa como azúcar, y está conformado por cuatro nucleótidos o bases: Adenina (A), Guanina (G), Citosina (C) y Timina (T).

La Figura 1 muestra la forma de doble hélice común del ADN de doble cadena (dsADN), la cual se produce cuando dos cadenas sencillas (ssADN) se unen a través de enlaces no covalentes llamados puentes de hidrógeno que se generan por la interacción entre una base purina y una pirimidina. La Adenina se une con la Timina a través de dos enlaces y la Citosina con la Guanina con tres enlaces; a estos pares se les conoce como pares complementarios, de acuerdo al modelo de Watson y Crick (Ignatova *et al.*, 2008).

A pesar de que las cadenas sencillas de ADN (ssADN) no son estables, se les puede encontrar en las células, sobre todo durante su replicación. En este proceso una molécula ssADN es utilizada como plantilla para la generación de una nueva cadena de ADN doble (dsADN) (Amos, 2005).

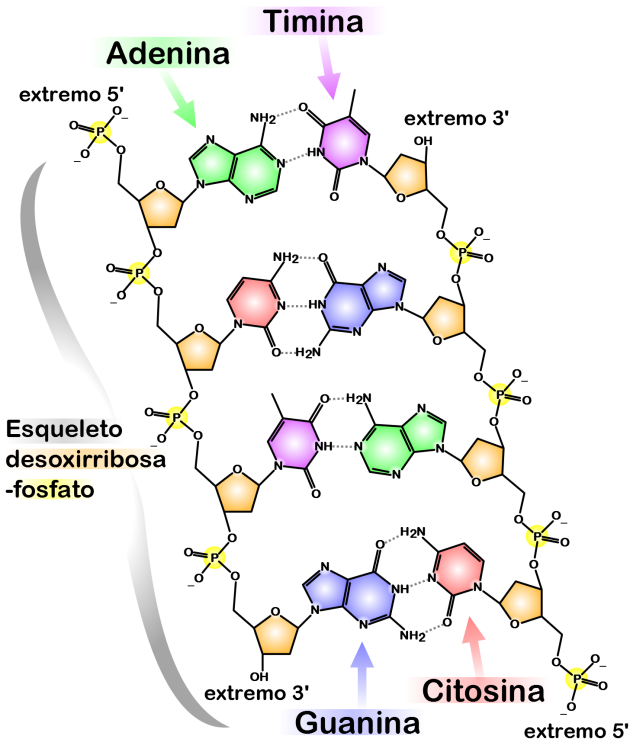


Figura 1: Estructura de la molécula de ADN (obtenida con licencia *Creative commons s/a* 3.0, 2.5, 2.0, 1.0).

3.3. Enzimas

Las enzimas son proteínas que se encuentran en las células y poseen una gran importancia como catalizadores en sus procesos metabólicos, incluyendo la replicación, transcripción y traducción del ADN. Algunas reacciones no se llevan a cabo sin la presencia de una enzima. Cada enzima cataliza un proceso distinto y su nombre está relacionado con dicho proceso, como las enzimas de restricción que localizan y eliminan secuencias en el ADN, las enzimas exonucleasas que eliminan nucleótidos al final de una cadena y las enzimas endonucleasas que eliminan segmentos en regiones dentro de las cadenas. También existe la enzima ligasa

que permite la creación de enlaces fosfodiester entre cadenas de ADN (Amos, 2005).

Algunos investigadores han aprovechado esta interacción entre enzimas y moléculas de ADN para realizar operaciones básicas de cómputo biomolecular.

3.4. Termodinámica

3.4.1. Hibridación y desnaturalización

Dos cadenas sencillas de ADN se unen o hibridan produciendo una cadena doble de ADN si son complementarias en dirección inversa, es decir, una tiene una dirección 5' a 3' y su cadena complementaria es su inversa en la dirección 3' a 5'. Una ssADN puede interactuar con otra a pesar de no ser totalmente complementarias produciendo una cadena doble imperfecta. La temperatura posee un papel importante en el proceso de hibridación de las cadenas de ADN, ya que al disminuir la temperatura disminuye también el rigor de hibridación, el cual es el grado de interacción entre los nucleótidos de las dos cadenas ssADN. Cuando el rigor de hibridación disminuye las cadenas pueden hibridar con más pares de bases aunque no sean complementarias, mientras que bajo condiciones de alto rigor (altas temperaturas) una cadena doble comienza a separarse en dos cadenas sencillas, esto es llamado desnaturalización. A la temperatura en la cual más de la mitad de las bases complementarias son desnaturalizadas se le conoce como temperatura de fusión $Tm(^{\circ}C)$ (Amos, 2005).

3.5. Modelo del vecino más cercano

Actualmente el método más utilizado para la predicción de reacciones de hibridación de complejos ADN/ADN es el vecino más cercano (SantaLucia y Hicks, 2004). Aunque existen diversos métodos dependiendo el tipo de complejo, el vecino más cercano posee una gran exactitud y es muy fácil de calcular. De acuerdo a este modelo, la estabilidad de una molécula dsADN depende básicamente del par de bases que conforman al vecino más cercano de entre 10 posibles interacciones que pueden ocurrir en cualquier cadena dsADN: AA/TT, AT/TA,

TA/AT, CA/GT, GT/CA, CT/GA, GA/CT, CG/GC, GC/CG, y GG/CC (con dirección 5'-3'/3'-5').

La estabilidad y el comportamiento de las cadenas respecto a su rigor de hibridación puede ser descrito por tres factores: la energía libre de Gibbs, la entalpía y la entropía.

3.6. Energía libre de Gibbs, entalpía y entropía

La energía libre de Gibbs ($\triangle G^\circ$) denota la posibilidad de que una reacción ocurra espontáneamente, mientras que la entalpía describe la facilidad con la cual la energía es liberada o absorbida por el sistema y la entropía es la medida de desorden del sistema.

La energía libre de Gibbs se puede calcular de la siguiente manera (Ignatova *et al.*, 2008):

$$\triangle G^\circ = \triangle H^\circ - T\triangle S^\circ, \tag{1}$$

donde $\triangle H^\circ$ es la entalpía y $\triangle S^\circ$ es la entropía del sistema.

En una molécula dsADN compuesta por una hebra $x = a_1...a_n$ y su cadena inversa complementaria $\bar{a_n}...\bar{a_1}$, la energía de Gibbs puede calcularse de la siguiente manera:

$$\triangle G^\circ(x) = \triangle g_i + \triangle g_s + \sum_{i=1}^{n-1} \triangle G^\circ(a_i a_{i+1}/\bar{a_i}\bar{a_{i+1}}), \tag{2}$$

donde $\triangle g_i$ corresponde a la energía de iniciación de hélice, $\triangle g_s$ es la corrección de simetría y $\triangle G^\circ(a_i a_{i+1}/\bar{a_i}\bar{a_{i+1}})$ es la energía de la dupla $a_i a_{i+1}/\bar{a_i}\bar{a_{i+1}}$. Por ejemplo, considerando los datos de la Tabla 8 y la siguiente molécula de ADN:

5' - GCAATGGC - 3'

3' - CGTTACCG - 5'

La energía libre de Gibbs se puede calcular como

$$\triangle G^\circ = \triangle g_i + \triangle g_s + \triangle G^\circ(GC/CG) + \triangle G^\circ(CA/GT) + \triangle G^\circ(AA/TT)$$
$$+ \triangle G^\circ(AT/TA) + \triangle G^\circ(TG/AC) + \triangle G^\circ(GG/CC) + \triangle G^\circ(GC/CG)$$

$$= 1.96 + 0.0 - 2.24 - 1.45 - 1.0 - 0.88 - 1.44 - 1.84 - 2.24$$

$$= -9.13 kcal/mol$$

Tabla 8: Termodinámica del vecino más cercano en 1M NaCl a $37°C$. Las unidades para $\triangle G°$ y $\triangle H°$ son $kcal/mol$ de interacción y las unidades de $\triangle S°$ son en cal/K por mol de interacción. La correción por simetría es aplicada solamente a duplas auto-complementarias. La penalización por terminal aplica a cada dupla que finaliza con la secuencia AT. Una dupla con ambas terminales cerradas por pares AT tiene una penalización de $+1.0$ $kcal/mol$ para $\triangle G°$ (Ignatova *et al.* 2008, p. 64).

Interacción	$\triangle H°$	$\triangle S°$	$\triangle G°$
AA/TT	-7.6	-21.3	-1.00
AT/TA	-7.2	-20.4	-0.88
TA/AT	-7.2	-21.3	-0.58
CA/GT	-8.5	-22.7	-1.45
GT/CA	-8.4	-22.4	-1.44
CT/GA	-7.8	-21.0	-1.28
GA/CT	-8.2	-22.2	-1.30
CG/GC	-10.6	-27.2	-2.17
GC/CG	-9.8	-24.4	-2.24
GG/CC	-8.0	-19.9	-1.84
Inicialización	+0.2	-5.7	+1.96
Penalización por terminal AT	+2.2	+6.9	+0.05
Correción de simetría	0.0	-1.4	+0.43

Capítulo 4. Cómputo biomolecular

4.1. Origenes del cómputo biomolecular

El cómputo biomolecular es un área de la computación en la cual se sustituye el uso de instrumentos basados en silicón por moléculas de origen biológico como ADN, ARN y proteínas. En el año 1959, el físico ganador del premio Nobel Richard Feynman propuso en su plática '*There's plenty of Room at the bottom*' por primera vez la idea de utilizar elementos a escala molecular para fabricar dispostivos funcionales, fundando con esto el área de la nanotecnología (Feynman, 1960). Las siguientes investigaciones realizadas en este tema fueron de carácter teórico, hasta que en 1994 Richard Adleman demostró como puede ser implementada una búsqueda aleatoria masiva en paralelo utilizando operaciones básicas en cadenas de ADN para resolver el problema del camino Hamiltoniano de 7 nodos, el cual pertenece a la clase NP-Completo, es decir, pertenece a la clase NP por lo cual su solución puede ser evaluada en tiempo polinomial y cualquier problema NP puede ser reducido a él (Adleman, 1994). Debido a que el modelo generaba todas las posibles soluciones y las filtraba para encontrar la solución con características deseadas, a este método se le conoce como modelo de filtrado.

El trabajo de Adleman es considerado básico para el cómputo molecular, de aquí se derivaron un gran número de trabajos de investigación utilizando moléculas biológicas, no obstante el modelo de Adleman no era un modelo formal computacionalmente, ya que era específico para el problema del camino Hamiltoniano. En 1995 Richard Lipton extendió el modelo de Adleman para que pudiera ser utilizado en cómputo biomolecular en general, no solamente en solución de problemas de caminos Hamiltonianos. En su artículo Lipton (1995) muestra cómo utilizar ADN para resolver problemas NP-completos al resolver el problema de satisfacibilidad (SAT), el cual es el punto de referencia para los problemas NP-completos. Juris Hartmanis, uno de los fundadores de la teoría de complejidad computacional publicó

un artículo (Hartmanis, 1995) sobre cómputo biomolecular en el cual calculaba que escalar el problema del experimento de Adleman de 7 a 200 nodos requeriría un conjunto de cadenas de ADN iniciales cuyo peso sería mayor que el del planeta Tierra.

A pesar de esta publicación, el cómputo biomolecular continuó progresando y actualmente existe un gran número de modelos, los cuales pueden ser agrupados en dos categorías principales en base a la forma en que se realizan las operaciones computacionales.

4.2. Modelos de cómputo biomolecular no autónomos

Los modelos no autónomos son aquellos que requieren que las operaciones sean realizadas en un laboratorio por un ser humano, los primeros modelos de cómputo biomolecular tenían este enfoque. Se utilizaron originalmente para resolver problemas NP-Completos en tiempo polinomial, lo cual a la fecha es imposible de realizar con cómputo basado en silicón, sin embargo presentaban los inconvenientes de utilizar operaciones lentas, ser susceptibles a errores y requerir grandes volúmenes de ADN, lo cual limitaba la implementación de problemas a gran escala (Hartmanis, 1995).

4.2.1. Modelo de filtrado

Propuesto por Adleman y extendido por Richard Lipton, en este método se crea un conjunto de cadenas de ADN que codifican las variables, se hibridan las variables para obtener secuencias dsADN y se filtran las cadenas obtenidas para encontrar una solución con las características deseadas.

4.2.2. Modelo de etiquetas

En 1996 Adleman introdujo el modelo de etiquetas (Roweis *et al.*, 1996), el cual es una implementación de la máquina de registros. Este modelo incluye una memoria de acceso aleatorio conformada por conjuntos de complejos de memoria (cadenas de ADN parcialmente hibridadas) llamados tubos y una serie de operaciones que pueden ser aplicadas a ellos:

- Unir. Combina el contenido de dos o más tubos y deposita la mezcla en un tubo nuevo.
- Separar. Dado un valor i, separa los complejos de memoria de un tubo dado depositando en un nuevo tubo los complejos cuyo i-ésimo bit está activado y en otro tubo nuevo los complejos con el i-ésimo bit no activado.
- Encender. Dado un valor i y un tubo dado, activa el i-ésimo bit de todos los complejos de memoria en el tubo seleccionado.
- Apagar. Dado un valor i y un tubo dado, desactiva el i-ésimo bit de todos los complejos de memoria en el tubo seleccionado. Esta es la operación que presenta mayor dificultad debido a la naturaleza del ADN, pero puede evitarse su uso sin comprometer la funcionalidad computacional.
- Desechar. Toma un tubo y desecha su contenido.

4.2.3. Modelo de concatenación (*splicing*)

Este método es basado en las operaciones propuestas por Tom Head (1987) para la recombinación de ADN. Se ha comprobado que el poder generativo de los sistemas de concatenación permiten simular cualquier máquina de Turing.

Se utiliza un tubo de ensayo que contiene la población inicial en forma de cadenas dsADN (al utilizar cadenas dobles se evitan problemas de hibridación errónea), enzimas de restricción y enzima ligasa. La recombinación consiste en la partición de las cadenas dsADN que se realiza con las enzimas de restricción en zonas de reconocimiento específico. Posteriormente, los segmentos generados se unen con otros a través de la ligasa, produciendo nuevas cadenas dsADN.

4.2.4. Cómputo biomolecular con superficies

Introducido por Robert Corn *et al.* en 1998 , en este modelo se realiza la codificación de todas las posibles soluciones de un problema a través de mezclas combinatorias complejas de

ADN que son adheridas a una superficie. El cómputo se realiza en N ciclos en cada uno de los cuales se realizan dos operaciones en secuencia:

- Marcado (*Mark*). Se agrega la mezcla combinatoria deseada de ADN a la superficie, las cadenas que encuentren a su complementaria hibridarán marcando esa cadena; aquellas cadenas que no la encuentren quedarán sin marcar.

- Destruir (*Destroy*). Se agrega una enzima exonucleasa específica para cadenas ssADN, por lo que todas las cadenas no marcadas son destruídas.

En cada uno de los ciclos se agregan oligonucleótidos complementarios a un subgrupo de las cadenas de la superficie para marcarlos, y luego se realiza la destrucción de las cadenas no marcadas. Este procedimiento se repite con nuevos subgrupos de oligonucleótidos hasta que las cadenas restantes en la superficie representan la solución al problema a resolver (si existe alguna).

4.3. Modelos de cómputo biomolecular autónomos

Estos modelos pertenecen a la segunda generación de cómputo biomolecular, la cual aprovecha las propiedades de auto-organización del ADN y en algunas ocasiones se utilizan enzimas en los procesos de cómputo, brindándoles autonomía y la capacidad de ser parcialmente programables. Los modelos básicos que pertenecen a esta categoría se describen a continuación:

4.3.1. Auto-ensamblamiento de ADN

Este método fue propuesto por Erik Winfree (Winfree *et al.*, 1996), basado en la teoría de autoensamblamiento de ADN. La idea es aprovechar la capacidad de hibridación inherente de las moléculas de ADN para realizar cómputo universal en un plano 2D. El cómputo es realizado a través de la creación de estructuras bidimensionales que pueden ser vistas como polígonos o mosaicos que poseen la capacidad de unirse a otras estructuras a través

de extremidades salientes (segmentos de nucleótidos de cadena sencilla). Estas estructuras pueden auto-ensamblarse de forma algorítmica formando un plano bidimensional.

4.3.2. Modelo de autómata de estado finito

El primer modelo de autómata de estado finito utilizando ADN automatizado fue introducido por Shapiro *et al.* en 2001 (Benenson *et al.*, 2001), basándose en la máquina de Turing creada por Rothemund en 1996 (Wilhelm y Rothemund, 1996). El autómata de Shapiro consiste en hardware, software y una entrada. El hardware se conforma de la enzima ligasa y la endonucleasa *FokI*, el software (reglas de transición) y la entrada se codifican en dsDNA y la programación es realizada seleccionando moléculas de software adecuadas.

Una vez que se mezclan las soluciones que contienen al hardware, software y entrada, el autómata procesa la entrada a través de una serie de cortes por la enzima de restricción, hibridación y ligación por la enzima ligasa. En cada corte, la cadena resultante expone el estado actual y un extremo pegajoso saliente que codifica la siguiente entrada. Esta molécula hibrida con la molécula de transición correspondiente y el proceso se repite hasta que no se encuentra una transición aplicable o se encuentra un terminador. El resultado producido es una molécula que codifica el estado final del autómata.

Este modelo presenta la desventaja de sólo poder utilizar dos símbolos, además que el estado final debe ser igual al estado inicial. Kuramochi y Sakakibara trataron de resolver este problema al introducir un modelo para autómata finito general (Kuramochi y Sakakibara, 2006), sin embargo no resuelven el problema del estado inicial y final. En 2009, Martínez Pérez *et al.* (Martínez-Pérez *et al.*, 2009), presentan un modelo de autómata finito con el cual resuelven el problema de que el estado final e inicial deban ser el mismo.

4.3.3. Modelos de lazo (*Hairpin*)

Este modelo fue presentado por Hagiya *et al.*(1997) y desarrollado por Erik Winfree (1998). Originalmente este método fue llamado *polymerization stop* debido a que extiende el

extremo 3' de un lazo en una molécula ssADN. Después el modelo fue modificado para que el ciclo en la molécula fuera eliminado con enzimas; hoy es llamado *whiplash PCR* (*polymerase chain reaction*).

4.3.4. Desplazamiento de cadenas

El desplazamiento de cadenas es una reacción termodinámica que se lleva a cabo gracias al intercambio de puntos de apoyo (*toeholds*). En ella, dos cadenas que son parcial o totalmente complementarias se unen desplazando cualquier cadena de menor tamaño que haya hibridado previamente con ellas; esto debido a que las cadenas con mayor número de nucleótidos complementarios poseen un mayor equilibrio termodinámico.

El proceso comienza en los *toeholds*, los cuales son dominios cortos complementarios de cadena sencilla y una vez que éstos se hibridan, se continúan uniendo a través de un procedimiento parecido a una caminata aleatoria llamado *branch migration* (migración de ramificaciones).

Figura 2: Mecanismo de desplazamiento de cadenas. Adaptado de Sainz de Murieta *et al.* (2011).

La Figura 2 presenta el mecanismo del desplazamiento de cadenas. En ella se muestra una cadena dsADN parcialmente hibridada compuesta de las cadenas A y B. Al introducirse la hebra dsADN A', la cual complementa totalmente a la cadena A, ambas cadenas reaccionan a través de sus puntos de apoyo (marcados en rojo) y la hebra A' desplaza a la hebra B' en un proceso que es parcialmente reversible hasta unirse completamente con su cadena complementaria A.

A diferencia de otros procesos, este método no necesita de enzimas, sin embargo su principal desventaja consiste en que una vez que se forma una cadena dsADN completa el proce-

dimiento es irreversible. Como se muestra en la Figura 2, una vez unidas las hebras A y A', el complejo generado no puede reaccionar con otros complejos.

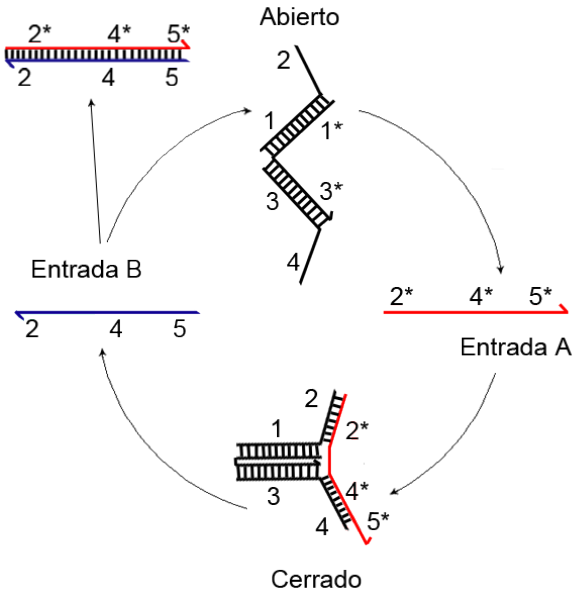


Figura 3: Funcionamiento de un par de pinzas moleculares. La entrada A hibrida con los dominios 2 y 4 de las pinzas forzándolas a mantenerse cerradas. Al agregar la entrada B, su dominio 5 se une al dominio 5* de la entrada A y comienza a desplazarse hasta que las entradas forman una cadena dsADN completa, consumiendo las entradas (combustible) y abriendo las pinzas. Adaptado de Yurke *et al.* (2000).

Bernard Yurke introdujo esta técnica diseñando un par de pinzas utilizando moléculas de ADN (Yurke *et al.*, 2000). Las pinzas tienen la capacidad de tomar, sujetar y liberar un cadenas sencillas de ADN y su funcionamiento está basado en el desplazamiento de cadenas.

Georg Seeling y colaboradores (2006), utilizaron desplazamiento de cadenas para implementar compuertas lógicas. Dicho trabajo fue pionero en el área y tras ellos se han realizado un gran número de investigaciones aprovechando esta técnica, entre ellas el algebra modelada por Luca Cardelli (2009) con la cual se puede realizar una descripción formal tanto de los complejos de ADN como de las operaciones de desplazamiento de cadenas, y que permitió la generación de un lenguaje de programación capaz de implementar operaciones de ADN *in silico* (Phillips y Cardelli, 2009).

4.4. Herramientas computacionales

4.4.1. El lenguaje DSD (*DNA Strand Displacement*)

El lenguaje DSD (Phillips y Cardelli, 2009) fue creado en base al álgebra de cadenas diseñado por Luca Cardelli (2009). Este lenguaje posee la sintaxis necesaria para representar estructuras de moléculas de ADN de una forma concisa. La representación de moléculas se limita a aquellas que no poseen estructuras secundarias, es decir, aquellas que poseen una cadena doble como base y adherida a ella sólo hay cadenas sencillas. Limitar la representación a este tipo de estructuras permite modelar un gran número de sistemas al mismo tiempo que simplifica la definición de la semántica al excluír estructuras complicadas.

La sintaxis del lenguaje está conformado por dominios de secuencias que pueden ser cortos (T^) o largos (T), dependiendo del número de nucleótidos que los conforman. Normalmente un dominio corto como un *toehold* posee de 4 a 10 nucleótidos, mientras que un dominio largo está compuesto por más de 20 nucleótidos. Una secuencia es formada por uno o más dominios y representa una cadena sencilla de ADN (ssADN). En la Figura 4(a) se muestra la cadena S que es una ssADN superior es decir, un secuencia con orientación de 5'—3', la

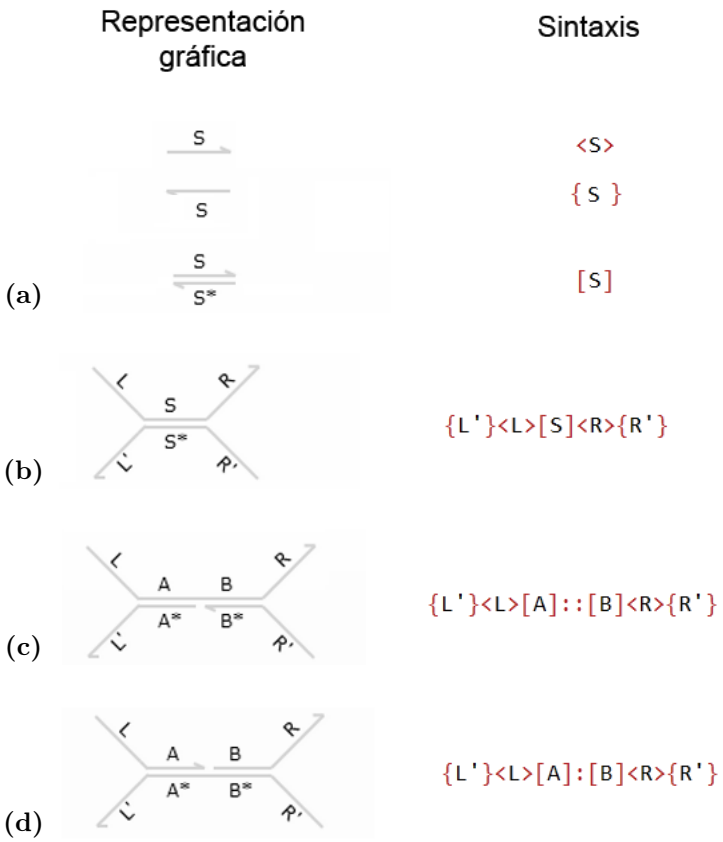


Figura 4: Representación de moléculas de ADN utilizando lenguaje DSD. (a) Cadena superior (con orientación de izquierda a derecha) S, cadena inferior (con orientación derecha a izquierda) S y cadena doble formada por las secuencias S y S^*. (b) Estructura creada al unir dos secuencias en un dominio complementario, una cadena ssADN sobresale en cada extremo de la estructura. (c) Complejo creado al unir dos compuertas a través de la cadena superior. (d) Estructura creada al unir dos compuertas utilizando la cadena inferior. Adaptado de Cardelli *et al.* (2012).

cual es representada en leguaje DSD como $<S>$. También se muestra una cadena inferior S la cual es una secuencia con orientación de 3'—5', representada en lenguaje DSD como $\{S\}$. Por último, una cadena doble de ADN S que consiste en una secuencia superior unida a una secuencia inferior complementaria, se presenta utilizando la instrucción $[S]$ para su codificación.

Una compuerta es una estructura que posee una forma parecida a la mostrada en la Figura 4(b). Dicha molécula consiste en dos segmentos ssADN unidos en un dominio complementario (S), resultando en una estructura central dsADN con cuatro extremos ssADN. En general, una compuerta puede ser creada al unir segmentos de secuencias o compuertas más sencillas, tal como se muestra en las Figuras 4(c) y 4(d) al unir compuertas a través de la cadena

superior e inferior, utilizando los operadores '::' y ':', respectivamente.

4.4.2. Visual DSD

La herramienta Visual DSD[1] es una implementación del lenguaje DSD que puede ser utilizado en modo web y consola en sistemas Windows y Mac, y únicamente en modo consola en sistemas Linux (utilizando OCaml Bytecode). Visual DSD brinda la capacidad de diseñar sistemas basados en desplazamiento de cadenas, incluye librerías de elementos básicos como dominios y *toeholds*, además de permitir al usuario introducir secuencias personalizadas.

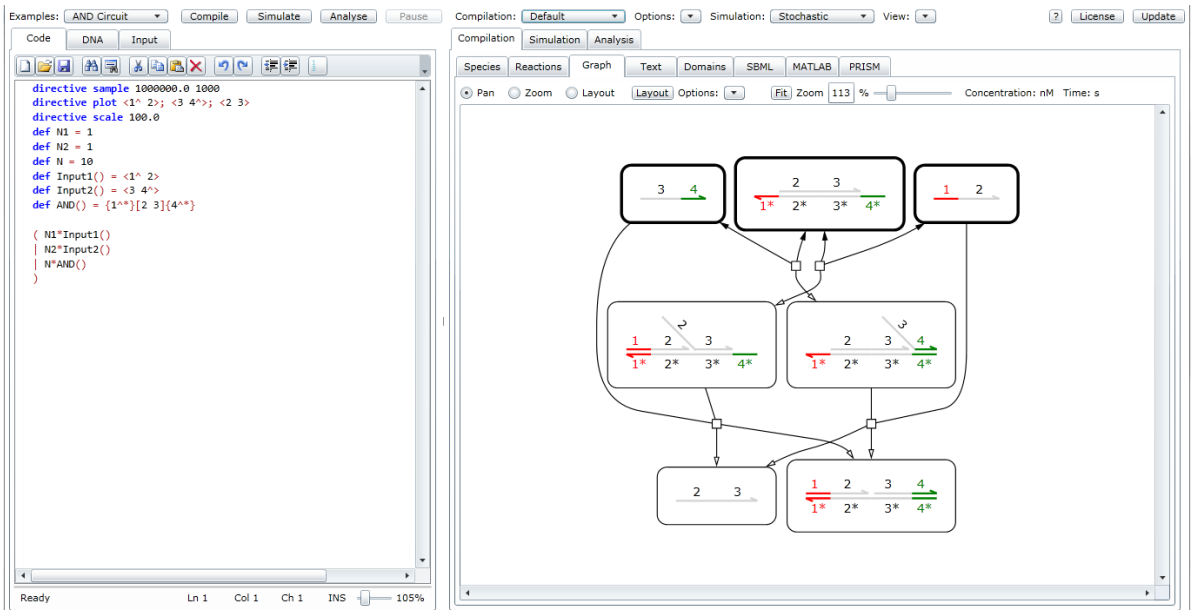


Figura 5: Captura de pantalla de la herramienta Visual DSD. En el extremo izquierdo se encuentra el editor en el cual se muestra el código del ejemplo de un circuito AND incluído en la herramienta. Al lado derecha se muestra la gráfica de reacciones generada al compilar el ejemplo.

La herramienta posee tres diferentes algoritmos de simulación. El simulador estocástico utiliza el algoritmo de Gillespie (Gillespie, 1977) para crear una trayectoria del sistema en el tiempo, una vez que la red de reacciones químicas ha sido calculada. El simulador determinístico utiliza el método de Runge-Kutta-Fehlberg (Fehlberg, 1969) para resolver un modelo de equaciones diferenciales ordinarias y producir una serie determinística suave de concentraciones evolucionando en el tiempo.

[1] http://lepton.research.microsoft.com/webdna/

La tercera opción es una extensión para utilizar polímeros de ADN, o para redes de reacciones químicas que son muy grandes. Estos sistemas pueden llegar a tomar un gran tiempo de compilación, por lo cual el simulador JIT (*Just In Time*) (Paulevé *et al.*, 2010), genera solamente la población inicial de las especies, y la simulación calcula la red de reacciones químicas gradualmente conforme se necesita en cada paso de la simulación estocástica. Debido a que el esquema de reacciones se crea gradualmente, ejecutar la misma simulación en múltiples ocasiones en modo JIT puede producir diferentes esquemas de reacciones.

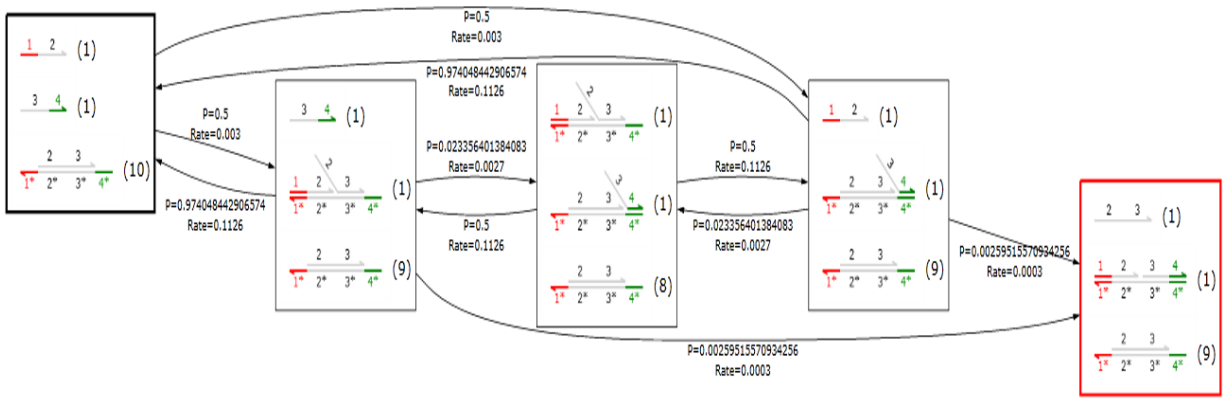


Figura 6: Diagrama de estados y transiciones generado por Visual DSD para el circuito AND incluído en la herramienta. El estado marcado en negro es el punto inicial, el marcado en rojo es el estado final.

Visual DSD posee varios tipos de visualización de las moléculas diseñadas, como la representación gráfica de las especies. Las reacciones pueden visualizarse en forma de secuencia o en forma gráfica como en la Figura 5. También es posible la visualización de la red de reacciones químicas (cadenas de Markov en tiempo contínuo). La Figura 6 muestra la visualización del diagrama de estados y transiciones para el ejemplo del circuíto AND inlcuído en la herramienta.

Los resultados de simulación se visualizan a través de gráficas de líneas de concenctración de especies en el tiempo. Además, es posible graficar funciones aritméticas simples realizadas sobre las poblaciones. Esta información también pueden ser exportada en archivos separados por comas para su posterior análisis. Los modelos generados en Visual DSD pueden ser

exportados en formato de redes de reacciones químicas (SBML) y las cadenas de Markov en tiempo contínuo pueden ser exportadas para su posterior revisión en el analizador de modelos probabilísticos (PRISM).

4.4.3. La herramienta PRISM (*Probabilistic Model Checker*)

La herramienta PRISM es un analizador de modelos probabilísticos desarrollado en las universidades de Birmingham y Oxford. Esta herramienta soporta entre otros el modelo de Cadenas de Markov en tiempo contínuo (CTMCs), el cual es utilizado para modelar sistemas de reacciones a nivel molecular debido a que las transiciones entre estados poseen probabilidades con valores reales positivos, provenientes de una distribución exponencial negativa.

El análisis de modelos es una técnica formal en la cual se construye y analiza exhaustivamente un modelo de estado finito del sistema a verificar. El modelo consiste en una serie de estados que representan las posibles configuraciones del sistema y una tabla de transiciones, donde cada transición representa un posible movimiento de un estado a otro. Un modelo puede ser ampliado con penalizaciones y recompensas, los cuales son valores reales que se asocian a un estado o transición. PRISM soporta solamente el uso de recompensas, debido a que la diferencia entre ambos consiste sólo en la percepción general de que uno es 'bueno' y el otro 'malo'. Una recompensa de estado puede representar la medida de una propiedad de interés en un tiempo determinado o la velocidad a la que una medida se acumula en el tiempo, mientras que una recompensa de transición se acumula cada vez que una transición ocurre.

Una de las funcionalidades más importantes de PRISM es la de verificar propiedades del modelo, ya sean expresadas en lógica CSL o en su extensión basada en recompensas (Kwiatkowska *et al.*, 2011); de este modo, si se desea verificar que 'el sistema siempre alcanza un estado final correcto', utilizando CSL se verifica 'cual es la probabilidad de que el sistema alcance un estado final correcto' o 'cual es la probabilidad de que se alcance un estado final correcto en un tiempo T'. Utilizando la verificación de propiedades se puede analizar si el

modelo es correcto. Para esto es necesario identificar los estados en el modelo en los cuales la finalización ha sido exitosa ("$Estado\_Final$"), las propiedades del sistema están expresadas en lógica temporal CTL (*Computation Tree Logic*) o TLT (*Linear-Time Temporal Logic*), por lo cual se pueden utilizar consultas en lógica temporal (CTL) para realizar la verificación del modelo:

$$\text{E [F "}deadlock\text{"]}$$

donde "$deadlock$" es cualquier estado en el que no puede realizarse una transición a otro; es decir, un estado final que puede o no ser deseado. E y F, son palabras reservadas del lenguaje CTL, E es el operador existencial y F evalúa si la condición se cumple eventualmente en el sistema. Esta fórmula evalúa si existe al menos una ruta en el sistema con la cual se puede alcanzar un estado final independientemente de si es deseado o no.

La verificación de propiedades cuantitativas se lleva a cabo utilizando las fórmulas de lógica temporal (CLS) de PRSIM y a diferencia de las verificaciones con lógica CTL, éstas producen un valor probabilístico como resultado, por ejemplo:

$$\text{P = ? [F "}deadlock\text{" \& "}Estado\_Final\text{"]}$$
$$\text{R \{ "}Tiempo\text{" \} = ? [F "}Estado\_Final\text{"]}$$

La primera consulta calcula la probabilidad de que se alcance un estado final ("$deadlock$") y que dicho estado cumpla las características de un estado final exitoso ("$Estado\_Final$"); la segunda consulta está basada en recompensas, se utiliza la etiqueta "$Tiempo$" para referirse a la recompensa acumulada por cada transición realizada en el sistema. Esta consulta calcula el tiempo en el cual se alcanzará un estado final exitoso ("$Estado\_Final$").

PRISM también permite la creación de experimentos en los cuales se realiza la verificación de propiedades sobre una variable. Por ejemplo, la consulta

$$\text{P = ? [F [T,T] "}deadlock\text{"]}$$

especifica $[T, T]$ que será evaluada sobre la variable T (tiempo) en el periodo dado, por lo cual realiza el cálculo de la probabilidad de alcanzar un estado final en cada unidad de tiempo. La interfaz permite especificar si se desea realizar el cálculo para un valor de la variable. También es posible elegir un valor inicial y final y especificar el tamaño de los incrementos a realizar para alcanzar el valor final. Cuando la verificación se lleva a cabo sobre una serie de valores se puede realizar una gráfica de los resultados. La Figura 7 muestra la gráfica generada para una serie de experimentos realizados sobre un modelo con múltiples estados finales.

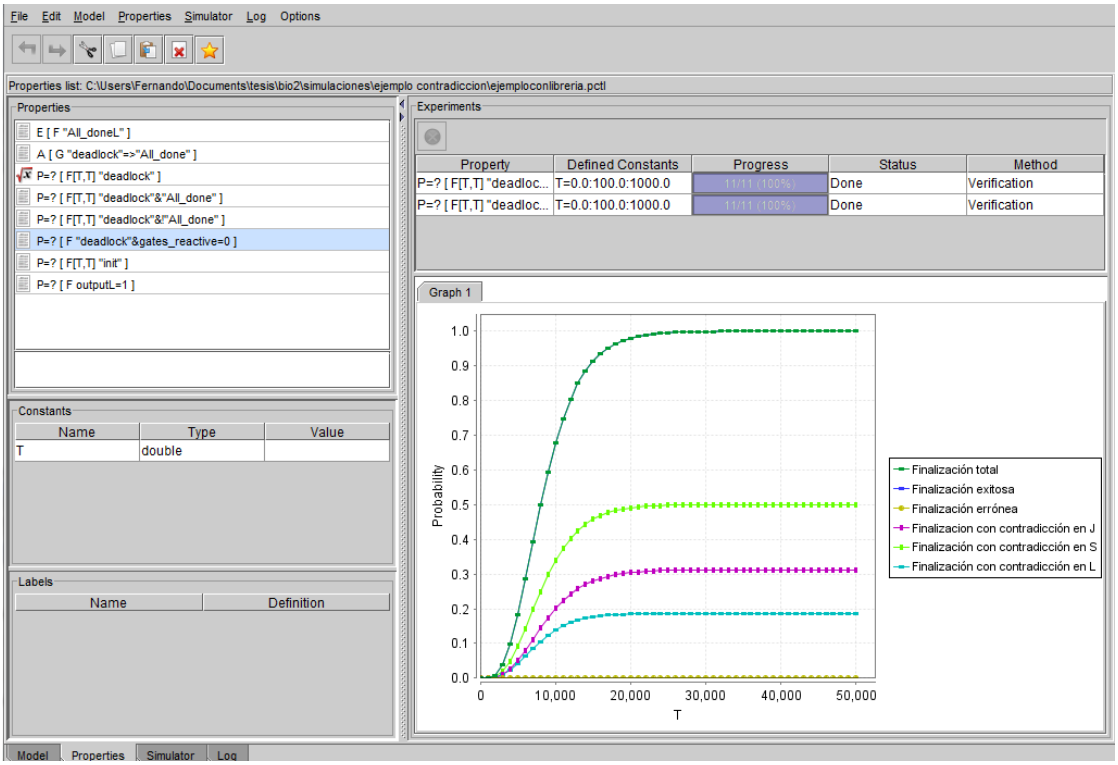


Figura 7: Captura de pantalla de la herramienta PRISM. Se muestra la lista de propiedades en lógica temporal al lado izquiero superior, en el lado izquierdo inferior se muestra la lista de variables, en la parte derecha arriba se muestran los experimentos realizados y en la gráfica se muestran los resultados de los experimentos.

Capítulo 5. Investigación previa

Los primeros modelos biomoleculares creados para generar inferencia lógica pertenecen a la categoría de modelos no autónomos y se enfocan principalmente en aplicaciones de sistemas expertos. El primero fue creado en 1998 por J. Mulawka (Mulawka *et al.*, 1998), quien desarrolló un modelo que hace uso del modelo de etiquetas para realizar encadenamiento hacia atrás, esto es, los complejos de memoria se utilizan para codificar las reglas y las etiquetas se utilizan para codificar hechos e hipótesis.

En 1999, Piotr Wasiewicz y colaboradores (1999), presentaron un modelo de generación de inferencias capaz de resolver reglas del tipo si-entonces utilizando moléculas de ADN circulares derivadas de plásmidos. En su modelo, las premisas y reglas se codifican en cadenas dsADN con un segmento o *sticky end* y se colocan en una solución sobre la cual se aplican operaciones de recocido, ligación, PCR y electroforesis en gel. El resultado consiste en la creación de una molécula circular perfecta.

Un año después, Piotr Wasiewicz, Tomasz Janczak y Jan J. Mulawka (2000) crearon un modelo para la generación de inferencias a través de reglas de producción (si-entonces), utilizando cadenas ssADN con orientación 3'—5' para la codificación de premisas y conclusiones, así como cadenas ssADN con orientación 5'—3' para las reglas. Éstas poseen nucleótidos complementarios a una premisa y una conclusión, de modo que al hibridarlas se obtiene una cadena con una conclusión en un extremo.

In-Hee y colaboradores (Lee *et al.*, 2003), desarrollaron un modelo de refutación por resolución para la comprobación de lógica proposicional utilizando codificación en cadenas ssADN de proposiciones ordenadas de manera que al hibridar producen una molécula dsADN. El mismo modelo fue extendido al uso de lazos de cadenas de ADN parcialmente hibridados (*hairpins*), sin embargo el modelo en ambos casos presenta un gran número de desventajas, como la formación de falsos positivos independientemente de la codificación, el tener que elegir la estructura de la cadena inicial o la posible hibridación de dos lazos.

5.1. Investigación previa con modelos autónomos

Tom Ran, Ehud Shapiro y Shai Kaplan publicaron en el año 2009 su investigación (Ran *et al.*, 2009) sobre implementación molecular de programas capaces de realizar deducciones lógicas simples. Este modelo se basa en la codificación en cadenas lineales dobles y *hairpins* con extremos salientes, además de cadenas ssADN lineales.

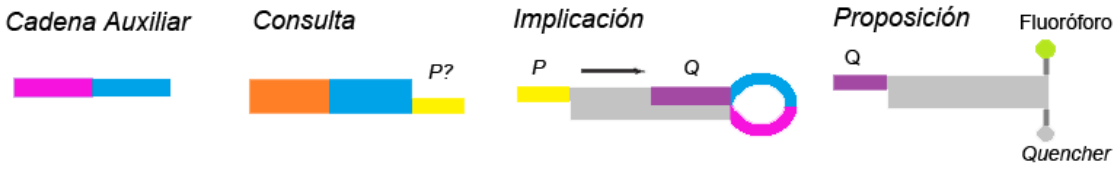


Figura 8: Codificación utilizada en el modelo de Ran, Shapiro y Kaplan. Adaptado de Ran *et al.* (2009).

En la Figura 8 se muestra la codificación utilizada: las proposiciones se codifican en una cadena doble con un extremo saliente que representa la proposición y en el otro extremo una molécula emisora de luz (fluoróforo) suprimida por una molécula que absorbe su energía (*quencher*). La consulta se codifica en una cadena dsADN con un extremo saliente que representa la proposición que se desea consultar y un dominio de reconocimiento para la enzima *FokI* (coloreado en azúl). La estructura de la implicación está conformada por un extremo saliente que representa a P (antecedente), una cadena doble de ADN dentro de la cual se encuentra un dominio que es complementario a Q (consecuente) y que se encuentra unido a una estructura de lazo, la cual al unirse con una cadena auxiliar posee un dominio de reconocimiento para la enzima *FokI* (marcado en azúl).

La Figura 9 describe el proceso a través del cual se realiza la inferencia utilizando la enzima endonucleasa *FokI*. Las consultas cuentan con una parte del segmento que la enzima es capaz de detectar: si su extremo saliente es complementario al de una molécula de proposición, el área de detección es completada y la enzima corta la molécula, separando el flourósforo del *quencher*, lo que produce luminiscencia como resultado de la inferencia.

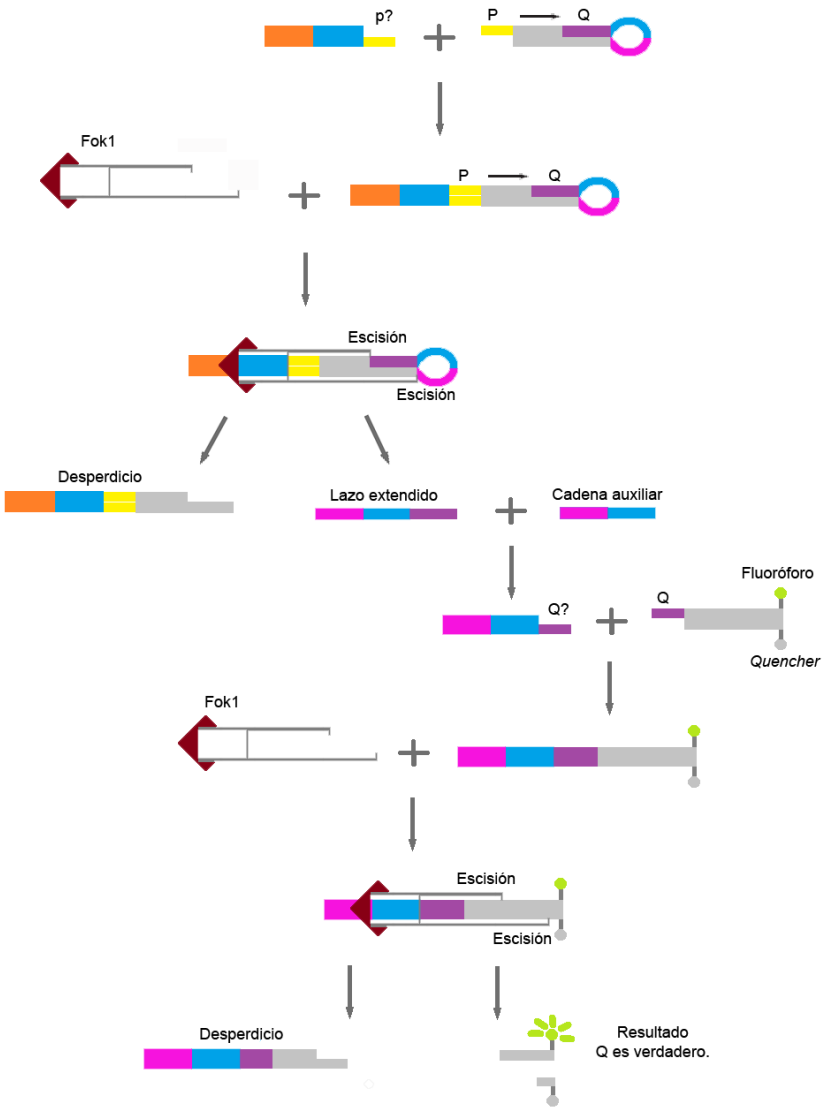


Figura 9: Resolución de la implicación $P \rightarrow Q$. La estructura de la implicación se une a través de su *sticky end* a la estructura de la consulta P? y atrae a la enzima *FokI* a la zona de detección marcada en azúl. La enzima libera la sección del lazo y el dominio complementario a Q y ésta se hibrida con la cadena auxiliar produciendo la consulta Q?. La consulta generada se une a través de su *sticky end* a la proposición Q y atrae a la enzima *FokI*, la cual corta la cadena doble separando al fluoróforo y al *quencher* produciendo luminiscencia. Adaptado de Ran *et al.* (2009).

Las implicaciones son generadas con moléculas de ADN en forma de lazo que poseen codificadas dos variables separadas por un segmento de nucleótidos, una variable se encuentra en el extremo de cadena sencilla y otra en el segmento hibridado. El proceso de inferencia se realiza al unirse una consulta con una molécula de implicación, se complementa el segmento de localización de la enzima y ésta corta la cadena dsADN liberando la variable y el ciclo del lazo. La molécula resultante hibrida con una de las cadenas auxiliares que se encuentran en la solución y se produce con ella una nueva molécula de consulta con la variable liberada en el extremo saliente.

Tras la presentación del modelo de Ran, otro trabajo publicó el bosquejo de un modelo para la generación de inferencias con moléculas de ADN y desplazamiento de cadenas para el *modus tollens* y *modus ponens* (Rodríguez-Patón *et al.*, 2010). Posteriormente, en el año 2011 presentaron una extensión a este modelo en el cual aplican las nociones de su bosquejo (Rodríguez-Patón *et al.*, 2011).

A diferencia del modelo de Ran, Shapiro y Kaplan, éste no requiere de enzimas de restricción, la codificación se realiza en forma normal conjuntiva, y se asigna una secuencia de nucleótidos a cada proposición por lo cual la negación de una variable equivale a la secuencia complementaria a su representación. La Figura 10 muestra la codificación utilizada.

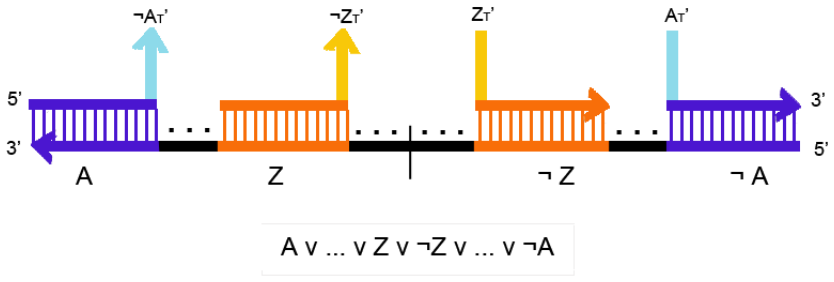


Figura 10: Codificación en ADN de las cláusulas con sus regiones de apoyo. Por cada cláusula se crea una secuencia ordenada, en la que se encuentran primero las cláusulas afirmadas en orden ascendente, seguidas de las cláusulas negadas de forma descendente. Cada cláusula tiene una cobertura complementaria con un *toehold* en el extremo 3' en las proposiciones afirmadas y en el extremo 5' en las proposiciones negadas. Adaptado de Rodríguez-Patón *et al.* (2011).

A cada una de las proposiciones se les crea una secuencia de cobertura con un punto de apoyo (*toehold*) en el extremo 3' cuando la secuencia es complementaria a una proposición afirmativa o en el extremo 5' cuando la proposición es negada. Cada secuencia de cobertura posee un flourósforo para la detección de resultados.

La inferencia se realiza al combinar las soluciones que contienen todos los elementos. Como puede verse en la Figura 11, en una proposición afirmativa, la secuencia de cobertura alinea su extremo 5' contraria al punto de apoyo con el extremo 3' de la proposición, mientras que en una proposición negada la secuencia de apoyo alinea su extremo 3' contrario al punto de apoyo con la región 3' de la proposición. El resultado se puede obtener al examinar la estructura de la molécula resultante.

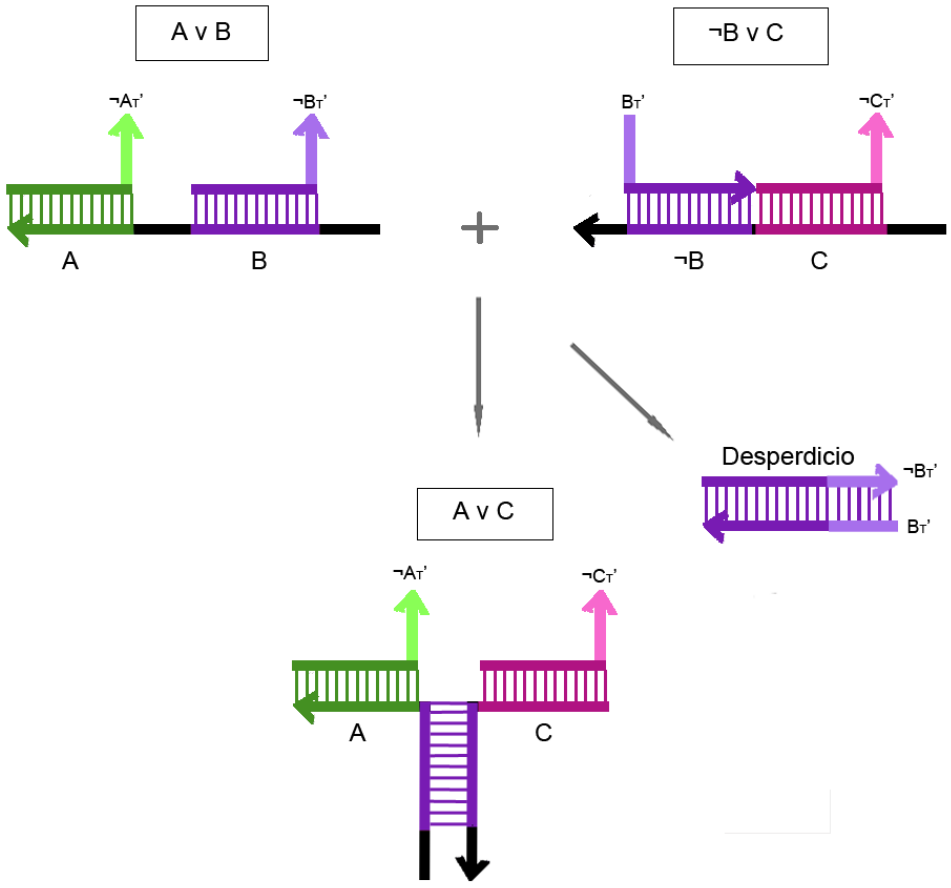


Figura 11: Resolución de las cláusulas $A \vee B$ y $\neg B \vee C$ mediante la contradicción de la variable B. Las coberturas de B y $\neg B$ ($\neg B_{T'}$) se unen al ser complementarias, dejando al descubierto las regiones que codifican B y $\neg B$. Éstas se hibridan, produciendo la molécula que codifica la cláusula resultado $A \vee C$. Adaptado de Rodríguez-Patón *et al.* (2011).

Capítulo 6. Modelo de inferencia lógica propuesto

Como se menciona en el Capítulo 4, existen algunos modelos para la generación de inferencia lógica con técnicas de cómputo biomolecular, de los cuales dos son modelos autónomos. El primer modelo hace uso de enzimas y ha sido probado en laboratorio satisfactoriamente, el otro modelo utiliza desplazamiento de cadenas para llevar a cabo las operaciones con ADN y es completamente teórico. En este capítulo se presenta el diseño de las estructuras propuestas realizado con el lenguaje DSD, y posteriormente se simula utilizando la herramienta visual DSD de Microsoft. Por último, se muestran los resultados obtenidos utilizando el analizador de modelos probabilísticos PRISM.

6.1. Convenciones utilizadas en la simulación con la herramienta Visual DSD

El lenguaje DSD posee normas que tuvieron que ser consideradas en el diseño de las estructuras utilizadas para la inferencia lógica. Primeramente, las proposiciones siempre se representan utilizando cadenas superiores sencillas de ADN, es decir, una ssADN con orientación 5'—3'. Las compuertas que se utilizan para representar los conectivos lógicos y reglas de inferencias son complejos formados por una cadena inferior de ADN parcialmente hibridada con una o más cadenas superiores ssADN. Los puntos de apoyos son dominios cortos de 4 a 10 nucleótidos, mientras que el resto de los dominios son secuencias largas de más de 20 nucleótidos, asegurando con esto que los *toeholds* sólo sean utilizados como puntos de apoyo para iniciar las reacciones, y la dinámica del desplazamiento de cadenas se base solamente en los dominios largos; además de darle estabilidad a las estructuras, ya que los dominios poseen suficientes nucleótidos para evitar que las hebras se separen.

Las pruebas realizadas en Visual DSD se llevaron a cabo con el simulador estocástico que implementa el algoritmo de Gillespie. Los parámetros utilizados incluyen una velocidad de migración de ramificaciones de 8,000 s^{-1} por nucleótido, dominios largos de 20 nucleótidos y puntos de apoyo de 6 nucleótidos, con una tasa de hibridación y desnaturalización de 3

x $10^{-4} nM^{-1}s^{-1}$ y 0.1126 s^{-1}, respectivamente. El tiempo y número de puntos de muestra depende del caso de prueba utilizado.

6.2. Convenciones utilizadas en el análisis de propiedades con la herramienta PRISM

El análisis realizado con la herramienta PRISM consiste en la verificación de corrección del modelo y el cálculo de las propiedades cuantitativas de las estructuras. Visual DSD permite la exportación del modelo de cadenas de Markov en tiempo continuo de los dispositivos diseñados al formato *sm* utilizado en PRISM. Previo al análisis de los elementos diseñados es necesario definir un estado final deseable; es decir, un estado en el cual no existan reacciones posibles y además se obtiene el resultado esperado. La definición de estado final utilizada fue tomada de Cardelli *et al.* (2012), en donde se considera que un estado final es deseable cuando todas las compuertas son inertes y las únicas cadenas reactivas existentes son aquellas generadas como resultado. Esto se escribe utilizando el lenguaje de PRISM de la siguiente manera:

```
label "all_done" = strands_reactive = output & output = N & gates_reactive =
0;
```

donde `label` es una palabra reservada que permite crear la etiqueta booleana "*all_done*", `strands_reactive` y `gates_reactive` son variables generadas por visual DSD en el modelo exportado, y contabilizan el número de cadenas y compuertas capaces de reaccionar. `Output` es una variable que contabiliza el número de resultados generados y `N` es la cantidad de resultados esperados.

La estabilidad del sistema se define con base en el estado final deseable. De acuerdo a Cardelli y colaboradores (2012), se considera que un modelo es correcto cuando todo estado final alcanzable cumple las condiciones del estado final deseable especificado por la etiqueta "*all_done*", y existe al menos una ruta en el sistema que permita alcanzar un estado con

dichas características. En PRISM se utiliza lógica temporal de árboles computacionales (*CTL*) para realizar esta verificación. Las condiciones anteriores se expresan utilizando lógica *CTL* de la siguiente manera:

$$\text{A [G } \text{“}Deadlock\text{”} \Rightarrow \text{“}all\_done\text{”]}$$

$$\text{E [F } \text{“}all\_done\text{”]}$$

donde “*Deadlock*.[es] una palabra reservada de PRISM que define a los estados finales del modelo, y las literales *A* y *E* son los operadores universal y existencial, respectivamente. La opción *G* especifica que la evaluación se llevará a cabo globalmente y *F* significa que la condición se espera que se cumpla eventualmente o en un futuro. El resultado de verificar estas propiedades es un valor booleano, y en el caso de que sea falso, PRISM demuestra su resultado presentando un contraejemplo. Estas evaluaciones no se presentan en las secciones de análisis de la herramienta para cada estructura, ya que exceptuando al modelo del *modus tollendo ponens* (cuyos resultados se discuten en la Sección 6.5.6.3), todas ellas han obtenido resultados verdaderos en ambas propiedades, con lo cual se prueba que sus modelos son correctos.

Además de comprobar el correcto funcionamiento de las estructuras, se utilizó PRISM para realizar evaluaciones probabilísticas de las propiedades cuantitativas. Para ello se definieron cuatro propiedades principales: la probabilidad de alcanzar un estado final total, independientemente de ser deseado o no, la probabilidad de alcanzar un estado final exitoso, la probabilidad de llegar a un estado final erróneo y la probabilidad de permanecer en el estado inicial. Estas propiedades se expresan utilizando lógica temporal *CLS* de la siguiente manera:

- P = ? [F [T , T] (correct_output)]
- P = ? [F [T , T] (“*deadlock*” & correct_output)]
- P = ? [F [T , T] (“*deadlock*” & ! correct_output)]

- P = ? [F [T , T] (“*init*”)]

Estas verificaciones se llevan a cabo sobre la variable T que representa unidades de tiempo, por lo cual se pueden realizar experimentos evaluando las propiedades en cada unidad de tiempo de un periodo definido para posteriormente ser graficadas.

6.3. Representación de proposiciones

En el diseño de este modelo se identificaron dos tipos de proposiciones: las proposiciones simples, las cuales representan una afirmación ya sea verdadera o falsa, y las proposiciones temporales, las cuales son resultado de un conectivo lógico y representan la afirmación de dicha operación.

6.3.1. Representación de proposiciones simples

Como se puede observar en la Figura 12, cada proposición se representa con una secuencia simétrica de nucleótidos con orientación 5’—3’. En cada extremo se encuentra un *toehold* o punto de apoyo genérico, seguido de un dominio único representativo de la proposición, y en el centro se encuentran dos dominios genéricos, auxiliares en el procedimiento de inferencia lógica.

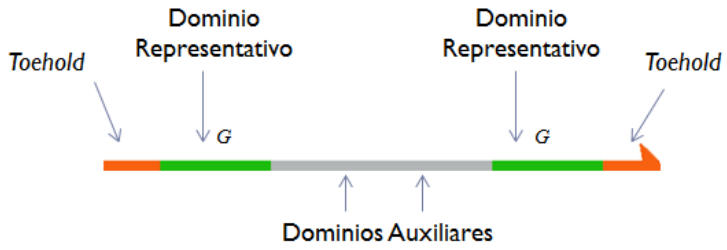


Figura 12: Representación de la proposición G. Se utiliza una cadena ssADN compuesta por dos puntos de apoyo o *toeholds* en los extremos, dos dominios que le representan y diferencian de las demás proposiciones, y en el centro dos dominios genéricos auxiliares.

Debido a las convenciones del leguaje DSD, no es posible representar la negación de una proposición utilizando su cadena complementaria, ya que ésta sería una cadena inferior con

orientación 3'—5', y en lenguaje DSD se utilizan cadenas superiores para la representación de variables, o en este caso, proposiciones. Por este motivo, la negación de una proposición se representa utilizando una nueva cadena ssADN superior con un dominio representativo distinto al de la proposicón original. La Figura 13 muesta una proposición G y su negación $\neg G$.

Figura 13: Representación de la proposición G y su negación $\neg(G)$ en dos cadenas superiores distintas.

6.3.2. Representación de proposiciones temporales

Este tipo de proposiciones son llamadas temporales, ya que no representan en sí una proposición, sino un resultado intermedio en la inferencia lógica. Dado que el resultado de la conjunción y disyunción es $A \wedge B$ y $A \vee B$, respectivamente, esta salida no puede ser representada utilizando una proposición simple, por lo que se utilizan proposiciones temporales.

Las proposiciones temporales, al igual que las simples, se representan utilizando una cadena ssADN superior simétrica, sin embargo, en este caso se utilizan puntos de apoyo únicos en los extremos como identificación, por lo cual carecen del dominio único representativo. La Figura 14 muestra la estructura de la cadena temporal producida por la conjunción $A \wedge B$. La negación de una proposición temporal se representa utilizando una nueva cadena ssADN, pero en este caso se utiliza un nuevo *toehold* para su identificación.

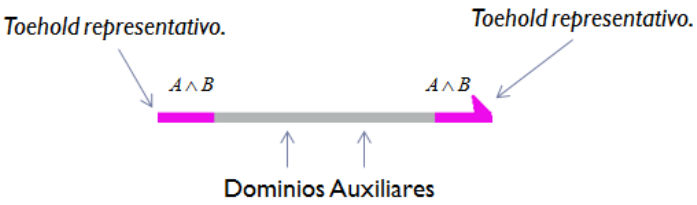


Figura 14: Representación de la proposición temporal producida por la conjunción $A \wedge B$. Esta cadena carece del dominio representativo de las proposiciones simples y posee un *toehold* único que lo identifica.

6.3.3. Lectura de resultados

La conclusión de una inferencia lógica siempre es dada en forma de proposiciones, por lo cual se propone utilizar una estructura con un *toehold* genérico y un dominio de detección para las proposiciones simples, o un *toehold* representativo en el caso de las premisas temporales. La estructura cuenta con dos dominios auxiliares genéricos complementarios a los utilizados en las proposiciones, uno de ellos posee un fluoróforo en un extremo y el otro una molécula que absorbe la energía producida por la molécula de fluorescencia (*quencher*). La Figura 15 muestra dicha estructura.

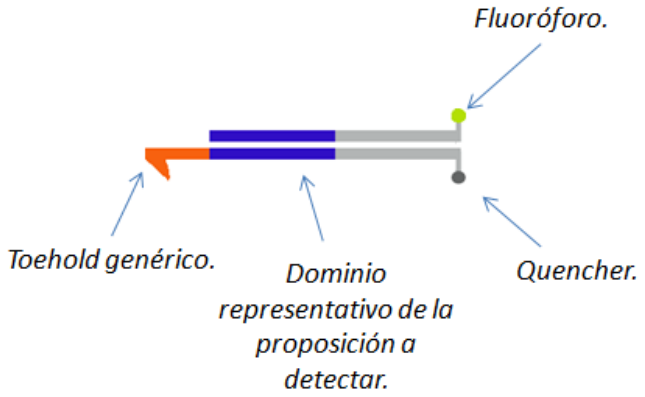


Figura 15: Representación de la estructura utilizada para la detección de resultados. En el lado izquierdo se encuentra una región de detección de la proposición que se desea identificar, del lado derecho se encuentra un fluoróforo y un *quencher*.

Como se puede observar en la Figura 16, cuando la proposición que se desea detectar se encuentra presente, su punto de apoyo se une a su complemento en la estructura y comienza

a desplazar por termodinámica a la cadena superior. Al separar los dominios auxiliares el fluoróforo y el *quencher* se separan produciendo luminiscencia.

Figura 16: Dinámica de la detección de resultados. La proposición B desplaza la cadena superior y separa al fluoroforo del *quencher* produciendo luminiscencia.

6.4. Conectivos lógicos

Los conectivos lógicos son unidades básicas en la lógica proposicional, ya que permiten unir dos o más proposiciones para crear proposiciones compuestas. Este modelo incluye los tres conectivos más utilizados: la conjunción, la disyunción y la proposición condicional.

6.4.1. Conjunción

Como se menciona en la Sección 2.3.1, la conjunción de A y B $(A \wedge B)$, es un operador que recibe dos proposiciones y su resultado sólo es verdadero cuando ambas premisas lo son. Para su diseño, se consideró que la presencia de una proposición indica que ésta es verdadera, además de que, por la naturaleza del resultado de la conjunción, es necesario utilizar una proposición temporal.

6.4.1.1. Diseño de la compuerta

La Figura 17 muestra la estructura diseñada para el conectivo $A \wedge B$. La compuerta posee dos áreas de reconocimiento: en el lado izquierdo se encuentra el complemento al *toehold* genérico y el dominio representativo de la proposición A, mientras que en el lado derecho se encuentra el complemento del punto de apoyo genérico y el dominio que identifica

a la proposición B. En el centro se encuentran cuatro dominios auxiliares, los dos de los extremos poseen una cadena de cobertura, mientras que los dos del centro son cubiertos por la proposición temporal que será liberada como resultado.

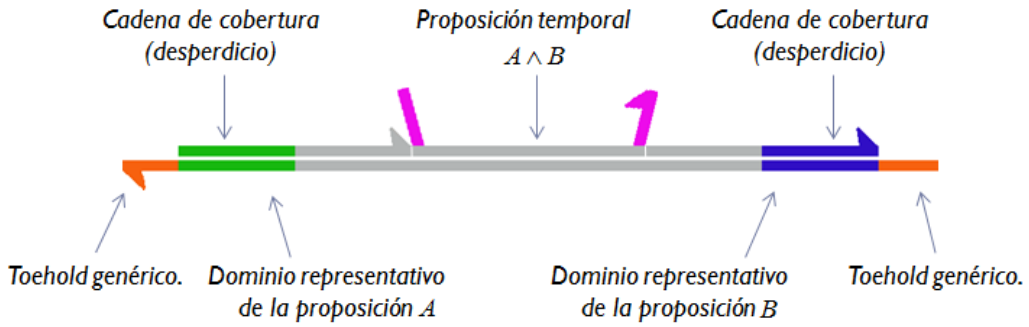


Figura 17: Estructura diseñada para la conjunción $A \wedge B$.

El funcionamiento de la estructura depende de los complementos del *toehold* genérico utilizado en las zonas de reconocimiento. El lenguaje DSD define que todas las reacciones deben llevarse a cabo a través de puntos de apoyo y ningún otro dominio complementario debe ser expuesto, por esto se cubren los complementos de los dominios representativos y las zonas auxiliares en cada lado de la estructura con cadenas de cobertura, que al finalizar el proceso de desplazamiento serán liberadas como desperdicio, ya que son cadenas inertes al carecer de un punto de apoyo.

Cualquier proposición puede reaccionar con los puntos de apoyo de la compuerta, sin embargo sólo aquellas que posean el dominio representativo complementario al utilizado en las zonas de reconocimiento (A y B en el ejemplo de la Figura 17) podrán realizar el desplazamiento de las cadenas de cobertura y producir un resultado. Las cadenas que no coincidan con la zona de reconocimiento eventualmente se desprenderán del punto de apoyo, ya que éste posee una cantidad de nucleótidos tan pequeña que no logra mantener una unión estable. Al agregar dos dominios auxiliares en cada extremo de la estructura, se asegura que no se produzcan resultados a menos que ambas proposiciones reaccionen con la compuerta, debido a que cada proposición sólo posee dos dominios auxiliares (una de ellas no puede desplazar más de uno de los dominios auxiliares del resultado).

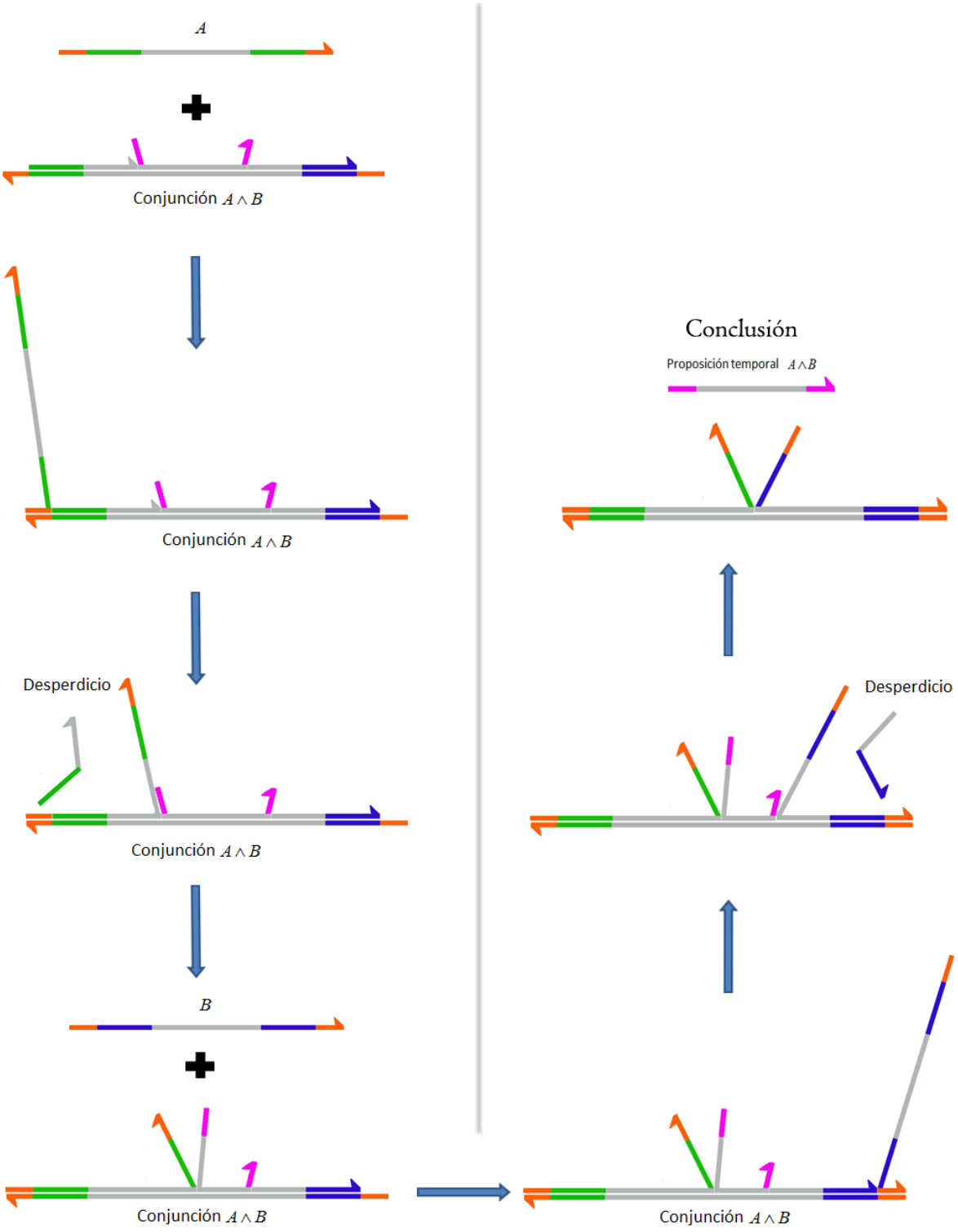


Figura 18: Diagrama de resolución de la conjunción $A \wedge B$. La proposición A se une a la zona de reconocimiento izquierda y desplaza la cadena superior y el primer dominio de la proposición temporal. La proposición B se une a la zona de reconocimiento de la derecha y se une a la compuerta hasta desprender la proposición $A \wedge B$.

En la Figura 18 se muestra el funcionamiento de la estructura propuesta. Una vez que la proposición A se encuentra con la compuerta, une su punto de apoyo con su complemento en el lado izquierdo y comienza a desplazar a la cadena de cobertura, primero en su dominio representativo, luego con el dominio auxiliar del extremo izquierdo y después con uno de los dominios centrales. En este momento la mitad de la cadena de salida ha sido desprendida. La proposición B une su *toehold* a la zona de reconocimiento del lado derecho y comienza a desplazar a la cadena de cobertura de su dominio representativo y del primer dominio auxiliar. Al desplazar el segundo dominio auxiliar la proposición temporal $A \wedge B$ se libera como resultado.

Visual DSD posee un módulo para realizar análisis sobre los sistemas diseñados. Esta opción permite calcular y visualizar el diagrama de los estados del sistema con las probabilidades de transición entre ellos. Como se muestra en la Figura A.01 del Apéndice , el sistema creado por la compuerta de conjunción y las proposiciones A y B cuenta con 4 estados y 4 transiciones, y sólo uno de éstos es terminal. Visual DSD también genera una visualización de los estados inicial y final del sistema, lo cual resulta útil en la posterior verificación de corrección del modelo. La Figura A.02 del Apéndice muestra el estado inicial y el estado final de la conjunción.

6.4.1.2. Simulación estocástica con Visual DSD

Para probar el funcionamiento del diseño de la estructura de conjunción, se llevaron a cabo simulaciones utilizando el algoritmo de Gillespie incluído en la herramienta Visual DSD. La Figura 19 muestra el promedio de 10 simulaciones de los cuatro casos de prueba correspondientes a la tabla de verdad, con 100,000 moléculas de la compuerta y las combinaciones de 100,000 moléculas de las proposiciones de entrada, utilizando como parámetros una duración de mil unidades de tiempo y 20,000 puntos de muestra. El inciso (a) muestra la gráfica de los resultados de la simulación del caso de prueba en el cual no existe ninguna molécula de las proposiciones de entrada. Como puede observarse, el número de compuertas se mantiene

constante ya que no existen proposiciones con las cuales pueda reaccionar, por lo cual no se lleva a cabo ninguna reacción y no se producen conclusiones como resultado.

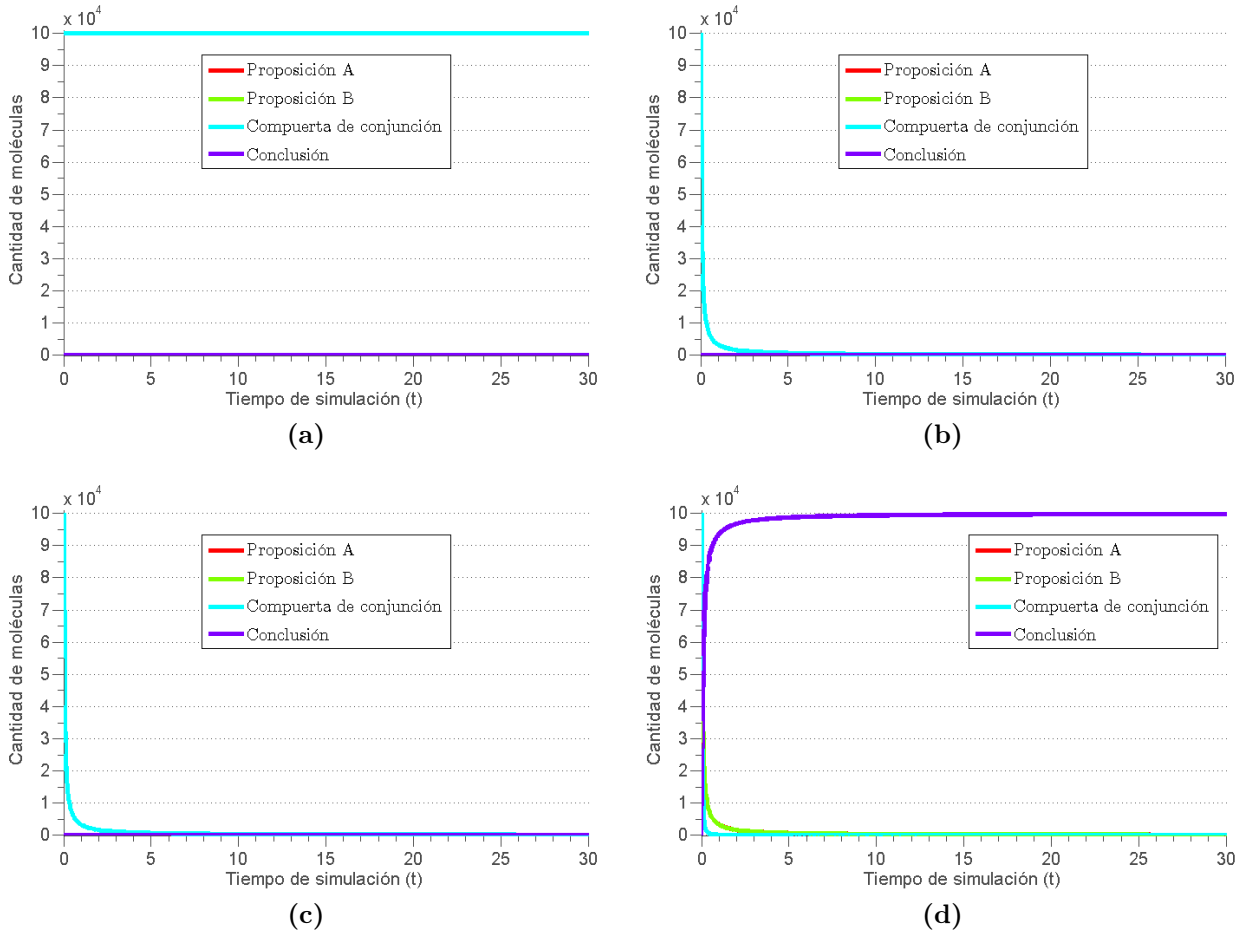


Figura 19: Promedio de 10 simulaciones de la conjunción $A \wedge B$. (a) Utilizando 0 moléculas de las proposiciones A y B. (b) Utilizando solamente 100,000 moléculas de la proposición B. (c) Utilizando solamente 100,000 moléculas de la proposición A. (d) Utilizando 100,000 moléculas de las proposiciones A y B.

El inciso (b) de la Figura 19 muestra la gráfica del los resultados de la simulación del caso de prueba en el cual existen 100,000 moléculas de la proposición B y no existen moléculas de la proposición A. La única línea visible corresponde a la compuerta de conjunción, dado que ésta y la premisa B se consumen mutuamente, poseen el mismo comportamiento. La compuerta de conjunción requiere ambas premisas para producir una conclusión, y al no encontrarse presente la proposición A, nunca se produce un resultado, sin embargo, la gráfica de la compuerta se muestra como consumida debido a que el simulador considera que se

consume en el momento en que una de las dos proposiciones se une a ella.

En el inciso (c) de la Figura 19 se muestra la gráfica del promedio de las simulaciones del caso de prueba con 100,000 moléculas de la proposición A y 0 moléculas de la proposición B. Al igual que en el inciso (b), sólo la línea de la compuerta de conjunción es visible, ya que ésta y la premisa A se consumen mutuamente y poseen el mismo comportamiento. Aunque las reacciones se llevan a cabo rápidamente, no se produce una conclusión debido a que no existe la premisa B y la compuerta de conjunción requiere a ambas proposiciones para producir un resultado.

Por último, el inciso (d) de la Figura 19 muestra la gráfica generada por el promedio de las simulaciones del caso de prueba con 100,000 moléculas de las proposiciones A y B. Las premisas y la compuerta se consumen antes de las 20 unidades de tiempo, produciendo 100,000 moléculas de conclusión como resultado. Se puede observar que las compuertas se consumen a una mayor velocidad que las premisas, ya que el simulador considera que esto sucede una vez que una de las proposiciones se une a ella.

6.4.1.3. Análisis de propiedades utilizando PRISM

Para confirmar el correcto funcionamiento de la compuerta de conjunción, se realizaron experimentos con la herramienta PRISM para verificar las probabilidades de alcanzar un estado final exitoso, terminar en un estado no deseado, finalizar en cualquier estado terminal, así como de encontrarse en el estado inicial después de un cierto periodo de tiempo. La Figura 20 muestra las probabilidades de todas las posibles finalizaciones del modelo de conjunción en un tiempo T. Como era de esperarse de un modelo que cuenta con sólo un estado terminal, la probabilidad de finalización total y la de finalización exitosa es la misma, y a partir de las 20,000 unidades de tiempo tiende a ser 1. Debido a lo anterior, la probabilidad de alcanzar un estado terminal erróneo se mantiene siempre en 0, y la probabilidad de mantenerse en el estado inicial disminuye al aumentar el tiempo y la probabilidad de finalización exitosa.

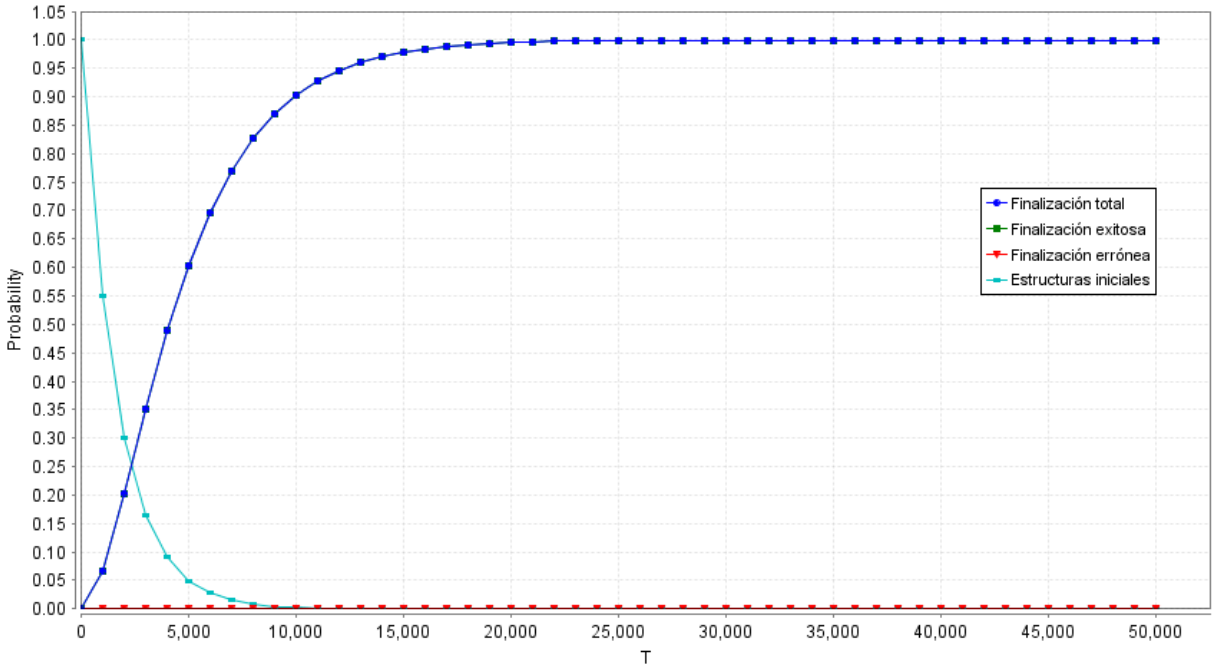


Figura 20: Gráfica de probabilidades de todas las posibles finalizaciones del modelo de conjunción generado con la herramienta PRISM.

6.4.2. Disyunción

Como se menciona en la sección 2.3.2, el conectivo lógico de disyunción es un operador que recibe dos proposiciones y produce un resultado verdadero cuando al menos alguna de ellas lo es. Al igual que en la conjunción, la disyunción utiliza una proposición temporal para la representación del resultado.

6.4.2.1. Diseño de la compuerta

En la Figura 21 se muestra la estructura diseñada para el operador de la disyunción $A \vee B$. Como se puede observar, es muy parecida a la utilizada en la conjunción: posee dos áreas de reconocimiento en cada extremo, en el lado izquierdo se lee la proposición A y en el derecho la B, y en el centro se encuentran dos dominios auxiliares unidos a la proposición temporal $A \vee B$ que será producida como resultado. La diferencia entre ambas estructuras reside en el número de dominios auxiliares utilizados. La disyunción sólo posee dos dominios auxiliares centrales, mientras que la conjunción posee dos centrales y uno al lado de cada

zona de reconocimiento. Al reducir el número de dominios, se permite que cualquiera de las dos proposiciones libere el resultado.

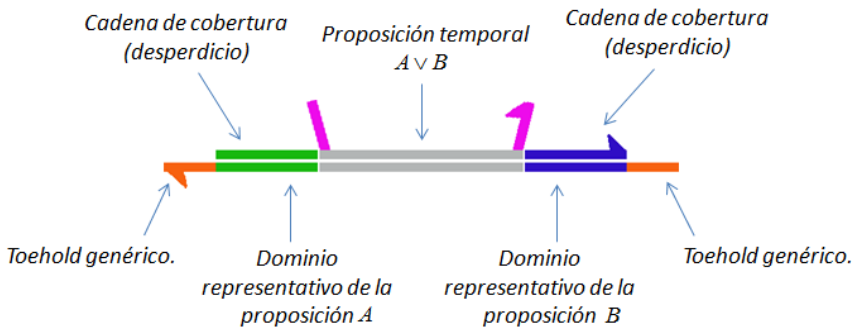


Figura 21: Representación de la disyunción $A \vee B$.

El funcionamiento de esta estructura depende de las zonas de reconocimiento conformadas por un complemento del *toehold* genérico a través del cual se realizan todas las reacciones, y un complemento del dominio representativo de la proposición que se desea leer, unido a una cadena de cobertura de acuerdo a las especificaciones del lenguaje DSD. Si una proposición se une a una zona de reconocimiento y posee el dominio complementario al de la estructura, se comenzará a realizar el desplazamiento de la cadena de cobertura, la cual será inerte al carecer de un punto de apoyo genérico. Posteriormente, la premisa unirá sus dominios auxiliares con los de la estructura, desplazando a la proposición temporal que estaba unida a ellos y producirá con esto el resultado esperado. Si la proposición no cuenta con el dominio complementario al de la zona de reconocimiento, no se realiza un desplazamiento de cadenas y eventualmente se desprende ya que los *toeholds* no poseen suficientes nucleótidos para mantener la unión estable. La Figura 22 muestra el funcionamiento de la disyunción $A \vee B$, utilizando la proposición A para su resolución.

El sistema creado por la compuerta de disyunción y las proposiciones A y B cuenta con 4 estados y 4 transiciones: un estado inicial, del cual se puede pasar a uno de dos estados intermedios, y por último llegar al estado final (Figura A.01 del Apéndice).

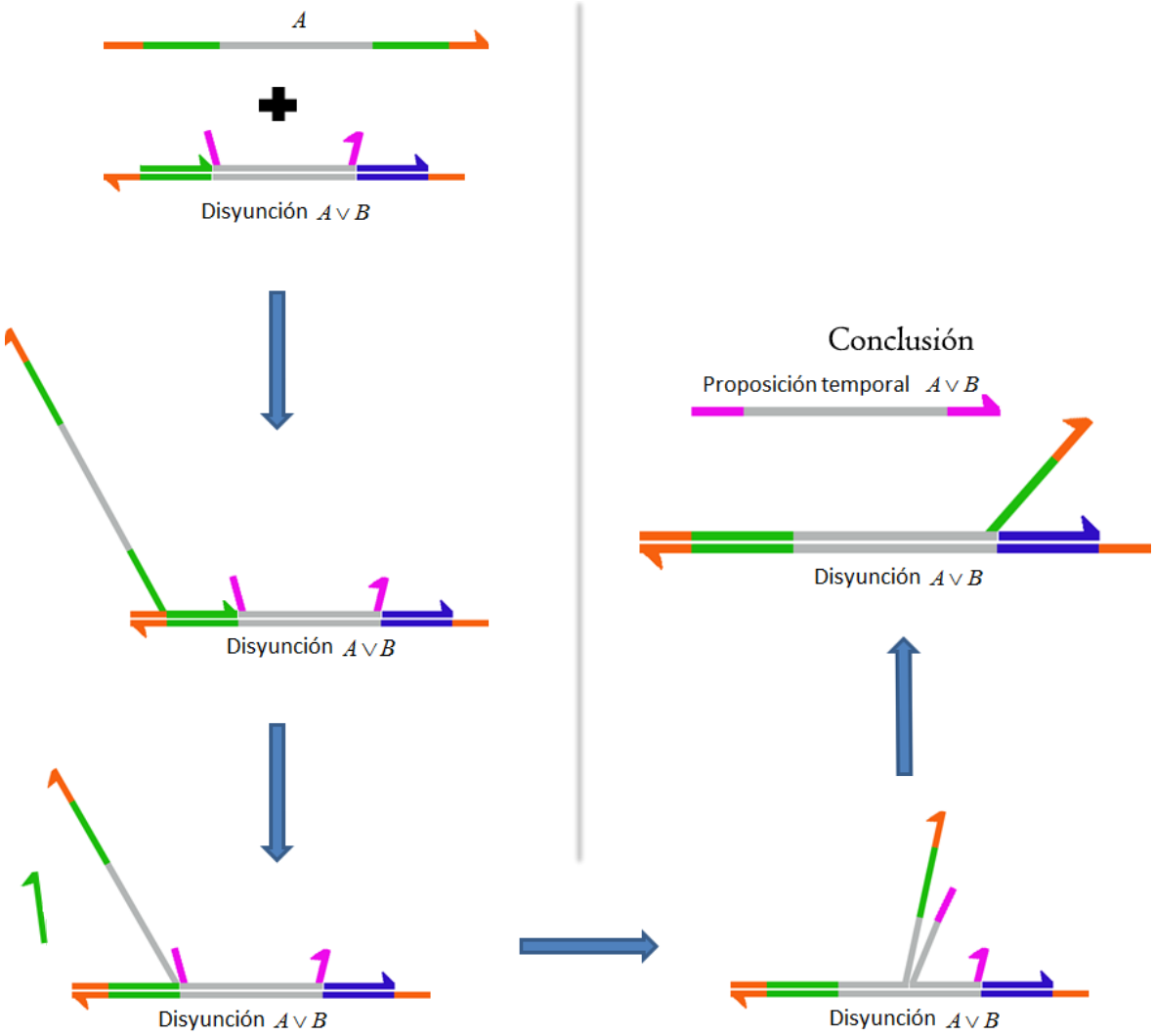


Figura 22: Procedimiento para la resolución de la disyunción $A \vee B$ utilizando la proposición A. La proposición une su punto de apoyo genérico con su complemento en la zona de reconocimiento izquierda y desplaza primero la cadena de cobertura del dominio representativo, para posteriormente unir sus dominios auxiliares con los de la estructura, liberando la proposición temporal $A \vee B$ como resultado. De manera similar funcionaría utilizando la proposición B.

6.4.2.2. Simulación estocástica con Visual DSD

Las simulaciones de la estructura de disyunción se realizaron utilizando el simulador estocástico de Visual DSD. En la Figura 23, se muestra el promedio de 10 simulaciones de los cuatro casos de prueba correspondientes a la tabla de verdad realizados utilizando 100,000 moléculas de la compuerta de disyunción y 0 ó 100,000 moléculas de las entradas A y B. Como parámetros se incluyen una duración de 1500 unidades de tiempo y 30,000 puntos de muestra. La gráfica (a) corresponde al caso donde no se utilizaron moléculas de las premisas. Observe que el número de compuertas se mantiene constante ya que no existen reacciones que las consuman y no se produce ninguna conclusión como resultado.

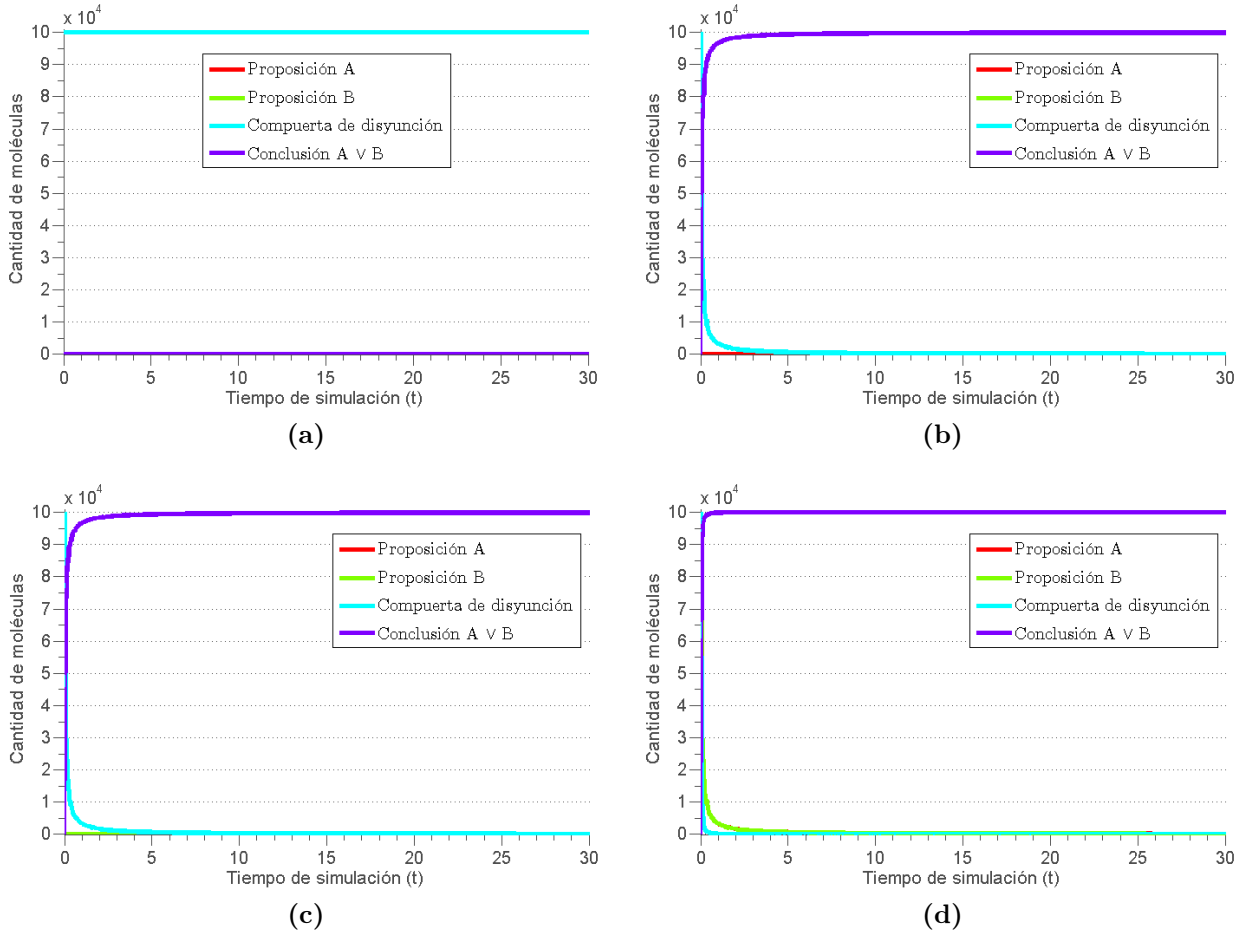


Figura 23: Promedio de 10 simulaciones de la disyunción $A \vee B$. (a) Utilizando 0 moléculas de las proposiciones A y B. (b) Utilizando solamente 100,000 moléculas de la proposición B. (c) Utilizando solamente 100,000 moléculas de la proposición A. (d) Utilizando 100,000 moléculas de las proposiciones A y B.

La gráfica (b) de la Figura 23 pertenece al caso de prueba en el que se utilizaron 100,000 moléculas de la premisa B, y no se incluye a la proposición A. La compuerta de disyunción es consumida rápidamente por la premisa, por lo cual poseen el mismo comportamiento y ésta última no es visible. A diferencia de la simulación de la conjunción, en ésta se produce un resultado, ya que ambas premisas no son necesarias para que esto suceda. En el inciso (c) se muestra el resultado de las simulaciones del caso en el cual se utilizan 100,000 moléculas de la premisa A y no existe la proposición B. En esta gráfica, se presenta un comportamiento parecido al caso anterior, la compuerta es consumida por la proposición A, por lo cual poseen el mismo comportamiento y ésta no es visible. Al producirse la reacción se genera la conclusión esperada, y antes de las 15 unidades de tiempo todas las estructuras iniciales han sido utilizadas y se han liberado 100,000 proposiciones temporales como resultado. Finalmente, el inciso (d) muestra la gráfica del caso de prueba donde se utilizaron 100,000 moléculas de las premisas A y B. Como se puede observar, en esta simulación se disminuye considerablemente la duración de la producción de conclusiones, debido a que las reacciones se llevan a cabo en las primeras unidades de tiempo por el gran número de moléculas que colisionan y al hecho de que existe el doble de moléculas de proposiciones que de compuertas, dado que en esta compuerta basta con una de las proposiciones para generar un resultado, por lo que prácticamente existe el doble de proposiciones que de compuertas, aumentando la probabilidad de que se produzca una reacción y se genere una conclusión.

6.4.2.3. Análisis de propiedades utilizando PRISM

A continuación se procederá a realizar un análisis de las propiedades que definen el correcto funcionamiento del conectivo lógico de disyunción utilizando la herramienta PRISM. Las propiedades utilizadas fueron: la probabilidad de alcanzar cualquier estado terminal, la probabilidad de finalizar en un estado deseado, la probabilidad de terminar en un estado no deseado y la probabilidad de permanecer en el estado inicial. La Figura 24 muestra la gráfica de evaluación de las propiedades cuantitativas en el tiempo T, utilizando un lapso de

50,000 unidades de tiempo. Observe que la probabilidad de que el sistema se mantenga en el estado inicial disminuye, mientras que la probabilidad de alcanzar un estado final aumenta. La relación entre estas dos variables no es proporcional, debido a que el estado inicial se abandona en el momento en que una de las dos proposiciones se une a la compuerta, mientras que el estado final exitoso por definición se alcanza hasta que ambas proposiciones han hibridado con la estructura. Esto hace que se alcance una probabilidad de alcanzar un estado final deseable de 1 entre los 15,000 y 20,000 unidades de tiempo, aunque la producción de resultados comienza tiempo antes.

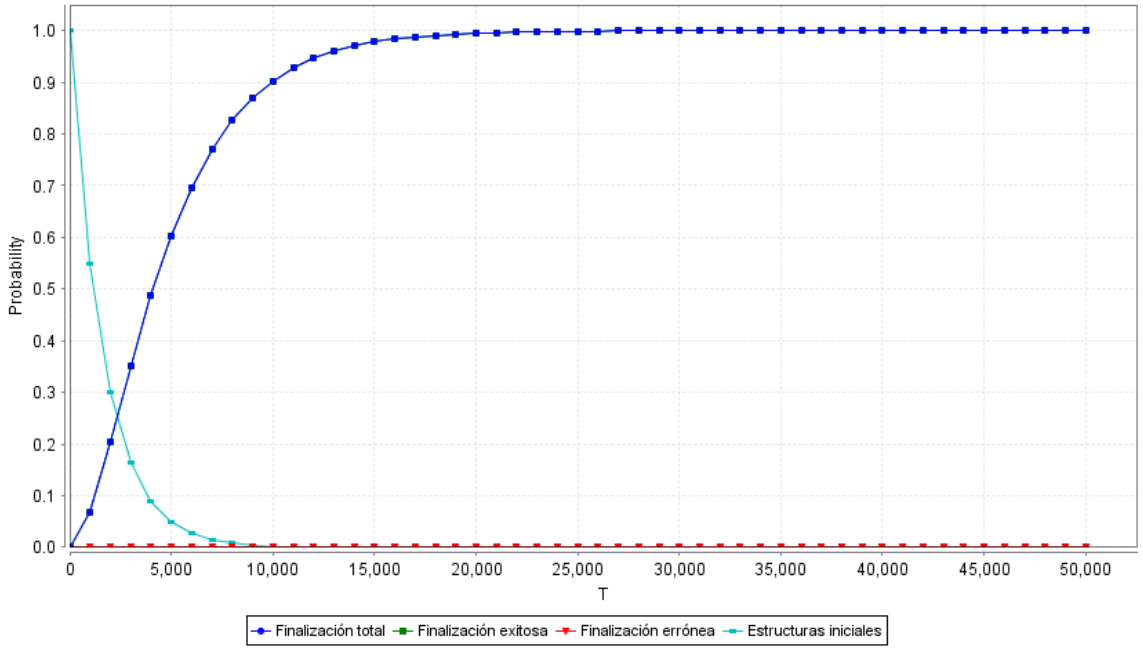


Figura 24: Gráfica generada por PRISM para los experimentos de verificación de las probabilidades de todas las finalizaciones posibles del modelo del conectivo de disyunción.

6.4.3. Proposición condicional

En la Sección 2.3.3, se define la proposición condicional como un operador unario, es decir, recibe una proposición como entrada y produce una nueva proposición. A diferencia de los conectivos de conjunción y disyunción, en éste se utiliza una proposición simple para la codificación del resultado.

6.4.3.1. Diseño de la compuerta

La estructura diseñada para la proposición condicional posee una zona de reconocimiento compuesta por el complemento del *toehold* genérico y un complemento del dominio representativo del antecedente, hibridado a su correspondiente cadena de cobertura. En el extremo opuesto se encuentra un dominio auxiliar unido a la proposición consecuente. La Figura 25 muestra la estructura de la proposición condicional $A \rightarrow B$.

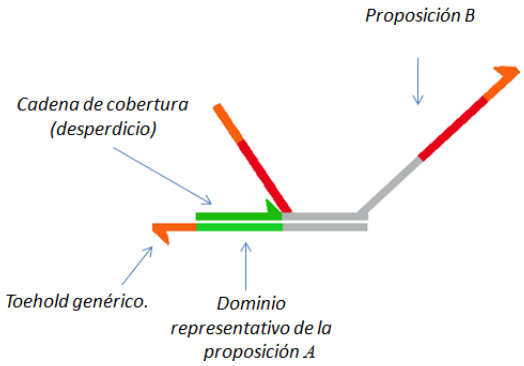


Figura 25: Representación de la proposición condicional $A \rightarrow B$. En el extremo izquierdo se encuentra una zona de reconocimiento conformada por un complemento del punto de apoyo genérico y el dominio representativo de la proposición antecedente A, a su derecha se encuentra un complemento del dominio auxiliar unido a la proposición consecuente B.

Al igual que en las estructuras anteriores, las reacciones se llevan a cabo a través de la zona de reconocimiento. Las proposiciones unen su punto de apoyo genérico a su complemento en el área de reconocimiento de la compuerta, si cuenta con el dominio representativo correspondiente, comienza a desplazar su cadena de cobertura. Posteriormente, la premisa desplaza a la proposición consecuente que está unida al dominio auxiliar, y con esto produce un resultado. Las proposiciones que no poseen el dominio representativo requerido, pueden unir su punto de apoyo genérico al de la estructura pero no realizan un desplazamiento de la cadena de cobertura y por lo tanto no producen un resultado. Eventualmente estas cadenas se liberan, ya que el *toehold* con el que están unidas es un dominio corto y carece de estabilidad.

En la Figura 26, se muestra la resolución de la proposición condicional $A \rightarrow B$. La

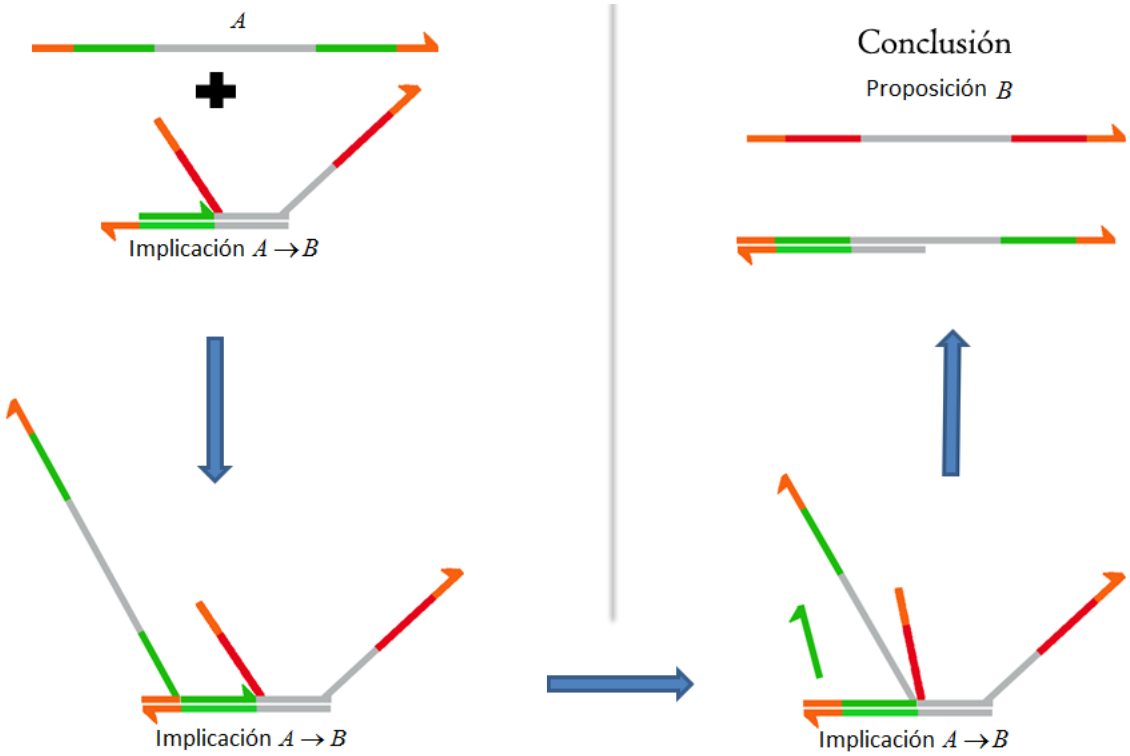


Figura 26: Resolución esquemática de la proposición condicional $A \rightarrow B$. La proposición A, une su punto de apoyo a su complemento en la zona de reconocimiento de la estructura. Posteriormente, desplaza a la cobertura del complemento del dominio representativo y continúa uniendo su dominio auxiliar al de la estructura, liberando en el proceso a la proposición consecuente B.

proposición A se une al área de reconocimiento de la compuerta utilizando su punto de apoyo genérico. Dado que su dominio representativo corresponde al complemento de la estructura, se une a él, desplazando a la cadena de cobertura con la que estaba hibridado. Posteriormente, une su primer dominio auxiliar al de la compuerta y con esto libera a la proposición B que representa el consecuente en la proposición condicional.

Analizando la estructura de proposición condicional con la herramienta Visual DSD, se obtuvo el diagrama de transición de estados mostrado en la Figura A.01 del Apéndice . El sistema cuenta solamente con dos estados: el inicial y el terminal, y una transición entre ellos.

6.4.3.2. Simulación estocástica con Visual DSD

Al igual que en las estructuras anteriores, las simulaciones se realizaron utilizando el algoritmo de Gillespie incluído en la herramienta Visual DSD de Microsoft. La Figura 27 muestra las gráficas generadas por el promedio de 10 simulaciones realizadas con 100,000 moléculas de la compuerta y diferente número de moléculas del antecedente. Los parámetros incluyen una duración de 1,000 unidades de tiempo y 30,000 puntos de muestra. En la simulación de la gráfica (a) no se utilizaron moléculas de la premisa A. Como es de esperarse, el número de moléculas de la compuerta se mantiene constante ya que no existen posibles reacciones a falta del antecedente y no se produce un resultado. En la gráfica (b) se observa el resultado de agregar 100,000 moléculas de la premisa A, en este caso todas las proposiciones B se liberan en las primeras 15 unidades de tiempo, dado que esta compuerta sólo posee una transición entre el estado inicial y final, siempre que existen moléculas del antecedente se produce un resultado. De manera similar a las simulaciones de las compuertas anteriores, la línea de la proposición A no es visible al consumirse al mismo tiempo que la compuerta, por lo que sus gráficas comparten el mismo comportamiento.

La gráfica (c) de la Figura 27 muestra el resultado 10 simulaciones realizadas utilizando 10,000 moléculas de la proposición A y 100,000 moléculas de la compuerta de proposición condicional. Como se puede observar, a pesar de utilizar un tiempo de simulación de 10,000

unidades, la simulación finaliza en el primer ciclo de tiempo. La razón es que al existir un mayor número de moléculas de la compuerta que de la proposición, aumenta la probabilidad de que reaccionen, produciendo un resultado en un menor tiempo. Por otro lado, observe que las moléculas de la proposición A consumen 10,000 estructuras de la compuerta, reduciendo su número a 90,000 y produciendo como resultado 10,000 cadenas de la proposición B. Al igual que en previas simulaciones, la proposición A no es visible, ya que se consume al mismo tiempo que la compuerta y por lo tanto sus gráficas son idénticas.

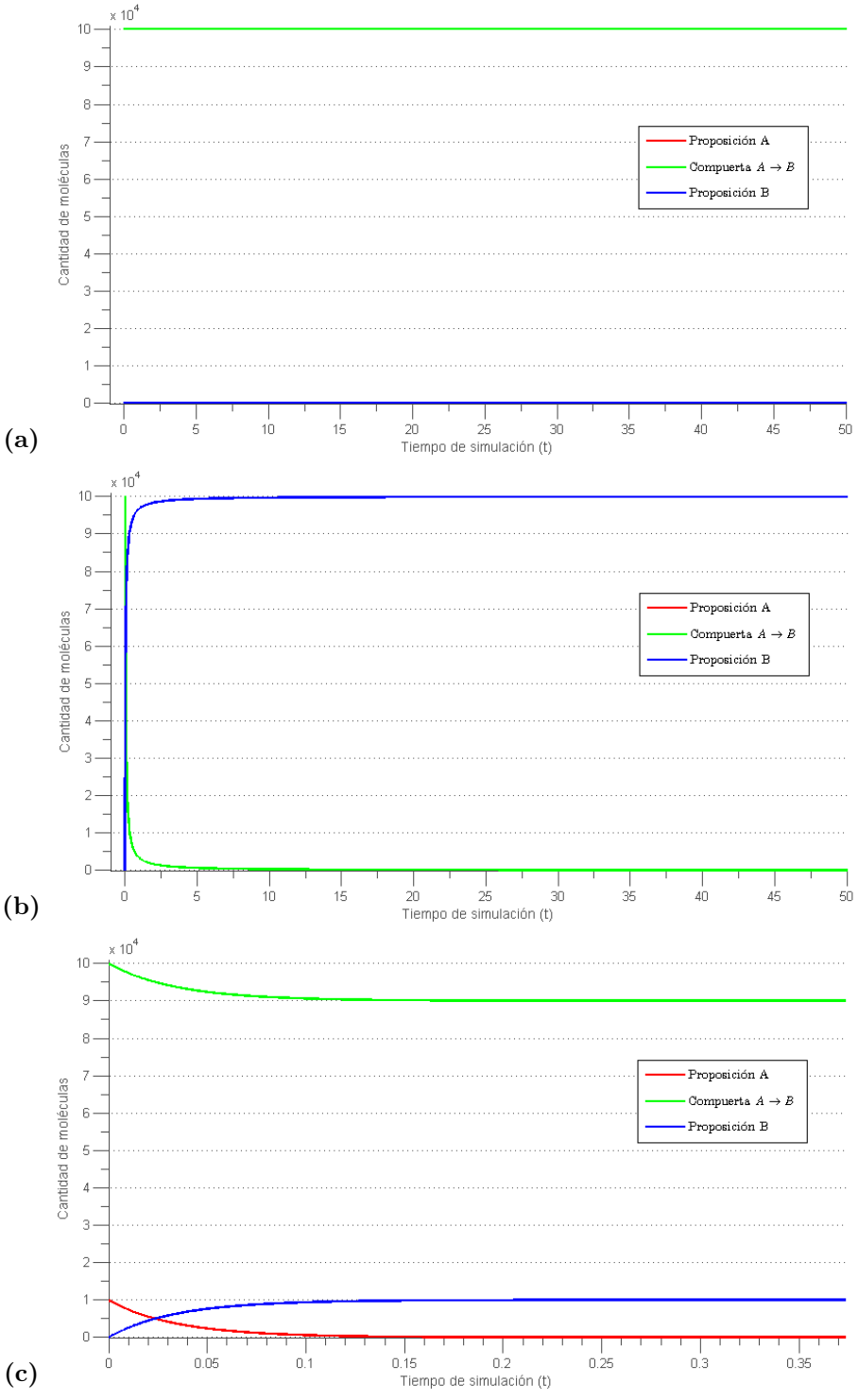


Figura 27: Promedio de 10 simulaciones de la estructura $A \rightarrow B$ realizadas con el simulador estocástico de Visual DSD. (a) Utilizando 0 moléculas de la premisa A y 100,000 moléculas de la compuerta. (b) Utilizando 100,000 moléculas de la proposición A y de la compuerta. (c) Utilizando 100,000 moléculas de la compuerta y 10,000 moléculas de la premisa A.

6.4.3.3. Análisis de propiedades utilizando la herramienta PRISM

Utilizando la herramienta PRISM se evaluaron las propiedades cuantitativas: la probabilidad de alcanzar cualquier estado terminal, la probabilidad de finalizar en un estado deseado, la probabilidad de terminar en un estado no deseado y la probabilidad de permanecer en el estado inicial. Se realizaron experimentos con cada una de las propiedades, evaluándolas en un lapso de 50,000 unidades de tiempo. En la Figura 28, se puede observar que la probabilidad de permanecer en el estado inicial disminuye rápidamente al paso del tiempo, hasta llegar a 0 alrededor de las 20,000 unidades de tiempo. La probabilidad de terminación errónea se mantiene constante en 0, ya que el sistema sólo posee una transición, por lo que es imposible que ocurra un error. Debido a lo anterior, la probabilidad de finalización exitosa y total son idénticas, y son inversamente proporcionales a la probabilidad de mantenerse en el estado inicial.

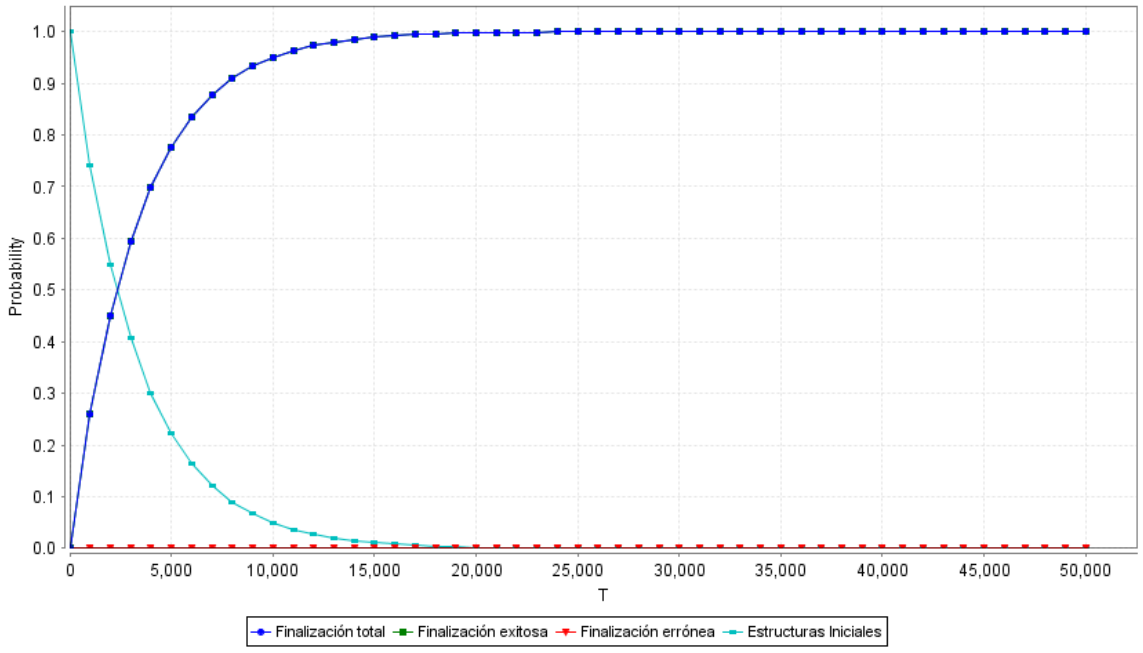


Figura 28: Probabilidad de todas las posibles finalizaciones del modelo de proposición condicional, en un periodo de 50,000 unidades de tiempo, calculado por PRISM.

6.5. Reglas de inferencia

Las reglas de inferencia crean relaciones sintácticas entre las premisas y la conclusión, y son utilizadas en el proceso de razonamiento para llegar a nuevos resultados a partir de hechos ya conocidos. En este modelo se incluyen estructuras para el uso de las reglas básicas de la lógica proposicional, como la simplificación, adición, silogismo hipotético, *modus ponens*, *modus tollens*, dilema constructivo y *modus tollendo ponens*.

6.5.1. *Modus ponens* y *modus tollens*

Como se menciona en las Secciones 2.6.1 y 2.6.2, las reglas *modus ponens* y *modus tollens*, consisten en una proposición condicional $A \rightarrow B$, la cual puede producir uno de dos resultados: En el *modus ponens*, se tiene como premisa que la proposición A es verdadera, por lo que la proposición B también lo es. En el *modus tollens*, la proposición B es falsa y se implica lógicamente que A también lo es.

6.5.1.1. Diseño de la compuerta

Debido a que las dos reglas de inferencia poseen un punto en común, para su diseño se consideraron como una sola estructura que produce un resultado distinto dependiendo de la proposición con la que reacciona. En la Figura 29, se muestra la compuerta diseñada, la cual está compuesta por dos proposiciones condicionales. El extremo izquierdo correspondiente al *modus ponens*, reacciona con la proposición A y libera B como resultado. El extremo derecho, equivalente al *modus tollens*, recibe $\neg B$ y produce la proposición $\neg A$. La estructura posee un *toehold* genérico central, con un dominio representativo a cada lado (a la izquierda el complemento de A y a la derecha el de $\neg B$). De esta manera, se asegura que sólo una de las dos proposiciones reaccione con la estructura y se produzca uno de los dos resultados posibles.

En la Figura 30 se muestra la resolución de la estructura a través del *modus ponens*. La proposición A se une a la estructura a través de su punto de apoyo central, dado que

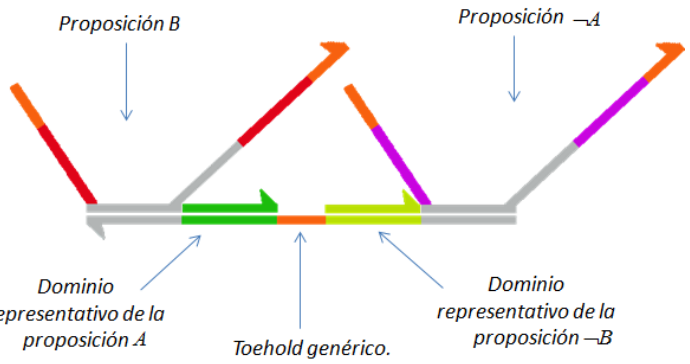


Figura 29: Representación de las reglas *Modus ponens* y *Modus tollens*. En el centro se encuentra una zona de reconocimiento, conformada por el complemento del punto de apoyo genérico con el complemento de un dominio representativo a cada lado. En el lado izquierdo se recibe la proposición A y se produce B, de acuerdo al *modus ponens*. El lado derecho representa al *modus tollens*: se lee la proposición $\neg B$ y se libera $\neg A$.

su dominio representativo corresponde al complemento izquierdo, también se adhiere a él, desplazando a la cadena de cobertura con la que estaba hibridado. Posteriormente, une su primer dominio auxiliar al de la compuerta, y con esto libera a la proposición B, que representa el consecuente en el *modus ponens*. Observe que cualquier proposición es capaz de unirse al *toehold* central. Sin embargo, sólo aquella que cuente con el dominio representativo correspondiente podrá realizar el desplazamiento de las cadenas de cobertura y producir un resultado. Las cadenas cuyo dominio no corresponda con el de la estructura se desprenderán eventualmente, debido a que el punto de apoyo es un dominio corto que no le brinda la suficiente estabilidad para mantenerla unida a la compuerta.

La resolución de la estructura a través del *modus tollens* se lleva a cabo de una forma similar. En este caso la proposición $\neg B$ reacciona con el punto de apoyo central, y dado que su dominio representativo corresponde al complemento de la zona de reconocimiento del lado derecho, desplaza su cadena de cobertura. Posteriormente, une su dominio auxiliar al de la compuerta, liberando la proposición $\neg A$ como resultado.

Los diagramas de estados obtenidos por la herramienta Visual DSD se muestran en la Figura B.01 (Apéndice). En ambos casos se encuentran sólo el estado inicial y un estado final, con una transición entre ellos con probabilidad de transición de 1.

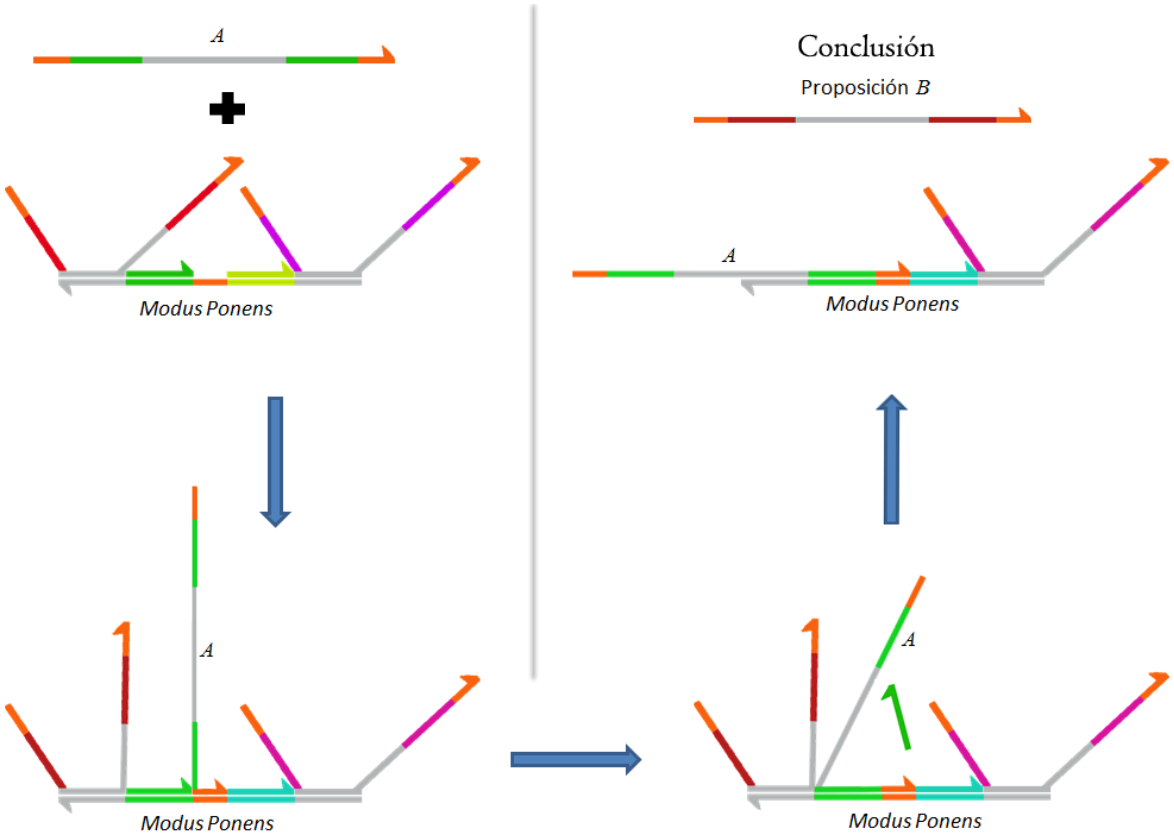


Figura 30: Procedimiento para la resolución de la regla *modus ponens*. La proposición A une su punto de apoyo al complemento en el centro de la estructura, y comienza a desplazar la cadena de cobertura del complemento del dominio representativo de la izquierda. Posteriormente, une su dominio auxiliar al de la compuerta, liberando B como resultado de la inferencia.

6.5.1.2. Simulación del *modus ponens*

Debido a que esta estructura incluye dos reglas de inferencia, las simulaciones se realizaron individualmente utilizando el algoritmo de Gillespie incluído en la herramienta Visual DSD. Se crearon 9 casos de prueba para cada regla de inferencia, cada uno de los cuales posee una combinación de 1,000, 10,000 ó 100,000 moléculas de la proposición y de la compuerta.

La Figura 31 muestra las gráficas del promedio de 10 simulaciones de la regla *modus ponens* utilizando el mismo número de moléculas tanto en la compuerta como en la proposición A. En (a) se utilizaron 1,000 moléculas de la proposición A y de la compuerta, una duración de 5,000 unidades de tiempo y 50,000 puntos de muestra. En la gráfica se muestran las primeras 1,500 unidades de tiempo, esto con el propósito de hacer un mayor énfasis a los primeros instantes de la simulación, en los cuales se llevan a cabo la mayor parte de las reacciones. Debido a lo anterior, la proposición A y la compuerta (que tienen el mismo comportamiento) parecen nunca agotarse totalmente; sin embargo, se alcanza a observar que disminuyen, y de hecho en la gráfica completa, eventualmente lo hacen. De manera similar, la gráfica de la proposición B aumenta conforme se consumen la compuerta y la proposición A, hasta que eventualmente alcanza un total de 1,000 moléculas.

El inciso (b) de la Figura 31, es el resultado de 10 simulaciones realizadas con el caso de prueba de 10,000 moléculas tanto de la compuerta como de la premisa A, los parámetros utilizados incluyen una duración de 5,000 unidades de tiempo y 30,000 puntos de muestra. Observe que la gráfica posee el mismo comportamiento que la del caso anterior, sin embargo, al aumentar la concentración de moléculas el tiempo de simulación disminuye considerablemente y al llegar a las 300 unidades de tiempo ya se han consumido todas las estructuras iniciales y se han producido 10,000 moléculas de la premisa B como conclusión, por lo cual se muestran solamente las primeras 500 unidades de tiempo. Al igual que en el caso anterior, la gráfica de la proposición A no es visible al poseer el mismo comportamiento que la compuerta. Por último, en (c) se muestra el resultado de 10 simulaciones realizadas con el caso de prueba de 100,000 moléculas tanto de la compuerta como de la proposición A. Los parámetros incluyen

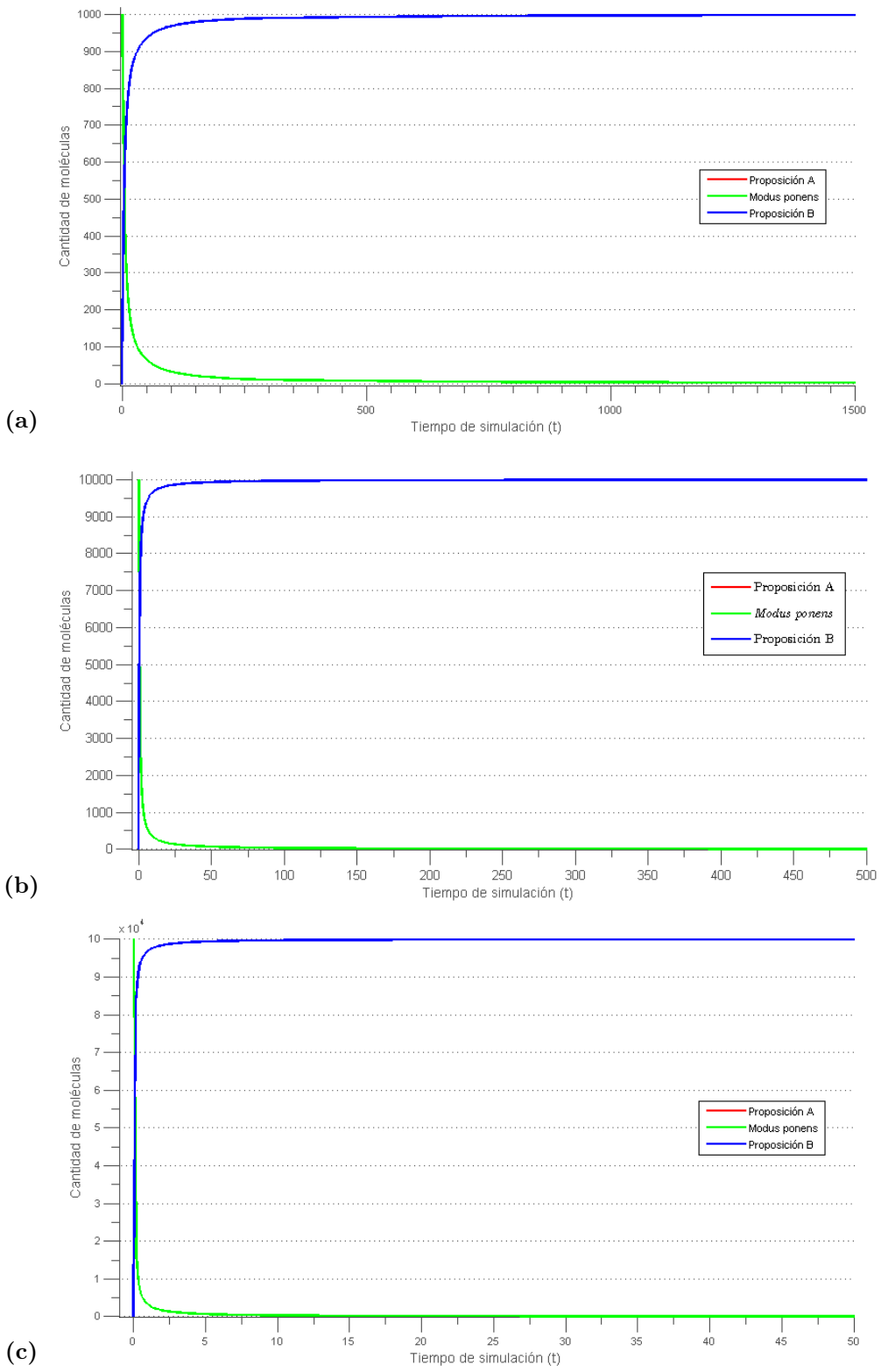


Figura 31: Promedio de 10 simulaciones de la estructura *modus ponens* realizadas con el simulador estocástico de Visual DSD. (a) Utilizando 1,000 moléculas de las premisa A y de la compuerta. (b) Utilizando 10,000 moléculas de la proposición A y de la compuerta. (c) Utilizando 100,000 moléculas de la compuerta y de la premisa A.

una duración de 5,000 unidades de tiempo y 30,000 puntos de muestra. En este caso de prueba, las estructuras de la compuerta y la proposición A se consumen más rápidamente y alrededor de las 10 unidades de tiempo ya no existen más reacciones, como es de esperarse al aumentar el número de moléculas. Las gráficas de la compuerta y la proposición A se encuentran encimadas debido a que se consumen mutuamente y por lo tanto, tienen el mismo comportamiento. La gráfica de la proposición B, producida como resultado de la inferencia lógica, aumenta rápidamente mientras las compuertas se consumen. Observe que en el tiempo 5 se han producido la mayoría de los resultados, y a las 10 unidades de tiempo se alcanzan las 100,000 moléculas, que es la totalidad de soluciones esperadas.

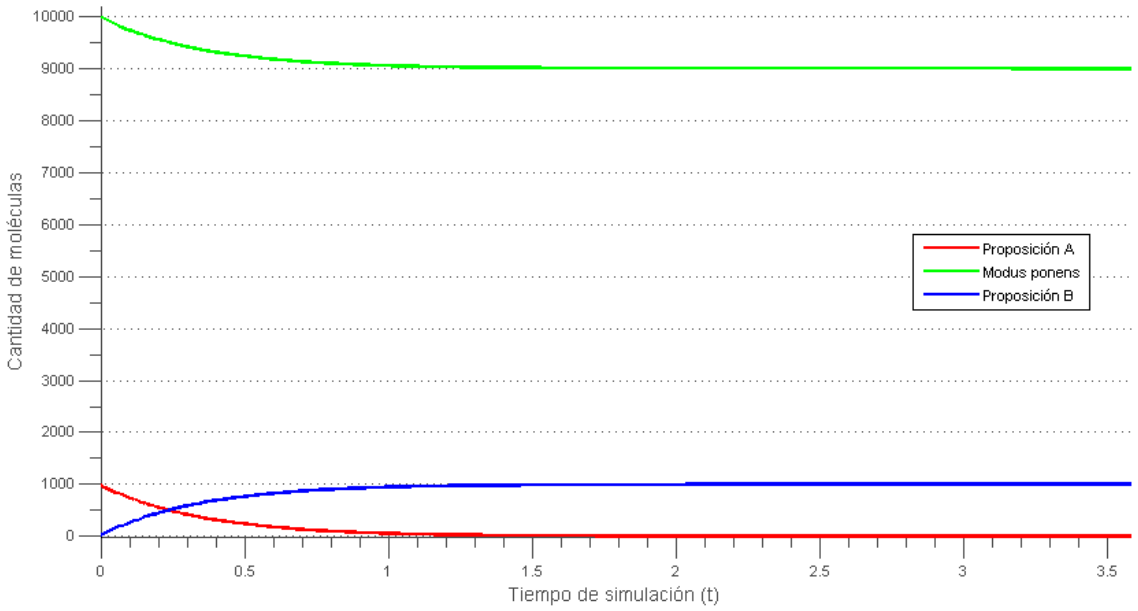


Figura 32: Promedio de 10 simulaciones realizadas en el simulador estocástico de Visual DSD con la estructura del *modus ponens*, utilizando una combinación de 1,000 moléculas de la proposición A y 10,000 moléculas de la compuerta.

El último caso de prueba del *modus ponens*, se muestra en la Figura 32. Esta gráfica, es el resultado de 10 simulaciones realizadas con diferente número de moléculas: 10,000 de la compuerta y 1,000 de la proposición A. Además, se utilizaron una duración de 5,000 unidades de tiempo y 1,000,000 de puntos de muestra. Al aumentar 10 veces el número de moléculas de la compuerta sobre el número de proposiciones A, se produce una reducción

drástica en el tiempo de simulación. Para la segunda unidad de tiempo se han consumido todas las moléculas de la proposición A, el número de compuertas se ha reducido a 9,000 y se han producido 1,000 moléculas de la proposición B, alcanzando la totalidad de resultados esperados.

6.5.1.3. Análisis del *modus ponens* con PRISM

El análisis del modelo de la estructura se realizó separando las dos reglas de inferencia, al igual que en la simulación, ya que ambos casos no pueden ocurrir al mismo tiempo. Las siguientes verificaciones se llevaron a cabo en forma de experimentos, evaluando las propiedades cuantitativas explicadas en la introducción del capítulo sobre un periodo de tiempo de 50,000 unidades. Estas propiedades son las probabilidades de todas las posibles finalizaciones del sistema: la probabilidad de finalizar en cualquier estado, la de terminar en un estado final deseado, la de finalizar en un estado no deseado y por último, la de permanecer en el estado inicial.

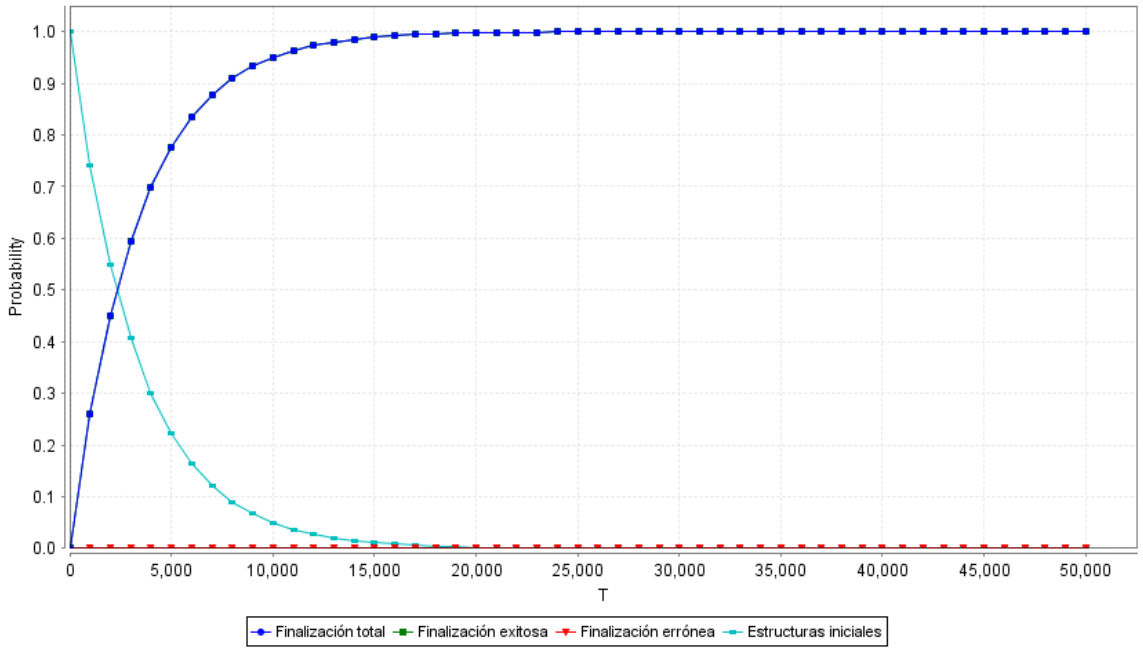


Figura 33: Gráfica de los experimentos realizados en PRISM al evaluar la probabilidad de todas las finalizaciones del modelo de la estructura de *modus ponens* sobre un periodo de 50,000 unidades de tiempo.

La Figura 33 muestra la gráfica generada por PRISM para los experimentos realizados. Puede observarse que la probabilidad de permanecer en el estado inicial disminuye rápidamente al paso del tiempo hasta llegar a 0, mientras que la probabilidad de terminación errónea se mantiene constante en 0. Por esto, la probabilidad de finalización exitosa y la total son idénticas, ya que todos los estados finales son deseados. Ambas gráficas tienen el mismo comportamiento, y la finalización total cubre la línea de finalización exitosa, evitando que ésta sea visible. Como es de esperarse en un sistema que cuenta con sólo dos estados y una transición entre ellos, la probabilidad de finalización exitosa aumenta al pasar el tiempo hasta alcanzar la probabilidad de 1, alrededor de las 20,000 unidades de tiempo, y a partir de ese punto se mantiene constante.

6.5.1.4. Simulación de la regla *modus tollens*

Las simulaciones del *modus tollens* se realizaron utilizando el simulador estocástico incluído en la herramienta Visual DSD. Al igual que en el *modus ponens*, se crearon 9 casos de prueba, cada uno de los cuales posee una combinación de 1,000, 10,000 ó 100,000 moléculas de la proposición $\neg B$ y de la compuerta.

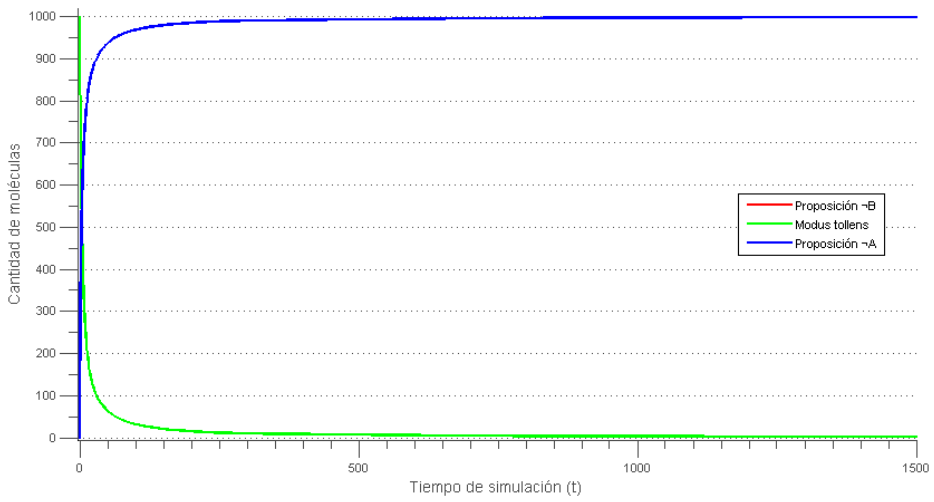
El primer caso de prueba se muestra en la gráfica (a) de la Figura 34, la cual corresponde al promedio de 10 simulaciones utilizando 1,000 moléculas de la proposición $\neg B$ y de la compuerta. Los parámetros incluyen una duración de 5,000 unidades de tiempo y 60,000 puntos de muestra. La gráfica muestra las primeras 1,500 unidades de tiempo, ya que la mayoría de las reacciones se llevan a cabo en los primeros instantes de la simulación. La línea de la proposición $\neg B$ no es visible, ya que es cubierta por la compuerta del *modus tollens*. Dado que se consumen mutuamente, poseen el mismo comportamiento. Ambas disminuyen al avanzar el tiempo de simulación, iniciando a una gran velocidad y alentándose conforme disminuye el número de moléculas reactivas, hasta que eventualmente se consumen todas las estructuras y no hay más reacciones posibles. La proposición $\neg A$ posee un comportamiento inversamente proporcional a la proposición $\neg B$ y a la compuerta, ya que por cada par de

éstas que se consumen, se produce un resultado. La gráfica (b) pertenece al caso del prueba en el cual se utilizaron 10,000 moléculas tanto de la premisa $\neg B$ como de la compuerta, así como una duración de 5,000 unidades de tiempo y 50,000 puntos de muestra. Al aumentar la concentración de moléculas se disminuye el tiempo necesario para que se consuman las estructuras iniciales. A diferencia de la simulación anterior, en la cual no es visible el tiempo en el cual se agotan las premisas $\neg B$ y compuertas, se puede observar que en esta simulación esto ocurre alrededor de las 200 unidades de tiempo.

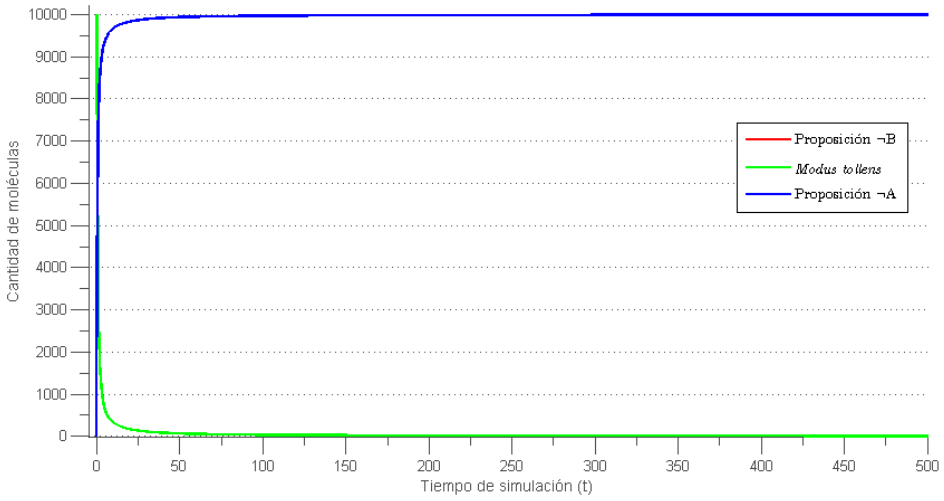
En la Figura 34(c), se muestra la gráfica del promedio de 10 simulaciones del caso de prueba con 100,000 moléculas de la proposición $\neg B$ y de la compuerta de *modus tollens*. Los parámetros incluyen una duración de 5,000 unidades de tiempo y 50,000 puntos de muestra. Lo primero que se observa en esta gráfica es que el tiempo necesario para la finalización de la simulación disminuye al aumentar el número de moléculas, debido a que existe una mayor probabilidad de que las reacciones se lleven a cabo al existir una mayor concentración. Al llegar al tiempo 15, las compuertas y proposiciones $\neg B$ se han consumido totalmente, y se han producido 100,000 moléculas de la proposición $\neg A$ como resultado de la inferencia lógica, alcanzando la totalidad de soluciones esperadas. Al igual que en las gráficas anteriores, la proposición $\neg B$ y la compuerta se consumen mutuamente y poseen el mismo comportamiento, por lo cual la línea de la proposición $\neg B$ no es visible, al encontrarse debajo de la línea de la compuerta.

Por último, en la Figura 35, se muestra la la gráfica del promedio de 10 simulaciones utilizando 1,000 moléculas de la proposición $\neg B$ y 10,000 de la compuerta. Los parámetros incluyen una duración de 5,000 unidades de tiempo y 1,000,000 de puntos de muestra. Observe que al utilizar un mayor número de moléculas de la compuerta que de la proposición inicial, la cantidad de reacciones que pueden llevarse a cabo está limitada por esta última, reduciéndose así el tiempo de simulación. Las estructura del *modus tollens* y la proposición $\neg B$ se consumen mutuamente entre el primer y segundo tiempo de la simulación, liberando como resultado 1,000 moléculas de la proposición $\neg A$.

(a)



(b)



(c)

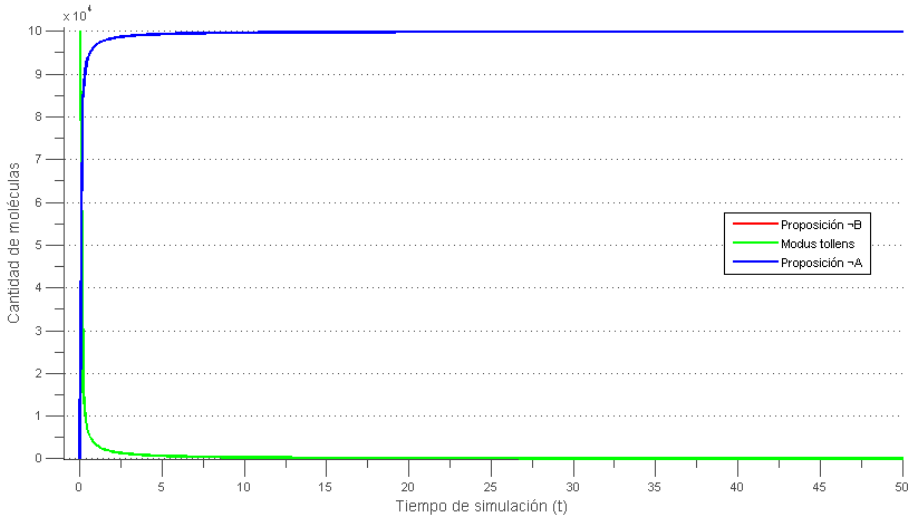


Figura 34: Promedio de 10 simulaciones de la estructura *modus tollens* realizadas con el simulador estocástico de Visual DSD. (a) Utilizando 1,000 moléculas de las premisa $\neg B$ y de la compuerta. (b) Utilizando 10,000 moléculas de la proposición $\neg B$ y de la compuerta. (c) Utilizando 100,000 moléculas de la compuerta y de la premisa $\neg B$.

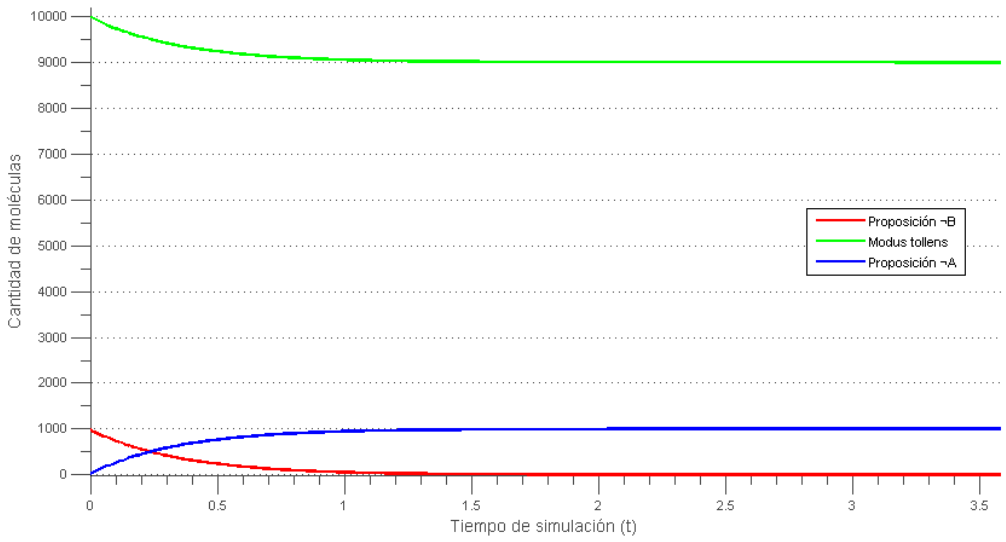


Figura 35: Promedio de 10 simulaciones de la estructura diseñada para el *modus tollens*, utilizando una combinación de 1,000 moléculas de la proposición $\neg B$ y 10,000 moléculas de la compuerta.

6.5.1.5. Análisis del *modus tollens* con PRISM

Posteriormente se realizó la verificación de las propiedades cuantitativas anteriormente definidas. Para esto, se llevaron a cabo experimentos con lógica temporal CSL, evaluándolas sobre un lapso de tiempo de 50,000 unidades. Las propiedades evaluadas son: la probabilidad de finalizar en cualquier estado, la de terminar en un estado final deseado, la de finalizar en un estado no deseado, y por último, la de permanecer en el estado inicial.

En la Figura 36, se muestra la gráfica generada por PRISM para los experimentos realizados. Como se puede observar, dado que el sistema cuenta sólo con dos estados y una transición, la probabilidad de alcanzar un estado final erróneo es siempre igual a 0, y la probabilidad de alcanzar un estado final exitoso es igual a la probabilidad de finalización total. Conforme pasa el tiempo, éstas aumentan hasta alcanzar la probabilidad de 1, mientras que la probabilidad de permanecer en el estado inicial es inversamente proporcional a la finalización total, y disminuye al avanzar el tiempo hasta alcanzar una probabiliad de 0 entre las 15,000 y 20,000 unidades de tiempo.

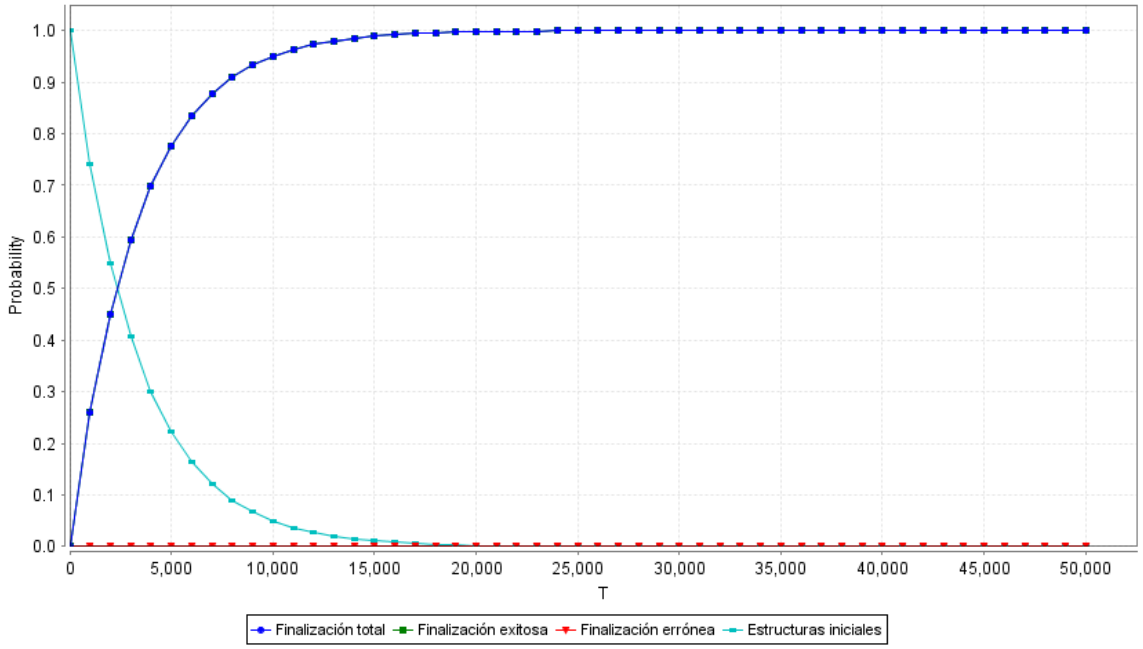


Figura 36: Resultado de los experimentos realizados en PRISM, evaluando la probabilidad de todas las posibles finalizaciones del modelo de la estructura diseñada para el *modus tollens* sobre 50,000 unidades de tiempo.

6.5.2. Adición

Como se menciona en la Sección 2.6.4, la regla de adición consiste en una proposición A, a la cual se une una proposición B (la cual puede o no existir actualmente) a través de un operador de disyunción, produciendo la premisa $A \vee B$ como resultado.

6.5.2.1. Diseño de la compuerta

La compuerta de adición posee una estructura similar a la de proposición codicional: una zona de reconocimiento compuesta por el complemento del *toehold* genérico y un complemento del dominio representativo de la proposición que se desea utilizar, hibridado a su correspondiente cadena de cobertura. La principal diferencia entre ambas compuertas consiste en que en la adición, el extremo derecho posee un dominio auxiliar unido a la proposición temporal que será producida como conclusión, mientras que el resultado de la proposición condicional consiste en una proposición simple. La Figura 37 muestra la estructura de la adición $A \implies A \vee B$.

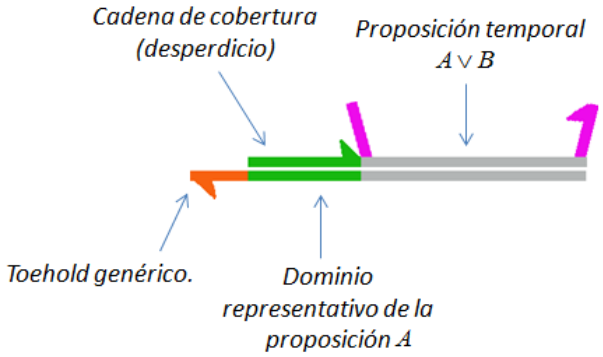


Figura 37: Representación de la estructura de adición. En el extremo izquierdo se encuentra una zona de reconocimiento conformada por un complemento del punto de apoyo genérico y el dominio representativo de la proposición A, a su derecha se encuentra un complemento del dominio auxiliar unido a la proposición temporal $A \vee B$ que será liberada como resultado.

Al igual que en las estructuras anteriores, las reacciones se llevan a cabo a través de la zona de reconocimiento. Las proposiciones unen su punto de apoyo genérico a su complemento en el área de reconocimiento de la compuerta, si cuenta con el dominio representativo correspondiente, comienza a desplazar su cadena de cobertura. Posteriormente, la premisa desplaza a la proposición consecuente que está unida al dominio auxiliar, y con esto produce un resultado. Las proposiciones que no poseen el dominio representativo requerido, pueden unir su punto de apoyo genérico al de la estructura pero no realizan un desplazamiento de la cadena de cobertura y por lo tanto no producen un resultado. Eventualmente estas cadenas se liberan, ya que el *toehold* con el que están unidas es un dominio corto y carece de estabilidad.

La Figura 38 muestra la resolución de la regla de adición $A \implies A \vee B$. La proposición A se une al área de reconocimiento de la compuerta utilizando su punto de apoyo genérico. Dado que su dominio representativo corresponde al complemento de la estructura, se une a él, desplazando a la cadena de cobertura con la que estaba hibridado. Posteriormente, une su primer dominio auxiliar al de la compuerta y con esto libera a la proposición temporal $A \vee B$ como conclusión.

Analizando la estructura de adición con la herramienta Visual DSD, se obtuvo el diagrama de transición de estados mostrado en la Figura B.01 del Apéndice . El sistema cuenta

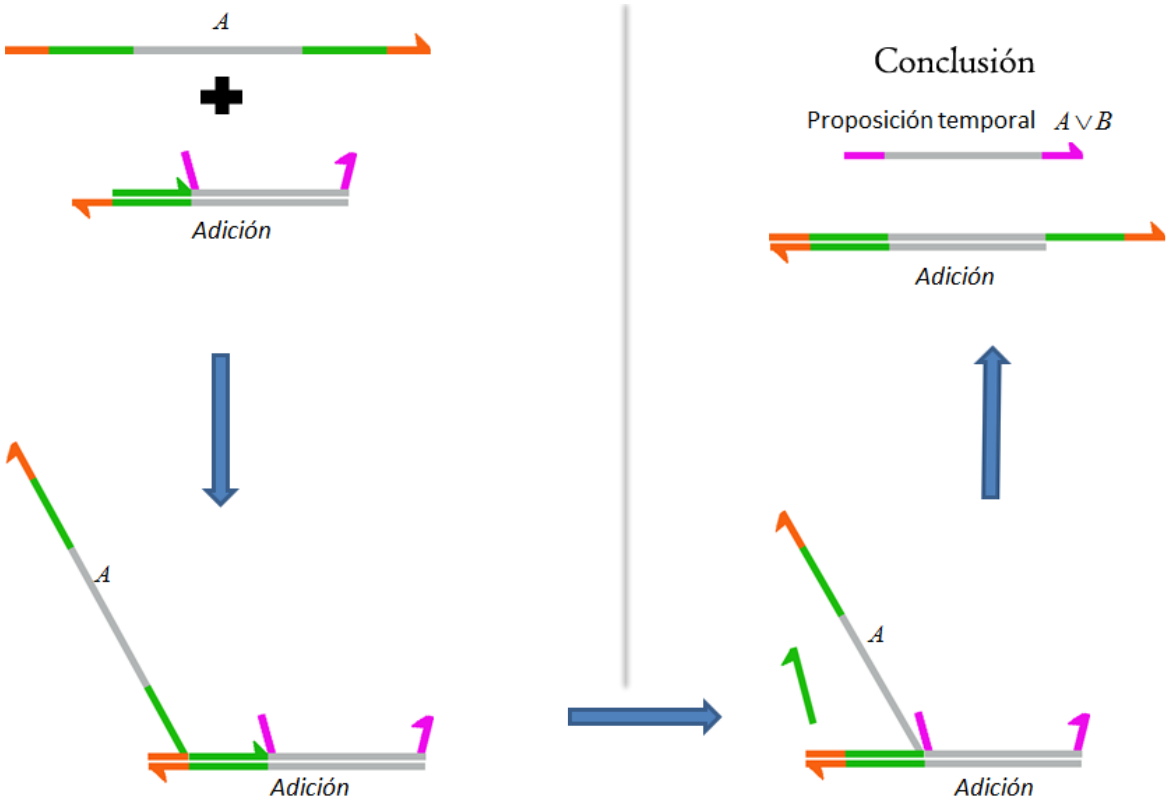


Figura 38: Resolución esquemática de la regla de adición $A \implies A \vee B$. La proposición A, une su punto de apoyo a su complemento en la zona de reconocimiento de la estructura. Posteriormente, desplaza la cobertura del complemento del dominio representativo y continúa uniendo su dominio auxiliar al de la estructura, liberando en el proceso a la proposición temporal $A \vee B$.

solamente con dos estados: el inicial y el terminal, y una transición entre ellos.

6.5.2.2. Simulación estocástica con Visual DSD

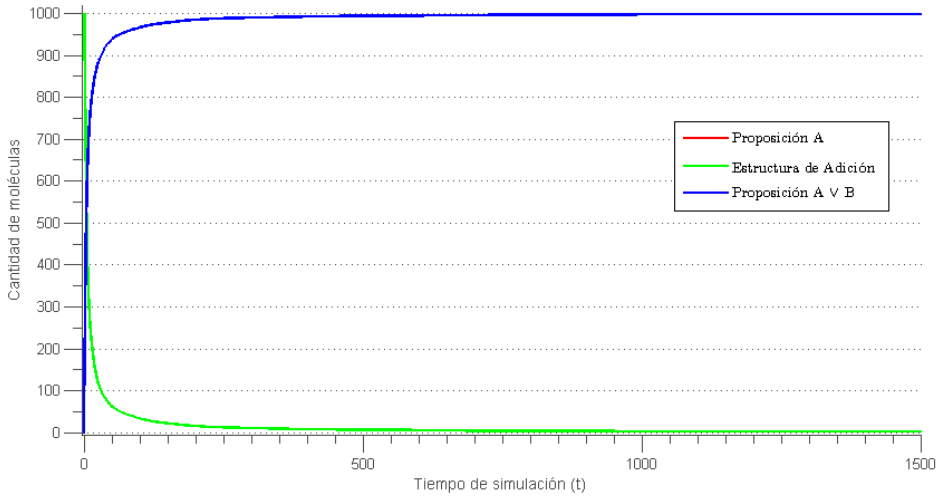
Las simulaciones de la estructura de adición se realizaron utilizando el algoritmo de Gillespie incluído en la herramienta Visual DSD de Microsoft. En total se crearon 9 casos de prueba, cada uno de los cuales es una combinación de 1,000, 10,000, ó 100,000 moléculas de concentración de la proposición y la compuerta.

La Figura 39 muestra las gráfica generada por el promedio de 10 simulaciones. En la gráfica (a) se utilizaron 1,000 moléculas de la premisa A y de la compuerta de adición, los parámetros incluyen una duración de 5,000 unidades de tiempo y 50,000 puntos de muestra. Como puede observarse, sólo se muestran las primeras 1,500 unidades de tiempo, con el fin de hacer énfasis en el inicio de la simulación, que es cuando se producen la mayoría de las reacciones, es por esto que la gráfica de la compuerta parece nunca consumirse totalmente, sin embargo, en la gráfica completa eventualmente lo hace. Alrededor de las 1,000 unidades de tiempo se han consumido la mayor parte de las estructuras iniciales y se han producido 1,000 moléculas de la proposición temporal $A \vee B$. Al igual que en simulaciones anteriores, la gráfica de la proposición A no es visible, al consumirse al mismo tiempo que la compuerta, por lo que sus gráficas comparte el mismo comportamiento.

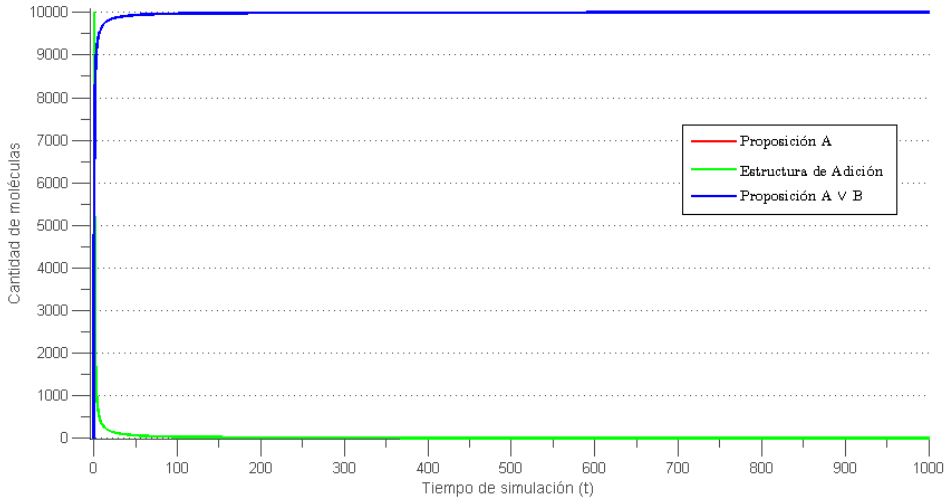
La gráfica 39(b) presenta el resultado de utilizar 10,000 moléculas tanto de la premisa A como de la compuerta de adición, una duración de 5,000 unidades de tiempo y 50,000 puntos de muestra. En este caso todas las conclusiones $A \vee B$ se liberan en las primeras 600 unidades de tiempo, dado que esta compuerta sólo posee una transición entre el estado inicial y final, siempre que existen moléculas de la premisa A y se le brinde el tiempo suficiente, se produce un resultado. Al igual que en la simulación anterior, la gráfica de la proposición A no es visible al consumirse al mismo tiempo que la compuerta. Por último, la gráfica (c) muestra el resultado de 10 simulaciones realizadas utilizando 100,000 moléculas de la proposición A y 100,000 moléculas de la compuerta de adición, con una duración de 5,000 unidades de

tiempo y 30,000 puntos de muestra. Como se puede observar, a pesar de utilizar un tiempo de simulación de 5,000 unidades, la simulación finaliza antes del tiempo 100. La razón es que al existir un mayor número de moléculas, aumenta la probabilidad de que se lleven a cabo las reacciones, produciendo un resultado en un menor tiempo. Al igual que en previas simulaciones, la proposición A no es visible, ya que se consume al mismo tiempo que la compuerta y por lo tanto sus gráficas son idénticas.

(a)



(b)



(c)

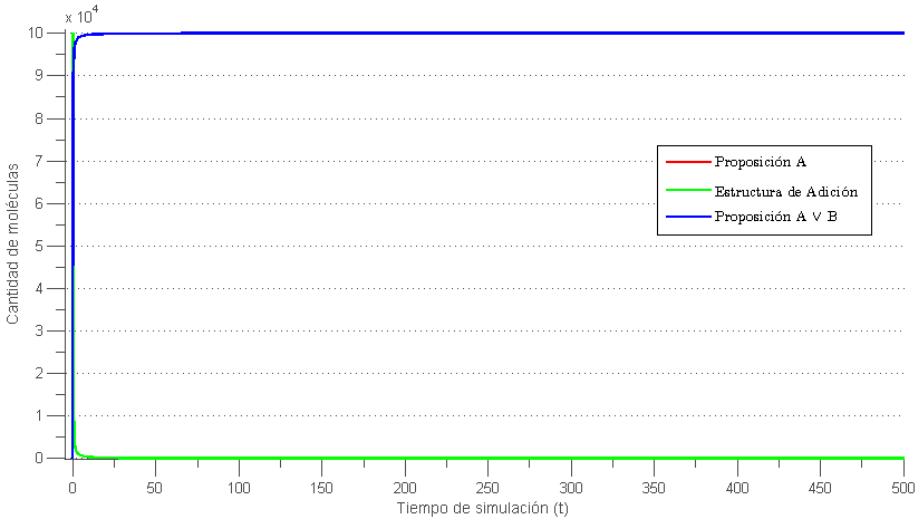


Figura 39: Promedio de 10 simulaciones de la estructura adición $A \implies A \vee B$ realizadas con el simulador estocástico de Visual DSD. (a) Utilizando 1,000 moléculas de las premisa A y 1,000 moléculas de la compuerta. (b) Utilizando 10,000 moléculas de la proposición A y de la compuerta. (c) Utilizando 100,000 moléculas tanto de la compuerta como de la premisa A.

6.5.2.3. Análisis de propiedades utilizando la herramienta PRISM

Para confirmar el correcto funcionamiento de la estructura de adición, se realizaron experimentos con la herramienta PRISM para verificar las probabilidades de alcanzar un estado final exitoso, terminar en un estado no deseado, finalizar en cualquier estado terminal, así como la probabilidad de encontrarse en el estado inicial después de un cierto periodo de tiempo. La Figura 40 muestra las probabilidades de todas las posibles finalizaciones del modelo de adición en un tiempo T, utilizando una duración de 50,000 unidades de tiempo. Como se espera de un modelo que cuenta sólo con un estado inicial y uno terminal, la probabilidad de finalización total y la de finalización exitosa son iguales y entre las 15,000 y 20,000 unidades de tiempo tienden a ser 1. Por lo cual la probabilidad de finalizar en un estado erróneo permanece en 0 constantemente, mientras que la probabilidad de mantenerse en el estado inicial disminuye al aumentar el tiempo y la probabilidad de finalización exitosa.

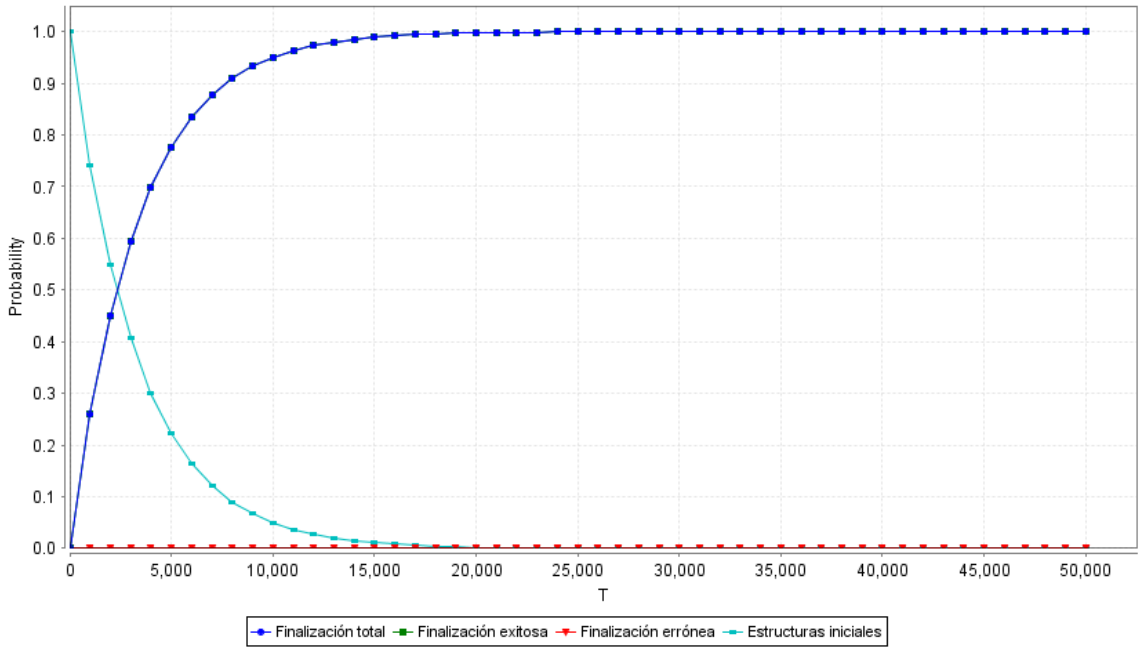


Figura 40: Gráfica de probabilidades de todas las posibles finalizaciones del modelo de adición generado con la herramienta PRISM.

6.5.3. Simplificación

En la Sección 2.6.3, se define a la simplificación como una regla de inferencia que permite separar una proposición compuesta por un conectivo de conjunción en sus proposiciones básicas. Por lo cual, al contrario de la adición, esta estructura recibe como entrada una proposición temporal y libera como conclusión a una proposición simple.

6.5.3.1. Diseño de la compuerta

Al igual que la compuerta de adición, la simplificación posee una estructura similar a la proposición condicional: en el extremo izquierdo posee una zona de reconocimiento, que en este caso está compuesta por el complemento del *toehold* único representativo de la proposición temporal que se utilizará como entrada. Del lado derecho posee dos dominios auxiliares unidos a la proposición simple que será liberada como conclusión. La Figura 41 muestra la estructura de la simplificación $A \wedge B \implies A$.

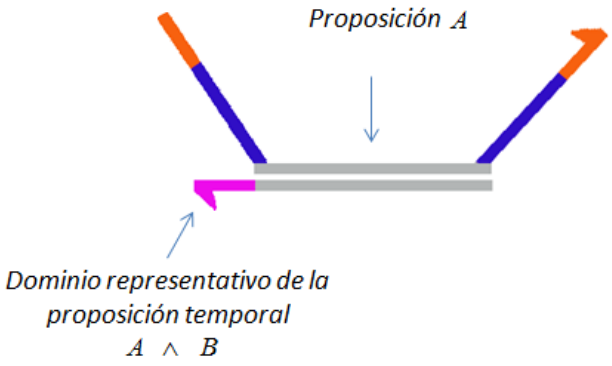


Figura 41: Representación de la estructura de simplificación. En el extremo izquierdo se encuentra una zona de reconocimiento conformada por un complemento de un punto de apoyo único representativo de la proposición temporal $A \wedge B$, a su derecha se encuentran dos complementos del dominio auxiliar unidos a la proposición A que será liberada como resultado.

En esta estructura las reacciones inician en la zona de reconocimiento del lado izquierdo, una vez que las proposiciones temporales se encuentran presentes, unen su punto de apoyo único a su complemento en la compuerta, que en éste es representativo de la proposición temporal deseada, por lo cual, cualquier premisa capaz de unirse a la zona de reconocimiento

es capaz de continuar uniendo sus dominimos auxiliares a los de la compuerta, desplazando en el proceso a la proposición unida a ellos y la libera como conclusión.

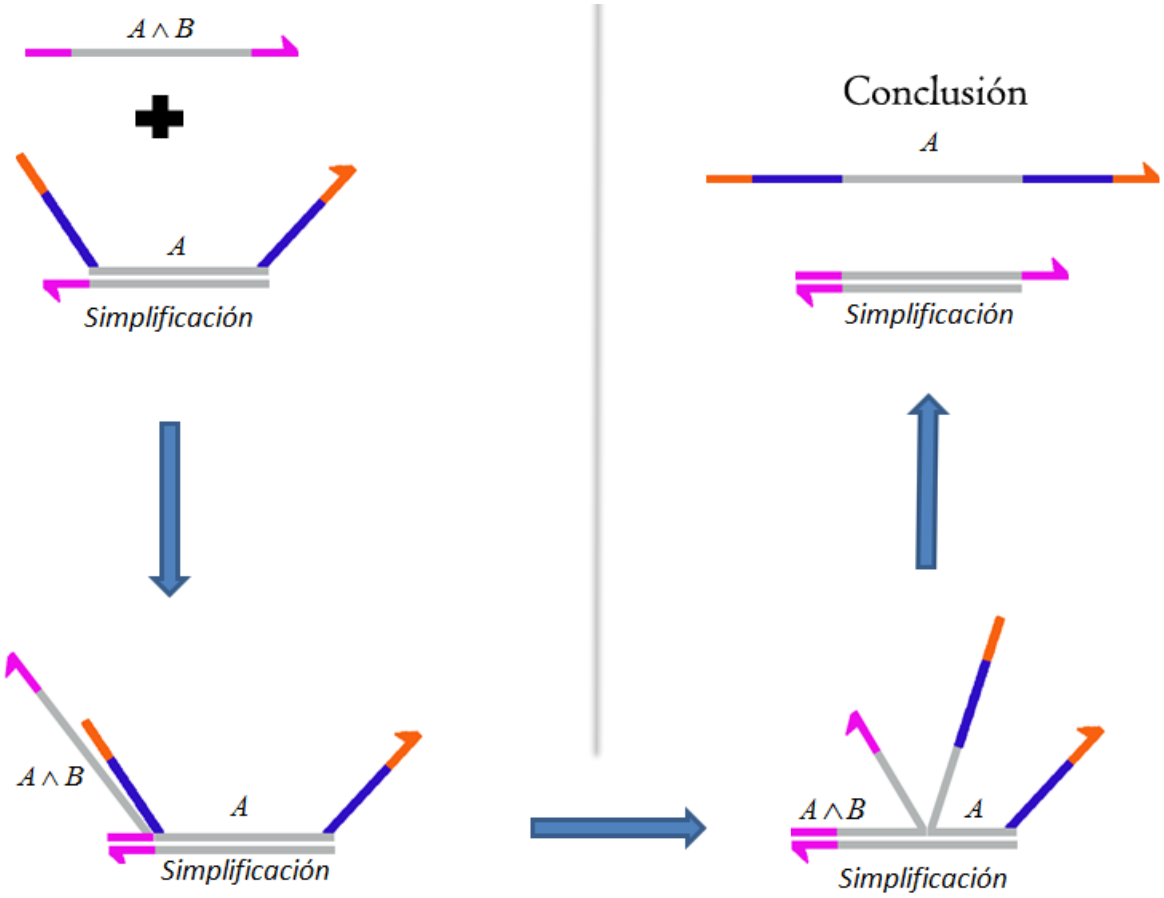


Figura 42: Resolución esquemática de la regla de simplificación $A \wedge B \implies A$. La proposición temporal $A \wedge B$, une su punto de apoyo a su complemento en la zona de reconocimiento de la estructura. Posteriormente, desplaza las coberturas de los dominios auxiliares de la estructura, liberando en el proceso a la proposición A.

Utilizando la herramienta Visual DSD para realizar el análisis de la estructura de simplificación se obtuvo el diagrama de transición de estados mostrados en la Figura B.01 del Apéndice . Como puede observarse, el sistema cuenta con dos estados: el inicial y el terminal, y una transición entre ellos.

6.5.3.2. Simulación estocástica con Visual DSD

Para llevar a cabo las pruebas de la estructura de simplificación se crearon un total de 9 casos de prueba, utilizando en cada uno de ellos una combinación de 1,000, 10,000 ó 100,000 moléculas en la proposición temporal y la compuerta. Posteriormente se utilizó el simulador estocástico basado en el algoritmo de Gillespie incluído en la herramienta Visual DSD para realizar las simulaciones de dichos casos de prueba.

En la Figura 43 se muestran 3 gráficas del promedio de 10 simulaciones de los casos de prueba en los cuales se utiliza la misma concentración de moléculas de la proposición temporal y de la estructura de simplificación. En la gráfica (a) se utilizaron 1,000 moléculas de la premisa $A \wedge B$ y de la compuerta de simplificación. Los parámetros incluyen una duración de 5,000 unidades de tiempo y 10,000 puntos de muestra. Al igual que en simulaciones de estructuras anteriores, en esta gráfica solamente se muestran las primeras 1,500 unidades de tiempo, con el fin de hacer énfasis en el inicio de la simulación, donde se llevan a cabo la mayoría de las reacciones. Como se puede observar, el comportamiento de la compuerta y la proposición temporal es idéntico e inversamente proporcional a la proposición A y alrededor de las 800 unidades de tiempo se han liberado 1,000 moléculas de conclusión.

La gráfica 43(b) corresponde al resultado de utilizar 10,000 moléculas tanto de la premisa temporal $A \wedge B$ como de la compuerta de simplificación, además de una duración de 5,000 unidades de tiempo y 30,000 puntos de muestra. Como es de esperarse al aumentar la concentración de las estructuras iniciales, las reacciones se llevan a cabo a una mayor velocidad y todas las estructuras iniciales se consumen en las primeras 250 unidades de tiempo, liberando en el proceso 10,000 moléculas de la conclusión A. Al igual que en la simulación anterior, la gráfica de la proposición temporal $A \wedge B$ no es visible al consumirse al mismo tiempo que la compuerta.

Por último, la gráfica (c) muestra el resultado de 10 simulaciones realizadas utilizando 100,000 moléculas de la proposición temporal $A \wedge B$ y 100,000 moléculas de la compuerta de simplificación. Los parámetros incluyen una duración de 5,000 unidades de tiempo y

30,000 puntos de muestra. A pesar de utilizar un tiempo de simulación de 5,000 unidades, las reacciones se llevan a cabo en las primeras 70 unidades de tiempo ya que existen un mayor número de moléculas y por lo tanto existe una mayor probabilidad de que reaccionen entre ellas. Al igual que en las simulaciones anteriores, las estructuras iniciales tienen el mismo comportamiento al consumirse mutuamente, por lo cual la proposicición temporal $A \wedge B$ no es visible.

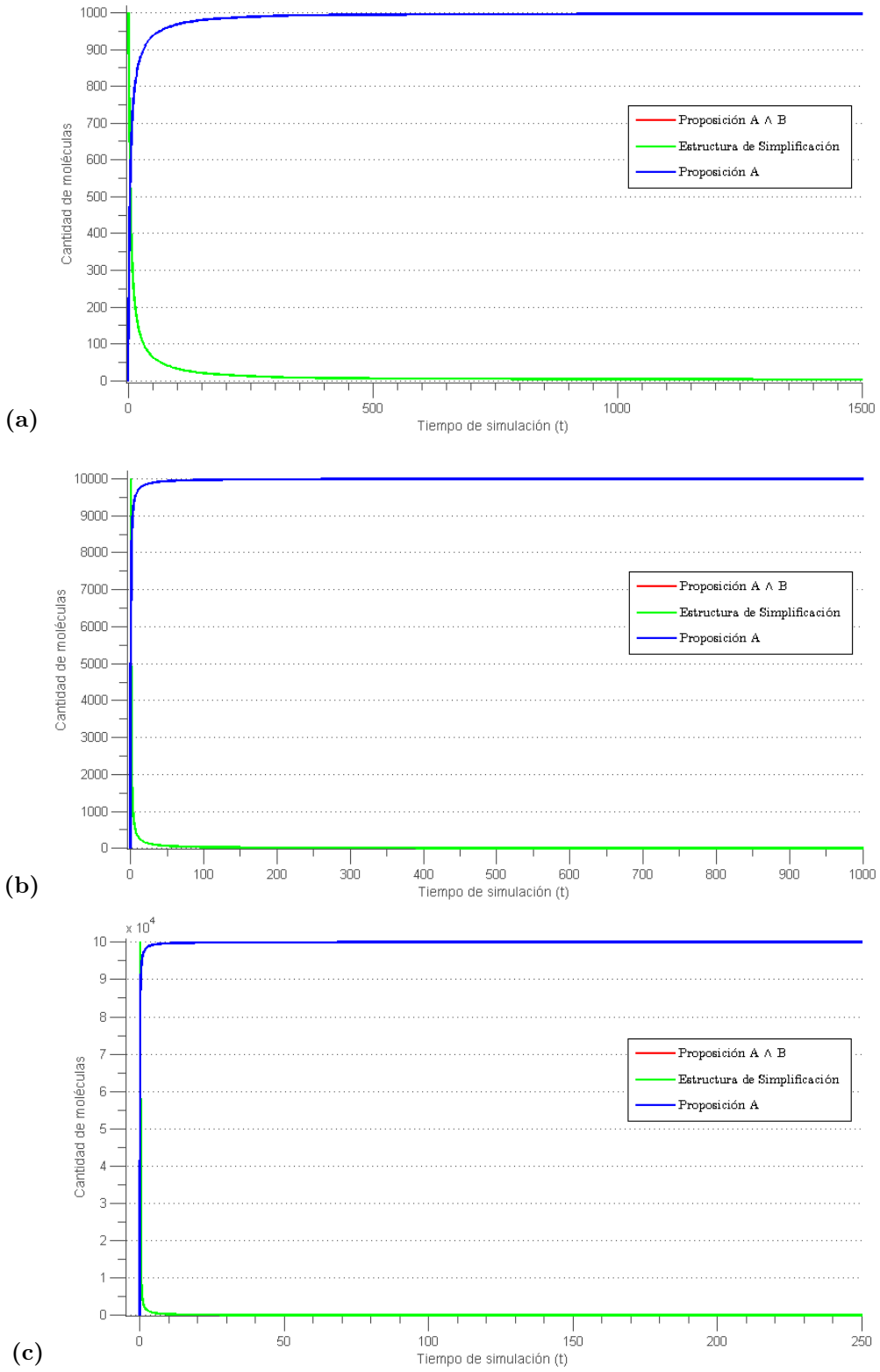


Figura 43: Graficas del promedio de 10 simulaciones de tres casos de la estructura simplificación $A \wedge B \implies B$ realizadas con el simulador estocástico de Visual DSD. (a) Utilizando 1,000 moléculas de las premisa $A \wedge B$ y 1,000 moléculas de la compuerta. (b) Utilizando 10,000 moléculas de la proposición $A \wedge B$ y de la compuerta. (c) Utilizando 100,000 moléculas tanto de la compuerta como de la premisa $A \wedge B$.

6.5.3.3. Análisis de propiedades utilizando la herramienta PRISM

Utilizando la herramienta PRISM se realizaron experimentos para confirmar el correcto funcionamiento de la estructura de simplificación, los cuales consistieron en la verificación de las probabilidades de alcanzar un estado terminal exitoso, finalizar en un estado no deseado, finalizar en cualquier estado terminal, y la probabilidad de encontrarse en el estado inicial después de un cierto periodo de tiempo. Estos experimentos se realizaron utilizando una duración de 50,000 unidades de tiempo. La Figura 44 muestra las probabilidades de todas las posibles finalizaciones del modelo de simplificación en un tiempo T. Nótese que la probabilidad de finalización total y la de finalización exitosa son iguales y tiende a 1. Dado que el modelo sólo cuenta con un estado inicial y un estado terminal (Apéndice , Figura B.01) y ya que el estado terminal cuenta con las propiedades de un estado final exitoso, el sistema siempre finaliza correctamente, por esta razón la gráfica de finalización errónea se mantiene constantemente en 0, mientras que la probabilidad de mantenerse en el estado inicial disminuye al aumentar el tiempo y la probabilidad de finalización exitosa.

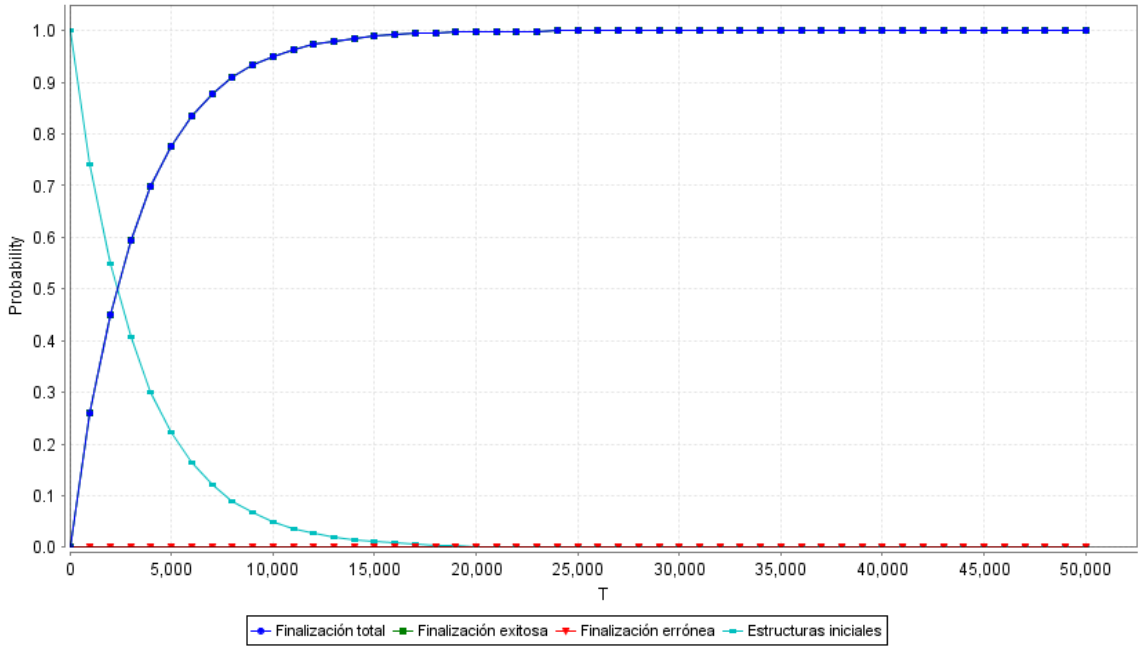


Figura 44: Gráfica de probabilidades de todas las posibles finalizaciones del modelo de simplificación generado con la herramienta PRISM.

6.5.4. Silogismo hipotético

Como se menciona en la Sección 2.6.6, el silogismo hipotético consiste de dos proposiciones condicionales $A \rightarrow B$ y $B \rightarrow C$, donde la conclusión de la primera proposición condicional es la premisa para la segunda, por lo cual, basta con que exista la premisa inicial (A) para producir la conclusión final (C).

6.5.4.1. Diseño de la compuerta

Dado que el silogismo hipotético está compuesto por dos proposiciones condicionales, en su diseño se utilizaron dos estructuras de proposición condicional (definidas en la Sección 6.4.3) unidas en una sola estructura, la cual se muestra en la Figura 29. El extremo izquierdo corresponde a $A \rightarrow B$ y reacciona con la proposición A liberando B como resultado. El extremo derecho equivale a $B \rightarrow C$, recibe B y produce la proposición C. La estructura posee un *toehold* genérico en cada una de las proposiciones condicionales, con un dominio representativo al lado (en la izquierda el complemento de A y en la derecha el de B).

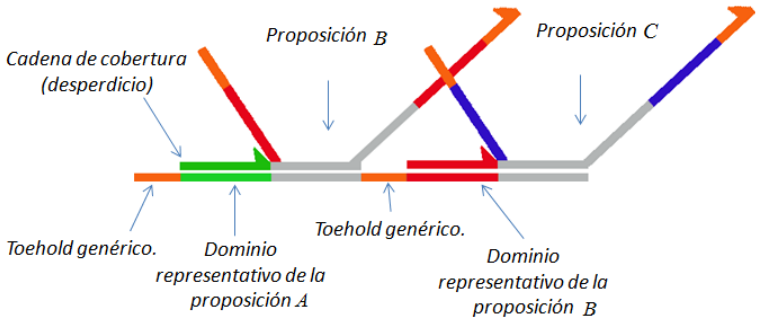


Figura 45: Representación del silogismo hipotético. En el extremo izquierdo se encuentra una zona de reconocimiento, conformada por el complemento del punto de apoyo genérico con el complemento de un dominio representativo de A y se produce B, de acuerdo a la proposición condicional $A \rightarrow B$. En el lado derecho se recibe la proposición B y libera C, de acuerdo a la proposición condicional $B \rightarrow C$.

En la Figura 46 se muestra la resolución del silogismo hipotético. La proposición A se une a la zona de reconocimiento izquierdo a través de su punto de apoyo, dado que su dominio representativo corresponde al complemento de la compuerta, también se adhiere

a él, desplazando a la cadena de cobertura con la que estaba hibridado. Posteriormente, une su primer dominio auxiliar al de la estructura, y con esto libera a la proposición B, que representa el consecuente de la primera proposición condicional ($A \rightarrow B$). Una vez liberada la premisa B, adhiere su punto de apoyo genérico al complemento de la zona de reconocimiento central, ya que su dominio representativo corresponde al de la compuerta, desplaza a su cadena de cobertura y se une a él. Por último, adhiere su dominio auxiliar al de la estructura y libera a la premisa C como conclusión de la inferencia. Observe que cualquier proposición es capaz de unirse a los *toeholds*. Sin embargo, solamente aquellas que cuenten con los dominios representativos correspondientes podrán realizar el desplazamiento de las cadenas de cobertura y producir un resultado. Las cadenas cuyo dominio no correspondan con los de la estructura se desprenderán eventualmente, debido a que el punto de apoyo es un dominio corto que no les brinda la suficiente estabilidad para mantenerlas unidas a la compuerta.

Los diagramas de estados obtenidos por la herramienta Visual DSD se muestran en la Figura B.01 Apéndice . El sistema creado por el silogismo hipotético posee tres estados: el inicial, el terminal y un estado intermedio; y dos transiciones entre ellos.

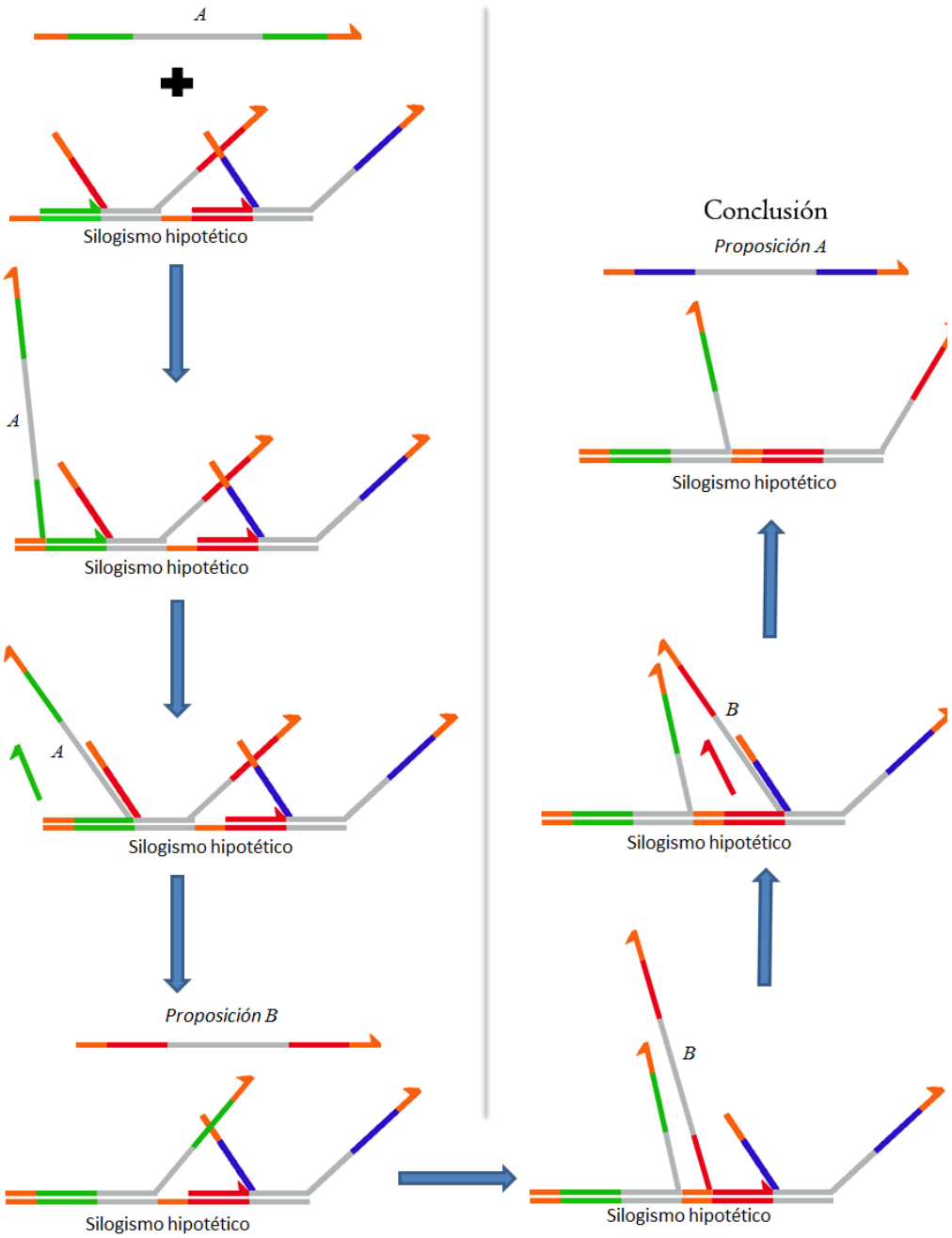


Figura 46: Resolución esquemática del silogismo hipotético. La proposición A une su punto de apoyo al complemento en la izquierda de la estructura, continúa con su dominio complementario y al unir su dominio auxiliar libera a la proposición B. La premisa B adhiere su punto de apoyo genérico al complemento central de la compuerta, continúa con su dominio complementario y al hibridar su dominio auxiliar libera a C como conclusión.

6.5.4.2. Simulación estocástica con Visual DSD.

La simulación de la estructura de silogismo hipotético fue realizada utilizando el simulador estocástico de la herramienta Visual DSD, el cual implementa el algoritmo de Gillespie. Para esto se crearon 9 casos de prueba, cada uno de los cuales es creado utilizando una combinación de 1,000, 10,000 ó 100,000 moléculas de la proposición inicial y de la compuerta.

La Figura 47 muestra las gráficas del promedio de 10 simulaciones de la estructura del silogismo hipotético utilizando el mismo número de moléculas tanto en la compuerta como en la proposición A. En (a) se utilizaron 1,000 moléculas de la proposición A y de la compuerta, una duración de 5,000 unidades de tiempo y 10,000 puntos de muestra. En la gráfica se muestran las primeras 1,500 unidades de tiempo, con el propósito de hacer énfasis a los primeros instantes de la simulación, en los cuales se llevan a cabo la mayor parte de las reacciones. Como se puede observar, la gráfica de la compuerta se consume con una mayor velocidad que las premisas A y B, dado que el simulador la considera consumida una vez que una de las proposiciones se une a ella, y la proposición B puede unirse a una estructura diferente de la que fue liberada. La conclusión C se produce más lentamente, ya que requiere que ambas premisas reaccionen con las estructuras para liberarlo, sin embargo, se acerca a las 1,000 moléculas, y en la gráfica completa, eventualmente se liberan la totalidad de los resultados esperados.

El inciso (b) de la Figura 47, es el resultado de 10 simulaciones realizadas con el caso de prueba de 10,000 moléculas tanto de la compuerta como de la premisa A, los parámetros utilizados incluyen una duración de 10,000 unidades de tiempo y 10,000 puntos de muestra. Observe que la gráfica posee el mismo comportamiento que la del caso anterior, sin embargo, al aumentar la concentración de moléculas el tiempo de simulación disminuye considerablemente y al llegar a las 200 unidades de tiempo ya se han consumido todas las estructuras iniciales y se han producido 10,000 moléculas de la premisa C como conclusión, por lo cual se muestran solamente las primeras 500 unidades de tiempo.

Por último, en (c) se muestra el resultado de 10 simulaciones realizadas con el caso de

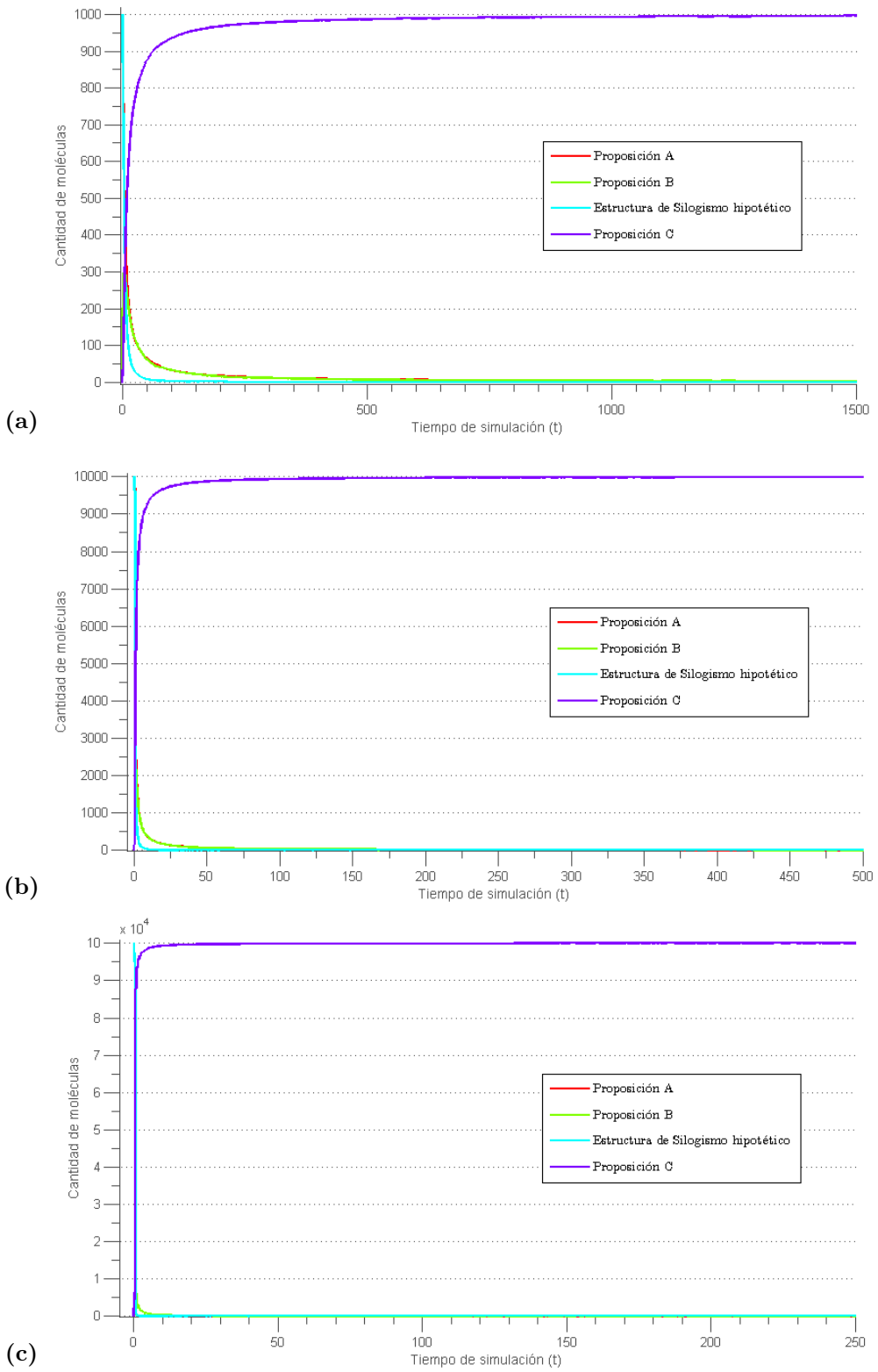


Figura 47: Promedio de 10 simulaciones de la estructura de silogismo hipotético realizadas con el simulador estocástico de Visual DSD. (a) Utilizando 1,000 moléculas de las premisa A y de la compuerta. (b) Utilizando 10,000 moléculas de la proposición A y de la compuerta. (c) Utilizando 100,000 moléculas de la compuerta y de la premisa A.

prueba de 100,000 moléculas tanto de la compuerta como de la proposición A. Los parámetros incluyen una duración de 5,000 unidades de tiempo y 10,000 puntos de muestra. En esta gráfica las estructuras de la compuerta y la proposición A se consumen más rápidamente y alrededor de las 50 unidades de tiempo ya no existen más reacciones, como es de esperarse al aumentar el número de moléculas. Al igual que en las simulaciones anteriores, la gráficas de la compuerta se consume a una mayor velocidad que las de las proposiciones A y B, mientras que la gráfica de la proposición C producida como resultado de la inferencia lógica, aumenta más lentamente, ya que la estructura requiere de ambas premisas para producir una conclusión.

6.5.4.3. Análisis de propiedades utilizando la herramienta PRISM

Para confirmar el correcto funcionamiento del modelo de silogismo hipotético se utilizó la herramienta PRISM, para realizar la verificación de las probabilidades de alcanzar un estado final exitoso, terminar en un estado no deseado, finalizar en cualquier estado terminal, así como la probabilidad de encontrarse en el estado inicial después de un cierto periodo de tiempo. Estos experimentos se realizaron utilizando un duración de 50,000 unidades de tiempo y su resultado se puede observar en la Figura 40, la cual muestra las probabilidades de todas las posibles finalizaciones del modelo de adición en un tiempo T. Al igual que en las estructuras anteriores, este sistema es correcto y por lo tanto siempre finaliza exitosamente, por lo cual, las gráficas de finalización total y finalización exitosas poseen el mismo comportamiento, sin embargo, a diferencia de los anteriores, este modelo posee tres estados y esto se refleja en el comportamiento de dichas probabilidades, que tiende a 1 alrededor de las 25,000 unidades de tiempo.

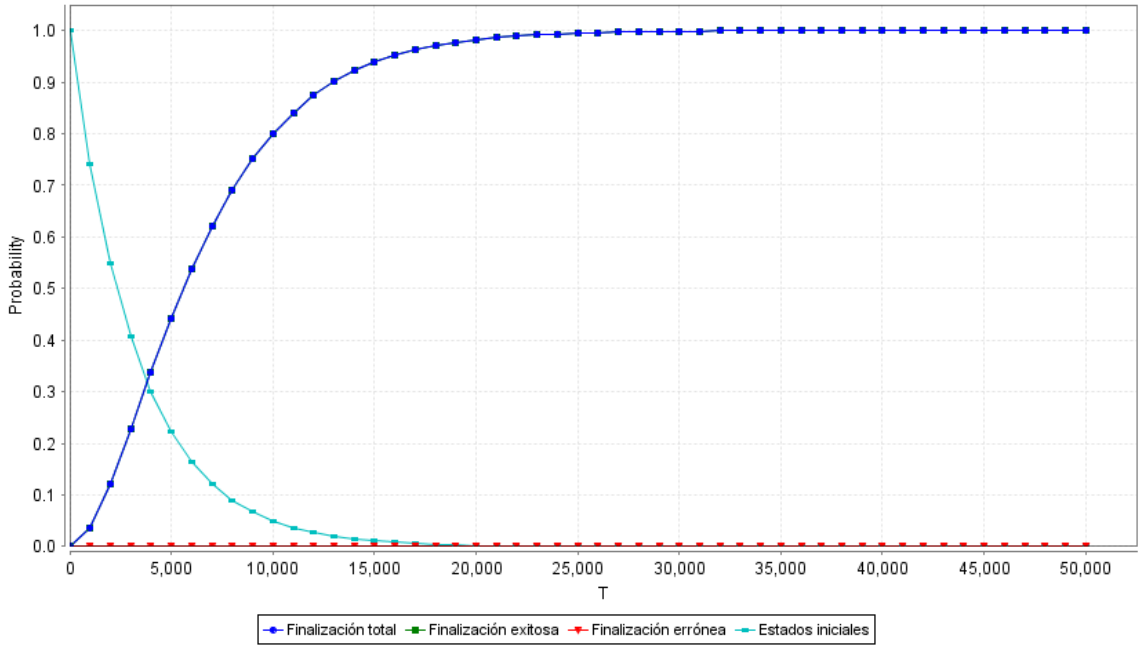


Figura 48: Gráfica de probabilidades de todas las posibles finalizaciones del modelo de silogismo hipotético generado con la herramienta PRISM.

6.5.5. Dilema constructivo

En la Sección 2.6.7 se define el dilema constructivo como una regla de inferencia compuesta por la conjunción de dos proposiciones condicionales que recibe una disyunción de sus antecedentes y produce como conclusión una premisa compuesta por la disyunción de sus consecuentes. Dado que la entrada y el resultado del dilema constructivo son proposiciones compuestas, es necesario representarlas utilizando una proposición temporal (de conjunción en la entrada y de disyunción en la conclusión).

6.5.5.1. Diseño de la compuerta

La estructura diseñada para el dilema constructivo es similar a la utilizada para las proposiciones condicionales, sin embargo, difieren en el tipo de proposiciones utilizadas como premisas y conclusiones. El dilema constructivo posee una zona de reconocimiento en el lado izquierdo, conformada por el complemento del punto de apoyo único que representa a la proposición temporal que será utilizada como entrada, seguida de dos dominios auxiliares unidos a la proposición temporal que será liberada como conclusión. Esta estructura se muestra en

la Figura 49.

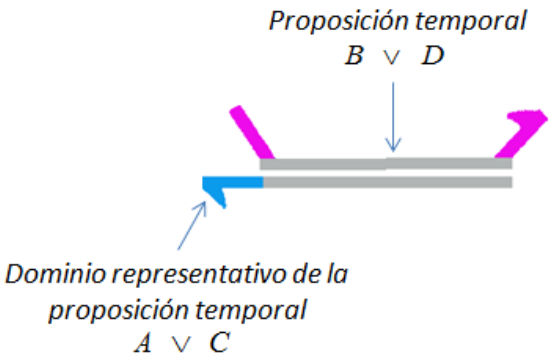


Figura 49: Representación del dilema constructivo. En el extremo izquierdo se encuentra el complemento del *toehold* representativo de la premisa, a la derecha se encuentran dos dominios auxiliares unidos a la proposición temporal que se producirá como resultado.

La Figura 50 muestra la resolución del dilema constructivo $(A \rightarrow B) \wedge (C \rightarrow D) \wedge (A \vee C) \implies B \vee D$. La proposición temporal $A \vee C$ une su punto de apoyo único a la zona de reconocimiento. Posteriormente une su primer dominio auxiliar a la compuerta, continua uniendo su segundo dominio auxiliar y libera a la proposición temporal $B \wedge C$ como conclusión. Dado que el *toehold* utilizado en la estructura es representativo de la premisa, ninguna otra proposición puede reaccionar con él.

Al realizar el análisis de la estructura del dilema constructivo con la herramienta Visual DSD se obtuvo el diagrama de transición de estados mostrado en la Figura B.01 del Apéndice . El sistema cuenta solamente con dos estados: el inicial y el terminal, y una transición entre ellos.

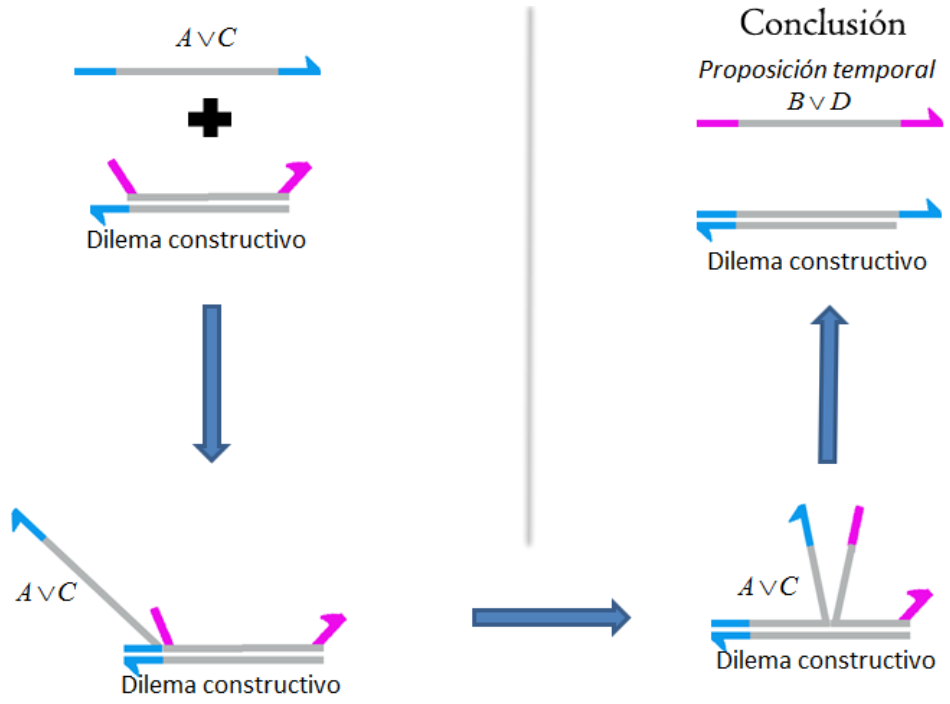


Figura 50: Resolución esquemática del dilema constructivo. La proposición temporal $A \vee C$ une su punto de apoyo representativo a su complemento en la estructura, une su primer dominio auxiliar a la estructura, y al unir su segundo dominio auxiliar libera a la proposición temporal $B \vee D$.

6.5.5.2. Simulación estocástica con Visual DSD

Las simulaciones de la estructura del dilema constructivo se realizaron utilizando el algoritmo de Gillespie incluído en la herramienta Visual DSD de Microsoft. En total se crearon 9 casos de prueba, cada uno de los cuales es una combinación de 1,000, 10,000, ó 100,000 moléculas de concentración de la proposición temporal y la compuerta.

En la Figura 51 se muestran las gráficas generadas por el promedio de 10 simulaciones realizadas utilizando el mismo número de moléculas tanto de la proposición temporal como de la compuerta. Los parámetros incluyen una duración de 5,000 unidades de tiempo y 50,000 puntos de muestra. En la gráfica (a) se utilizaron 1,000 moléculas de cada una de las estructuras iniciales. Al igual que en simulaciones de estructuras anteriores, solamente se muestran las primeras 1,500 unidades de tiempo, ya que la mayoría de las reacciones ocurren en los primeros instantes de la simulación. Como se puede observar, el número de moléculas

de la conclusión aumenta mientras que las estructuras iniciales se consumen, y antes de las 10,000 unidades de tiempo se han producido las 1,000 moléculas esperadas. La gráfica de la compuerta y la proposición temporal $A \vee C$ se consumen mutuamente y poseen el mismo comportamiento, por esta razón, solamente es visible la gráfica de la estructura. En (b) se observa el resultado de utilizar 10,000 moléculas de la premisa inicial y de la compuerta. En este caso todas las estructuras iniciales se consumen alrededor de las 200 unidades de tiempo, dado que este sistema solamente posee una transición entre el estado inicial y final, siempre que se encuentra presente la premisa $A \vee C$ se produce la conclusión $B \vee D$. Además, al aumentar el número de moléculas se disminuye el tiempo necesario para que se lleven a cabo las reacciones. De manera similar a la simulación anterior, la línea de la proposición temporal $A \vee C$ no es visible al consumirse al mismo tiempo que la compuerta, por lo que sus gráficas comparten el mismo comportamiento.

La gráfica (c) muestra el resultado 10 simulaciones realizadas utilizando 100,000 moléculas de la proposición temporal $A \vee C$ y 100,000 moléculas de la compuerta. Como se puede observar, posee el mismo comportamiento que las simulaciones anteriores, sólo que en este caso el tiempo necesario para llevar a cabo las reacciones se ha reducido a 80 unidades de tiempo, a pesar de utilizar un tiempo de simulación de 5,000 unidades. La razón es que al existir un mayor número de moléculas de las estructuras iniciales, aumenta la probabilidad de que reaccionen, produciendo un resultado en un menor tiempo. Al igual que en las simulaciones anteriores, la gráfica de la proposición $A \vee C$ no es visible, ya que se consume al mismo tiempo que la compuerta y por lo tanto sus gráficas son idénticas.

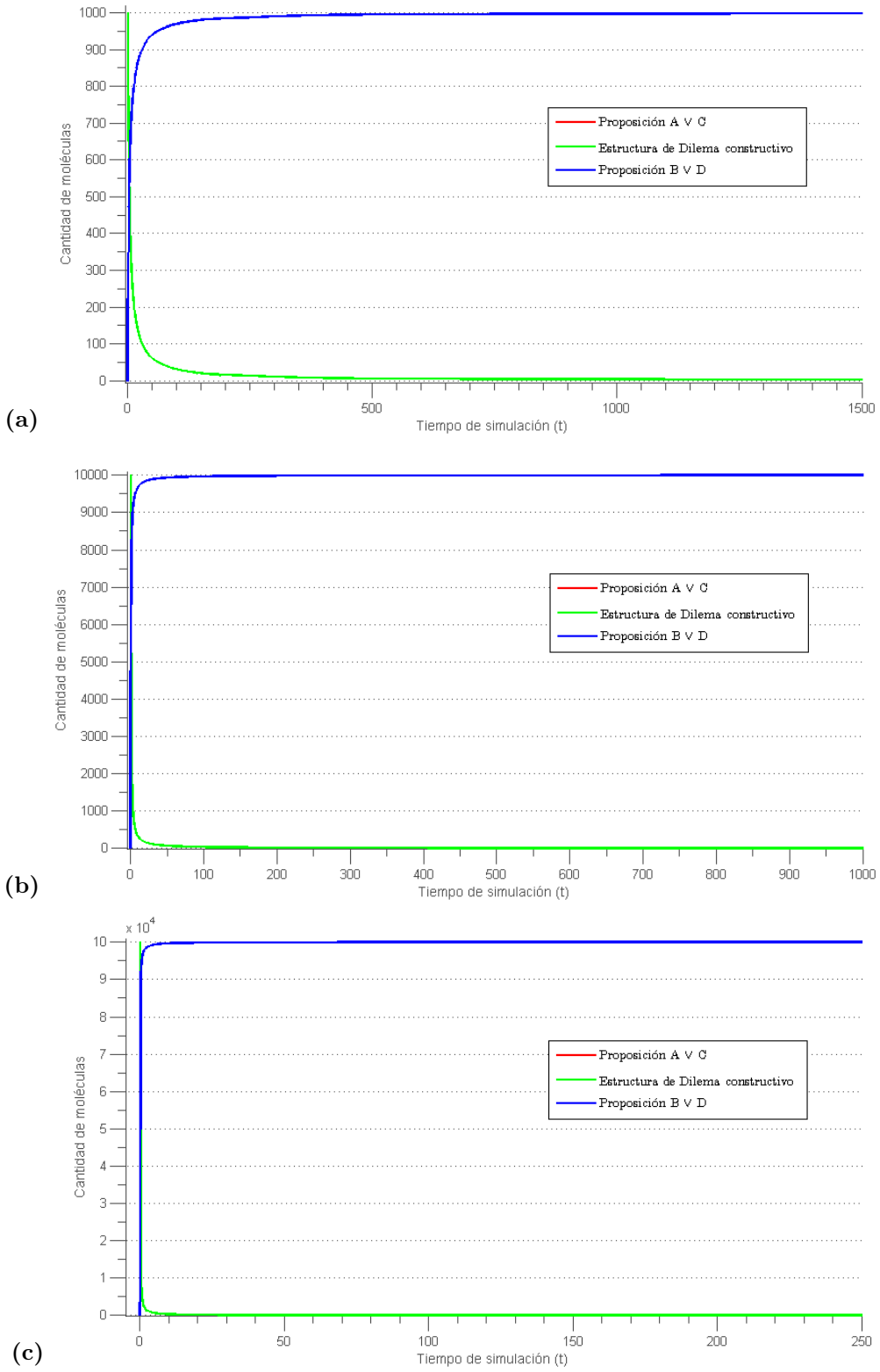


Figura 51: Promedio de 10 simulaciones de la estructura del dilema constructivo realizadas con el simulador estocástico de Visual DSD. (a) Utilizando 0 moléculas de las premisa $A \vee C$ y 100,000 moléculas de la compuerta. (b) Utilizando 100,000 moléculas de la proposición $A \vee C$ y de la compuerta. (c) Utilizando 100,000 moléculas de la compuerta y 10,000 moléculas de la premisa $A \vee C$.

6.5.5.3. Análisis de propiedades utilizando la herramienta PRISM

Para confirmar el correcto funcionamiento del modelo de dilema constructivo se utilizó la herramienta PRISM, con la cual se realizó la verificación de las propiedades cuantitativas (la probabilidad de alcanzar cualquier estado terminal, la probabilidad de finalizar en un estado deseado, la probabilidad de terminar en un estado no deseado y la probabilidad de permanecer en el estado inicial), evaluándolas en un lapso de 50,000 unidades de tiempo. El resultado de dichas evaluaciones se muestra en la Figura 52. Observe que las probabilidades de finalización total y finalización exitosa poseen el mismo comportamiento y a partir de las 19,000 unidades de tiempo tienden a 1, ya que el sistema solamente posee una transición y el estado terminal es correcto, por lo que es imposible que ocurra un error. Por esto, la probabilidad de terminación errónea se mantiene constante en 0.

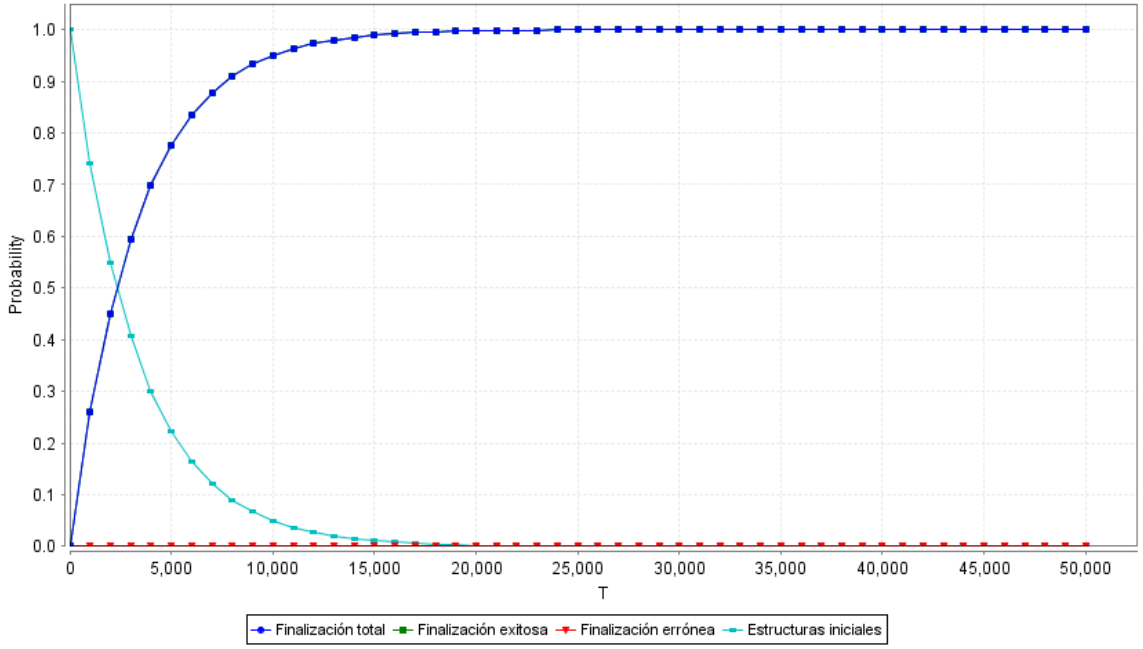


Figura 52: Probabilidad de todas las posibles finalizaciones del modelo de dilema constructivo en un periodo de 50,000 unidades de tiempo, calculado por la herramienta PRISM.

6.5.6. *Modus tollendo ponens*

Como se menciona en la Sección 2.6.5, la regla *modus tollendo ponens*, consiste en dos proposiciones unidas por una disyunción, y la premisa es la negación de una de ellas, por lo

cual se produce la afirmación de la premisa restante como conclusión.

6.5.6.1. Diseño de la compuerta

Para el diseño del *modus tollendo ponens*, se creó una estructura que recibe una proposición negada y una proposición temporal, y produce como resultado una proposición simple. En la Figura 53, se muestra la compuerta diseñada. En el centro se encuentra una zona de reconocimiento con un complemento del *toehold* genérico, rodeado por dos dominios representativos de las proposiciones $\neg B$ y $\neg A$. Posteriormente, se encuentran un dominio auxiliar y una nueva zona de reconocimiento con el complemento del punto de apoyo único representativo de $A \vee B$, ambos unidos a una sola cadena de cobertura. Por último, en cada extremo se encuentran dos complementos de los dominios auxiliares unidos a las proposiciones que serán liberadas como conclusión: en el lado izquierdo la proposición A y en el dado derecho la proposición B.

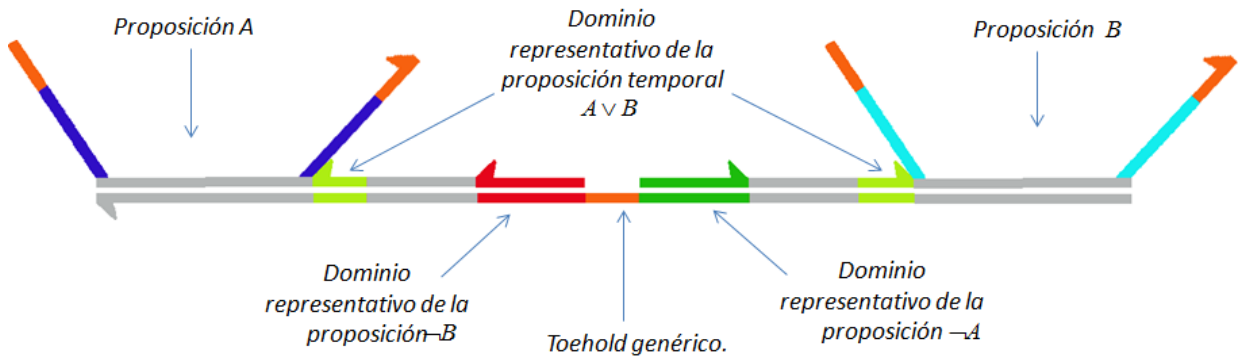


Figura 53: Representación de la regla *Modus tollendo ponens*. En el centro se encuentra una zona de reconocimiento, conformada por el complemento del punto de apoyo genérico con el complemento de un dominio representativo a cada lado, seguido de un dominio auxiliar y un punto de apoyo único con una cadena que cubre a ambos. En el lado izquierdo se recibe la proposición $\neg B$ y $A \vee B$ y se produce A. El lado derecho reacciona con las proposiciones $\neg A$ y $A \vee B$ y produce la proposición B como conclusión.

Las reacciones inician en el punto de apoyo central, ya que los puntos de apoyo únicos han sido cubiertos y requieren de la negación de la proposición para liberarse, esto con el fin de evitar que se produzcan resultados con una sola proposición. En la Figura 54 se muestra la

resolución de la estructura a través de la proposición $\neg A$, la cual se une a la estructura a través de su punto de apoyo central, dado que su dominio representativo corresponde al complemento derecho, también se adhiere a él, desplazando a la cadena de cobertura con la que estaba hibridado. Posteriormente, une su primer dominio auxiliar al de la compuerta, y con esto libera la segunda zona de reconocimiento, a la cual se une la proposición temporal $A \vee B$, y continúa uniendo sus dominios auxiliares a los complementos en la compuerta, liberando a la proposición B en el proceso. Observe que cualquier proposición es capaz de unirse al *toehold* central. Sin embargo, solamente aquella que cuente con el dominio representativo correspondiente podrá realizar el desplazamiento de las cadenas de cobertura, mientras que con el punto de apoyo único solamente pueden reaccionar las proposiciones $A \vee B$. Las cadenas cuyo dominio no corresponda con el de la estructura se desprenderán eventualmente, debido a que el punto de apoyo es un dominio corto que no le brinda la suficiente estabilidad para mantenerla unida a la compuerta.

En la Figura B.01 del Apéndice se muestra el diagrama de transición de estados obtenido por la herramienta Visual DSD para la estructura de la regla *modus tollendo ponens*. A diferencia de los modelos anteriores, éste posee 5 estados: el incial, el final y tres estados intermedios, y 6 transiciones. Observe que el número de transiciones es mayor que el número de estados, ya que dos de los estados intermedios son reversibles, sin embargo, sólo existe un estado terminal en el cual se produce la conclusión esperada.

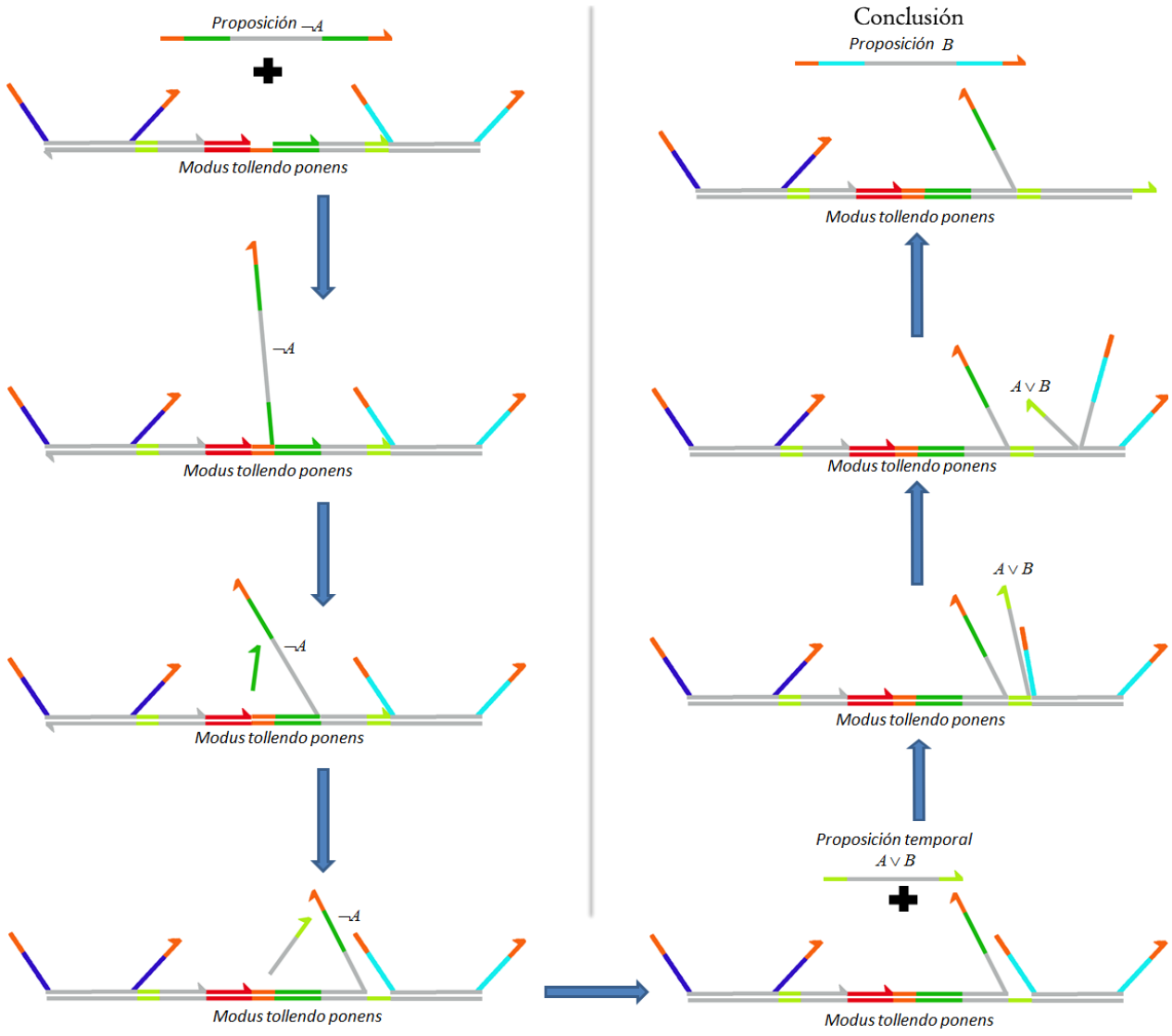


Figura 54: Procedimiento para la resolución de la regla *modus tollendo ponens*. La proposición $\neg A$ une su punto de apoyo al complemento en el centro de la estructura, desplaza la cadena de cobertura del dominio auxiliar y libera al *toehold* único. La proposición temporal $A \vee B$ reacciona con la zona de reconocimiento de la estructura y comienza a unir sus dominios auxiliares, liberando en el proceso a la proposición B como conclusión.

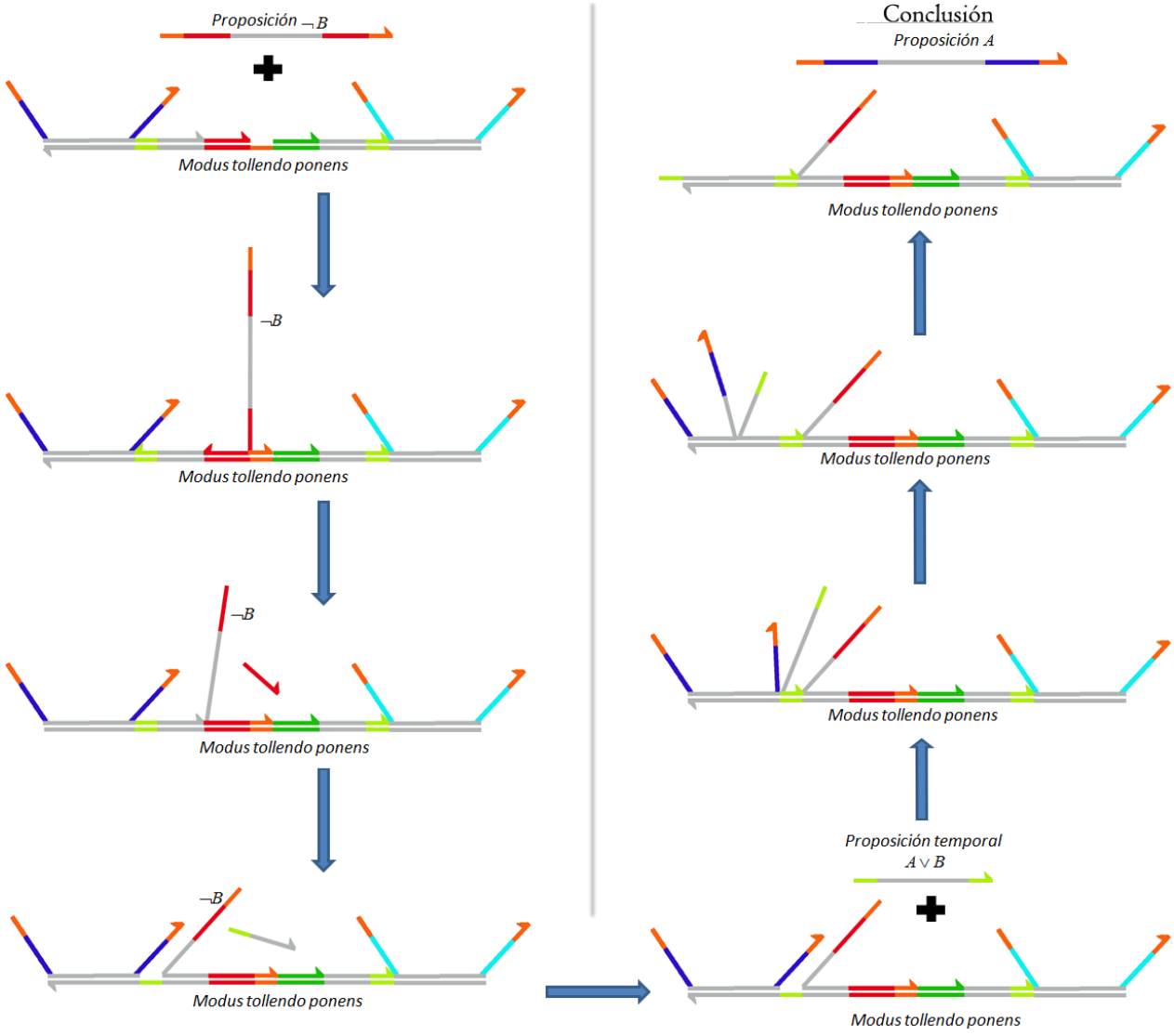


Figura 55: Procedimiento para la resolución de la regla *modus tollendo ponens*. La proposición $\neg B$ reacciona con el complemento de su punto de apoyo en el centro de la estructura, comienza a desplazar la cadena de cobertura del dominio auxiliar y libera al *toehold* único. La proposición temporal $A \vee B$ une su punto de apoyo a la zona de reconocimiento de la estructura y comienza a unir sus dominios auxiliares, liberando en el proceso a la proposición A como conclusión.

6.5.6.2. Simulación estocástica con Visual DSD

Para llevar a cabo las pruebas de la estructura de *modus tollendo ponens* se crearon un total de 27 casos de prueba, utilizando en cada uno de ellos una combinación de 1,000, 10,000 ó 100,000 moléculas de la proposición simple, la proposición temporal y la compuerta. Posteriormente se utilizó el simulador estocástico basado en el algoritmo de Gillespie incluído en la herramienta Visual DSD para realizar las simulaciones de dichos casos de prueba.

En la Figura 56 se muestran 3 gráficas del promedio de 10 simulaciones de los casos de prueba en los cuales se utiliza la misma concentración de moléculas de la proposiciones y de la compuerta. En la gráfica (a) se utilizaron 1,000 moléculas de las estructuras iniciales. Los parámetros incluyen una duración de 15,000 unidades de tiempo y 10,000 puntos de muestra. A diferencia de otras simulaciones, en ésta se muestra la gráfica completa, ya que las reacciones se llevan a cabo más lentamente, dado que el sistema posee un mayor número de estados y algunos de ellos son reversibles. Como se puede observar, el comportamiento de la compuerta y la proposición $\neg A$ es idéntico ya que se consumen mutuamente, mientras que la proposición temporal $A \vee B$ se consume a una menor velocidad ya que para que esto suceda es necesario que primero se libere el punto de apoyo único de la estructura. Sin embargo, aunque sucede a una menor velocidad, alrededor de las 6,000 unidades de tiempo se agotan las estructuras iniciales y se han producido 1,000 moléculas de la proposición B como conclusión de la inferencia lógica.

La gráfica 56(b) corresponde al resultado de utilizar 10,000 moléculas tanto de las premisas $\neg A$ y $A \wedge B$ como de la compuerta de *modus tollendo ponens*, además de una duración de 10,000 unidades de tiempo y 10,000 puntos de muestra. Como es de esperarse al aumentar la concentración de las estructuras iniciales, las reacciones se llevan a cabo a una mayor velocidad, y aunque en esta simulación la reducción del tiempo necesario para agotar las reacciones es menor, todas las estructuras iniciales se consumen alrededor de las 4,000 unidades de tiempo, liberando en el proceso 10,000 moléculas de la conclusión B. Al igual que en la simulación anterior, la gráfica de la proposición temporal $\neg A$ no es visible al consumirse al

mismo tiempo que la compuerta, y la proposición temporal $A \vee B$ se consume más lentamente dado que la reacción se lleva a cabo una vez que el punto de apoyo único de la estructura es liberado por la proposición $\neg A$.

Por último, la gráfica (c) muestra el resultado de 10 simulaciones realizadas utilizando 100,000 moléculas de las proposiciones $\neg A$ y $A \wedge B$ y 100,000 moléculas de la compuerta de *modus tollendo ponens*. Los parámetros incluyen una duración de 10,000 unidades de tiempo y 1,000 puntos de muestra. A pesar de utilizar un mayor número de moléculas, las reacciones se agotan alrededor de las 3,000 unidades de tiempo, liberando 100,000 moléculas de la conclusión B. Observe que aumentado el número de moléculas la gráfica de la compuerta disminuye a una mayor velocidad que en simulaciones anteriores, mientras que la gráfica de la proposición temporal $A \vee B$ presenta el mismo comportamiento, esto ocurre porque los estados intermedios entre ambas reacciones son reversibles. Al igual que en las simulaciones anteriores, las estructuras iniciales tienen el mismo comportamiento al consumirse mutuamente, por lo cual la proposicición $\neg A$ no es visible.

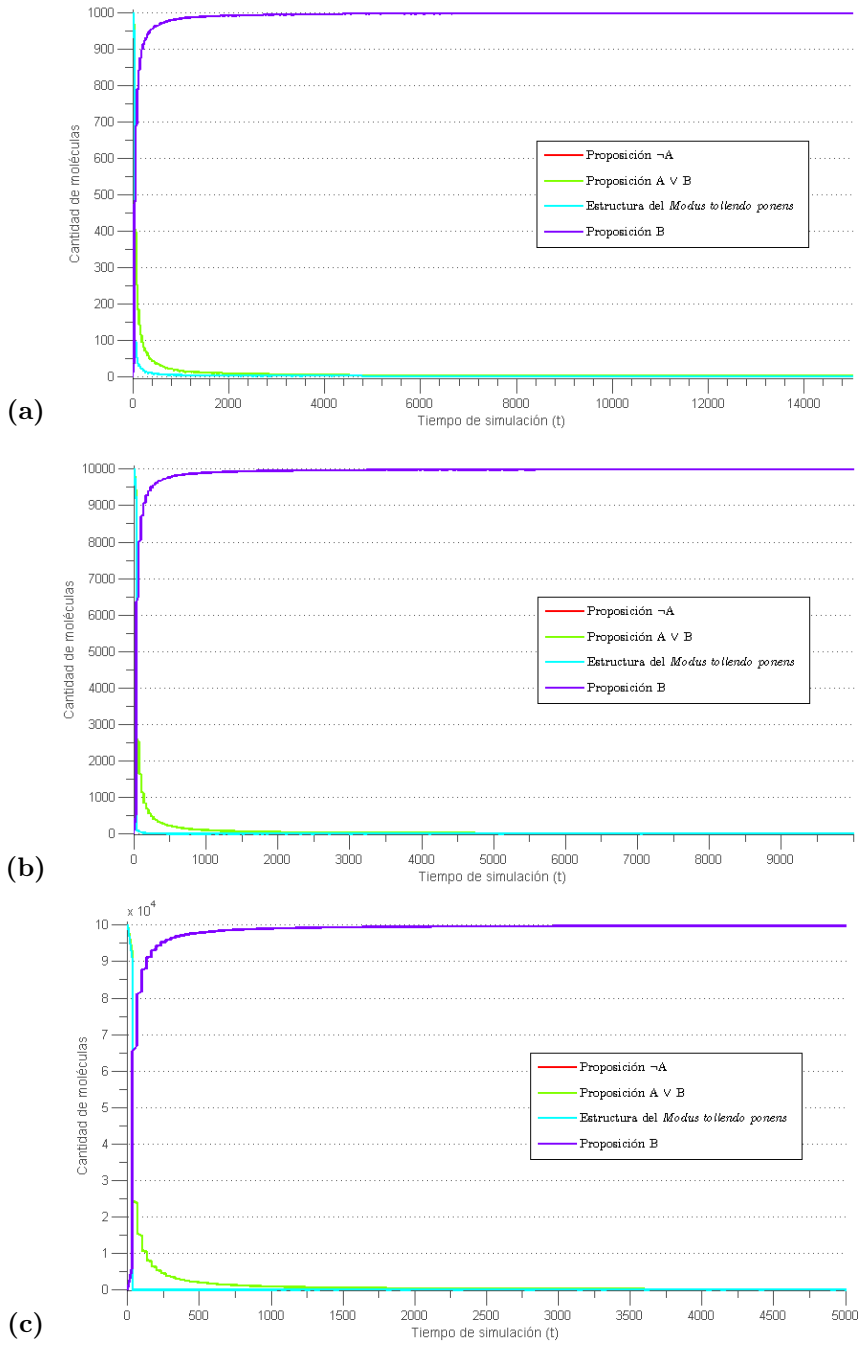


Figura 56: Graficas del promedio de 10 simulaciones de tres casos de la estructura *modus tollendo ponens* realizadas con el simulador estocástico de Visual DSD. (a) Utilizando 1,000 moléculas de las premisas y de la compuerta. (b) Utilizando 10,000 moléculas de las proposiciones y de la compuerta. (c) Utilizando 100,000 moléculas tanto de la compuerta como de la premisas.

6.5.6.3. Análisis de propiedades utilizando la herramienta PRISM

Para confirmar el correcto funcionamiento de la estructura de *modus tollendo ponens* se utilizó la herramienta PRISM, con la cual se realizó la verificación de las probabilidades de alcanzar un estado terminal exitoso, finalizar en un estado no deseado, finalizar en cualquier estado terminal, y la probabilidad de encontrarse en el estado inicial después de un cierto periodo de tiempo. Estos experimentos se realizaron utilizando una duración de 50,000 unidades de tiempo. La Figura 44 muestra las probabilidades de todas las posibles finalizaciones del modelo en un tiempo T. Nótese que la probabilidad de finalización total y la de finalización exitosa son iguales. Dado que el modelo cuenta con 5 estados y dos de ellos son reversibles (Apéndice , Figura B.01), se alcanza la probabilidad 1 hasta las 25,000 unidades de tiempo. Ya que el estado terminal cuenta con las propiedades de un estado final exitoso, el sistema siempre finaliza correctamente, por esta razón la gráfica de finalización errónea se mantiene constantemente en 0, mientras que la probabilidad de mantenerse en el estado inicial disminuye al aumentar el tiempo a una mayor velociad que la probabilidad de finalización exitosa puesto que se abandona el estado inicial en el momento en que la primera proposición reacciona con la estructura, y se alcanza el estado final hasta que ambas lo han hecho.

Anteriormente en la Sección se mencionaba que el modelo de *modus tollendo ponens* es el único que no cuenta con las condiciones de un modelo correcto, esto ocurre porque en la primera reacción se libera la cadena de cobertura del dominio auxiliar y el punto de apoyo único, dicha cadena de cobertura posee las características de una proposición temporal (es una cadena superior con un punto de apoyo único y un dominio auxiliar), por lo cual puede reaccionar con otras estructuras y es responsable de que uno de los estados del modelo sea reversible, ya que puede volver a unirse a su posición inicial y eventualmente es desplazada nuevamente por la proposición $\neg A$. Para realizar los experimentos de la Figura 57 se consideró que dicha cadena es inerte, ya que no evita que se alcance un estado terminal correcto, y el único efecto negativo que posee es el de retardar la producción de la conclusión. Sin embargo,

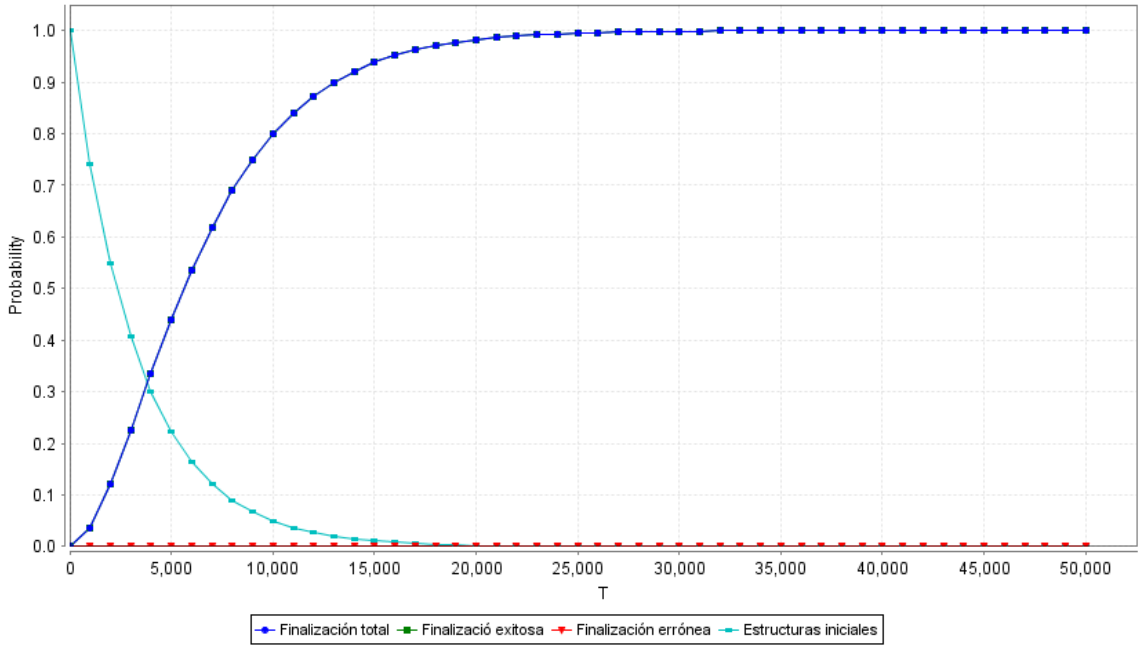


Figura 57: Gráfica de probabilidades de todas las posibles finalizaciones del modelo del *modus tollendo ponens* generado con la herramienta PRISM.

cuando el *modus tollendo ponens* es utilizado con otras estructuras es necesario tener en cuenta este problema, ya que si una de las estructuras utilizadas recibe como entrada a la proposición temporal ($A \vee B$ en el caso del ejemplo), la cadena de cobertura podría iniciar la reacción y producir un resultado impreciso.

Para solucionar el problema anterior se probaron varias soluciones, la primera consiste en intercambiar la posición de las zonas de reconocimiento evitando el uso de la cadena de cobertura con un punto de apoyo y que se produzcan interacciones con otras estructuras, sin embargo, esto produce un efecto contraproducente ya que en dicha estructura la proposición temporal tiene dos posibles reacciones iniciales, cada una de las cuales es irreversible, por lo cual se producen dos estados finales, uno correcto y uno eróneo (donde la proposición temporal reacciona hacia el lado de la proposición $\neg B$). Esta estructura nuevamente es un modelo no correcto y como se observa en la Figura 58 posee sólo una probabilidad de finalización exitosa de 0.5.

Otra posible solución consiste en cambiar las estructuras que utilizan a la proposición temporal ($A \vee B$), de modo que la zona de reconocimiento utilice ambos dominios auxiliares,

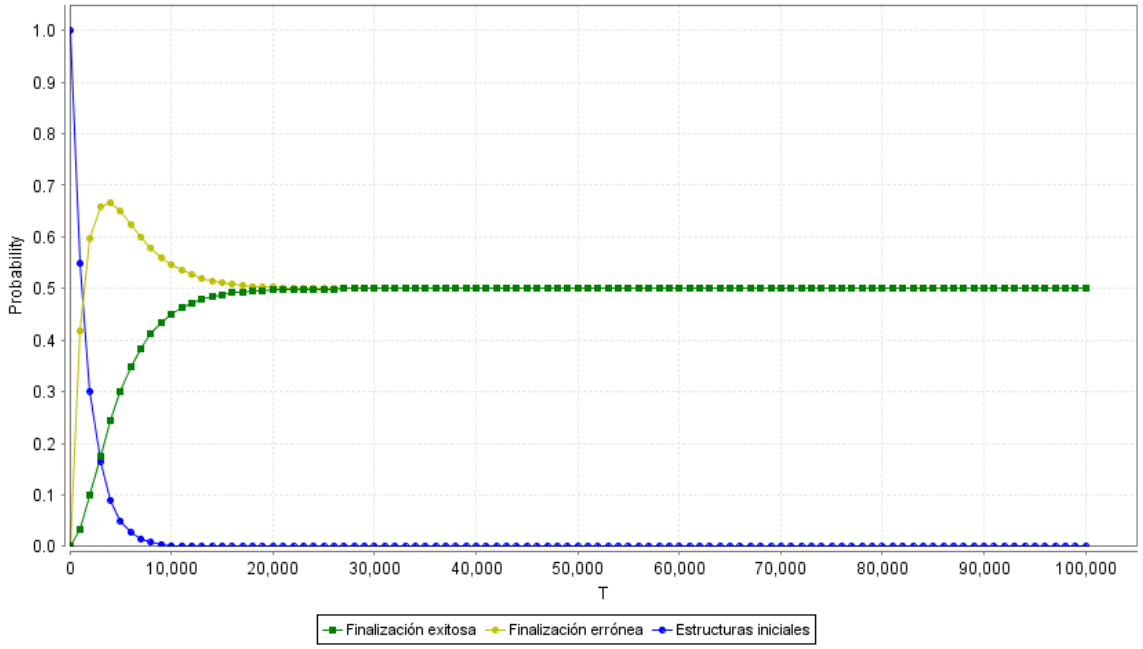


Figura 58: Gráfica de probabilidades de todas las posibles finalizaciones del modelo del *modus tollendo ponens* intercambiando la posición de las zonas de reconocimiento.

esto evita que la cadena de cobertura produzca un resultado pero las compuertas a las que se adhiera la cadena de cobertura dejarán de ser productivas, ya que es un proceso irreversible.

Uno de los principales beneficios de este modelo consiste en la implementación de las reglas de inferencia en cascada. Para demostrar que esto es posible se han tomado dos ejemplos de la literatura y se han modelado con las estructuras mostradas en el Capítulo 6.

7.1. Caso 1. Resolución de un razonamiento válido

El primer ejemplo fue tomado de la publicación de Tom Ran, Ehud Shapiro y Shai Kaplan (2009), en la cual fue utilizado para probar la implementación molecular de programas capaces de realizar deducciones lógicas simples utilizando lazos y enzimas, y consiste en los siguientes enunciados:

'Si Platón es griego, entonces Platón es hombre.

Si Platón es hombre, entonces Platón es mortal.

Platón es griego.

Por lo tanto Platón es mortal.'

Formalizando el problema se extraen las siguientes proposiciones:

G='Platón es griego'.

H='Platón es hombre'.

M='Platón es mortal'.

y se reescribe el ejemplo nuevamente de la siguiente manera:

$G \rightarrow H$

$H \rightarrow M$

G

$\therefore M.$

Este ejemplo es válido ya que es derivable utilizando dos *modus ponens*:

1. $G \rightarrow H$
2. $H \rightarrow M$
3. G

$\therefore M$

4. H (1),(3), *Modus ponens*
5. M (2),(4), *Modus ponens.*

Sin embargo, también es derivable utilizando silogismo hipotético en lugar de dos *modus ponens*, como se muestra a continuación:

1. $G \rightarrow H$
2. $H \rightarrow M$
3. G

$\therefore M$

4. M (1),(2),(3), silogismo hipotético.

7.1.1. Modelación del problema.

Como se menciona en la sección anterior, este ejemplo se puede derivar utilizando dos reglas *modus ponens* o un silogismo hipotético, sin embargo, la resolución a través del silogismo hipotético solamente utiliza una estructura, por lo cual se utilizará solamente la resolución a través dos *modus ponens* para complicar un poco más el ejemplo. La Figura 59 muestra las estructuras utilizadas para la modelación del ejemplo. En el inciso (a) se representa la premisa $G \rightarrow H$ ('Si Platón es griego, entonces Platón es hombre') a través de una estructura de *modus ponens* y *modus tollens*. El inciso (b) muestra el complejo *modus ponens* y *modus tollens* utilizado para la modelación de la premisa $H \rightarrow M$ ('Si Platón es hombre, entonces Platón es mortal'); y por último, el inciso (c) muestra la representación de la premisa G ('Platón es griego') a través de una proposición simple.

La Figura 60 muestra el procedimiento a través del cual se lleva a cabo la resolución de la inferencia lógica. Primero la premisa G une su punto de apoyo a la zona de reconocimiento de la compuerta $G \rightarrow H$ y se desplaza sobre ella hasta liberar a la proposición H. Posteriormente ésta se adhiere a través de su punto de apoyo genérico a la zona de reconocimiento central de la estructura $H \rightarrow M$ y dado que su dominio representativo corresponde al de la estructura, comienza a realizar el desplazamiento de la cadena de cobertura. A continuación une su dominio auxiliar al de la compuerta y con esto libera a la conclusión M, con lo cual se infiere lógicamente que Platón es mortal.

En la Figura C.01 del Apéndice se muestra el diagrama de estados y transiciones del modelo creado para la resolución de este ejemplo, utilizando la herramienta Visual DSD. Como se puede observar, el sistema cuenta con tres estados: el estado inicial, un estado intermedio en el cual se ha llevado a cabo la primera reacción y se ha liberado la premisa H, y un estado final, en el cual se han producido todas las reacciones y se llega a la conclusión.

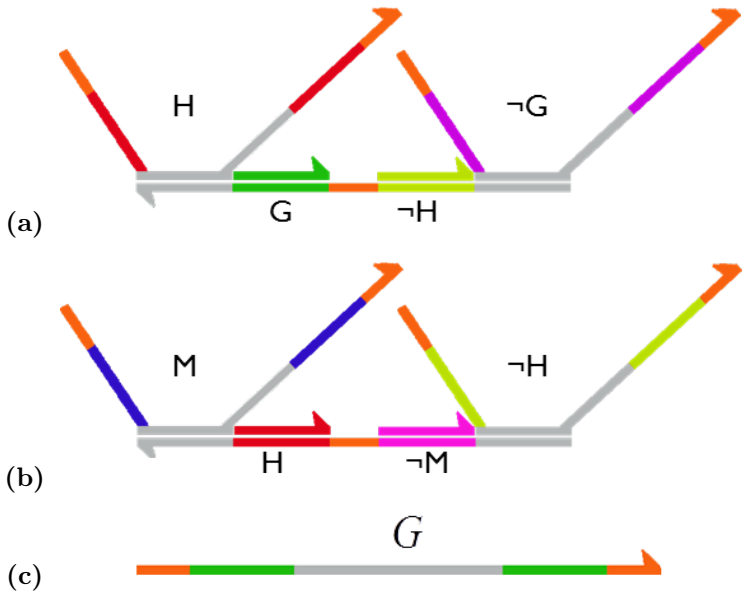


Figura 59: Estructuras utilizadas para la modelación del caso 1. (a) Representación de la premisa 'Si Platón es griego, entonces Platón es hombre' utilizando una estructura de *modus ponens* $G \rightarrow H$. (b) Representación de la premisa 'Si Platón es hombre, entonces Platón es mortal' utilizando una estructura de *modus ponens* $H \rightarrow M$. (c) Representación de la premisa 'Platón es griego' utilizando una proposición simple G.

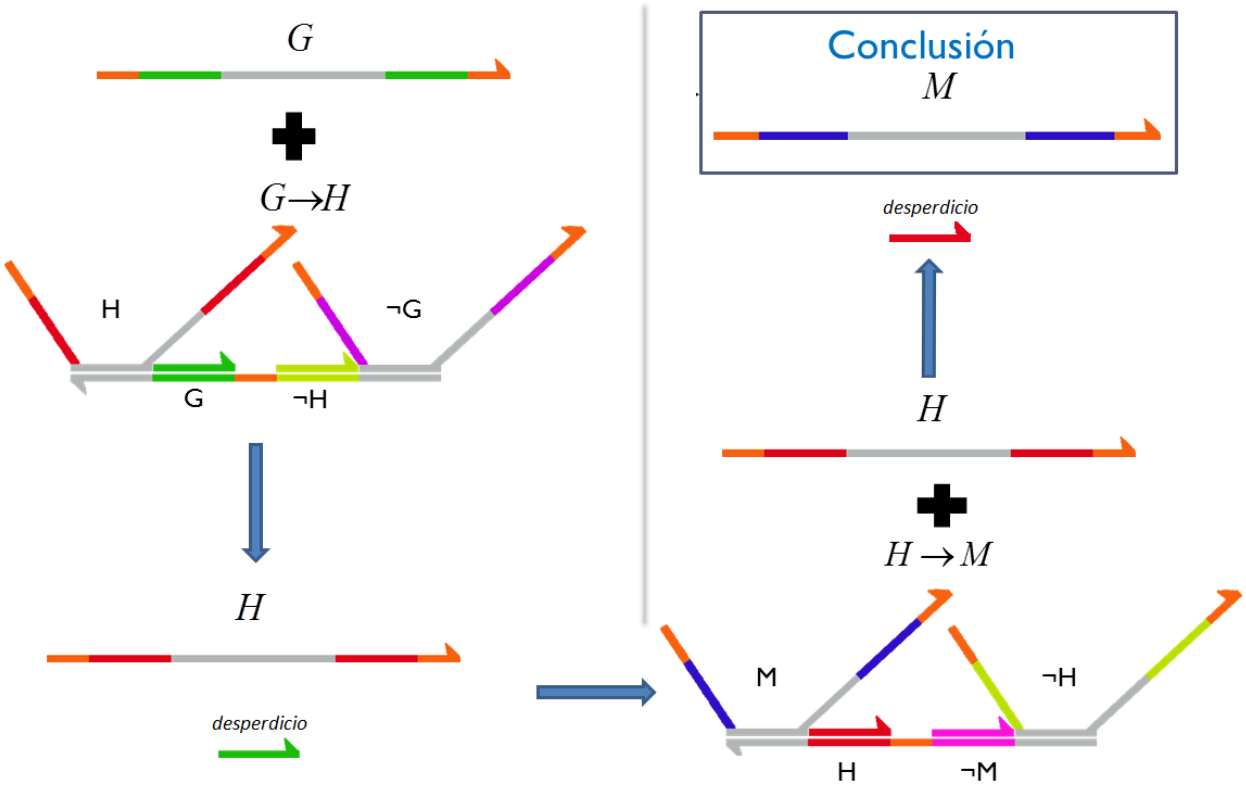


Figura 60: Resolución esquemática del modelo creado para el caso 1. La proposición G reacciona con la estructura $G \rightarrow H$ y libera la proposición H como resultado, ésta a su vez reacciona con el complejo $H \rightarrow M$ y produce la conclusión M.

7.1.2. Simulación estocástica con Visual DSD

La simulación de este modelo fue realizada utilizando el algoritmo de Gillespie incluído en la herramienta Visual DSD, para esto se crearon 27 escenarios, cada uno de los cuales posee una combinación de 1,000, 10,000 ó 100,000 moléculas de la premisa G, y de las compuertas.

Tabla 9: Escenarios de simulación para el caso 1. Número de moléculas empleadas por premisa para cada uno de los escenarios de simulación.

Escenario	Premisa G	Premisa $G \rightarrow H$	Premisa $H \rightarrow M$
1	1,000	1,000	1,000
2	1,000	1,000	10,000
3	1,000	1,000	100,000
4	1,000	10,000	1,000
5	1,000	10,000	10,000
6	1,000	10,000	100,000
7	1,000	100,000	1,000
8	1,000	100,000	10,000
9	1,000	100,000	100,000
10	10,000	1,000	1,000
11	10,000	1,000	10,000
12	10,000	1,000	100,000
13	10,000	10,000	1,000
14	10,000	10,000	10,000
15	10,000	10,000	100,000
16	10,000	100,000	1,000
17	10,000	100,000	10,000
18	10,000	100,000	100,000
19	100,000	1,000	1,000
20	100,000	1,000	10,000
21	100,000	1,000	100,000
22	100,000	10,000	1,000
23	100,000	10,000	10,000
24	100,000	10,000	100,000
25	100,000	100,000	1,000
26	100,000	100,000	10,000
27	100,000	100,000	100,000

La Figura 61 muestra el resultado del promedio de 10 simulaciones del escenario en el cual se utilizaron 100,000 moléculas de estructuras iniciales. Los parámetros incluyen una duración de 1,000 unidades de tiempo y 10,000 puntos de muestra. Como se puede observar, todas las reacciones se llevan a cabo en las primeras 10 unidades de tiempo, produciendo 100,000 moléculas de la conclusión M. La gráfica de proposición G no es visible, ya que al consumirse

con la estructura $G \rightarrow H$ poseen el mismo comportamiento, al igual que la proposición H que se consume con la estructura $H \rightarrow M$ al instante en que es liberada. También se observa que la gráfica de la compuerta $G \rightarrow H$ se consume a una mayor velocidad que la estructura $H \rightarrow M$, ya que ésta última requiere el resultado de la primera para poder consumirse. Se presenta solamente este escenario, ya que en él se utiliza la concentración más alta de las estructuras iniciales, y se espera que su comportamiento sea más cercano al producido en una prueba de laboratorio.

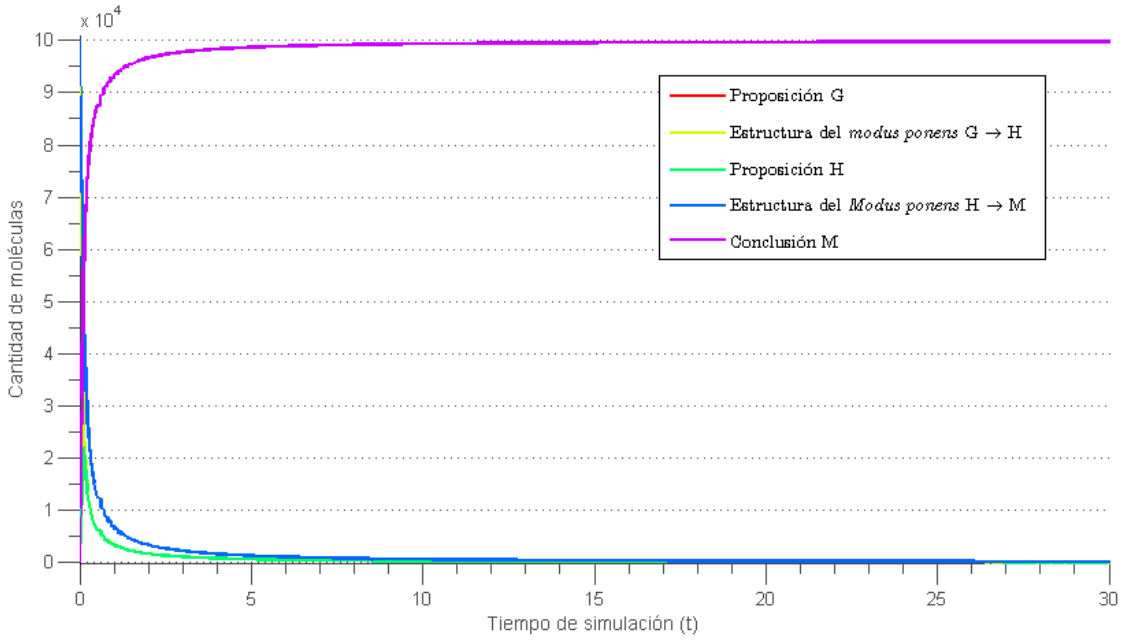


Figura 61: Promedio de 10 simulaciones del modelo creado para el escenario 1, realizadas con el simulador estocástico de Visual DSD, utilizando 100,000 moléculas de las estructuras iniciales.

7.1.3. Análisis de propiedades con la herramienta PRISM

Además de la simulación, en la cual se obtuvieron resultados exitosos, se realizó el analisis del correcto funcionamiento del modelo utilizando la herramienta PRISM para verificar las probabilidades de alcanzar cualquier estado terminal, de finalizar en un estado deseado, de terminar en un estado no deseado y de permanecer en el estado inicial despúes de periodo de tiempo. Estos experimentos se realizaron sobre un lapso de 50,000 unidades de tiempo, y los resultados se muestran en la Figura 62. Notese que la gráfica de finalización exitosa

y finalización total poseen el mismo comportamiento y a partir de las 25,000 unidades de tiempo tienden a 1, esto debido a que el sistema sólo posee un estado final y éste cumple con las condiciones de un estado terminal exitoso. Por esta razón la finalización errónea se mantiene en 0 durante las 50,000 unidades de tiempo. Ésto significa que el modelo es correcto y confiable, ya que siempre que exista la premisa G se producirá el resultado esperado (H).

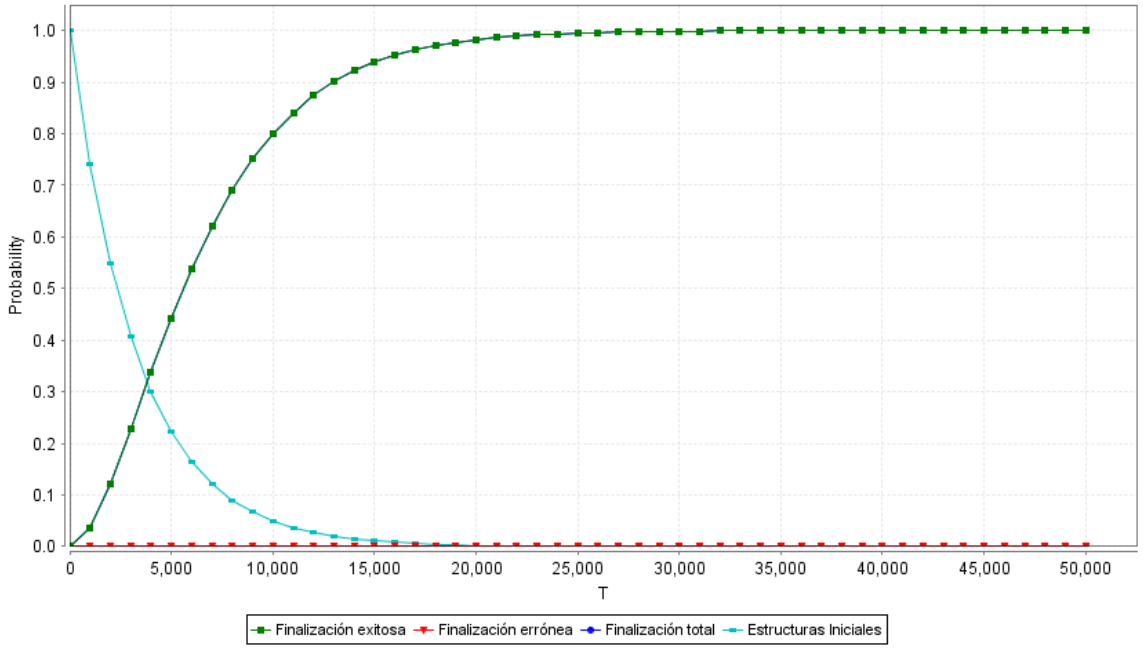


Figura 62: Probabilidad de todas las posibles finalizaciones del modelo creado para el caso 1, calculado por PRISM en un periodo de 50,000 unidades de tiempo.

7.2. Caso 2. Razonamiento inconsistente (contradicción en premisas)

Este ejemplo fue tomado del libro *Proof and fundamentals* (Bloch, 2000), y es utilizado para ilustrar la derivación de sistemas que poseen contradicciones. Como se puede observar, este argumento no posee una conexión aparente entre la conclusión y las premisas. Sin embargo se trata de un argumento válido ya que no existen valores de verdad para las premisas que sean verdaderos y produzcan una conclusión falsa. El ejemplo consiste en el siguiente razonamiento:

‘Juan no toca la guitarra, o Susana toca el violín.

Si Leslie no toca el piano, entonces Susana no toca el violín.

Juan toca la guitarra, y Leslie no toca el piano.

Por lo tanto, Fernando toca el acordeón.'

Formalizando estos enunciados, se extraen las siguientes proposiciones:

J='Juan toca la guitarra'.

S='Susana toca el violín'.

L='Leslie toca el piano'.

F='Fernando toca el acordeón'.

y con esto se problema se puede modelar como:

$\neg J \vee S$

$J \wedge \neg L$

$\neg L \rightarrow \neg S$

$\therefore F$.

Este ejemplo es derivable, sin embargo, la proposición F no existe en ninguna premisa, por lo que debe introducirse utilizando la regla de adición:

1. $\neg J \vee S$
2. $J \wedge \neg L$
3. $\neg L \rightarrow \neg S$

$\therefore F$.

4. J (2), simplificación

5. $J \vee F$ (4), adición

6. $\neg L$ (2), simplificación

7. $\neg S$ (3),(6), *modus ponens*

8. $\neg J$ (1),(7) *modus ponens*

Como se puede observar, a pesar de realizar la derivación del razonamiento y llegar a la conclusión deseada, las premisas 4 y 8 se contradicen, lo cual significa que forman una contradicción. Este tipo de premisas son llamadas *inconsistentes* y no significan que están mal, simplemente no son de ningun uso para fines matemáticos ya que se puede derivar cualquier cosa a partir de ellas, por lo cual pueden producir errores lógicos en argumentos formales e informales. Por lo cual este caso es importante para la revisión del funcionamiento del modelo propuesto, ya que si este no es capaz de detectar la inconsistencia de las premisas, los resultados producidos no son confiables.

7.2.1. Modelación del problema

Este ejemplo se modeló utilizando solamente las estructuras de las tres premisas iniciales: $\neg J \vee S$, $J \wedge \neg L$ y $\neg L \rightarrow \neg S$; y las tres reglas de inferencia con las que pueden interactuar: simplificación de $J \wedge \neg L$, *modus ponens* $\neg L \rightarrow \neg S$ y la regla *modus tollendo ponens* de $\neg J \vee S$. No se utilizó la adición ya que ésta obliga al sistema a producir la salida esperada, y agrega una mayor complejidad al incluír un mayor número de moléculas. Este experimento se realizó con el fin de probar la reacción y la conclusión del sistema, por lo cual se espera la identificación de la contradicción, no la generación de la conclusión F.

En la Figura 63 se muestran las estructuras utilizadas. El inciso (a) corresponde a la disyunción $\neg J \vee S$, la cual es representada con una proposición temporal; (b) muestra la conjunción $J \wedge \neg L$ modelada a través de una proposición temporal; en (c) se observa la compuerta *modus ponens* y *modus tollens* que codifica a la premisa $\neg L \rightarrow \neg S$. Los incisos

(d), (e) y (f) corresponden a las estructuras adicionales con las que pueden reaccionar las premisas anteriores: (d) y (e) son los dos casos de la regla de simplificación para la proposición compuesta $J \wedge \neg L$, en los cuales se producen las conclusiones $\neg L$ y J, respectivamente. En (f) se presenta la estructura *modus tollendo ponens* que puede interactuar con $\neg J \vee S$ para liberar las proposiciones $\neg J$ o S.

La Figura 64 muestra la resolución esquemática del modelo diseñado. Primero la proposición temporal $J \wedge \neg L$ se une a las compuertas de simplificación y libera a las conclusiones J y $\neg L$. Posteriormente la premisa J adhiere su punto de apoyo genérico a la zona de reconocimiento central de la estructura de *modus tollendo ponens*, revelando su punto de apoyo único, con el cual reacciona la proposición temporal $\neg J \vee S$, liberando a la cadena S. Ésta se consume al interactuar con la estructura de *modus ponens* y genera la conclusión L, que se contradice con la premisa $\neg L$ producida previamente por una de las estructuras de simplificación.

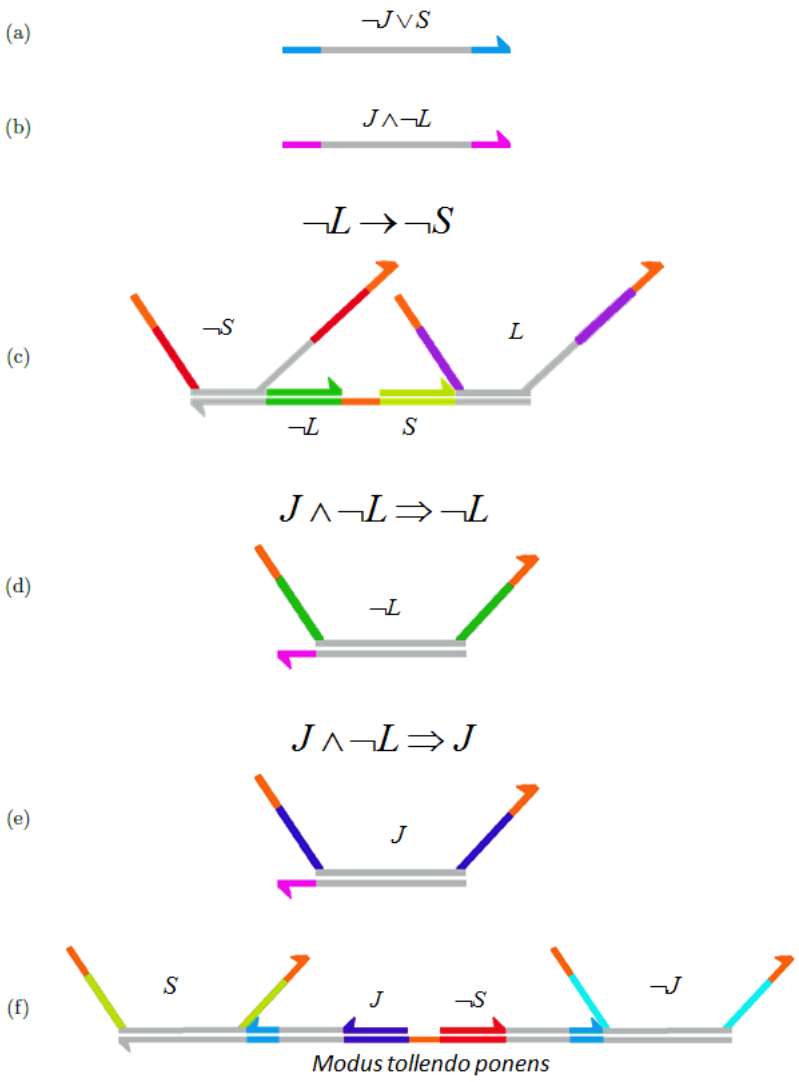


Figura 63: Estructuras diseñadas para el modelo del caso 2. (a) Representación de la premisa $\neg J \vee S$ utilizando una proposición temporal. (b) Codificación de la conjunción $J \wedge \neg L$ con una proposición temporal. (c) Modelación de la premisa $\neg L \rightarrow \neg S$ a través de una estructura de *modus ponens*. (d) y (e) Diseño de los dos casos de la simplificación de $J \wedge \neg L$, donde se obtiene $\neg L$ y J, respectivamente. (f) Codificación del *modus tollendo ponens* de $\neg J \vee S$.

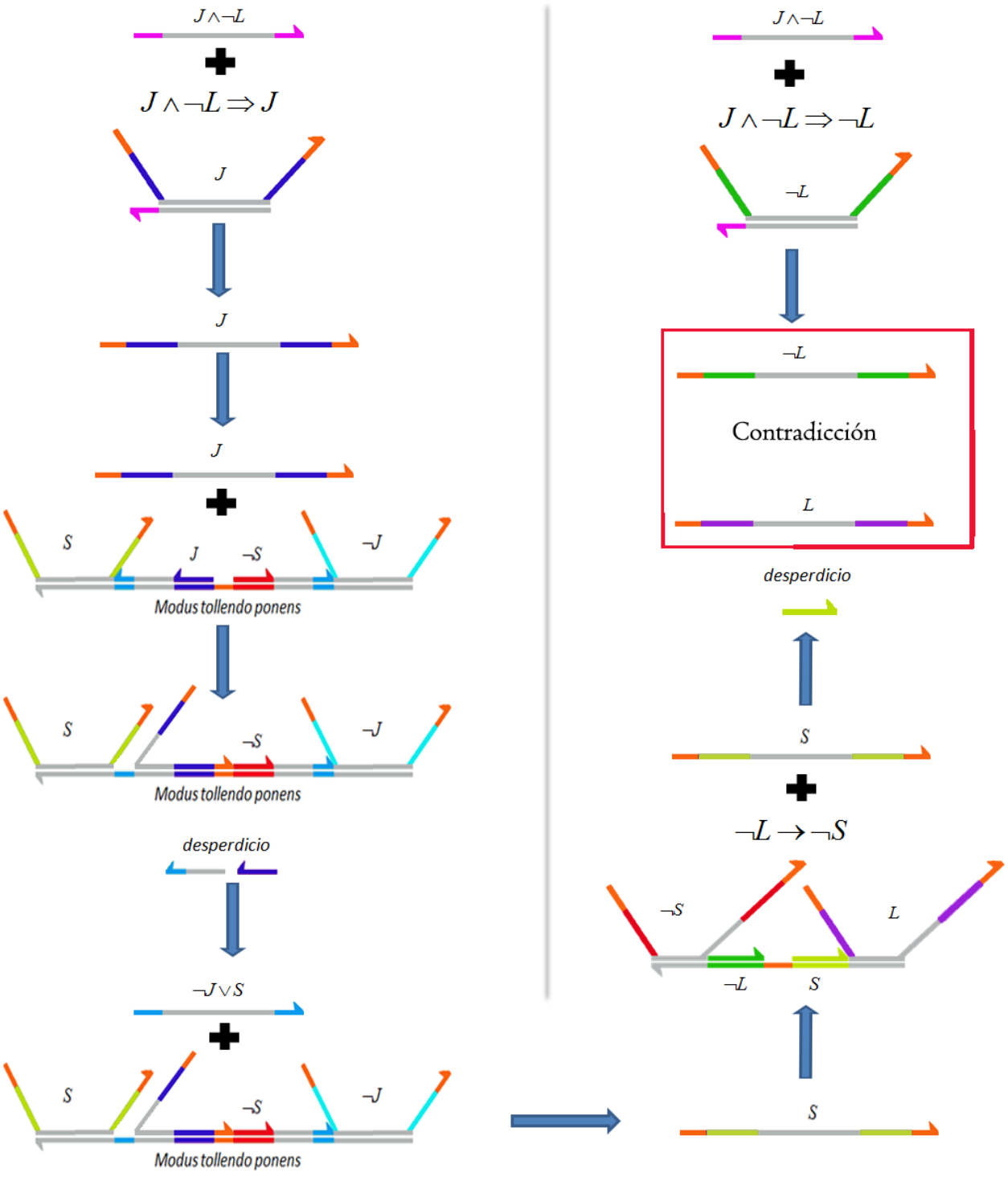


Figura 64: Resolución esquemática del modelo creado para el caso 2. La proposición temporal $J \wedge \neg L$ reacciona con las estructuras de simplificación y libera a las proposiciones J y $\neg L$ como resultado. Las premisas J y $\neg J \vee S$ interactúan con la compuerta de *modus tollendo ponens* y genera la conclusión S, que a su vez consume al *modus ponens* y genera la solución L, con lo cual se contradice el resultado generado anteriormente en la simplificación de $J \wedge \neg L$.

7.2.2. Simulación estocástica con Visual DSD

La simulación de este modelo fue realizada utilizando el algoritmo de Gillespie incluído en la herramienta Visual DSD. La Figura 65 muestra el promedio de 10 simulaciones realizadas utilizando 200,000 moléculas de la proposición temporal $J \wedge \neg L$ y 100,000 moléculas de las estructuras iniciales restantes. Los parámetros incluyen una duración de 1,000 unidades de tiempo y 1,000 puntos de muestra. El inciso (a) presenta la gráfica de las estructuras iniciales. Note que todas las compuertas se consumen en las primeras unidades de tiempo, excepto por la proposición temporal $\neg J \vee S$ que interactúa con la estructura de *modus tollendo ponens* y para ello requiere que las premisas J o $\neg S$ sean producidas y liberen el punto de apoyo único de la compuerta. Las estructuras inician en la mitad de la gráfica porque se utilizaron el doble moléculas de la proposición temporal $J \wedge \neg L$, dado que ésta reacciona con las dos estructuras de simplificación. En el inciso (b) se muestra la gráfica de proposiciones generadas en la simulación. Como se mencionó anteriormente, se esperaba la producción de las proposiciones L y $\neg L$ para demostrar que existe una contradicción, sin embargo, no solamente se generan las proposiciones esperadas, sino que al utilizar un gran número de moléculas se producen también reacciones intermedias que generan a las proposiciones J, $\neg J$, S y $\neg S$, produciendo no una sino tres contradicciones.

En laboratorio, la experimentación se puede llevar a cabo al agregar las moleculas de las estructuras propuestas a un tubo de ensayo y dejando reaccionar el tiempo necesario para llevar a cabo todas las inferencias lógicas posibles. Mientras que el resultado puede ser analizado utilizando la estructura mostrada en 6.3.3 con un fluoróforo de distinto color para cada una de las premisas que pueden ser producidas, así la proposición L puede ser detectada en rojo, $\neg L$ en verde y la conclusión F en amarillo. Otro posible método para la detección de la contradicción consiste en la utilización de la conjunción, es decir, utilizar la conjunción $L \wedge \neg L$ y una estructura de lectura de resultados con un fluoróforo único para la detección de la contradicción de las premisas L y $\neg L$.

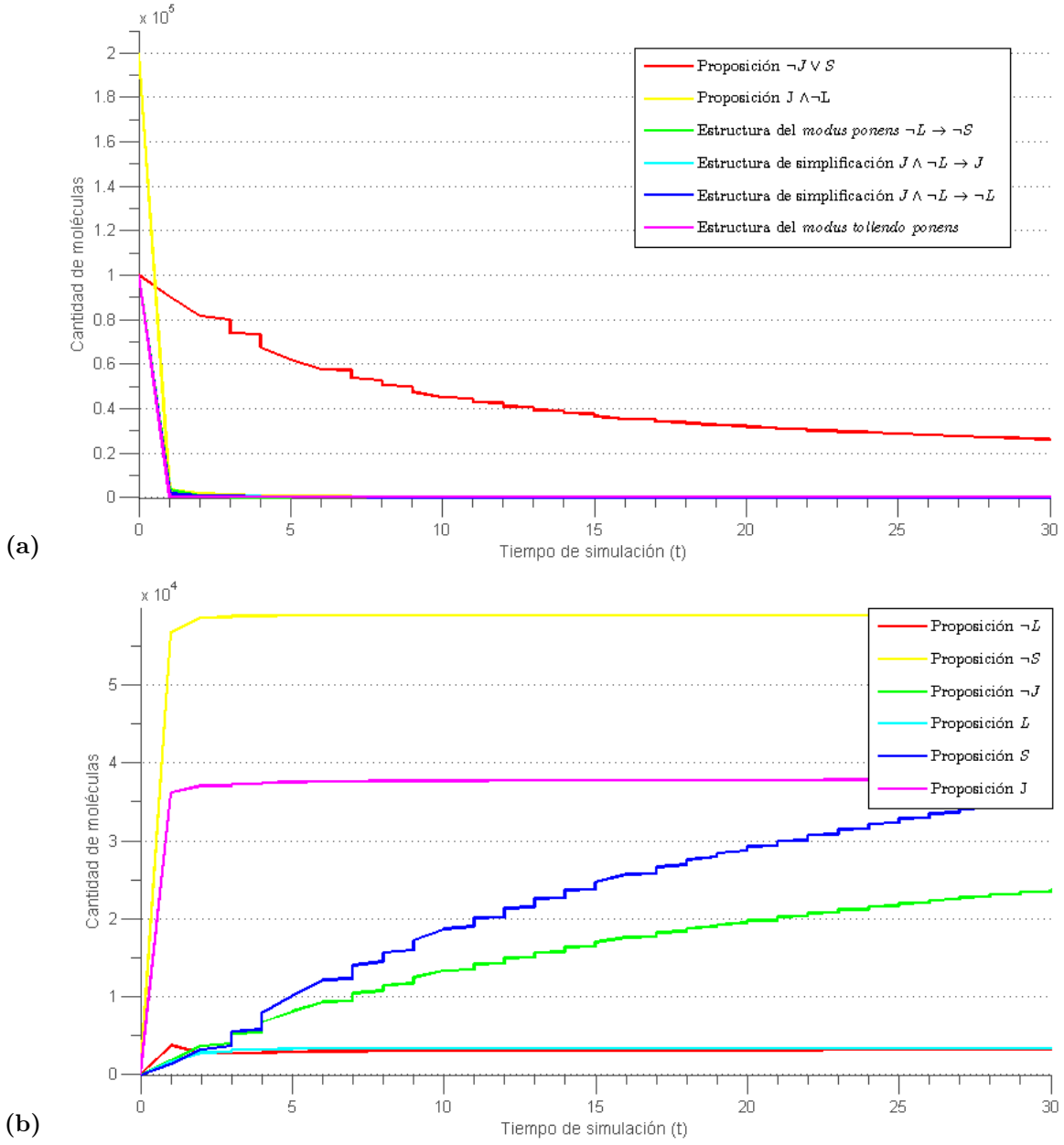


Figura 65: Promedio de 10 simulaciones de las estructuras diseñadas para el ejemplo con contradicción, realizadas con el simulador estocástico de Visual DSD. (a) Gráfica del comportamiento de las estructuras iniciales. (b) Comportamiento de la generación de proposiciones en la simulación, donde se comprueba que existen tres contradiciones al producirse las proposiciones L y $\neg L$, J y $\neg J$ y S y $\neg S$.

7.2.3. Análisis de propiedades con la herramienta PRISM

Además de las simulaciones, se utilizó la herramienta PRISM para analizar el comportamiento del modelo, el cual claramente no es correcto ya que tiene más de un estado terminal, y ninguno de ellos cumple las condiciones de un estado final exitoso. A pesar de esto, se realizó la verificación de sus propiedades con fines ilustrativos. Las propiedades evaluadas consisten en las probabiliades de alcanzar un cualquier estado terminal, la probabilidad de finalizar en cada uno de los estados terminales y la probabiliad de permanecer en el estado inicial después de un periodo de tiempo. Estos experimentos se realizaron con un modelo que cuenta solamente con una molécula de cada estructura inicial (excepto por la conjunción $J \wedge \neg L$, de la cual hay dos moléculas) sobre un lapso de 50,000 unidades de tiempo, los resultados generados se muestran en la Figura 66. Observe que no se utilizaron las evaluaciones de finalización exitosa y errónea, ya que no existen estados finales exitosos y todas las finalizaciones son erróneas. Notese también que la finalización total está dada por la suma de las probabilidades de finalización de los tres estados terminales.

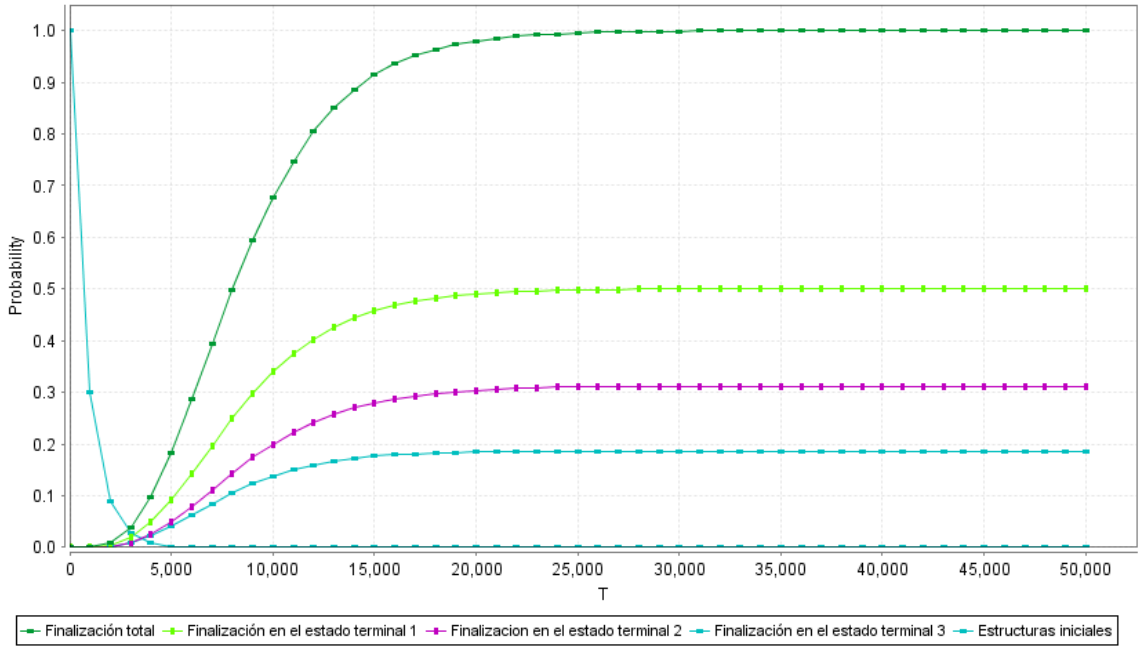


Figura 66: Probabilidad de todas las posibles finalizaciones del modelo con contradicción calculado por PRISM en un periodo de 50,000 unidades de tiempo.

Capítulo 8. Conclusiones

8.1. Sumario

En este trabajo se presenta un modelo de inferencia lógica basado en las propiedades termodinámicas de las moléculas de ADN, utilizando lenguaje DSD para su diseño y las herramientas Visual DSD y PRISM para su posterior simulación y verificación. Las simulaciones se realizaron en casos de prueba creados con base en la relación de la concentración de las entradas y las compuertas, con el fin de demostrar que individualmente las estructuras siempre producen el resultado esperado.

El análisis de propiedades con PRISM permitió mostrar que individualmente los modelos de todas las estructuras son correctos, es decir, siempre finalizan produciendo el resultado esperado. El único diseño que presenta un problema con esto es la regla *modus tollendo ponens*, ya que produce una cadena de cobertura que podría causar problemas si se utiliza con otras estructuras que requieran a la proposición temporal correspondiente a la cadena de cobertura. Sin embargo, se mencionan dos posibles soluciones implementadas. Además de la verificación de corrección de los modelos, se realizó la evaluación de propiedades probabilísticas, con las cuales se probó que todas las estructuras poseen una probabilidad de finalización exitosa igual a 1, lo cual sustenta los resultados obtenidos en las simulaciones.

Una vez realizadas las pruebas inviduales de las estructuras, se utilizaron problemas de razonamiento lógico tomadas de diversas fuentes literarias para demostrar que las compuertas funcionan correctamente en cascada. El primer ejemplo corresponde a un razonamiento válido que utiliza las reglas *Modus ponens* y *modus tollens* para llegar a una conclusión. El análisis de las simulaciones y propiedades prueba que no solo es un modelo correcto que siempre produce el resultado esperado, sino que además estas estructuras pueden coexistir sin producir reacciones inesperadas.

El segundo caso de prueba correspondió a un razonamiento no válido, el cual posee una

contradicción, y con él se esperaba probar que las compuertas además de interactuar correctamente tiene la capacidad de detectar la producción de premisas que se contradicen. La simulación demostró que el sistema no solo produce una contradicción, sino que todas las premisas generadas se contradicen, lo cual es visible en las gráficas de resultados y puede ser detectado en laboratorio utilizando una compuerta de conjunción cuya conclusión sirva como entrada a la estructura de detección de resultados propuesta.

El análisis de propiedades con PRISM confirma el resultado de las simulaciones, al comprobar que el modelo no es correcto al poseer tres estados terminales, por lo que la probabilidad de finalización es dividida en tres posibles terminaciones. Así, se muestra que el modelo propuesto es capaz de realizar inferencias en cascada, utilizando una o más compuertas sin problema, y además es capaz de detectar cuando se produce una contradicción.

8.2. Conclusiones

- Los resultados generados con las herramientas Visual DSD y PRISM demuestran que es posible crear un modelo que permita realizar inferencias lógicas con moléculas de ADN y desplazamiento de cadenas que es correcto y que siempre produce un resultado esperado, ya que todas las estucturas individualmente son correctas, y solamente la regla *modus tollendo ponens* no lo es al ser utilizada con otras estructuras, para lo cual se presentan dos posibles soluciones.

- El modelo propuesto permite implementar la mayoría de las reglas básicas de la lógica proposicional, además de los conectivos lógicos principales, mientras que los modelos existentes solo utilizan la proposición condicional y la regla *modus ponens*.

- Este modelo posee las ventajas de no utilizar enzimas y que no genera estructuras secundarias, lo cual permite que sea evaluado *in silico*, y fue diseñado de esta manera debido a la falta de instalaciones necesarias para realizar los experimentos requeridos para evaluar su desempeño.

- Por último, una de las principales ventajas es la facilidad con que se obtienen los resultados, ya que si bien el modelo de (Ran *et al.*, 2009) posee un método igual de sencillo, el modelo de (Rodríguez-Patón *et al.*, 2011) puede generar estructuras complejas que deben ser analizadas para obtener los resultados correspondientes.

8.3. Posibles aplicaciones

Como se mencionaba anteriormente, la inferencia lógica posee un gran número de aplicaciones en el cómputo biomolecular, como la resolución de razonamientos lógicos para fines educativos, lo cual se ha demostrado en la Sección 7 que es posible realizarse fácilmente utilizando el modelo propuesto. Otra aplicación consiste en la aplicación de las estructuras de conectivos lógicos para diseñar circuitos lógicos báscios, ya que este modelo posee los operadores básicos (conjunción y disyunción), además es posible extenderlo para agregar las compuertas lógicas restantes. Al igual que en los primeros modelos de inferencia lógica molécular, tambiés es posible utilizar estas estructuras para implementar sistemas expertos a nivel molecular, ya que este modelo no solo posee las reglas de inferencia lógicas necesarias, también posee la capacidad de ser utilizado en cascada.

Por último, la aplicación que presenta una mayor importancia e interés es la detección de enfermedades causadas por mutaciones, como cáncer o enfermedades hereditarias, como se muestra en (Kahan-Hanum *et al.*, 2013) es posible seleccionar marcadores genéticos de células enfermas que consisten en ciertos tipos de0 microARNs y oncogenes, los cuales están probados que se relacionan con cáncer de mama, sin embargo, la detección de estos marcadores no implica directamente que la enfermedad se encuentra presente, para esto es necesario utilizar un razonamiento lógico complejo, para lo cual actualmente existe un número de enfoques limitado.

8.4. Trabajo futuro

A pesar de que los resultados de las pruebas realizadas al modelo han sido satisfactorias, aún existe trabajo por hacer, tanto en el diseño de estructuras como en su evaluación. En el diseño, se puede mejorar buscando una solución definitiva al problema del *modus tollendo ponens*; por ejemplo, se podría crear una estructura para agregar la regla de condicional-bicondicional, crear un método para implementar equivalencias lógicas con estas compuertas o incluso utilizar un método distinto para crear una nueva librería de reglas de inferencia lógica con ADN. En lo que respecta a la evaluación de las estructuras, es posible calcular la estabilidad de los complejos propuestos o incluso llevar a cabo pruebas en un laboratorio, ya que para este trabajo no se contó con las instalaciones necesarias para hacerlo.

Esta área del cómputo biomolecular es reciente y existen pocas investigaciones en ella, por lo cual existe un gran número de enfoques que pueden ser utilizados, uno de ellos podría ser el uso de compuertas *seesaw*, las cuales han sido utilizadas previamente para la implementación de circuitos lógicos (Qian y Winfree, 2009, 2011). Otro método considerado inicialmente consiste en el uso de estructuras secundarias que al reaccionar con las premisas, se descomponen en cadenas dobles de ADN y producen una conclusión, sin embargo éste posee la desventaja de no poder ser simulado utilizando la herramienta Visual DSD, por lo cual fue descartado.

Referencias bibliográficas

Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, **266**(5187): 1021–4.

Amos, M. (2005). *Theoretical and Experimental DNA Computation*. Natural Computing Series. Springer. p. 172.

Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., y Shapiro, E. (2001). Programmable and autonomous computing machine made of biomolecules. *Nature*, **414**(6862): 430–4.

Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., y Shapiro, E. (2004). An autonomous molecular computer for logical control of gene expression. *Nature*, **429**(6990): 423–9.

Bloch, E. D. (2000). *Proofs and Fundamentals: A First Course in Abstract Mathematics*. Birkhauser. p. 424.

Cardelli, L. (2009). Strand algebras for dna computing. En: *DNA Computing and Molecular Programming*, Lecture Notes in Computer Science, pp. 12–24. Springer.

Cardelli, L., Kwiatkowska, M., Lakin, M. R., Parker, D., y Phillips, A. (2012). Design and analysis of DNA strand displacement devices using probabilistic model checking. *Journal of The Royal Society Interface*, **9**: 1470–1485.

Fehlberg, E. (1969). Low-order classical runge-kutta formulas with stepsize control and their application to some heat transfer problem. Technical Report R-315, NASA. Recuperado de http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19690021375.pdf.

Feynman, R. P. (1960). There's Plenty of Room at the Bottom. *Caltech Engineering and Science*, **23**(5): 22–36.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, **81**(25): 2340–2361.

Hagiya, M., Arita, M., Kiga, D., y Sakamoto, K. (1997). Towards Parallel Evaluation and Learning of Boolean μ-Formulas with Molecules. *Proceedings of Third Annual Meeting on DNA Based Computers.*, **48**: 105–114.

Hartmanis, J. (1995). On the Weight of Computations. *Bulletin of the EATCS*, **55**: 136–138.

Head, T. (1987). Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, **47**(6): 737–759.

Ignatova, Z., Martínez-Pérez, I., y Zimmermann, K. H. (2008). *DNA Computing Models*. Springer. p. 288.

Kahan-Hanum, M., Douek, Y., Adar, R., y Shapiro, E. (2013). A library of programmable dnazymes that operate in a cellular environment. *Scientific reports*, **3**: 1–6.

Kuramochi, J. y Sakakibara, Y. (2006). Intensive in vitro experiments of implementing and executing finite automata in test tube. En: *DNA Computing*, Lecture Notes in Computer Science, pp. 193–202. Springer.

Kwiatkowska, M., Norman, G., y Parker, D. (2011). Prism 4.0: verification of probabilistic real-time systems. En: *Proceedings of the 23rd international conference on Computer aided verification*, CAV'11, pp. 585–591. Springer.

Labean, T. H., Shetty, G., Yan, H., Schules, E. A., Chandran, H., y Reif, J. H. (2009). Target DNA detection by strand displacement and Deoxyribozymogen amplification. En: *Fundations of Nanoscience*, pp. 82–83.

Lee, I. H., Park, J. Y., Jang, H. M., Chai, Y. G., y Zhang, B. T. (2003). Dna implementation of theorem proving with resolution refutation in propositional logic. En: *DNA Computing*, Lecture Notes in Computer Science, pp. 156–167. Springer.

Lipton, R. J. (1995). DNA Solution of Hard Computational Problems. *Science*, **268**(5210): 542–545.

Martínez-Pérez, I. M., Zhang, G., Ignatova, Z., y Zimmermann, K. H. (2007). Computational genes: a tool for molecular diagnosis and therapy of aberrant mutational phenotype. *BMC bioinformatics*, **8**(365): 1–9.

Martínez-Pérez, I. M., Zimmermann, K. H., y Ignatova, Z. (2009). An autonomous DNA model for finite state automata. *International Journal of Bioinformatics Research and Applications*, **5**(1): 81–96.

Mulawka, J., Weglenski, P., y Borsuk, P. (1998). Implementation of the inference engine based on molecular computing technique. En: *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence*, pp. 493–498.

Paulevé, L., Youssef, S., Lakin, M. R., y Phillips, A. (2010). A generic abstract machine for stochastic process calculi. En: *Computational Methods in Systems Biology*, pp. 43–54. ACM.

Phillips, A. y Cardelli, L. (2009). A programming language for composable dna circuits. *Journal of The Royal Society Interface*, **6**(4): S419–S436.

Qian, L. y Winfree, E. (2009). A simple dna gate motif for synthesizing large-scale circuits. En: *DNA Computing*, Lecture Notes in Computer Science, pp. 70–89. Springer.

Qian, L. y Winfree, E. (2011). Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades. *Science*, **332**(6034): 1196–1201.

Ran, T., Kaplan, S., y Shapiro, E. (2009). Molecular implementation of simple logic programs. *Nature Nanotechnology*, **4**(10): 642–648.

Rodríguez-Patón, A., Larrea, J., y Sainz de Murieta, I. (2010). Inference with dna molecules. En: *Unconventional Computation*, Lectures Notes in Computer Sciene, pp. 192–192. Springer.

Rodríguez-Patón, A., Sainz de Murieta, I., y Sosík, P. (2011). Autonomous resolution based on dna strand displacement. En: *DNA Computing and Molecular Programming*, Lecture Notes in Computer Science, pp. 190–203. Springer.

Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N. V., Goodman, M. F., Rothemund, P. W. K., y M., A. L. (1996). A sticker based model for dna computation. *Journal of Computational Biology*, **5**(4): 415–440.

Sainz de Murieta, I., Miro-Bueno, J., y Rodríguez-Patón, A. (2011). Biomolecular computers. *Current Bioinformatics*, **6**(2): 173–184.

SantaLucia, J. y Hicks, D. (2004). The thermodynamics of DNA structural motifs. *Annual Review of Biophysics and Biomolecular Structure*, **33**: 415–440.

Seeling, G., Soloveichik, D., Zhang, D. Y., y Winfree, E. (2006). Enzyme-free nucleic acid logic circuits. *Science*, **314**: 1585–1588.

Wasiewicz, P., Janczak, T., y Mulawka, J. J. (1999). The inference via dna computing. En: *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 988–993.

Wasiewicz, P., Janczak, T., Mulawka, J. j., y Plucienniczak, A. (2000). The inference based on molecular computing. *Cybernetics and systems: an international journal.*, **31**(3): 283–315.

Wilhelm, P. y Rothemund, K. (1996). A DNA and restriction enzyme implementation of Turing Machines. En: *DIMACS Workshop DNA Based Computers*, pp. 75–119.

Winfree, E. (1998). Whiplash PCR for O(1) Computing. Reporte técnico, California Institute of Technology. Recuperado de http://www.dna.caltech.edu/Papers/.

Winfree, E., Yang, X., y Seeman, N. C. (1996). Universal computation via self-assembly of dna: Some theory and experiments. En: *DNA Based Computers II, volume 44 of DIMACS*, pp. 191–213. American Mathematical Society.

Yurke, B., Turber, A. J., Mills, A. P., Simmel, F. C., y Neumann, J. L. (2000). A DNA-fuelled molecular machine made of DNA. *Letters to Nature*, **406**: 605–608.

Apéndice A

Estructura del conectivo de conjunción lógica

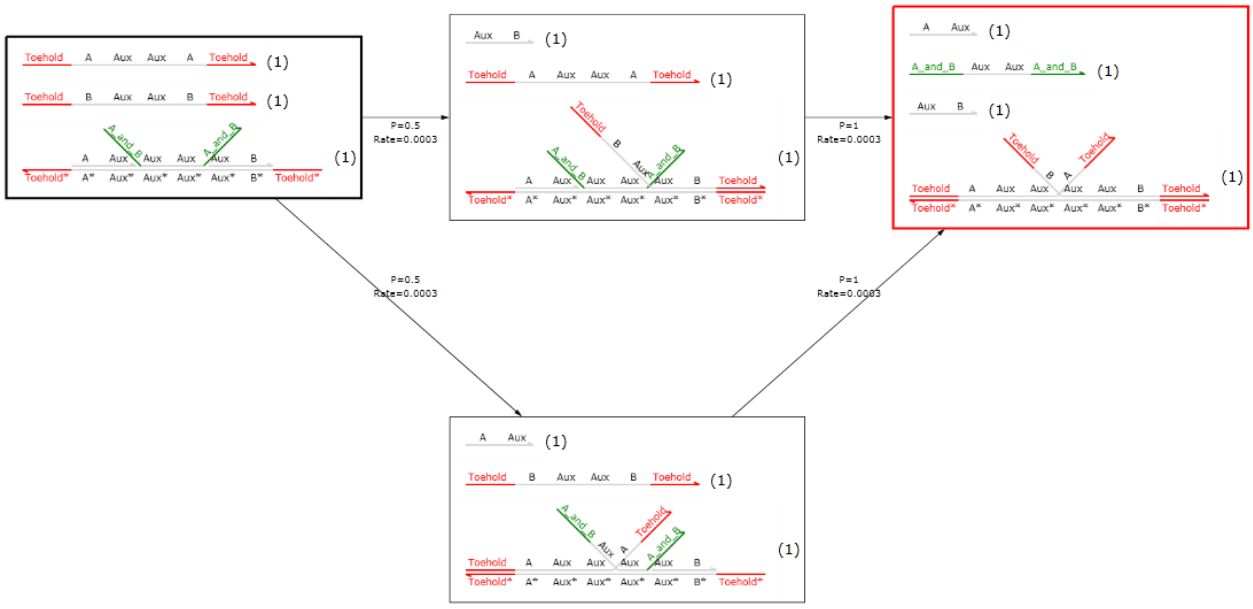


Figura A.01: Diagrama de transición de los estados de la conjunción $A \wedge B$ obtenido utilizando Visual DSD. El recuadro marcado en negro simboliza estado inicial, el recuadro en rojo representa el estado final de la conjunción.

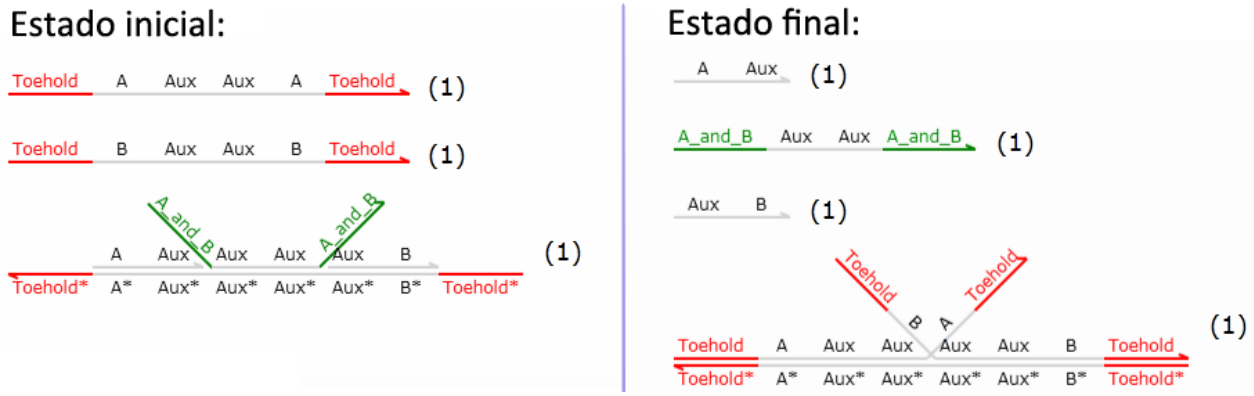


Figura A.02: Análisis utilizando Visual DSD en la compuerta de la conjunción $A \wedge B$. En el lado izquierdo se muestra el estado inicial del sistema conformado por la estructura de conjunción y las proposiciones A y B. A la derecha se muestra el estado final, una vez realizado el desplazamiento de cadenas se obtienen la compuerta inerte, la proposición temporal $A \wedge B$ y las dos cadenas de cobertura como desperdicio.

Tabla A.01: Estructuras utilizadas en la simulación de la conjunción lógica

Premisa/Regla de inferencia	Estructura
A	<Toehold^A Aux Aux A Toehold^>
B	<Toehold^B Aux Aux B Toehold^>
$A \wedge B$	<A_and_B^Aux Aux A_and_B^>
Conjunción	{Toehold^*}[A Aux]:<A_and_B^>[Aux Aux]<A_and_B^>:[Aux B]{Toehold^*}

Tabla A.02: Secuencias utilizadas para la codificación de los dominios usados para la simulación de la conjunción lógica

Dominio	Secuencias
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
B	(5') CCCTTTACATTACATAACAA (3')
A	(5') CCCAAAACAAAACAAAACAA (3')
A_or_B^	(5') GCTA (3')

Estructura del conectivo de disyunción lógica

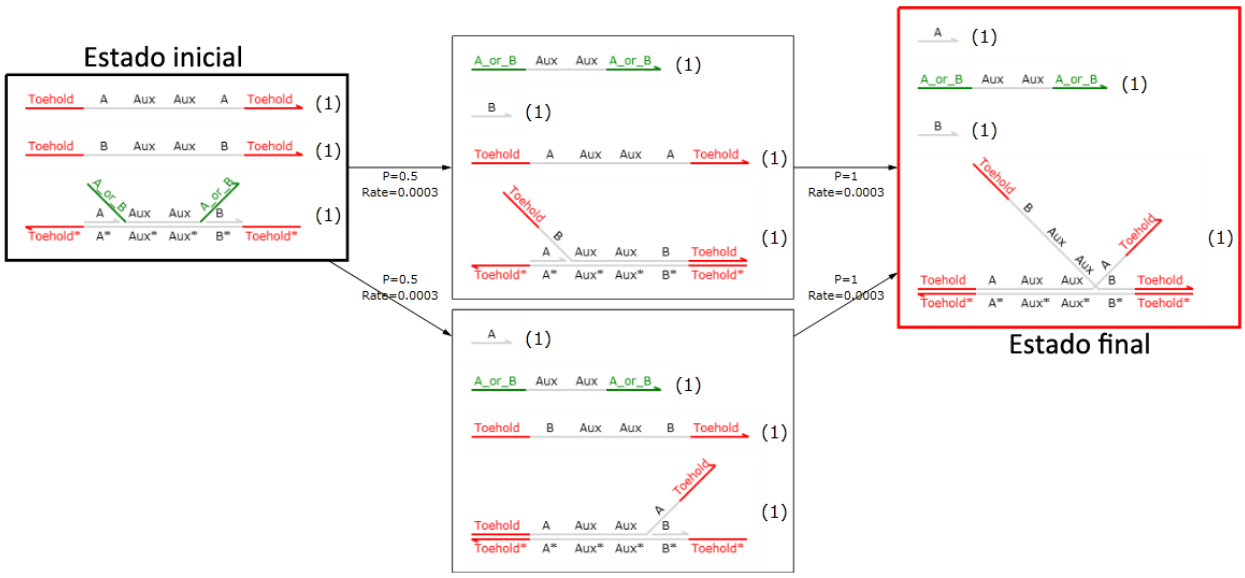


Figura A.01: Diagrama de transición de los estados de la disyunción $A \vee B$ obtenido utilizando el módulo de análisis de Visual DSD.

Tabla A.01: Estructuras utilizadas en la simulación de la disyunción lógica

Premisa/Regla de inferencia	Estructura
A	<Toehold^A Aux Aux A Toehold^>
B	<Toehold^B Aux Aux B Toehold^>
$A \vee B$	<A_or_B^Aux Aux A_or_B^>
Disyunción	{Toehold^*}[A]:<A_or_B^>[Aux Aux]<A_or_B^>:[B]{Toehold^*}

Tabla A.02: Secuencias utilizadas para la codificación de los dominios usados para la simulación de la disyunción lógica

Dominio	Secuencias
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
B	(5') CCCTTTACATTACATAACAA (3')
A	(5') CCCAAAACAAAACAAAACAA (3')
A_or_B^	(5') GCTA (3')

Estructura del conectivo de proposición condicional

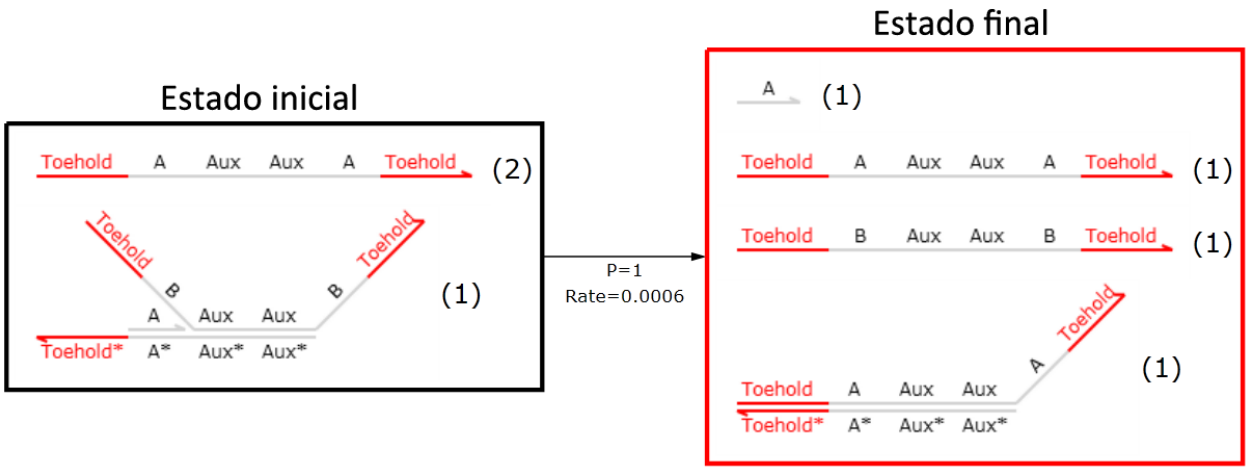


Figura A.01: Diagrama de transición de los estados del modelo de la proposición condicional $A \rightarrow B$ obtenido utilizando el módulo de análisis de Visual DSD.

Tabla A.01: Estructuras utilizadas en la simulación de la implicación lógica.

Premisa/Regla de inferencia	Estructura
A	<Toehold^A Aux Aux A Toehold^>
B	<Toehold^B Aux Aux B Toehold^>
Implicación lógica	{Toehold^*}[A]:<Toehold^B Aux>[Aux]<B Toehold^>

Tabla A.02: Secuencias utilizadas para la codificación de los dominios usados para la simulación de la implicación lógica

Dominio	Secuencias
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
B	(5') CCCTTTACATTACATAACAA (3')
A	(5') CCCAAAACAAAACAAAACAA (3')

Apéndice B

Estructura de las reglas *modus ponens* y *modus tollens*

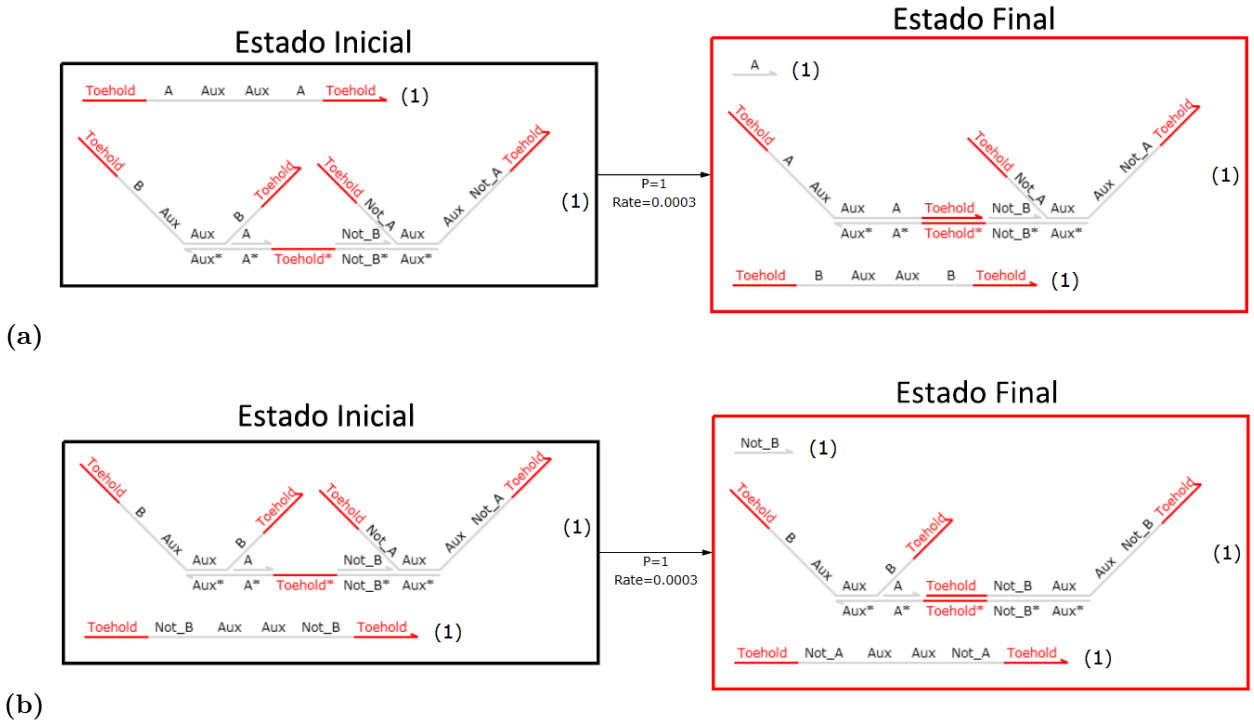


Figura B.01: Diagrama de transición de estados, generado por el módulo de análisis de Visual DSD para las reglas (a) *modus ponens* y (b)*modus tollens*.

Tabla B.01: Estructuras utilizadas en la simulación del *modus ponens* y *modus tollens*

Premisa/Regla de inferencia	Estructura
A	<Toehold^A Aux Aux A Toehold^>
$\neg A$	<Toehold^Not_A Aux Aux Not_A Toehold^>
B	<Toehold^B Aux Aux B Toehold^>
$\neg B$	<Toehold^Not_B Aux Aux Not_B Toehold^>
Modus ponens y *Modus tollens*	<Toehold^B Aux>[Aux]<B Toehold^>:[A]{Toehold^*}:[Not_B]:<Toehold^Not_A>[Aux]<Aux Not_A Toehold^>

Tabla B.02: Secuencias utilizadas para la codificación de los dominios usados para la simulación del *modus ponens* y *modus tollens*

Dominio	Secuencias
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
B	(5') CCCTTTACATTACATAACAA (3')
A	(5') CCCTTATCATATCAATACAA (3')
A_or_B^	(5') TATTCC (3')
Not_B	(5') CCCAAAACAAAACAAAACAA (3')
Not_A	(5') CCCTTAACTTAACAAATCTA (3')

Estructura de la regla de adición

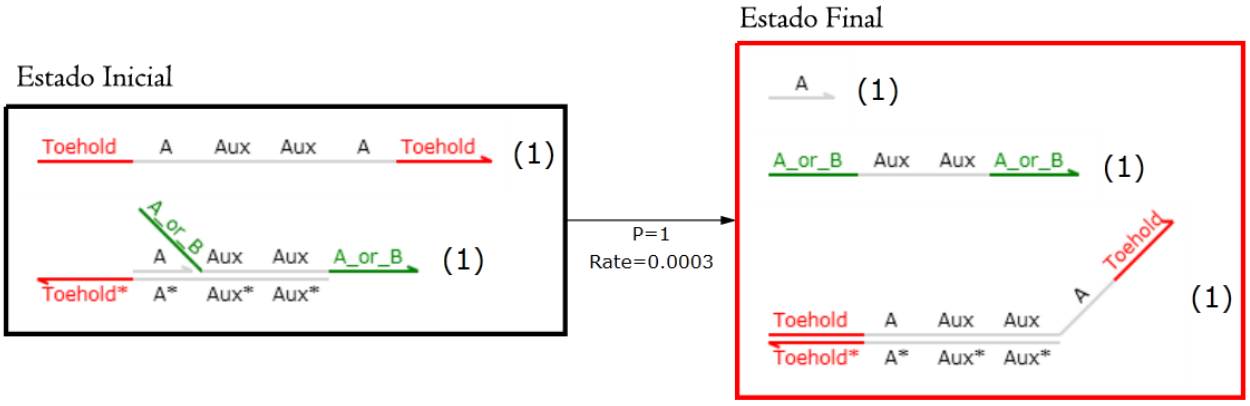


Figura B.01: Diagrama de transición de los estados del modelo de la regla de adición $A \implies A \vee B$ obtenido utilizando el módulo de análisis de Visual DSD. El estado inicial se encuentra remarcado en negro, mientras que el estado final posee un borde marcado en rojo.

Tabla B.01: Estructuras utilizadas en la simulación de la regla de adición

Premisa/Regla de inferencia	Estructura
A	<Toehold^A Aux Aux A Toehold^>
$A \vee B$	<A_or_B^Aux Aux A_or_B^>
Adición	{Toehold^*}[A]:<A_or_B^>[Aux Aux]<A_or_B^>

Tabla B.02: Secuencias utilizadas para la codificación de los dominios utilizados para la simulación de la regla de adición

Dominio	Secuencias
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
A	(5') CCCAAAACAAAACAAAACAA (3')
A_or_B^	(5') GCTA (3')

Estructura de la regla de simplificación

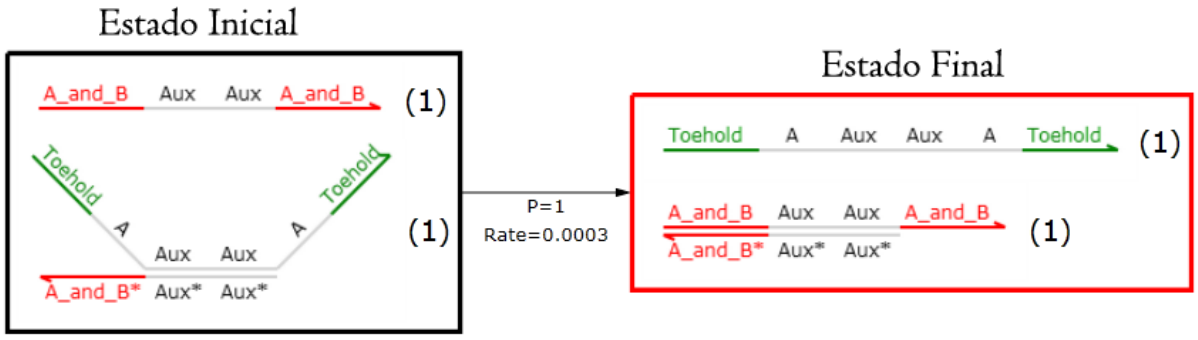


Figura B.01: Diagrama de transición de los estados del modelo de la regla de simplificación $A \wedge B \implies A$ obtenido utilizando el módulo de análisis de Visual DSD. El estado inicial se encuentra remarcado en negro, mientras que el estado final posee un borde marcado en rojo.

Tabla B.01: Estructuras utilizadas en la simulación de la simplificación

Premisa/Regla de inferencia	Estructura
$A \vee B$	<Toehold^A Aux Aux A Toehold^>
$A \wedge B$	<A_and_B^Aux Aux A_and_B^>
B	<Toehold^B Aux Aux B Toehold^>
Simplificación	{A_and_B^*} <Toehold^A>[Aux Aux]<A Toehold^>

Tabla B.02: Secuencias utilizadas para la codificación de los dominios utilizados para la simulación de la simplificación

Dominio	Secuencias
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
B	(5') CCCTTTACATTACATAACAA (3')
A	(5') CCCAAAACAAAACAAAACAA (3')
A_and_B^	(5') GCTA (3')

Estructura de la regla de silogismo hipotético

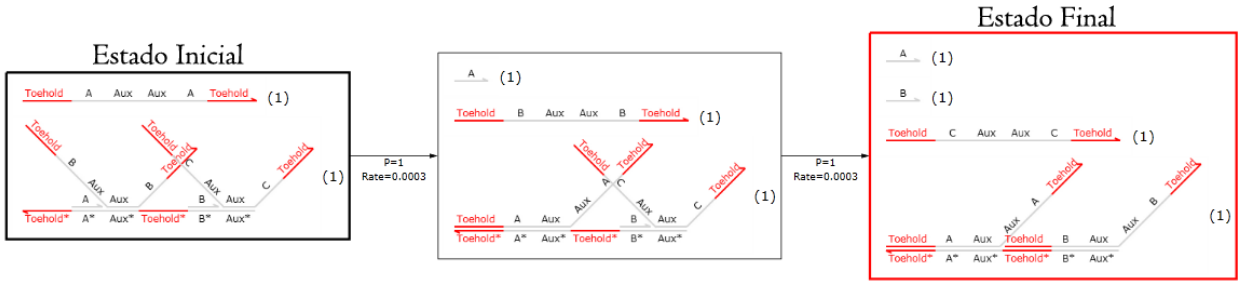


Figura B.01: Diagrama de transición de los estados del modelo de silogismo hipotético obtenido utilizando el módulo de análisis de Visual DSD. El estado inicial se encuentra remarcado en negro, mientras que el estado final posee un borde marcado en rojo.

Tabla B.01: Estructuras utilizadas en la simulación del silogismo hipotético

Premisa/Regla de inferencia	Estructura
A	<Toehold^A Aux Aux A Toehold^>
B	<Toehold^B Aux Aux B Toehold^>
C	<Toehold^C Aux Aux C Toehold^>
Silogismo hipotético	{Toehold^*}[A]:<Toehold^B Aux>[Aux]<B Toehold^> {Toehold^*}:[B]:<Toehold^C Aux>[Aux]<C Toehold^>

Tabla B.02: Secuencias utilizadas para la codificación de los dominios utilizados para la simulación del *modus tollendo ponens*

Dominio	Secuencias
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
B	(5') CCCTTTACATTACATAACAA (3')
A	(5') CCCAAAACAAAACAAAACAA (3')
A_or_B^	(5') TATTCC (3')
C	(5') CCCTTATCATATCAATACAA (3')

Estructura de la regla dilema constructivo

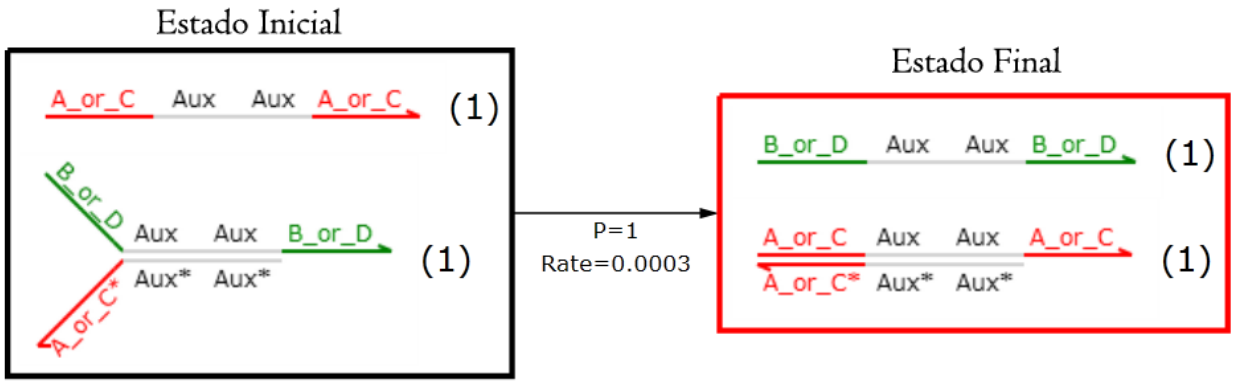


Figura B.01: Diagrama de transición de los estados del modelo del dilema constructivo obtenido utilizando el módulo de análisis de Visual DSD. El estado inicial se encuentra remarcado en negro, mientras que el estado final posee un borde marcado en rojo.

Tabla B.01: Estructuras utilizadas en la simulación del dilema constructivo

Premisa/Regla de inferencia	Estructura
$A \vee C$	<A_or_C^Aux Aux A_or_C^>
$B \vee D$	<B_or_D^Aux Aux B_or_D^>
Dilema constructivo	{A_or_C^*} <B_or_D^>[Aux Aux]<B_or_D^>

Tabla B.02: Secuencias utilizadas para la codificación de los dominios usados para la simulación del dilema constructivo

Dominio	Secuencias
Aux	(5') CCCAAAACAAAACAAAACAA (3')
A_or_C^	(5') TATTCC (3')
B_or_D^	(5') GCTA (3')

Estructura de la regla *modus tollendo ponens*

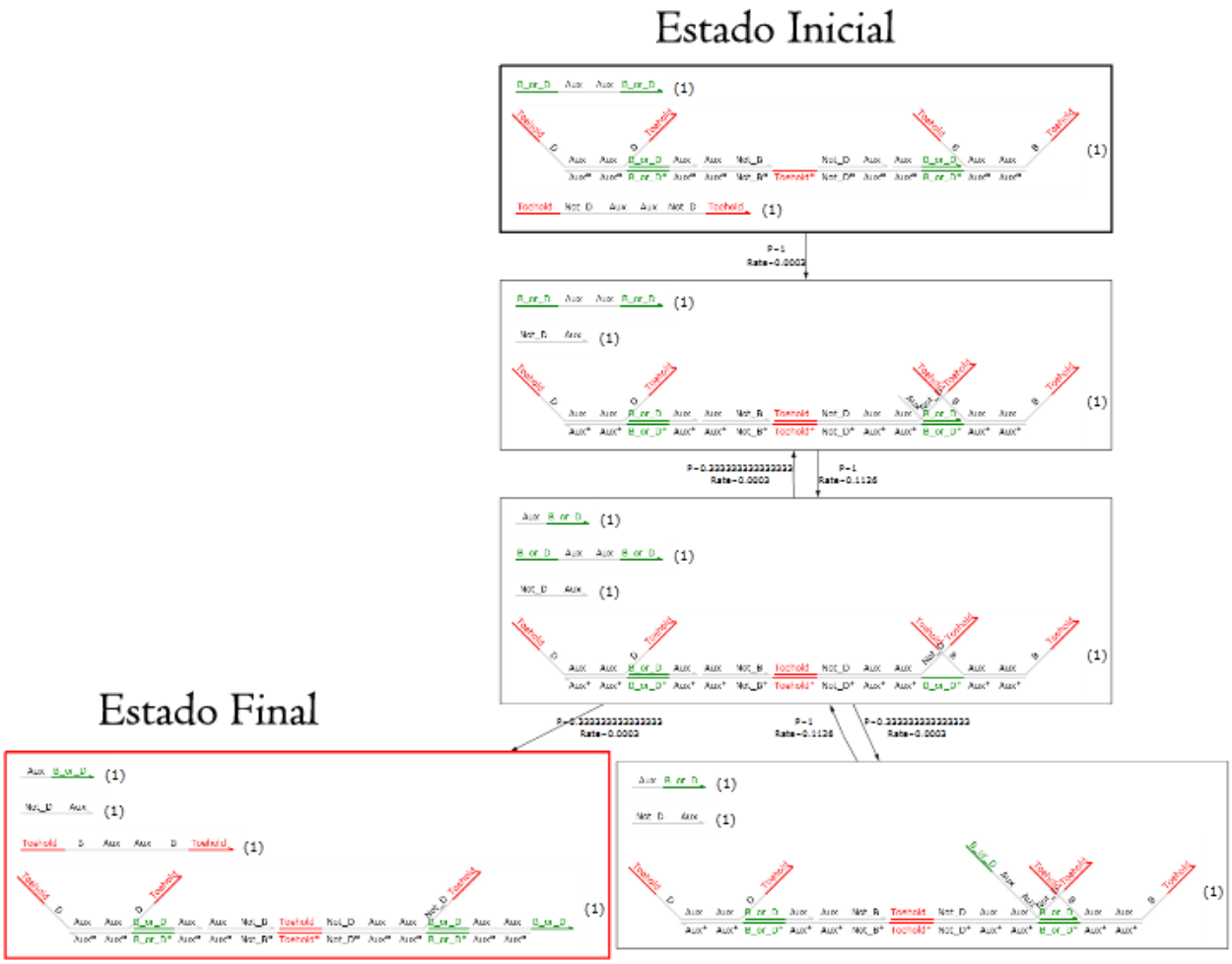


Figura B.01: Diagrama de transición de los estados del modelo de la regla *modus tollendo ponens* obtenido utilizando el módulo de análisis de Visual DSD. El estado inicial se encuentra remarcado en negro, mientras que el estado final posee un borde marcado en rojo.

Tabla B.01: Estructuras utilizadas en la simulación del *modus tollendo ponens*

Premisa/Regla de inferencia	Estructura
$A \vee B$	<A_or_B^Aux Aux A_or_B^>
$\neg A$	<Toehold^Not_A Aux Aux Not_A Toehold^>
B	<Toehold^B Aux Aux B Toehold^>
Modus tollendo ponens	<Toehold^B>[Aux Aux]<B Toehold^>:[A_or_B^Aux]:[Aux Not_A]{Toehold^*}:[Not_B Aux]:[Aux A_or_B^]:<Toehold^A>[Aux Aux]<A Toehold^>

Tabla B.02: Secuencias utilizadas para la codificación de los dominios utilizados para la simulación del *modus tollendo ponens*

Dominio	Secuencias
Toehold^	(5') GCTA (3')
Aux	(5') CCCAAAACAAAACAAAACAA (3')
B	(5') CCCTTTACATTACATAACAA (3')
A	(5') CCCTTAACTTAACAAATCTA (3')
A_or_B^	(5') TATTCC (3')
Not_A	(5') CCCTTTTCTAAACTAAACAA (3')
Not_B	(5') CCCTTATCATATCAATACAA (3')

Apéndice C

Ejemplo de razonamiento lógico: Caso 1.

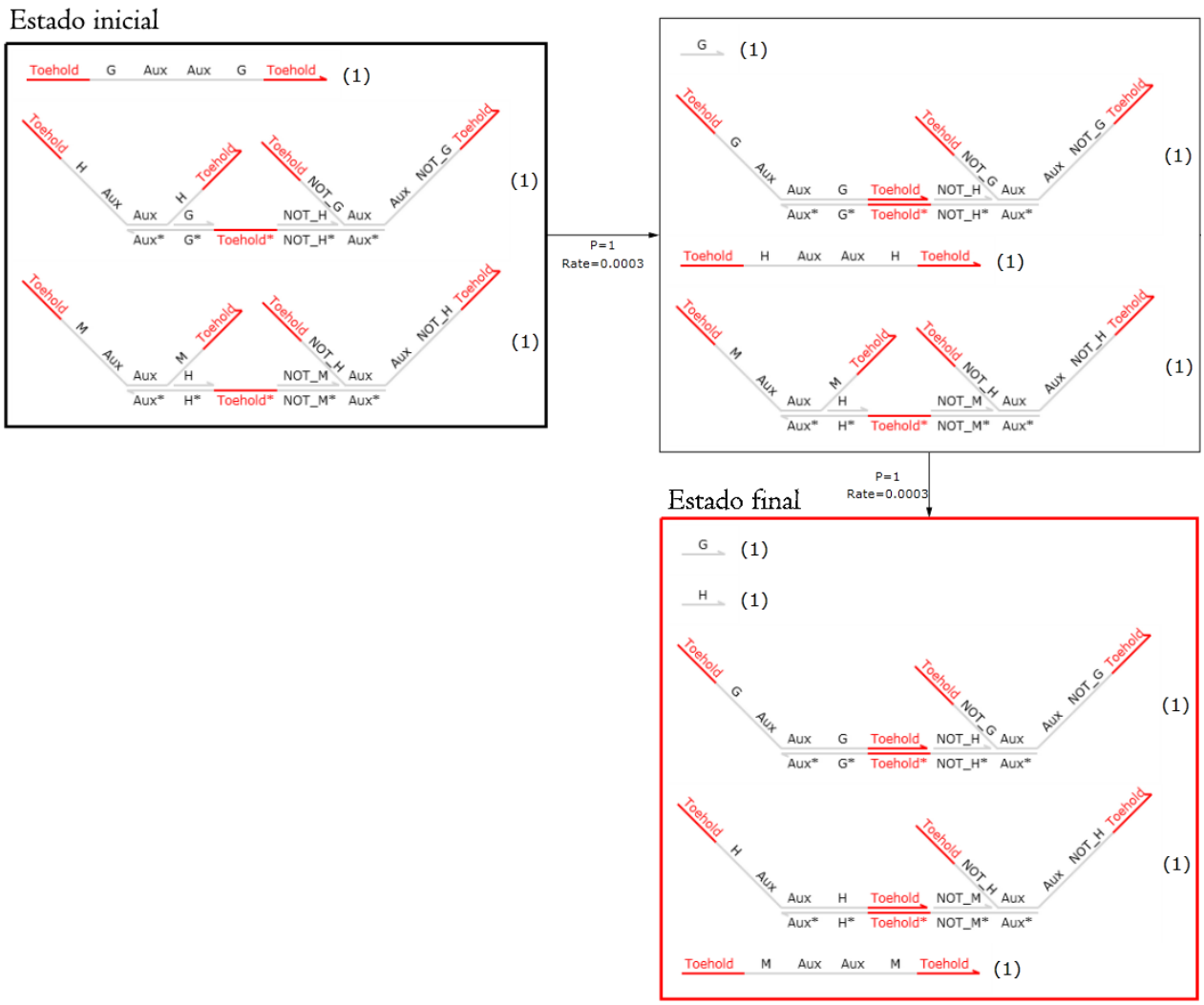


Figura C.01: Diagrama de transición de los estados del modelo creado para el ejemplo de inferir lógicamente si Platón es mortal, obtenido utilizando el módulo de análisis de Visual DSD. El estado inicial se encuentra remarcado en negro, mientras que el estado final posee un borde marcado en rojo.

Tabla C.01: Estructuras utilizadas en la simulación del ejemplo de razonamiento lógico válido

Premisa/Regla de inferencia	Estructura
G	<Toehold^G Aux Aux G Toehold^>
H	<Toehold^H Aux Aux H Toehold^>
M	<Toehold^M Aux Aux M Toehold^>
Modus ponens $H \rightarrow M$	<Toehold^M Aux >[Aux]<M Toehold^>:[H]:{Toehold^*}[NOT_M]:<Toehold^NOT_H>[Aux]<Aux NOT_H Toehold^>
Modus ponens $G \rightarrow H$	<Toehold^H Aux>[Aux]< H Toehold^>:[G]:{Toehold^*}[NOT_H]:<Toehold^NOT_G >[Aux]<Aux NOT_G Toehold^>

Tabla C.02: Secuencias utilizadas para la codificación de los dominios utilizados para la simulación del ejemplo de razonamiento lógico válido

Dominio	Secuencias
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
G	(5') CCCAAAACAAAACAAAACAA (3')
H	(5') CCCTTTACATTACATAACAA (3')
M	(5') CCCTTATCATATCAATACAA (3')
Not_G	(5') CCCTTAACTTAACAAATCTA (3')
Not_H	(5') CCCTATTCAATTCAAATCAA (3')
Not_M	(5') CCCTATACTATACAATACTA (3')

Ejemplo de razonamiento lógico: Caso 2.

Tabla C.01: Estructuras utilizadas en la simulación del caso 2

Premisa/Regla de inferencia	Estructura
$\neg J \vee S$	<Not_J_or_S^Aux Aux Not_J_or_S>
$J \wedge \neg L$	<J_and_Not_L^Aux Aux J_and_Not_L^>
Modus ponens/*Modus tollens* $\neg L \rightarrow \neg S$	<Toehold^Not_S Aux>[Aux]<Not_S Toehold^>:[Not_L]:{Toehold^*}:[S]:<Toehold^L>[Aux]<Aux L Toehold^>
Simplificación $J \wedge \neg L \implies J$	<Toehold^J >J_and_Not_L^*[Aux Aux]<J Toehold^>
Simplificación $J \wedge \neg L \implies \neg L$	<Toehold^Not_L >J_and_Not_L^*[Aux Aux]<Not_L Toehold^>
Modus tollendo ponens $\neg J \vee S, J/\neg S \implies S/\neg J$	<Toehold^S>[Aux Aux]<S Toehold^>:[Not_J_or_S^Aux]:[Aux J]{Toehold^*}:[Not_S Aux]:[Aux Not_J_or_S^]:<Toehold^Not_J >[Aux Aux]<Not_J Toehold^>

Tabla C.02: Secuencias utilizadas para la codificación de los dominios de las estructuras utilizadas para la simulación del ejemplo del caso 2

Dominio	Secuencias
J_and_Not_L^	(3')GTCA (5')
Not_J_or_S^	(5') GCTA (3')
Toehold^	(5') TATTCC (3')
Aux	(5') CCCTTTTCTAAACTAAACAA (3')
F	(5') CCCTTAACTTAACAAATCTA (3')
J	(5') CCCAAAACAAAACAAAACAA (3')
L	(5') CCCTTATCATATCAATACAA (3')
S	(5') CCCTTTACATTACATAACAA (3')
Not_F	(5') CCCTAAACTTATCTAAACAT (3')
Not_J	(5') CCCTATTCAATTCAAATCAA (3')
Not_L	(5') CCCTAATCTAATCATAACTA (3')
Not_S	(5') CCCTATACTATACAATACTA (3')