UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Simulación de modelos computacionales mediante computación con plásmidos

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: García López, Pablo

Tutor/a: Sempere Luna, José María

CURSO ACADÉMICO: 2023/2024

Resum

Aquest Treball de Fi de Grau té com a objectiu principal demostrar que la computació amb plàsmids constitueix un model de computació complet, capaç de simular màquines de registres i, per tant, màquines de Turing. Per a això, es desenvolupen llibreries que permeten la simulació de sistemes P amb plàsmids i s'implementa un compilador que transforma codi de màquina de registres en sistemes P equivalents. A més, es presenta un simulador amb una interfície gràfica que facilita la visualització i execució d'aquests sistemes. Finalment, es valida l'efectivitat del simulador amb diferents casos de prova.

Aquest treball no només amplia l'enteniment dels models de computació, sinó que també obre noves possibilitats en l'àmbit de la biocomputació, aprofitant les capacitats úniques dels plàsmids per realitzar operacions complexes de manera eficient i a escala molecular.

Paraules clau: Computació amb membranes, Computació amb plàsmids, Llibreries de programació d'alt nivell, Màquines de registres, Models de computació universals, Sistemes P

Resumen

Este Trabajo de Fin de Grado tiene como objetivo principal demostrar que la computación con plásmidos constituye un modelo de computación completo, capaz de simular máquinas de registros y, por ende, máquinas de Turing. Para ello, se desarrollan librerías que permiten la simulación de sistemas P con plásmidos y se implementa un compilador que transforma código de máquina de registros en sistemas P equivalentes. Además, se presenta un simulador con una interfaz gráfica que facilita la visualización y ejecución de estos sistemas. Finalmente se valida la efectividad del simulador con diferentes casos de prueba.

Este trabajo no solo amplía el entendimiento de los modelos de computación, sino que también abre nuevas posibilidades en el ámbito de la biocomputación, aprovechando las capacidades únicas de los plásmidos para realizar operaciones complejas de manera eficiente y a escala molecular.

Palabras clave: Computación con membranas, Computación con plásmidos, Librerías de programación de alto nivel, Máquinas de registros, Modelos de computación universales, Sistemas P

Abstract

The main objective of this Final Year Project is to demonstrate that plasmid computing constitutes a complete model of computation capable of simulating register machines, and thus Turing machines. To achieve this, libraries are developed that allow the simulation of P systems with plasmids, and a compiler is implemented that transforms register machine code into equivalent P systems. Additionally, a simulator with a graphical interface is presented to facilitate the visualization and execution of these systems. Finally, the effectiveness of the simulator is validated with various test cases.

This work not only expands the understanding of computational models but also opens new possibilities in the field of biocomputation, leveraging the unique capabilities of plasmids to perform complex operations efficiently and at the molecular scale.

Key words: Membrane computing, Plasmid computing, High-level programming libraries, Register machines, Universal computing models, P systems

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

La computación ha evolucionado significativamente desde sus inicios en la década de 1930, cuando matemáticos y científicos como Kurt Gödel, Alonzo Church, Stephen Kleene y Alan Turing sentaron las bases de la teoría de la computabilidad. En este contexto, los modelos de computación han surgido como descripciones teóricas de máquinas o sistemas que ejecutan un conjunto definido de operaciones para llevar a cabo algoritmos. Uno de los modelos más influyentes es la Máquina de Turing, presentada por Alan Turing en 1936, que permitió clasificar los problemas en decidibles y no decidibles, estableciendo un criterio fundamental en este campo: un problema o función es Turing-computable si una Máquina de Turing puede resolverlo en un tiempo finito.

Desde entonces, han surgido numerosos modelos de computación, incluyendo la computación natural, las funciones recursivas, el cálculo lambda y los modelos de computación cuántica, todos ellos con el objetivo de ser equivalentes a la Máquina de Turing. En este Trabajo de Fin de Grado (TFG), se busca demostrar que un modelo basado en los sistemas P, al cual se le han añadido plásmidos como variante, es equivalente a la Máquina de Turing, es decir, Turing-completo.

1.1 Motivación

Las máquinas de registros son modelos de computación completos que surgieron en las décadas de 1960 y 1970. Al ser modelos de computación completos, pueden simular a las máquinas de Turing. Este hecho les otorga una gran relevancia en el estudio de la teoría de la computación, ya que permiten explorar los límites de lo computable y proporcionan una base sólida para el desarrollo de nuevos modelos de computación.

La motivación principal de este trabajo es demostrar que la computación con plásmidos también constituye un modelo de computación completo. Se plantea la hipótesis de que, mediante la computación con plásmidos, es posible simular a las máquinas de registros y, por ende, a las máquinas de Turing. De confirmarse, esto no solo ampliaría el entendimiento actual de los modelos de computación, sino que también abriría nuevas posibilidades en el ámbito de la biocomputación, aprovechando las capacidades únicas de los plásmidos para realizar operaciones complejas de manera eficiente y a escala molecular.

Mi interés por este trabajo comenzó durante mi tercer año de Ingeniería Informática, en la asignatura de Computabilidad y Complejidad. Fue allí donde me sumergí en conceptos como los modelos de computación, incluyendo la famosa Máquina de Turing, entre otros. Durante las prácticas, dirigidas por el tutor de este proyecto, exploramos modelos de computación natural, como el basado en membranas (Sistemas P), el cual me llamó especialmente la atención. Y el hecho de poder relacionarlos haciendo una equivalencia entre ambos modelos me parece fascinante. Creo firmemente que este TFG puede abrir la puerta a grandes proyectos utilizando los Sistemas P, y estoy encantado de poder contribuir a ello.

Además, la parte de implementación del simulador es algo que me entusiasma especialmente. Siempre me ha gustado la programación y poder diseñar y desarrollar esta aplicación utilizando diversas lógicas de programación y tecnologías avanzadas, representa para mí un reto muy interesante y una excelente manera de concluir mi cuarto año académico con este Trabajo de Fin de Grado.

1.2 Objetivos

Los objetivos de este trabajo son los siguientes:

1. Desarrollar unas librerías que permiten la simulación de la computación con plásmidos.
2. Demostrar que la computación con plásmidos es un modelo de computación completo. Para ello, se llevará a cabo la simulación de máquinas de registros mediante el uso de plásmidos, mostrando que estas pueden emular el comportamiento de una máquina de Turing.
3. Desarrollar y validar algoritmos específicos para la simulación de máquinas de registros utilizando plásmidos. Se diseñarán procedimientos detallados que permitan replicar las operaciones básicas de las máquinas de registros mediante el uso de plásmidos.
4. Contribuir al cuerpo de conocimiento existente sobre modelos de computación no convencionales, proporcionando una base teórica y práctica que pueda ser utilizada en investigaciones futuras sobre computación biológica y molecular.

1.3 Estructura de la memoria

La estructura de la memoria se organiza de la siguiente manera:

Primero, en la sección de "Conceptos Básicos", se explican los fundamentos necesarios para comprender el trabajo. Se introducen las máquinas de registros y los sistemas P, detallando su estructura, reglas de evolución y procesos de computación.

En la siguiente sección, "Computación con Plásmidos", se define el concepto de plásmido y se describe detalladamente su aplicación en la computación. Se

presentan las reglas específicas que rigen su uso y se formaliza el modelo computacional basado en plásmidos.

Después, se aborda la sección "Equivalencia entre Máquinas de Registros y Computación con Plásmidos", la cual establece la relación entre ambos conceptos, demostrando su equivalencia en términos de capacidad computacional.

Luego de la explicación teórica, se desarrolla la sección de "Implementación", detallando las tecnologías y herramientas empleadas para llevar a cabo el modelo computacional propuesto. Se describen las librerías de sistemas P con plásmidos y el compilador desarrollado para este fin.

Seguidamente, tras la implementación del simulador, se presenta la sección "Casos de Prueba", donde se exponen diversos casos que validan el modelo computacional propuesto. Se muestran los resultados obtenidos y se discuten las implicaciones derivadas de estos casos.

Finalmente, en la sección de "Conclusiones", se resumen los hallazgos obtenidos en el trabajo. Se incluye también la sección de "Bibliografía", donde se citan las fuentes utilizadas para el desarrollo del trabajo, y los "Apéndices", que contienen los sistemas P con plásmidos generados en los casos de prueba y los objetivos de desarrollo sostenible abordados.

CAPÍTULO 2
Conceptos básicos

2.1 Máquina de registros

Una máquina de registros es un modelo teórico de computadora que se basa en la idea de que la memoria se divide en registros. Estos registros se utilizan para almacenar y manipular datos. Cada registro tiene una dirección representada por un número natural, que permite acceder directamente a su contenido, también representado por un número natural. Así, el conjunto de direcciones corresponde al conjunto de los números naturales. Se denota por R_i al registro i.

La programación de las máquinas de registros se realiza mediante instrucciones que especifican las operaciones a realizar sobre los valores almacenados en los registros. Un programa consiste en una serie finita de instrucciones simples y básicas, numeradas consecutivamente. La ejecución siempre comienza en la primera instrucción y termina cuando se intenta acceder a una instrucción inexistente.

El conjunto básico de instrucciones es el siguiente:

- **suc(i):** Incrementa en una unidad el contenido de R_i y continúa con la siguiente instrucción.
- **pre(i, k):** Si $R_i > 0$, decrementa en una unidad el contenido de R_i y continúa con la siguiente instrucción; de lo contrario, salta a la instrucción k.
- **goto(n):** Ejecuta incondicionalmente la instrucción n-ésima.

Estas tres instrucciones son suficientes para realizar cualquier computación. Además, permiten implementar instrucciones más complejas, como las siguientes:

- **cer(i):** Asigna cero al contenido de R_i.

```
n:      pre(i, n+2)
n+1:    goto(n)
```

- **asi(k, i):** Asigna el valor k a R_i.

```
        cer(i)
n+1:    suc(i)
        ......
n+k:    suc(i)
```

- **cop(j, i):** Copia el contenido de R_j en R_i.

```
        cer(i)
        cer(k)
n:      pre(j, n+4)
n+1:    suc(i)
n+2:    suc(k)
n+3:    goto(n)
n+4:    pre(k, n+7)
n+5:    suc(j)
n+6:    goto(n+4)
```

- **sum(p, q, m):** Almacena en R_m la suma de los contenidos de R_p y R_q.

```
        cop(p, i)
        cop(q, j)
n:      pre(j, n+3)
n+1:    suc(i)
n+2:    goto(n)
n+3:    cop(i, m)
```

- **mul(p, q, m):** Almacena en R_m el resultado de la multiplicación de los contenidos de R_p y R_q.

```
        cer(r)
        cop(p, i)
n:      pre(j, n')
        sum(r, q, r)
n':     goto(n)
n'+1:   cop(r, m)
```

- **div(p, q, m):** Almacena en R_m el resultado de la división de R_p entre R_q.

```
        cer(r)
        cer(k)
        cop(p, i)
        cop(q, j)
n:      pre(j, error)
n+1:    suc(j)
n+2:    pre(j, n+6)
```

```
n+3:   pre(j, n+10)
n+4:   suc(k)
n+5:   goto(n+2)
n+6:   suc(r)
n+7:   pre(k, n+2)
n+8:   suc(j)
n+9:   goto(n+7)
n+10:  cop(r, m)
error: ...
```

- **mei(p, q, m1, m2):** Comprueba si R_p es menor o igual a R_q; en caso afirmativo, salta a la instrucción $m1$, en caso contrario, salta a la instrucción $m2$.

```
       cop(p, i)
       cop(q, j)
n:     pre(i, m1)
n+1:   pre(j, m2)
n+2:   goto(n)
```

- **igu(p, q, m1, m2):** Comprueba si R_p es igual a R_q; en caso afirmativo, salta a la instrucción $m1$, en caso contrario, salta a la instrucción $m2$.

```
       cop(p, i)
       cop(q, j)
n:     pre(i, n+3)
n+1:   pre(j, m2)
n+2:   goto(n)
n+3:   pre(j, m1)
n+4:   goto(m2)
```

2.2 Sistemas P

Los sistemas celulares de computación con membranas forman parte de la computación natural. Su objetivo es crear modelos computacionales, conocidos como sistemas de membranas (o sistemas P), basándose en la observación de la estructura y funciones de elementos como las células, los tejidos celulares, los órganos, y las poblaciones celulares.

Las primeras investigaciones en este campo comenzaron en 1998 y rápidamente ganaron aceptación dentro de la comunidad científica. Tanto es así que, en 2003, el Instituto Thomson for Scientific Information identificó la computación con membranas como una nueva área emergente en las ciencias de la computación.

Los sistemas de membranas también se denominan sistemas P, probablemente en honor a la inicial del apellido de su creador, George Păun.

La computación con membranas se originó a partir de la búsqueda de un modelo inspirado en el funcionamiento y la estructura de una célula viva, particularmente en el papel que desempeñan las membranas dentro de la célula. Las membranas celulares, como es bien sabido, separan y compartimentalizan las funciones metabólicas, aislando los diversos componentes biológicos que intervienen a lo largo del ciclo de vida celular [3].

2.2.1. Estructura de los Sistemas P

Un sistema P se compone de varias membranas jerárquicamente anidadas, formando una estructura tipo árbol. Cada membrana define una región y puede contener objetos, reglas de evolución y otras membranas.

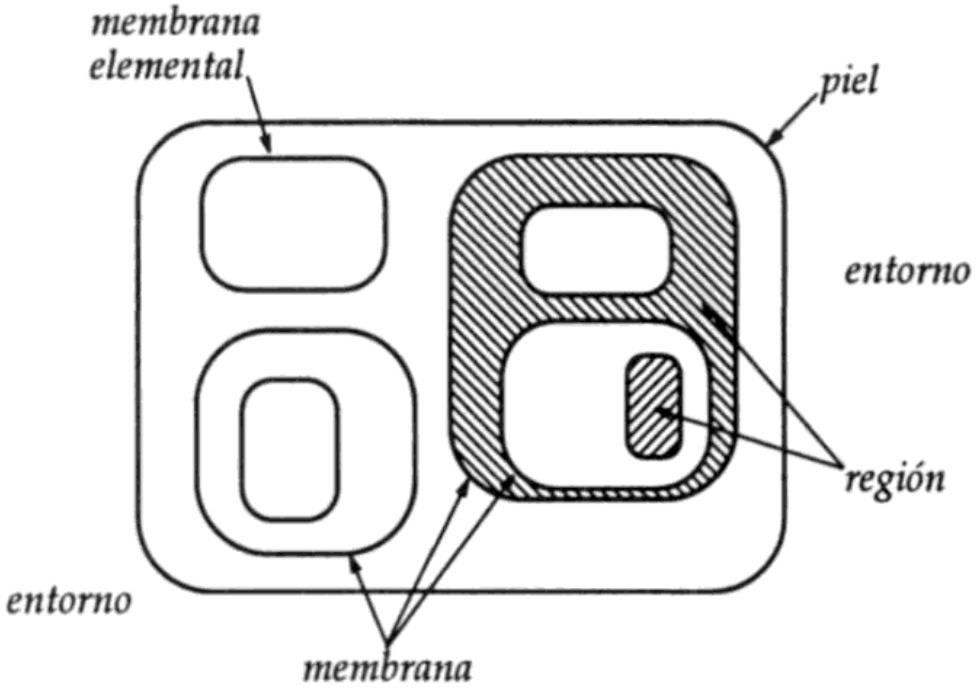

Figura 2.1: Representación gráfica de un sistema de membranas a través de un Diagrama Venn [6]

En las regiones delimitadas por las membranas se encuentran objetos que pueden representarse mediante símbolos de un alfabeto finito. Estos objetos evolucionan según reglas específicas.

2.2.2. Reglas de Evolución

Las reglas de evolución pueden ser de varios tipos:

- **Reescritura:** Transforman un objeto en otro.

- **Transporte:** Mueven objetos de una membrana a otra.
- **Disolución:** Disuelven una membrana, liberando sus objetos a la membrana circundante.

Se aplican de manera paralela y simultánea en cada paso de la computación. Las reglas están sujetas a una relación de prioridad que determina el orden de aplicación cuando varias reglas son aplicables a un mismo objeto.

La evolución de un sistema P puede describirse mediante una secuencia de configuraciones, donde cada configuración es un estado del sistema en un momento dado. La configuración incluye la distribución de objetos en las diferentes membranas y el estado de las membranas (si están disueltas o no).

2.2.3. Proceso de Computación

El modelo opera bajo el principio de ”máximo paralelismo”, lo que implica que en cada paso de ejecución se aplican todas las reglas posibles de cada membrana. Las reglas pueden ejecutarse siempre que no exista un conflicto entre ellas; es decir, cuando dos o más reglas no requieran el mismo objeto para su ejecución. En caso de conflicto, se selecciona una de las reglas de manera no determinista, es decir, aleatoriamente. El concepto de paso de ejecución está determinado por un reloj global, que sincroniza la ejecución de todas las membranas simultáneamente y en paralelo.

Debido a estos pasos de ejecución, la computación comienza con una configuración inicial que consiste en una distribución específica de objetos en las membranas. El sistema evoluciona aplicando las reglas de evolución en paralelo hasta alcanzar una configuración en la que no se pueden aplicar más reglas (configuración final).

La computación puede terminar de dos maneras:

- **Terminación normal:** La computación termina cuando se alcanza una configuración final donde ninguna regla es aplicable.
- **No terminación:** La computación no puede detenerse porque siempre hay alguna regla aplicable, lo que indica que el sistema no puede producir una respuesta definitiva.

El resultado de la computación se interpreta a partir de la configuración final, generalmente contando el número de ciertos objetos en una membrana específica o evaluando el estado de la estructura de membranas.

2.2.4. Formalización de los Sistemas P

Un sistema de computación por membranas se puede representar mediante una formulación matemática, lo que permite mantener la claridad y precisión incluso al aumentar la complejidad del sistema.

La formulación matemática de un Sistema P de transición se define mediante la tupla:

$$\Pi = (V, \Lambda, \mu, w_1, \ldots, w_n, (R_1, \rho_1), \ldots, (R_n, \rho_n), i_0),$$

donde:

(i) V: es un alfabeto cuyos elementos son denominados objetos.

(iii) Λ: es un alfabeto de etiquetas de las membranas.

(ii) μ: es una estructura de membranas de grado n, con las membranas y regiones etiquetadas individualmente con elementos de Λ.

(iv) $w_i, 1 \leq i \leq n$: son cadenas de V^* que representan los multiconjuntos sobre V asociados con las regiones 1, 2, ..., n de μ.

(v) $(R_i, \rho_i), 1 \leq i \leq n$:

- R_i: son conjuntos finitos de *reglas de evolución* sobre objetos de V asociadas a las regiones 1, 2, ..., n.
- ρ_i: es una relación de orden parcial sobre R_i, que especifica una relación de prioridad entre las reglas de la región.

Dependiendo del tipo de regla, se pueden clasificar en tres categorías:

a) $u \rightarrow v_{here}$, $u, v \in \mathrm{V}^*$ (Regla de reescritura).

b) $u \rightarrow v_{out}$, $u, v \in \mathrm{V}^*$ (Regla de transporte, extraer a la membrana padre).

c) $u \rightarrow v_{in_i}$, $u, v \in \mathrm{V}^*$ (Regla de transporte, introducir en una membrana hija).

En el caso de que una regla contenga el símbolo especial δ, se considera también como una regla de disolución, que tras aplicarse disuelve la membrana, haciendo que los objetos en su interior pasen a pertenecer a la membrana que la rodeaba (su membrana padre) y sus reglas desaparezcan.

(vi) i_0: es un numero entre 1 y n que determina la membrana de salida de Π.

2.2.5. Aplicaciones y Variantes de los Sistemas P

Los sistemas P han demostrado ser útiles en diversas áreas de investigación, incluyendo:

- **Biología computacional:** Modelado de procesos biológicos como la replicación del ADN, la dinámica de poblaciones y la interacción entre células.
- **Teoría de la computación:** Estudio de problemas de decisión y problemas NP-completos, proporcionando nuevos enfoques para algoritmos paralelos.

- **Inteligencia artificial:** Desarrollo de algoritmos inspirados en la naturaleza para resolución de problemas, optimización y aprendizaje automático.

Los sistemas P representan un modelo computacional potente y flexible, que aprovecha la paralelización y la distribución natural observada en los sistemas biológicos. Este enfoque ha demostrado ser útil en diversas áreas de investigación, ofreciendo nuevas perspectivas y herramientas para el estudio y la resolución de problemas complejos.

2.2.6. Ejemplos de Sistemas P

Para comprender mejor cómo se definen los Sistemas P veamos dos ejemplos de ellos. Para simplificar la notación se ha decidido omitir el objetivo 'here':

Ejemplo 1: Considerando un Sistema P de grado 4

$$
\begin{aligned}
&\Pi_1 = (V, \mu, w_1, \ldots, w_4, (R_1, \rho_1), \ldots, (R_4, \rho_4), 4),\\
&V = \{a, b, x, c, f\},\\
&\mu = [_1[_2[_3]_3[_4]_4]_2]_1,\\
&w_1 = \lambda, R_1 = \emptyset, \rho_1 = \emptyset,\\
&w_2 = \lambda, R_2 = \{x \to b,\ b \to b(c,\ in_4),\ r_1 :\ ff \to af,\ r_2 :\ f \to a\delta\}, \rho_2 = \{r_1 > r_2\},\\
&w_3 = af, R_3 = \{a \to ax,\ a \to x\delta,\ f \to ff'\}, \rho_3 = \emptyset,\\
&w_4 = \lambda, R_4 = \emptyset, \rho_4 = \emptyset.
\end{aligned}
$$

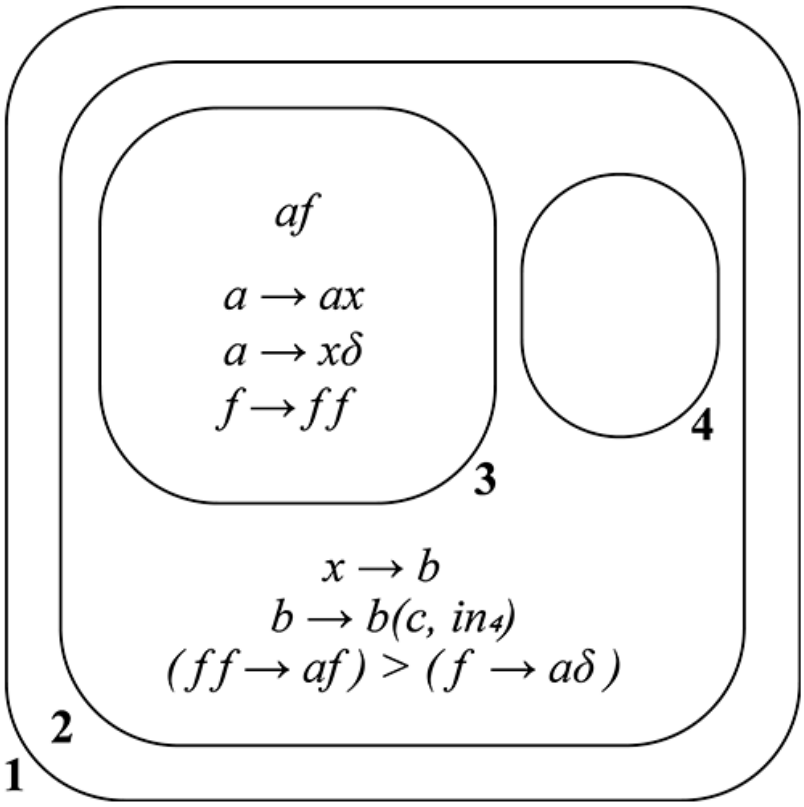

Figura 2.2: Diagrama Venn de un Sistema P que genera $n^2, n \geq 1$

Este Sistema P genera n^2 para $n \geq 1$. Inicialmente, calcula el valor de n dentro de la membrana 3. Una vez que la membrana 3 se disuelve, el cálculo de n^2 se lleva a cabo en la membrana 2. Al finalizar el cálculo, el resultado se almacena en la membrana 4.

Ejemplo 2: Considerando un Sistema P de grado 3

$$
\begin{aligned}
\Pi_2 &= (V, \mu, w_1, \ldots, w_3, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), 3), \\
V &= \{a, c, x, d\}, \\
\mu &= [_1[_2]_2[_3]_3]_1, \\
w_1 &= \lambda, R_1 = \{dcx \to a_{in_3}\}, \rho_1 = \emptyset, \\
w_2 &= a^n c^k d, R_2 = \{r_1 : ac \to x, r_2 : ax \to c, r_3 : d \to d\delta\}, \rho_2 = \{r_1, r_2 > r_3\} \\
w_3 &= a, R_3 = \emptyset, \rho_3 = \emptyset.
\end{aligned}
$$

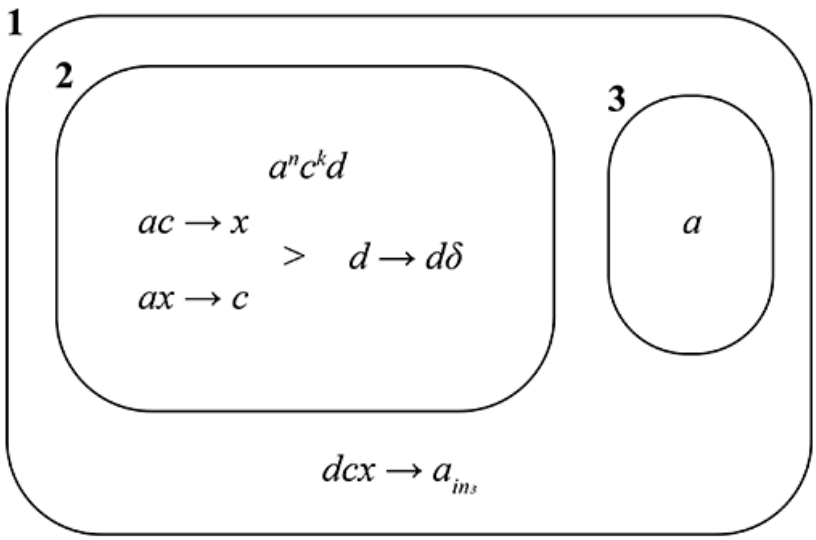

Figura 2.3: Diagrama Venn de un Sistema P que decide si k divide a n

Este sistema tiene una tarea de decisión en la que se introducen en la configuración de entrada dos números, n y k, y su objetivo es decidir si n es un múltiplo de k. En caso afirmativo, en la membrana de salida finalizará con un único objeto ($i_0 = a$); en caso negativo habrán dos objetos ($i_0 = aa$).

CAPÍTULO 3
Computación con Plásmidos

Los Sistemas P, como se describió en la sección 2.2, son modelos computacionales basados en una estructura jerárquica de membranas inspirada en la estructura y el comportamiento de una célula. Estos sistemas ofrecen diversas perspectivas y aplicaciones en múltiples áreas.

En este capítulo, exploraremos la integración de los plásmidos en los Sistemas P. Los plásmidos son diminutos elementos biológicos que, debido a sus propiedades únicas, representan un mecanismo versátil e interesante de transferencia de información. La incorporación de plásmidos en los Sistemas P tiene como objetivo aumentar la flexibilidad de los modelos de computación con membranas existentes y expandir sus posibles aplicaciones.

Comenzaremos con una descripción detallada del plásmido como elemento biológico, seguida de una explicación de los Sistemas P con plásmidos. Se justificará la introducción de los plásmidos en los Sistemas P, destacando las diferencias entre esta nueva variante y la versión tradicional.

3.1 Definición de Plásmido

Un plásmido es una pequeña estructura molecular compuesta de ADN, habitualmente en forma de una molécula circular, que se encuentra predominantemente en bacterias y algunos otros microorganismos biológicos. A diferencia del ADN cromosómico, los plásmidos se replican de manera independiente, lo que les permite duplicarse sin depender de la replicación del cromosoma bacteriano. Esta capacidad de replicación autónoma fue descubierta por el biólogo Joshua Lederberg en 1952.

Los plásmidos suelen contener un número reducido de genes, muchos de los cuales proporcionan ventajas significativas a los organismos que los albergan. Por ejemplo, algunos plásmidos incluyen genes que otorgan resistencia a los antibióticos (figura 3.1), permitiendo a las bacterias sobrevivir en presencia de estos medicamentos. Otros plásmidos pueden conferir habilidades especiales, como la capacidad de metabolizar nutrientes poco comunes o producir toxinas.

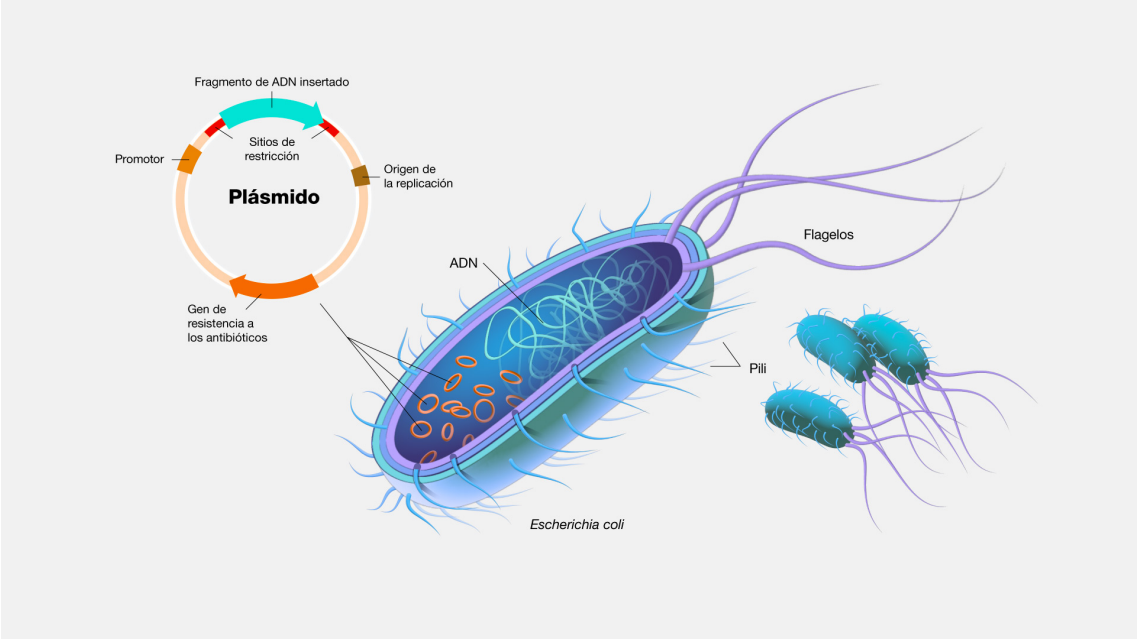

Figura 3.1: Estructura de un plásmido introducido en una bacteria *e coli* [7].

Además de su capacidad de replicación independiente, los plásmidos también pueden transferirse de una célula a otra. Este proceso de transferencia horizontal es crucial para la rápida diseminación de características beneficiosas, como la resistencia a antibióticos, entre poblaciones bacterianas.

La capacidad de los plásmidos para replicarse independientemente y transferirse entre células los convierte en herramientas valiosas en biotecnología y genética. Una de las principales aplicaciones es la clonación de ADN. Este proceso implica insertar una secuencia de ADN de interés en un plásmido, que luego se introduce en una célula huésped, como una bacteria. A medida que el plásmido se replica dentro de la célula, también lo hace la secuencia de ADN insertada, permitiendo la producción masiva de la misma.

Un ejemplo destacado de la aplicación de los plásmidos es la producción de proteínas específicas, como la insulina o ciertos antibióticos. Utilizando métodos de ADN recombinante, los científicos pueden insertar genes que codifican estas proteínas en plásmidos, facilitando su producción eficiente en organismos huésped.

3.2 Modelo computacional con Plásmidos

En la sección anterior, se destacó que los plásmidos son herramientas poderosas en áreas como la medicina, permitiendo la producción de ciertos medicamentos. Las ventajas prácticas de los plásmidos son igualmente aplicables a contextos teóricos que simulan estos escenarios, como los Sistemas P. Por esta razón, se propone integrar los plásmidos en modelos computacionales de este tipo. Al considerar las posibilidades de los plásmidos en un contexto real, se pueden identificar los beneficios de su incorporación en un modelo de Sistemas P con plásmidos, los cuales serían:

1. **Replicación independiente:** Similar a su comportamiento en bacterias, un objeto que emule un plásmido en un modelo computacional con membranas debería ser capaz de replicarse de manera autónoma. Esta capacidad permitiría al sistema replicar y transferir información e instrucciones internamente, incrementando la flexibilidad y eficiencia operativa. Esta función modificaría la configuración de un modelo de Sistemas P, haciéndolo más dinámico y colaborativo al permitir la compartición de instrucciones entre varias membranas, de manera análoga a cómo las bacterias comparten genes a través de plásmidos. Aunque los Sistemas P son un modelo teórico universal capaz de resolver cualquier problema computable por una Máquina de Turing, la inclusión de plásmidos no amplía esta capacidad, pero sí mejora la eficiencia del sistema gracias a la flexibilidad adicional.

2. **Almacenamiento de información:** Dado que los plásmidos pueden almacenar diversos tipos de información, su integración en un sistema informático permite una replicación sencilla y eficiente de la información, característica altamente deseada por su alta disponibilidad y tolerancia a fallos, asegurando la conservación de datos ante posibles fallos del sistema.

3. **Modularidad y adaptabilidad:** La modularidad es una característica crucial para los sistemas informáticos. Los plásmidos, considerados como módulos de información dentro de un sistema computacional, facilitan la adición, sustracción o distribución de estos módulos de manera rápida y sencilla, otorgando al sistema una mayor adaptabilidad a entornos dinámicos. Esto mejora la seguridad del sistema frente a ataques maliciosos o errores internos.

4. **Investigación:** Además de las ventajas aplicables a sistemas informáticos, un modelo de Sistemas P con plásmidos ofrece a la comunidad científica una mejor comprensión del funcionamiento y manejo de plásmidos en contextos biológicos.

Después de analizar las ventajas, se concluye que la integración de plásmidos en los Sistemas P no solo ofrece mejoras prácticas en términos de eficiencia, flexibilidad y seguridad, sino que también abre nuevas vías para la investigación científica, permitiendo una exploración más profunda de estos elementos biológicos en entornos simulados. A continuación, avanzaremos hacia la descripción del modelo propuesto y su comparación con el modelo original. Aunque las diferencias no son numerosas, lo principal es la introducción del objeto "Plásmido" en el sistema, junto con las reglas para gestionarlo.

Teniendo en cuenta la definición de membrana aportada en la sección 2.2, puede ser considerada como un objeto que alberga en su interior un conjunto de objetos, un conjunto de reglas que definen su funcionalidad y una subestructura embebida de membranas hijas, teniendo en cuenta que el conjunto de reglas de la membrana es una abstracción del ADN de una célula, podemos deducir que la estructura de un plásmido es similar.

La habilidad más sobresaliente que aporta un plásmido a una célula es su capacidad de replicarse y funcionar como una unidad genética independiente dentro de la célula huésped. Una vez dentro de una célula, el ADN del plásmido

se integra al material genético de la célula y se replica por sí solo, incluso cuando la célula no se está replicando. Además, el plásmido puede abandonar la célula sin dañar el material genético de ésta. Esto es posible debido a que el material genético del plásmido es independiente al de la célula.

Por lo tanto, el plásmido se considera una membrana que sólo contiene un conjunto de reglas. Estos plásmidos tienen la capacidad de moverse entre membranas, lo que permite que las reglas que portan sean integradas como parte de las reglas de la membrana en la que se encuentran, aportando funcionalidades adicionales que normalmente no estarían disponibles. Además, el plásmido puede replicarse dentro de una membrana o moverse a una membrana adyacente.

3.2.1. Reglas de plásmidos

En el modelo propuesto, definimos un plásmido como una colección finita de reglas de evolución. Dichas reglas siguen la estructura presentada en la sección 2.2.2, reflejando la similitud entre los conceptos de membrana y plásmido. Por ende, las reglas pueden clasificarse en los siguientes tipos:

a) $u \to v_{here}$, $u, v \in \mathrm{V}^*$ (Regla de reescritura).

b) $u \to v_{out}$, $u, v \in \mathrm{V}^*$ (Regla de transporte, extraer a la membrana padre).

c) $u \to v_{in_i}$, $u, v \in \mathrm{V}^*$ (Regla de transporte, introducir en una membrana hija).

Dentro de este marco computacional, la gestión de plásmidos se rige por un conjunto de reglas específicas. Estas reglas están diseñadas para facilitar la transferencia de plásmidos entre membranas adyacentes. Además, se contemplan reglas para la replicación de plásmidos, análogas a la mitosis en células. No obstante, para simplificar la representación del modelo, se omitirán las reglas de replicación.

Se añaden a los tipos de reglas del modelo, mencionado en la sección 2.2.4, las siguientes reglas que permiten el manejo de los plásmidos dentro del sistema:

a) $p_i t [_k a]_k \to r [_k p_i b]_k$, $a,b,r,t \in \mathrm{V}^*$, y $1 \leq i \leq q$ (Regla 'in symport').

b) $t [_k p_i a]_k \to p_i r [_k b]_k$, $a,b,r,t \in \mathrm{V}^*$, y $1 \leq i \leq q$ (Regla 'out symport').

c) $p_i t [_k p_j a]_k \to p_j r [_k p_i b]_k$, $a,b,r,t \in \mathrm{V}^*$, y $1 \leq i \leq q$ (Regla 'anty-port').

Para facilitar la comprensión de las reglas de manejo de plásmidos, vamos a desglosarlas y explicarlas de una manera más sencilla.

Las reglas se dividen en dos tipos principales: "Symport" y "Antyport". Estas reglas no son nuevas, ya que se han aplicado en otras variantes de Sistemas P. Ambas representan formas de transporte de sustancias que la biología molecular también utiliza.

El "Symport" y el "Antyport" se refieren al movimiento de moléculas o iones a través de una membrana. En el caso del "Symport", dos moléculas o iones se

mueven en la misma dirección. Por otro lado, el "Antyport" implica un movimiento similar. Ambos, los *simportadores* y los *antiportadores*, son proteínas transmembranales que forman un grupo conocido como *cotransportadores* (ver figura 3.2).

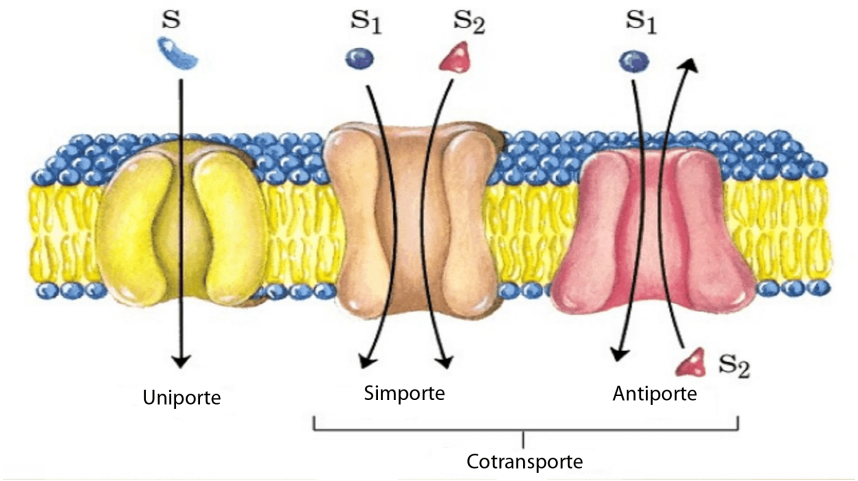

Figura 3.2: Tipos de transporte membranal

Las reglas "Symport" y "Antyport" tienen aplicaciones específicas. La regla "Symport" se refiere al movimiento de un plásmido de una región a otra. En las reglas "out symport", el destino sería la región circundante, mientras que en las reglas "in symport", el destino es la región especificada en la regla. Como resultado de la ejecución de esta regla, el plásmido se mueve de una región a otra, y los objetos tanto en la región de origen como en la de destino experimentan una evolución.

Las reglas "Antyport", por otro lado, realizan las mismas acciones que las reglas "Symport", pero con una adición: también mueven otro plásmido en la misma dirección transmembranal, pero en sentido contrario.

3.2.2. Formalización de Sistemas P con Plásmidos

Un modelo de computación de Sistemas P con plásmidos de grado n y q plásmidos, $m \geq 1$ y $q \geq 0$, se define mediante la tupla:

$$\Pi = (V, \Lambda, \mu, w_1, \ldots, w_n, (R_1, \rho_1), \ldots, (R_n, \rho_n), p_1, p_2, \ldots, p_q, i_0),$$

donde:

(i) V: es un alfabeto cuyos elementos son denominados objetos.

(iii) Λ: es un alfabeto de etiquetas de las membranas.

(ii) μ: es una estructura de membranas de grado n, con las membranas y regiones etiquetadas individualmente con elementos de Λ.
La estructura se puede representar como un arbol (estructura de datos en programación), en el que la raíz sería el entorno y los demás nodos serán las

membranas. De esta forma, los nodos hoja representan las membranas que no tienen ninguna membrana hija. es decir, membranas elementales (ver figura 2.1).

(iv) $w_i, 0 \leq i \leq n$: son cadenas de V^* que representan los multiconjuntos sobre V asociados con las regiones 1, 2, ..., n de μ. Si $i = 0$, entonces está asociado con el entorno.

(v) $(R_i, \rho_i), 0 \leq i \leq n$:

- R_i: son conjuntos finitos de *reglas de evolución* sobre objetos de V asociadas a las regiones 1, 2, ..., n.
- ρ_i: es una relación de orden parcial sobre R_i, que especifica una relación de prioridad entre las reglas de la región.

Dependiendo del tipo de regla, se pueden clasificar en los seis tipos de reglas (reescritura, transporte *out*, transporte *in*, 'in symport', 'out symport', 'anty-port'). Si $i = 0$, entonces está asociado con el entorno.

(vi) p_i, $1 \leq i \leq q$: son los plásmidos del sistema. Un plásmido equivale a una región sin objetos, solo contiene reglas de evolución, $p_i \equiv [_i R_i]_i$.

(vii) i_0: es un numero entre 1 y n que determina la membrana de salida de Π.

3.2.3. Proceso de Computación

La computación del modelo que incluye plásmidos no difiere significativamente del modelo tradicional sin plásmidos descrito en la sección 2.2.3. En este punto se discutirán las diferencias con la sección 2.2.3 y se destacarán los aspectos más importantes.

Se mantiene el principio de "máximo paralelismo", donde se aplican todas las reglas posibles en paralelo en todas las regiones. La primera diferencia es que, al seleccionar las reglas, se consideran tanto las reglas de la región como las reglas de los plásmidos presentes en esa región.

La segunda diferencia se presenta cuando dos reglas entran en conflicto por un objeto. En este caso, se sigue eligiendo de forma no determinista, sin importar si alguna de las reglas que compite es una regla contenida en un plásmido. Es importante mencionar que las reglas contenidas en los plásmidos no tienen preferencia o prioridad sobre las demás, no están marcadas por prioridades como pueden estarlo las reglas de la región; pueden ejecutarse en cualquier momento, excepto cuando compiten por un objeto. En ese caso, se elegirá una regla ganadora de forma no determinista, como se mencionó anteriormente.

Por último, este sistema asume una configuración inicial respecto a los plásmidos, donde todos los plásmidos inician en el entorno, es decir, fuera de la membrana que actúa como piel. En el modelo, el entorno está representado como la región 0. Solo puede haber un plásmido de cada tipo, lo que significa que cuando un plásmido se introduce en una región, abandona la anterior, llevándose consigo las reglas que contiene. Se podría modificar el modelo para permitir múltiples

copias del mismo plásmido. En ese caso, el modelo ampliado se definiría por la siguiente tupla:

$$\Pi = (V, \Lambda, \mu, w_1, \ldots, w_n, (R_1, \rho_1), \ldots, (R_n, \rho_n), (p_1, n_1), (p_2, n_2), \ldots, p_q, i_0),$$

siendo n_i el número de copias del plásmido p_i .

3.2.4. Ejemplos de Sistemas P con plásmidos

Al igual que hicimos al presentar los Sistemas P, veamos dos ejemplos de Sistemas P con plásmidos para comprender mejor su definición:

Ejemplo 1: Considerando un Sistema P de grado 4 y dos plásmidos

$$\begin{aligned}
\Pi_1 &= (V, \mu, w_0, w_1, w_2, w_3, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), P_1, P_2, 1),\\
V &= \{a, b, c, p, q\},\\
\mu &= [_1[_2]_2[_3]_3]_1,\\
w_0 &= \lambda, R_0 = \{P_1[_1p]_1 \to p[P_1], P_2[_1q]_1 \to q[P_2]\}\\
w_1 &= pq, R_1 = \{P_1[_2]_2 \to [_2P_1]_2, P_2[_3]_3 \to [_3P_2]_3, a \to a_{in_3}\}, \rho_1 = \emptyset,\\
w_2 &= a^n, R_2 = \emptyset, \rho_2 = \emptyset,\\
w_3 &= b^m, R_3 = \emptyset, \rho_3 = \emptyset,\\
P_1 &= \{a \to a_{out}\},\\
P_2 &= \{ab \to c_{out}\}
\end{aligned}$$

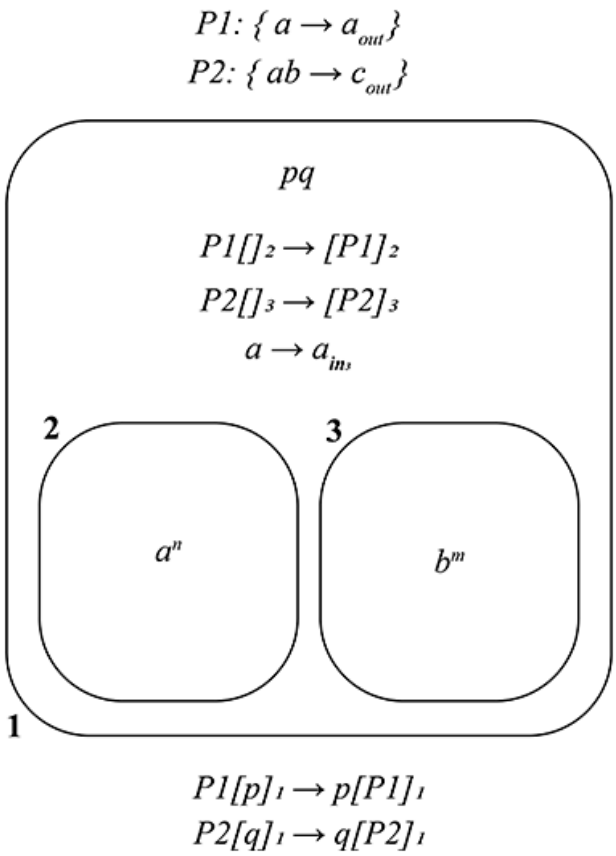

Figura 3.3: Diagrama Venn de un Sistema P que realiza la operación $m - n$

Este Sistema P realiza una resta aritmética de m (el número de objetos b en la membrana 3) menos n (el número de objetos a en la membrana 2). Esto se logra mediante la transferencia de los objetos a de la membrana 2 a la membrana 1, y posteriormente a la membrana 3, donde cada par ab es reemplazado por un objeto c que se extrae a la membrana 1. Al finalizar, el resultado se almacena en la membrana 1 como la cantidad de objetos c presentes. Los plásmidos permiten la operación dentro de las membranas 2 y 3 al aportar las reglas necesarias para ello.

Ejemplo 2: Considerando un Sistema P de grado 4 y dos plásmidos

$$
\begin{aligned}
&\Pi_2 = (V, \mu, w_0, w_1, w_2, w_3, (R_1, \rho_1), (R_2, \rho_2), (R_3, \rho_3), P_1, P_2, 1),\\
&V = \{a, b, p, x, q, r, t, s\},\\
&\mu = [_1[_2]_2[_3]_3]_1,\\
&w_0 = \lambda, R_0 = \{P_1[_1p]_1 \to [P_1x], P_2[_1x]_1 \to [P_2q]\}\\
&w_1 = p,\\
&R_1 = \{P_1q[_2a]_2 \to r[_2P_1a]_2, r[_2P_1]_2 \to P_1s[_2]_2, P_2s[_3]_3 \to t[_2P_2]_2, t[_2P_2]_2 \to P_2q[_2]_2\},\\
&\rho_1 = \emptyset,\\
&w_2 = ba^n, R_2 = \emptyset, \rho_2 = \emptyset,\\
&w_3 = ba^m, R_3 = \emptyset, \rho_3 = \emptyset,\\
&P_1 = \{ba \to b\},\\
&P_2 = \{a \to ab_{out}\}
\end{aligned}
$$

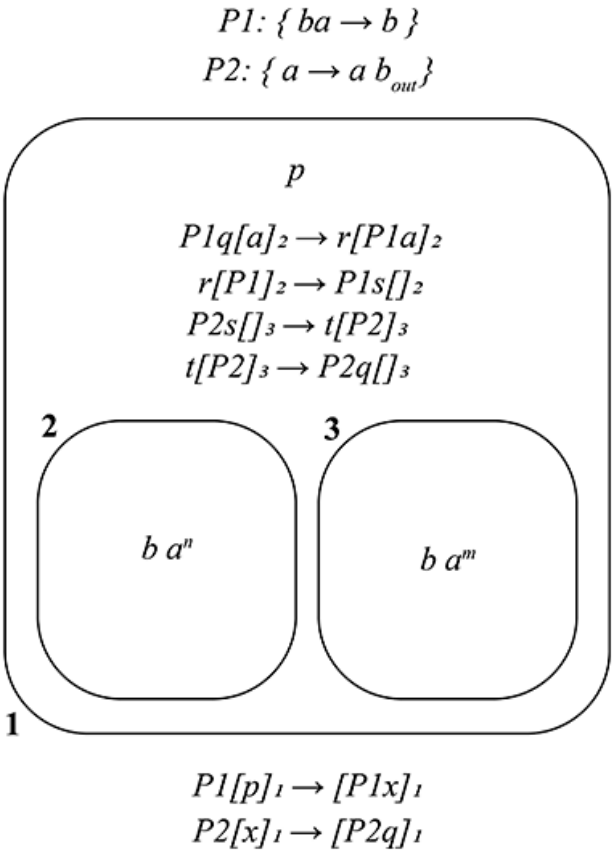

Figura 3.4: Diagrama Venn de un Sistema P que realiza la operación $m \cdot n$

Este Sistema P realiza la operación matemática de multiplicación de dos números, n (cantidad de objetos a en la membrana 2) y m (cantidad de objetos a en la membrana 3). El sistema utiliza los objetos a de la membrana 2 como contador. Para cada objeto a en la membrana 3, introduce m objetos b en la membrana 1. Este proceso se repite n veces, resultando en un total de $m \cdot n$ objetos b en la membrana 1.

CAPÍTULO 4
Equivalencia Máquinas de Registros y computación con Plásmidos

Es posible establecer una equivalencia entre cada una de las instrucciones de una Máquina de registros y los Sistemas P con plásmidos. Para cada instrucción, se ha diseñado un sistema P que, al finalizar la computación, genera un resultado equivalente al de la instrucción de la Máquina de registros.

Para ejecutar correctamente las instrucciones, se añaden objetos que actúan como controladores de las mismas. Estos objetos se representan con la letra "L" seguida de la posición de la instrucción a ejecutar. Por ejemplo, si se trata de la primera instrucción, será L_1; si es la décima instrucción, será L_{10}.

Los sistemas P equivalentes a las instrucciones explicadas en la sección 2.1 son:

- **suc(i)** : $[i] \leftarrow [i] + 1$

Este sistema P añade un objeto a a la membrana i.

$$\begin{aligned}\Pi_{suc} &= (V, \mu, w_0, \ldots, w_i, \ldots, w_n, (R_1, \rho_1), 1),\\ V &= \{L_x, L_{x+1}, a, t\},\\ \mu &= [_1[_i]_i]_1,\\ w_1 &= L_x t, R_1 = \{L_x t \rightarrow L_{x+1} t[_i a]_i\}, \rho_1 = \emptyset\end{aligned}$$

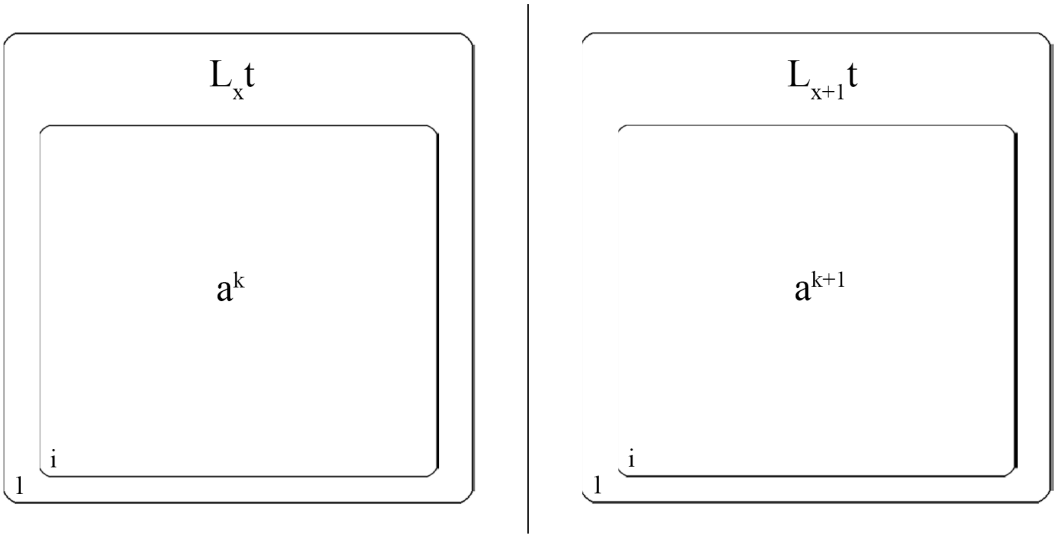

Figura 4.1: Estados inicial y final de un sistema P que realiza el sucesor de i

- **goto(k)** : ir a la instrucción k

  Este sistema P salta a la instrucción k.

$$\begin{aligned} \Pi_{goto} &= (V, \mu, w_0, \ldots, w_n, (R_1, \rho_1), 1), \\ V &= \{L_x, L_k\}, \\ \mu &= [_1]_1, \\ w_1 &= L_x, R_1 = \{L_x \to L_k\}, \rho_1 = \emptyset \end{aligned}$$

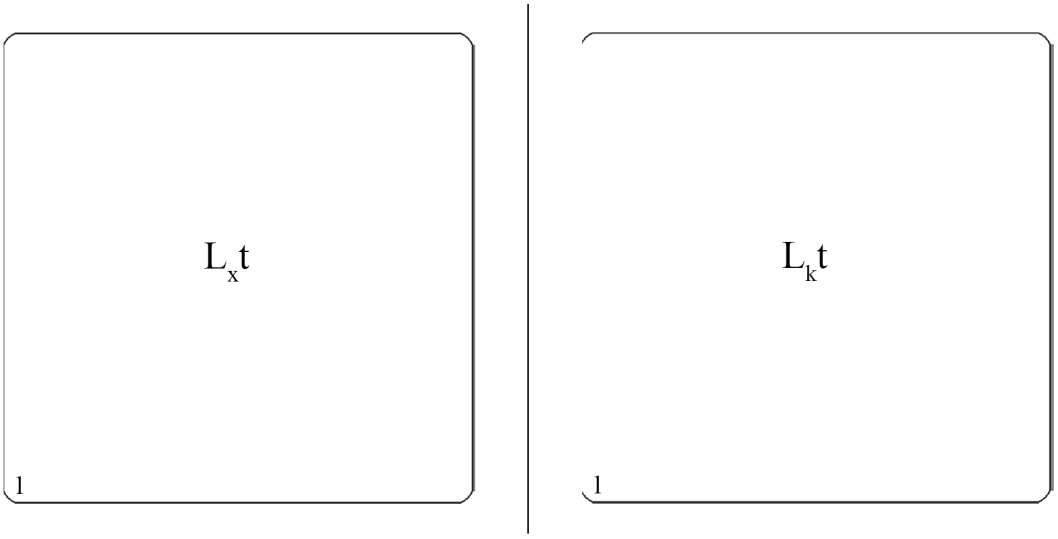

Figura 4.2: Estados inicial y final de un sistema P que realiza un salto a otra instrucción

- **pre(i, k)** : si $[i] > 0$ entonces $[i] \leftarrow [i] - 1$ sino $goto(k)$

Este sistema P quita un objeto a a la membrana i, en el caso de que no pueda quitar ningún objeto a porque ya no hay ninguno salta a la instrucción k.

$$\begin{aligned}
\Pi_{pre} &= (V, \mu, w_0, \ldots, w_i, \ldots, w_n, (R_1, \rho_1), 1), \\
V &= \{L_x, L_{x+1}, a, t\}, \\
\mu &= [_1[_i]_i]_1, \\
w_1 &= L_x t, \\
R_1 &= \{r_1 : L_x t[_i a]_i \rightarrow L_{x+1} t, r_2 : L_x t \rightarrow L_k t\}, \rho_1 = \{r_1 > r_2\}
\end{aligned}$$

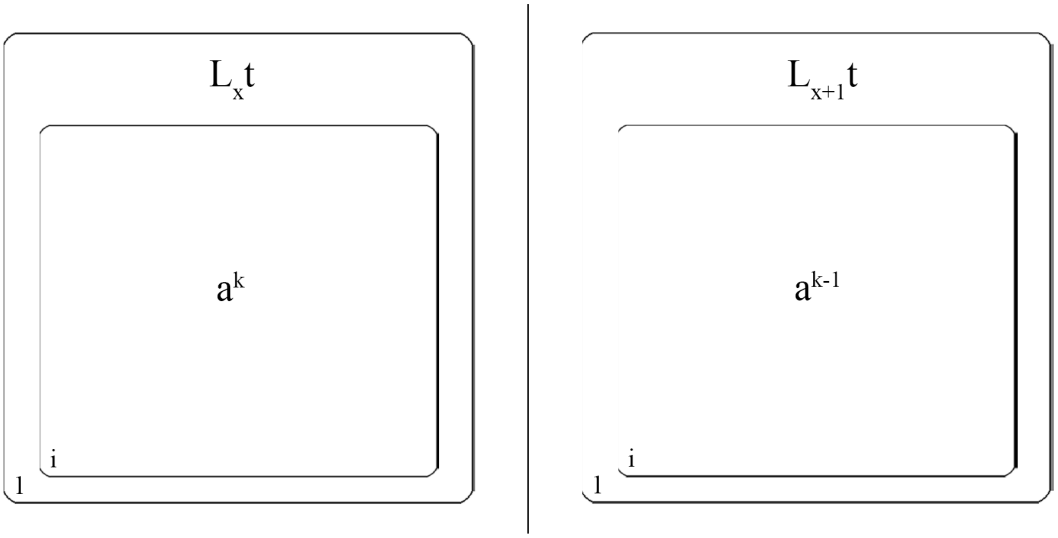

Figura 4.3: Estados inicial y final de un sistema P que realiza el predecesor en i

- **cer(i)** : $[i] \leftarrow 0$

  Este sistema P pone a cero la membrana i. Para ello, introduce el plásmisdo P_{cer} en la membrana 1, después se introduce P_{cer} en el registro membrana i y elimina sus objetos a.

  - P_{cer}: Se encarga de eliminar los objetos a de la membrana en el que se introduce.

$$\begin{aligned}
\Pi_{cer} &= (V, \mu, w_0, \ldots, w_i, \ldots, w_n, (R_0, \rho_0), (R_1, \rho_1), P_{cer}, 1), \\
V &= \{L_x, L_{x+1}, a, h, t\}, \\
\mu &= [_1[_i]_i]_1, \\
w_0 &= \lambda, R_0 = \{P_{cer}[_1 L_x t]_1 \rightarrow [_1 P_{cer} L_x t]_1, [_1 P_{cer} L_{x+1} h]_1 \rightarrow P_{cer}[_1 L_{x+1} t]_1\}, \rho_0 = \emptyset, \\
w_1 &= L_x t, \\
R_1 &= \{P_{cer} L_x t \rightarrow L_x[_i P_{cer}]_i, L_x[_i P_{cer}]_i \rightarrow P_{cer} L_{x+1} h\}, \rho_1 = \emptyset, \\
P_{cer} &= \{a \rightarrow \lambda\}
\end{aligned}$$

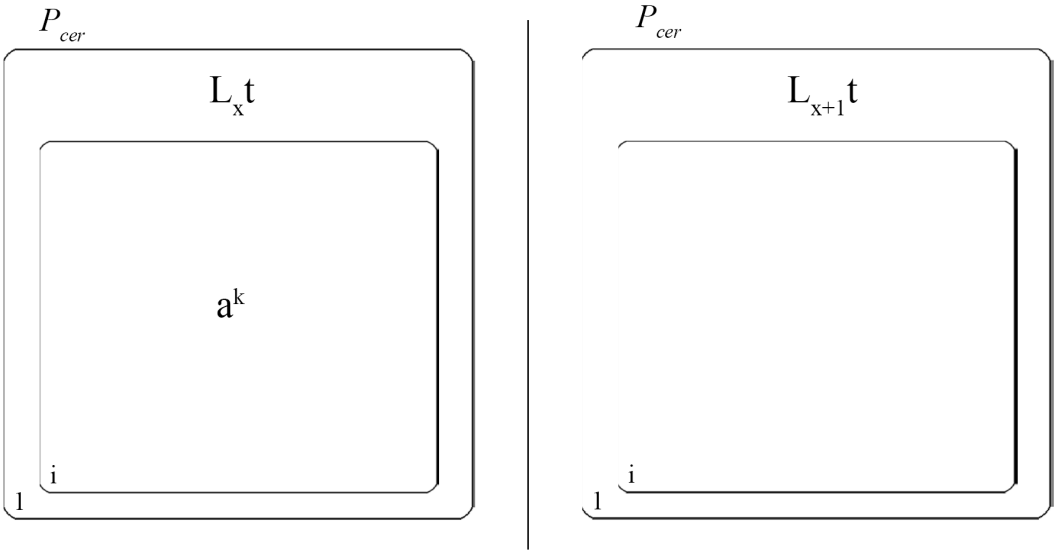

Figura 4.4: Estados inicial y final de un sistema P con plásmidos que pone *i* a cero

- **asi(k,i)** : $[i] \leftarrow k$

  Este sistema P asigna a la membrana i una cantidad k de objetos a. Para ello utiliza el plásmido P_{asi_k}, que entra en la membrana 1, luego en la membrana i y termina aplicando su regla.

  - P_{asi_k}: Se encarga de introducir en el registro membrana 2 k objetos a.

$$
\begin{aligned}
\Pi_{asi_k} &= (V, \mu, w_0, \ldots, w_i, \ldots, w_n, (R_0, \rho_0), (R_1, \rho_1), P_{asi_k}, 1),\\
V &= \{L_x, L_{x+1}, a, h, r, s, t\},\\
\mu &= [_1[_i]_i]_1,\\
w_0 &= \lambda, R_0 = \{P_{asi_k}[_1 L_x t]_1 \rightarrow [_1 P_{asi_k} L_x t]_1, [_1 P_{asi_k} L_{x+1} h]_1 \rightarrow P_{asi_k}[_1 L_{x+1} t]_1\},\\
\rho_0 &= \emptyset,\\
w_1 &= L_x t,\\
R_1 &= \{P_{asi_k} L_x t \rightarrow L_x [_i P_{asi_k} r]_i, L_x [_i P_{asi_k} s]_i \rightarrow P_{asi_k} L_{x+1} h\}, \rho_1 = \emptyset,\\
P_{asi_k} &= \{r \rightarrow s a^k\}
\end{aligned}
$$

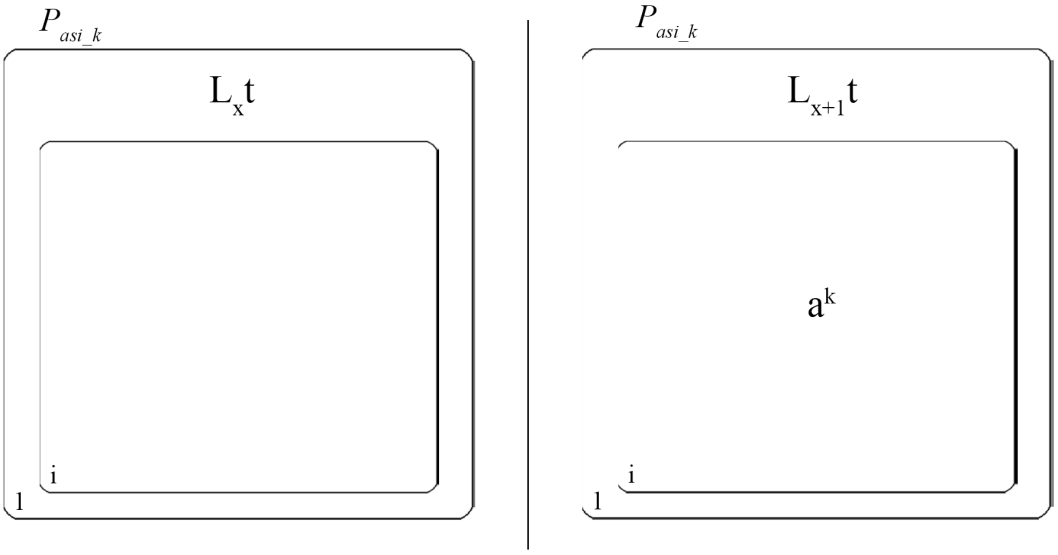

Figura 4.5: Estados inicial y final de un sistema P con plásmidos que asigna k objetos a i

- **cop(j,i)** : $[i] \leftarrow [j]$

  Este sistema P copia el contenido de la membrana j en el registro membrana i. Para ello utiliza el plásmido P_{cop}, que se introduce primero en la membrana 1 y después en la membrana j, entonces saca a la membrana 1 tantos objetos auxiliares x como objetos a se encuentran en la membrana j. Después, introduce en la membrana i tantos objetos a como objetos x se encuentra en la membrana 1.

  - P_{cop}: Se encarga de sacar a la membrana padre tantos objetos auxiliares x como objetos a se encuentran en la membrana registro en la que se encuentre.

$$
\begin{aligned}
\Pi_{cop} &= (V, \mu, w_0, \ldots, w_i, \ldots, w_j, \ldots, w_n, (R_0, \rho_0), (R_1, \rho_1), P_{cop}, 1),\\
V &= \{L_x, L_{x+1}, a, h, r, s, t, u\},\\
\mu &= [_1[_i]_i[_j]_j]_1,\\
w_0 &= \lambda, R_0 = \{P_{cop}[_1 L_x t]_1 \rightarrow [_1 P_{cop} L_x t]_1, [_1 P_{cop} L_{x+1} h]_1 \rightarrow P_{cop}[_1 L_{x+1} t]_1\},\\
\rho_0 &= \emptyset,\\
w_1 &= L_x t,\\
R_1 &= \begin{array}{l}\{r_1 : P_{cop} L_x t \rightarrow L_x t[_j P_{cop} r]_j,\\ \quad r_2 : L_x t[_j P_{cop} s]_j \rightarrow P_{cop} L_{x+1} u,\\ \quad r_3 : L_x r \rightarrow L_x s[_i a]_i,\\ \quad r_4 : L_x ss \rightarrow L_x s,\\ \quad r_5 : L_x us \rightarrow L_{x+1} h,\\ \quad r_6 : L_x u \rightarrow L_{x+1} h\},\end{array}\\
\rho_1 &= (r_1, r_2, r_3, r_4 > r_5), (r_1, r_2, r_3, r_4, r_5 > r_6),\\
P_{cop} &= \{a \rightarrow a, r_{out}\}
\end{aligned}
$$

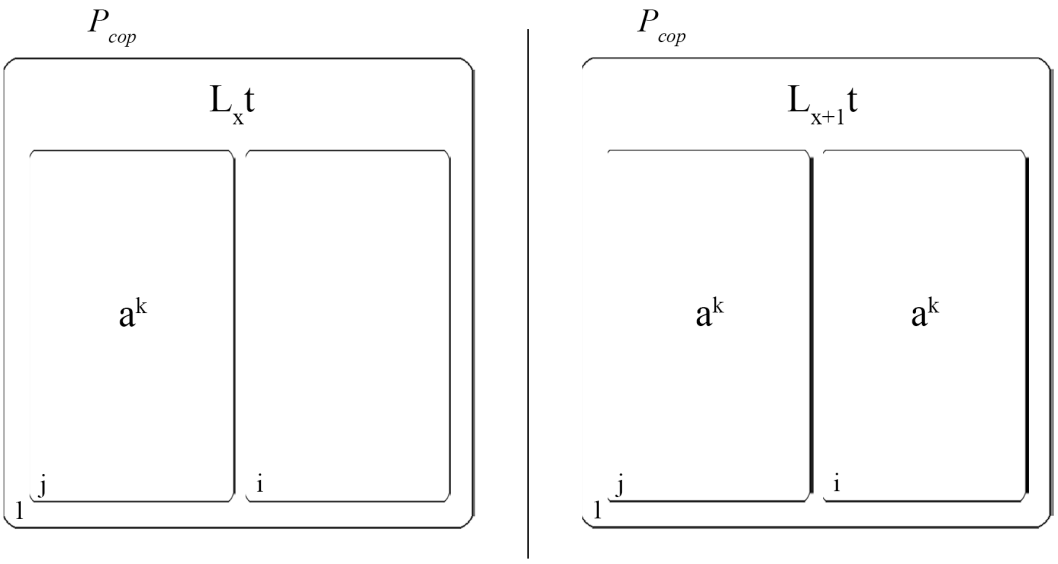

Figura 4.6: Estados inicial y final de un sistema P con plásmidos que copia *j* en *i*

- **sum(p,q,m)** : $[m] \leftarrow [p] + [q]$

  Este sistema P realiza la suma del contenido de las membranas p y q y la almacena en la membrana m. Para ello utiliza el plásmido P_{sum}, que se introduce primero en la membrana 1 y después en la membrana p, entonces saca a la membrana 1 tantos objetos auxiliares x como objetos a se encuentra en la membrana p. Después, el plásmido P_{sum} se introduce en la membrana q y realiza lo mismo que en la membrana p. Tras tener la suma en la membrana 1 introduce los objetos x como objetos a en la membrana m.

  - P_{sum}: Se encarga de saca a la membrana padre tantos objetos auxiliares x como objetos a se encuentra en la membrana registro en la que se encuentra.

$$
\begin{aligned}
\Pi_{sum} &= (V, \mu, w_0, \ldots, w_p, \ldots, w_q, \ldots, w_m, \ldots, w_n, (R_0, \rho_0), (R_1, \rho_1), P_{cop}, 1),\\
V &= \{L_x, L_{x+1}, a, h, r, s, t, x\},\\
\mu &= [_1[_p]_p[_q]_q[_m]_m]_1,\\
w_0 &= \lambda, R_0 = \{P_{sum}[_1L_xt]_1 \rightarrow [_1P_{sum}L_xt]_1, [_1P_{sum}L_{x+1}h]_1 \rightarrow P_{sum}[_1L_{x+1}t]_1\},\\
\rho_0 &= \emptyset,\\
w_1 &= L_xt,\\
R_1 &= \begin{array}{l}\{r_1 : P_{sum}L_xt \rightarrow L_xt[_pP_{sum}r]_p,\\ \quad r_2 : L_xt[_pP_{sum}]_p \rightarrow P_{sum}L_xs,\\ \quad r_3 : P_{sum}L_xs \rightarrow L_xs[_qP_{sum}]_q,\\ \quad r_4 : L_xs[_qP_{sum}]_q \rightarrow P_{sum}L_xr,\\ \quad r_5 : L_xrx \rightarrow L_xr[_ma]_m,\\ \quad r_6 : L_xr \rightarrow L_{x+1}h\},\end{array}\\
\rho_1 &= \{r_1, r_2, r_3, r_4, r_5 > r_6\},\\
P_{sum} &= \{a \rightarrow a, x_{out}\}
\end{aligned}
$$

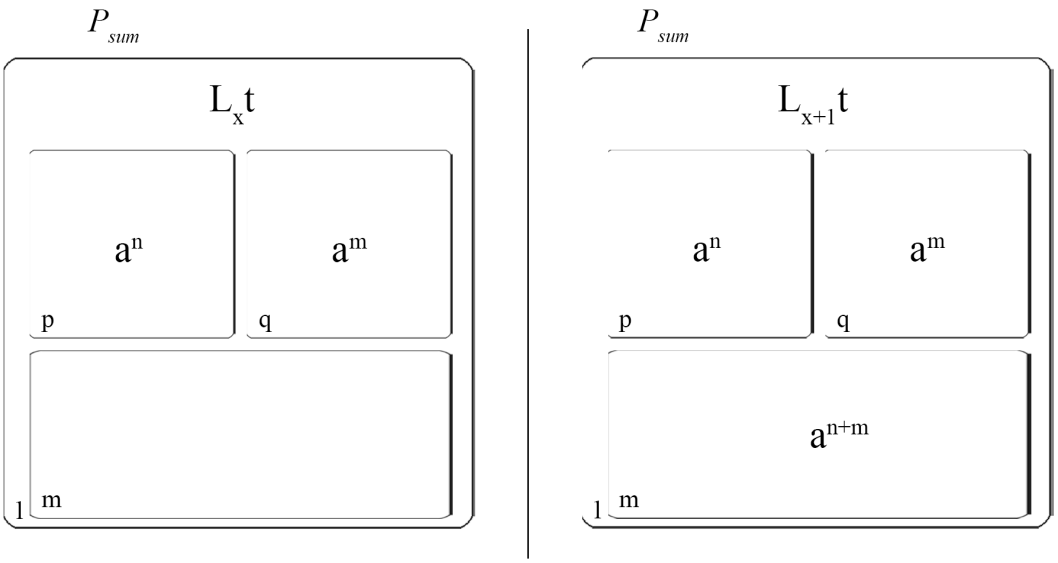

Figura 4.7: Estados inicial y final de un sistema P con plásmidos que suma p y q y almacena el resultado en m

- **mul(p,q,m)** : $[m] \leftarrow [p] \cdot [q]$

  Este sistema P realiza la multiplicación de las membranas p y q y la almacena en la membrana m. Para ello utiliza los plásmidos P_{mul_1}, P_{mul_2} y P_{mul_3}, se introducen en la membrana 1 y después, el plásmido P_{mul_1} se introduce en la membrana p aplica su regla y sale, tras eso, introduce los objetos obtenidos por la aplicación de la regla en la membrana auxiliar k. Después intercala la introducción del plásmido P_{mul_2} en la membrana auxiliar k y el plásmido P_{mul_3} en la membrana q. Los objetos que irá sacando el plásmido P_{mul_3} se introducirán en la membrana m. La aplicación de la regla del plásmido P_{mul_2} en la membrana auxiliar k se utiliza como contador para hacer tantas sumas objetos a tenga la membrana p. Tras estos pasos hace una limpieza de los objetos residuales en la membrana 1.

  - P_{mul_1}: Se encarga de sacar a la membrana padre tantos objetos auxiliares x como objetos a se encuentran en la membrana registro en la que se encuentre.
  - P_{mul_2}: Se encarga de bajar en 1 el contador de la membrana en la que se encuentre.
  - P_{mul_3}: Se encarga de sacar a la membrana padre tantos objetos auxiliares y como objetos a se encuentran en la membrana registro en la que se encuentre.

$$
\begin{aligned}
\Pi_{mul} =& (V, \mu, w_0, \ldots, w_p, \ldots, w_q, \ldots, w_m, \ldots, w_k, \ldots, w_n, (R_0, \rho_0), (R_1, \rho_1), \\
& P_{mul_1}, P_{mul_2}, P_{mul_3}, 1), \\
V =& \{L_x, L_{x+1}, P_1, P_2, P_3, P_4, a, h, o, p, q, r, s, t, u, y, x\}, \\
\mu =& [_1[_p]_p[_q]_q[_m]_m[_k]_k]_1, \\
w_0 =& \lambda, \\
R_0 =& \begin{array}{l} \{r_1 : P_{mul_1}[_1 L_x t]_1 \to [_1 P_{mul_1} L_x P_1]_1, \\ \ r_2 : P_{mul_2}[_1 L_x P_1]_1 \to [_1 P_{mul_2} L_x P_2]_1, \\ \ r_3 : P_{mul_3}[_1 L_x P_2]_1 \to [_1 P_{mul_3} L_x t]_1, \\ \ r_4 : [_1 P_{mul_1}, L_{x+1} P_3]_1 \to P_{mul_1}[_1 L_{x+1} P_4]_1, \\ \ r_5 : [_1 P_{mul_2}, L_{x+1} P_4]_1 \to P_{mul_2}[_1 L_{x+1} h]_1, \\ \ r_6 : [_1 P_{mul_1}, L_{x+1} h]_1 \to P_{mul_1}[_1 L_{x+1} t]_1\}, \end{array} \\
\rho_0 =& \emptyset, \\
w_1 =& L_x t, \\
R_1 =& \begin{array}{l} \{r_1 : P_{mul_1} L_x t \to L_x p[_p P_{mul_1}]_p, \\ \ r_2 : L_x p[_p P_{mul_1}]_p \to P_{mul_1} L_x q, \\ \ r_3 : P_{mul_2} L_x q[_k a]_k \to L_x r[_k P_{mul_2} a]_k, \\ \ r_4 : L_x r[_k P_{mul_2}]_k \to P_{mul_2} L_x s, \\ \ r_5 : P_{mul_3} L_x s \to L_x o[_q P_{mul_3}]_q, \\ \ r_6 : L_x o[_q P_{mul_3}]_q \to P_{mul_3} L_x q, \\ \ r_7 : L_x x \to L_x u[_k a]_k, \\ \ r_8 : L_x y \to L_x u[_m a]_m, \\ \ r_9 : L_x uu \to L_x u, \\ \ r_{10} : L_x qu \to L_{x+1} P_3, \\ \ r_{11} : P_{mul_2} L_x q \to P_{mul_2} L_x qu\}, \end{array} \\
\rho_1 =& \begin{array}{l} \{(r_1, r_2, r_3, r_4, r_5, r_6, r_7 > r_8), \\ \ (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8 > r_9), \\ \ (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9 > r_{10}),, \\ \ (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9 > r_{11}), \\ \ (r_6 > r_2)\}, \end{array} \\
w_k =& \{b\} P_{mul_1} = \qquad \{a \to a, x_{out}\}, \\
P_{mul_2} =& \{ba \to b\}, \\
P_{mul_3} =& \{a \to a, y_{out}\},
\end{aligned}
$$

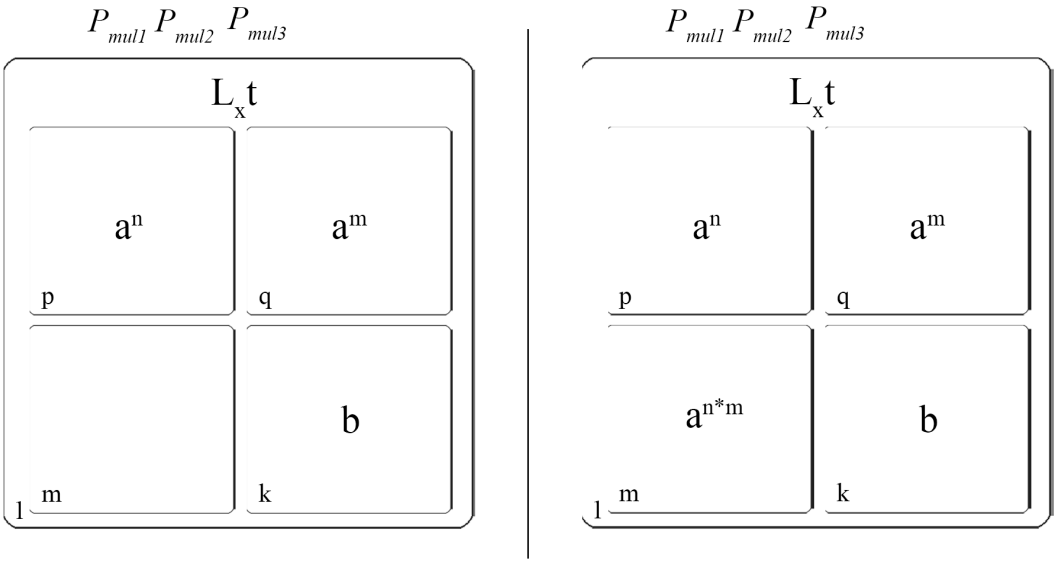

Figura 4.8: Estados inicial y final de un sistema P con plásmidos que multiplica p y q y almacena el resultado en m

- **div(p,q,m)** : $[m] \leftarrow [p]/[q]$

  Este sistema P divide la membrana p entre la membrana q y almacena el resultado en m, en el caso de resultado no entero redondea hacia arriba. Para ello utiliza los plásmidos P_{div_1} y P_{div_2}, los cuales se introducen en la membrana 1. Luego, el plásmido P_{div_1} se introduce en la membrana p, aplica la regla y sale, luego el P_{div_2} se va introduciendo en la membrana q, aplica su regla y sale. Por cada vez que se sale el plásmido P_{div_2} de la membrana q se introduce un objeto a en la membrana m. Tras estos pasos hace una limpieza de los objetos residuales en la membrana 1.

  - P_{div_1}: Se encarga de sacar a la membrana padre tantos objetos auxiliares x como objetos a se encuentran en la membrana registro en la que se encuentre.

  - P_{div_2}: Se encarga de sacar a la membrana padre tantos objetos auxiliares y como objetos a se encuentran en la membrana registro en la que se encuentre.

$$
\begin{aligned}
\Pi_{div} &= (V, \mu, w_0, \ldots, w_p, \ldots, w_q, \ldots, w_m, \ldots, w_n, (R_0, \rho_0), (R_1, \rho_1), P_{div_1}, P_{div_2}, 1),\\
V &= \{L_x, L_{x+1}, L_{err}, P_1, P_2, a, b, h, r, s, t, y, x\},\\
\mu &= [_1[_p]_p[_q]_q[_m]_m]_1,\\
w_0 &= \lambda,\\
R_0 &= \begin{array}{l}\{r_1 : P_{div_1}[_1 L_x t]_1 \to [_1 P_{div_1} L_x P_1]_1,\\ \ r_2 : P_{div_2}[_1 L_x P_1]_1 \to [_1 P_{div_2} L_x t]_1,\\ \ r_3 : [_1 P_{div_1}, L_{x+1} P_2]_1 \to P_{div_1}[_1 L_{x+1} h]_1,\\ \ r_4 : [_1 P_{div_2}, L_{x+1} h]_1 \to P_{div_2}[_1 L_{x+1} t]_1\},\end{array}\\
\rho_0 &= \emptyset,\\
w_1 &= L_x t,\\
R_1 &= \begin{array}{l}\{r_1 : P_{div_1} L_x t[_p a]_p \to L_x t[_p P_{div_1} a]_p,\\ \ r_2 : L_x t[_p P_{div_1}]_p \to P_{div_1} L_x s,\\ \ r_3 : P_{div_2} L_x s[_q a]_q \to L_x s b[_q P_{div_2} a]_q,\\ \ r_4 : L_x s[_q P_{div_2}]_q \to P_{div_2} L_x r,\\ \ r_5 : L_x r b x y \to L_x s, a_{in_m},\\ \ r_6 : L_x x y \to L_x b,\\ \ r_7 : L_x s x \to L_{err},\\ \ r_8 : L_x b \to L_x,\\ \ r_9 : L_x y \to L_x,\\ \ r_{10} : L_x r \to L_{x+1} P_2,\\ \ r_{11} : P_{div_1} L_x t \to P_{div_1} L_{x+1} P_2\},\end{array}\\
\rho_1 &= \begin{array}{l}\{(r_1, r_2, r_3, r_4, r_5, r_6, r_7 > r_8),\\ \ (r_1, r_2, r_3, r_4, r_5, r_6, r_7 > r_9),\\ \ (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9 > r_{10}),\\ \ (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{10} > r_{11}),\\ \ (r_3, r_4, r_6, > r_7),\\ \ (r_6 > r_3),\\ \ (r_5 > r_6)\},\end{array}\\
P_{div_1} &= \{a \to a, x_{out}\}\\
P_{div_2} &= \{a \to a, y_{out}\}
\end{aligned}
$$

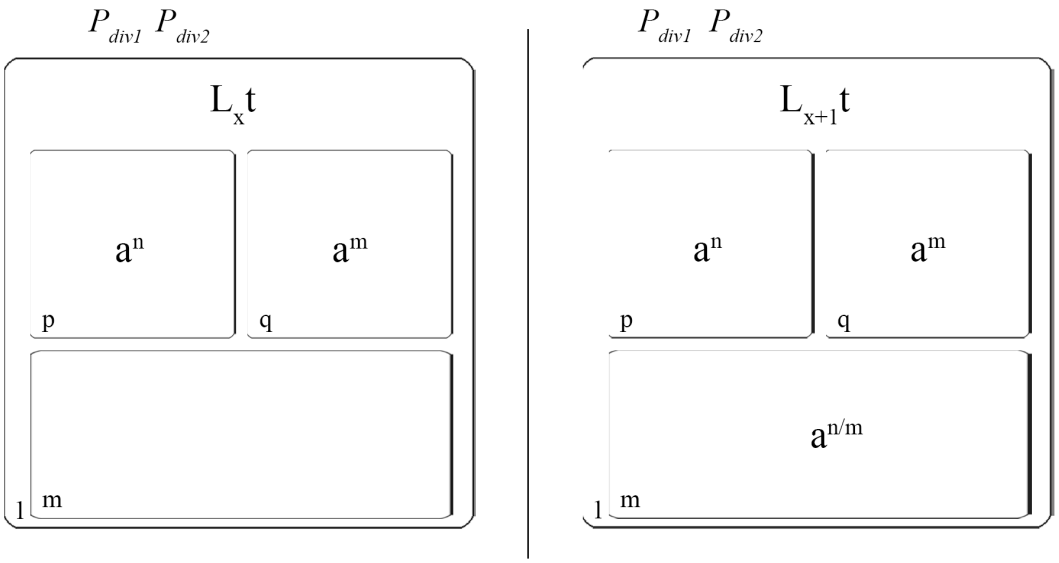

Figura 4.9: Estados inicial y final de un sistema P con plásmidos que divide p y q y almacena el resultado en m

- **mei(p,q,m1,m2)** : si $[p] \leq [q]$ entonces $goto(m1)$ sino $goto(m2)$

  Este sistema P comprueba si el contenido de la membrana registro p es menor o igual que el de la membrana q, en el caso de que sea afirmativo salta a la instrucción L_{m1}, es decir sustituye el objeto L_x por L_{m1}, en caso negativo, salta a la instrucción L_{m2}. Para ello utiliza los plásmidos P_{mei_1} y P_{mei_2}, los cuales se introducen en la membrana 1, luego se introducen en las membrana p y q respectivamente. Estos plásmidos sacan tantos objetos auxiliares, x o y, como objetos a se encuentren en los registros. Luego por cada par de objetos xy los elimina, en el caso de que no queden objetos o solo queden objetos y significará que es afirmativo, en caso de que solo queden objetos x significará que es negativo. Tras estos pasos hace una limpieza de los objetos residuales en la membrana 1.

  - P_{mei_1}: Se encarga de sacar a la membrana padre tantos objetos auxiliares x como objetos a se encuentran en la membrana registro en la que se encuentre.

  - P_{mei_2}: Se encarga de sacar a la membrana padre tantos objetos auxiliares y como objetos a se encuentran en la membrana registro en la que se encuentre.

$$
\begin{aligned}
\Pi_{mei} &= (V, \mu, w_0, \ldots, w_p, \ldots, w_q, \ldots, w_n, (R_0, \rho_0), (R_1, \rho_1), P_{mei_1}, P_{mei_2}, 1),\\
V &= \{L_x, L_{m1}, L_{m2}, P_1, P_2, P_3, b, h, s, t, y, x\},\\
\mu &= [_1[_p]_p[_q]_q]_1,\\
w_0 &= \lambda,\\
R_0 &= \begin{array}{l}\{r_1 : P_{mei_1}[_1L_xt]_1 \to [_1P_{mei_1}L_xP_1]_1,\\ \ r_2 : P_{mei_2}[_1L_xP_1]1 \to [_1P_{mei_2}, L_xt]_1,\\ \ r_3 : [_1P_{mei_1}L_{m1}P_3]_1 \to P_{mei_1}[_1L_{m1}h]_1,\\ \ r_4 : [P_{mei_1}L_{m2}P_3]_1 \to P_{mei_1}[_1L_{m2}h]_1,\\ \ r_5 : [P_{mei_2}L_{m1}h]1 \to P_{mei_2}[_1L_{m1}t]_1,\\ \ r_6 : [_1P_{mei_2}L_{m2}h]_1 \to P_{mei_2}[_1L_{m2}t]_1\},\end{array}\\
\rho_0 &= \emptyset,\\
w_1 &= L_xt,\\
R_1 &= \begin{array}{l}\{r_1 : P_{mei_1}L_xt \to L_xt[_pP_{mei_1}]_p,\\ \ r_2 : L_xt[_pP_{mei_1}]_p \to P_{mei_1}L_xs,\\ \ r_3 : P_{mei_2}L_xs \to L_xs[_qP_{mei_2}]_q,\\ \ r_4 : L_xs[_qP_{mei_2}]_q \to P_{mei_2}L_xb,\\ \ r_5 : L_xbxy \to L_xb,\\ \ r_6 : L_xbxx \to L_xbx,\\ \ r_7 : L_xbyy \to L_xby,\\ \ r_8 : L_xbx \to L_{m2}P_3,\\ \ r_9 : L_xby \to L_{m1}P_3,\\ r_{10} : L_xb \to L_{m1}P_3\},\end{array}\\
\rho_1 &= \begin{array}{l}\{(r_5 > r_6, r_7, r_8, r_9, r_{10}),\\ \ (r_6 > r_8),\\ \ (r_7 > r_9),\\ \ (r_8 > r_{10}),\\ \ (r_9 > r_{10})\},\end{array}\\
P_{mei_1} &= \{a \to a, x_{out}\}\\
P_{mei_2} &= \{a \to a, y_{out}\}
\end{aligned}
$$

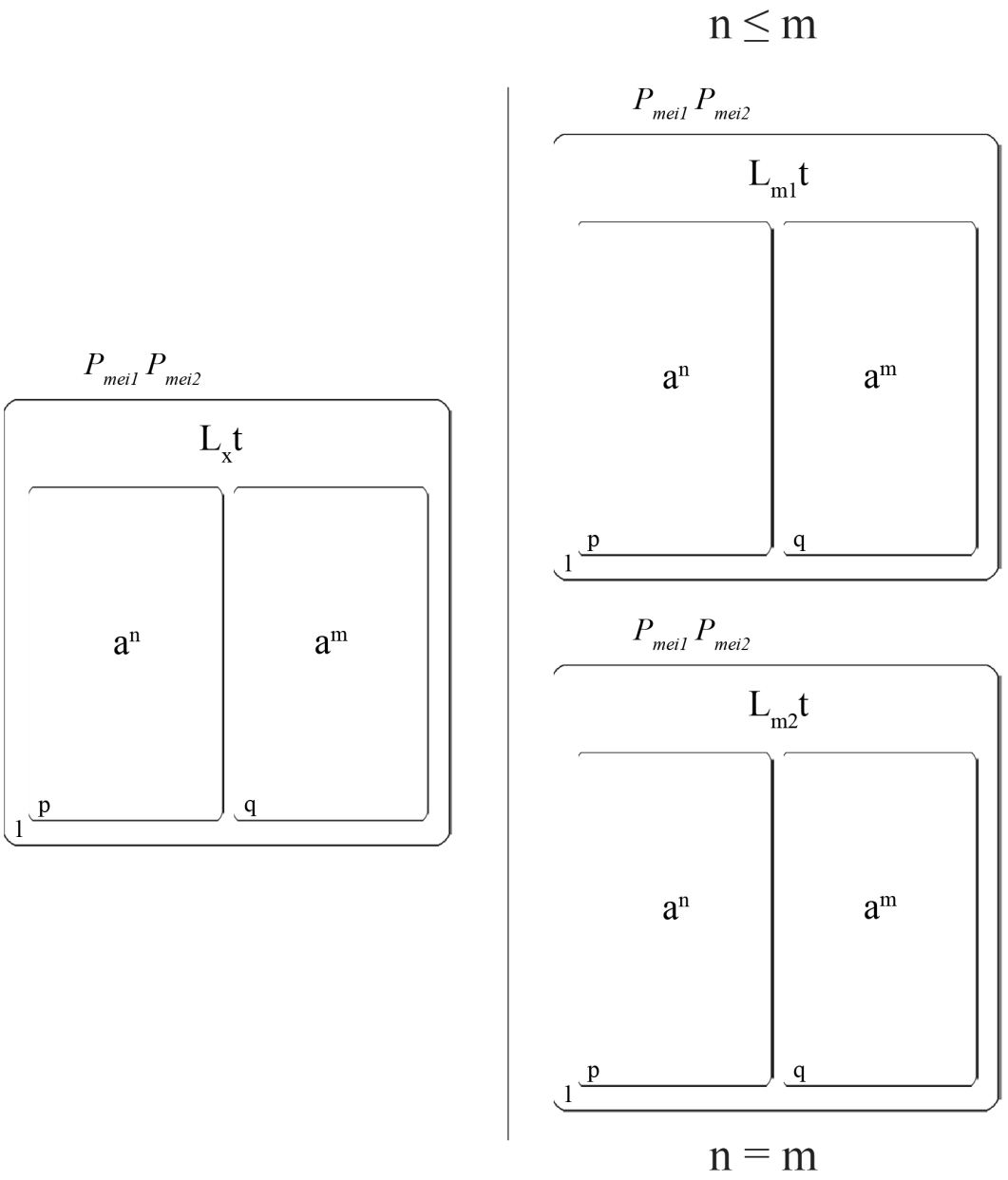

Figura 4.10: Estados inicial y final de un sistema P con plásmidos que si p es menor o igual que q salta a la instrucción $m1$, en caso contrario a $m2$

- **ig(p,q,m1,m2)** : si $[p] = [q]$ entonces $goto(m1)$ sino $goto(m2)$

  Este sistema P comprueba si el contenido de la membrana p es igual que el de la membrana q, es muy similar que el anterior (menor o igual), con la única diferencia que si queda algún objeto x o y en la membrana 1, será negativo y, en caso contrario, afirmativo.

$$
\begin{aligned}
\Pi_{ig} &= (V, \mu, w_0, \ldots, w_p, \ldots, w_q, \ldots, w_n, (R_0, \rho_0), (R_1, \rho_1), P_{ig_1}, P_{ig_2}, 1),\\
V &= \{L_x, L_{m1}, L_{m2}, P_1, P_2, P_3, b, h, s, t, y, x\},\\
\mu &= [_1[_p]_p[_q]_q]_1,\\
w_0 &= \lambda,\\
R_0 &= \begin{array}{l}\{r_1 : P_{ig_1}[_1L_xt]_1 \to [_1P_{ig_1}L_xP_1]_1,\\ \quad r_2 : P_{ig_2}[_1L_xP_1]1 \to [_1P_{ig_2}, L_xt]_1,\\ \quad r_3 : [_1P_{ig_1}L_{m1}P_3]_1 \to P_{ig_1}[_1L_{m1}h]_1,\\ \quad r_4 : [P_{ig_1}L_{m2}P_3]_1 \to P_{ig_1}[_1L_{m2}h]_1,\\ \quad r_5 : [P_{ig_2}L_{m1}h]1 \to P_{ig_2}[_1L_{m1}t]_1,\\ \quad r_6 : [_1P_{ig_2}Lm2h]_1 \to P_{ig_2}[_1L_{m2}t]_1\},\end{array}\\
\rho_0 &= \emptyset,\\
w_1 &= L_xt,\\
R_1 &= \begin{array}{l}\{r_1 : P_{ig_1}L_xt \to L_xt[_pP_{ig_1}]_p,\\ \quad r_2 : L_xt[_pP_{ig_1}]_p \to P_{ig_1}L_xs,\\ \quad r_3 : P_{ig_2}L_xs \to L_xs[_qP_{ig_2}]_q,\\ \quad r_4 : L_xs[_qP_{ig_2}]_q \to P_{ig_2}L_xb,\\ \quad r_5 : L_xbxy \to L_xb,\\ \quad r_6 : L_xbxx \to L_xbx,\\ \quad r_7 : L_xbyy \to L_xby,\\ \quad r_8 : L_xbx \to L_{m2}P_3,\\ \quad r_9 : L_xby \to L_{m2}P_3,\\ \quad r_{10} : L_xb \to L_{m1}P_3\},\end{array}\\
\rho_1 &= \begin{array}{l}\{(r_5 > r_6, r_7, r_8, r_9, r_{10}),\\ \quad (r_6 > r_8),\\ \quad (r_7 > r_9),\\ \quad (r_8 > r_{10}),\\ \quad (r_9 > r_{10})\},\end{array}\\
P_{ig_1} &= \{a \to a, x_{out}\}\\
P_{ig_2} &= \{a \to a, y_{out}\}
\end{aligned}
$$

CAPÍTULO 5
Implementación

A continuación, se explicará el desarrollo de la implementación del simulador de modelos computacionales mediante computación con plásmidos. Para ello, primero se describirán las tecnologías empleadas durante el desarrollo, incluyendo el lenguaje de programación utilizado y el entorno de desarrollo seleccionado.

En segundo lugar, se describirán las librerías que han sido necesarias crear para simular la computación de los Sistemas P con plásmidos.

Finalmente, se presentará el compilador diseñado para interpretar el código de las máquinas de registros y transformarlo en un Sistema P capaz de ejecutar la operación indicada por dicho código.

5.1 Tecnologías utilizadas

5.1.1. Lenguaje de desarrollo: Python

Python se elige como el lenguaje principal para el desarrollo de las librerías en este Trabajo de Fin de Grado debido a varias razones que abarcan tanto sus características técnicas como su amplia adopción en la comunidad científica y tecnológica.

Ventajas de Python

- **Simplicidad y Legibilidad del Código**:
  - **Sintaxis Clara**: La sintaxis de Python es simple y directa, lo que facilita la escritura y comprensión del código. Esto reduce el tiempo de desarrollo y minimiza errores.
  - **Lenguaje Interpretado**: Python es interpretado, lo que permite una rápida iteración en el desarrollo, haciendo más fácil probar y depurar el código.
- **Amplia Biblioteca Estándar**:

  - **Bibliotecas Incorporadas**: Python cuenta con una extensa biblioteca estándar que cubre muchas funcionalidades básicas, desde manipulación de cadenas hasta interacción con el sistema operativo.

- **Comunidad Activa y Soporte**:
  - **Documentación Abundante**: La vasta cantidad de documentación y recursos educativos disponibles facilita el aprendizaje y solución de problemas.
  - **Soporte Comunitario**: La comunidad de Python es muy activa, ofreciendo soporte a través de foros, grupos de discusión y conferencias.

- **Multiplataforma**:
  - **Compatibilidad**: Python es compatible con la mayoría de sistemas operativos, incluyendo Windows, macOS y Linux, lo que garantiza que las librerías desarrolladas funcionen en diferentes entornos.

Desventajas de Python

- **Rendimiento**:
  - **Velocidad de Ejecución**: Al ser un lenguaje interpretado, Python es generalmente más lento en comparación con lenguajes compilados como C++ o Java. Esto puede ser un inconveniente para aplicaciones que requieren alto rendimiento.
  - **Gestión de Memoria**: Python maneja la memoria de forma automática (*garbage collection*), lo que a veces puede llevar a problemas de eficiencia en aplicaciones de gran escala.

- **Limitaciones en Computación de Bajo Nivel**:
  - **Control de Hardware**: Python no es ideal para tareas que requieren control de hardware a bajo nivel o tiempo real, ya que no permite un manejo tan directo como otros lenguajes de programación de sistemas.

- **Dependencias de Terceros**:
  - **Compatibilidad de Paquetes**: La gran cantidad de bibliotecas de terceros puede ocasionar problemas de compatibilidad y dependencia, especialmente cuando las versiones de las librerías no están alineadas.

Conclusión

A pesar de las desventajas mencionadas, como el control limitado de hardware a bajo nivel o la posible incompatibilidad de paquetes, estas no afectan al desarrollo de este proyecto. La naturaleza del TFG no requiere optimización extrema de rendimiento ni interacción directa con hardware especializado. Por ello, las ventajas de Python, como su facilidad de uso, la vasta cantidad de recursos disponibles y robusto ecosistema.

La elección de Python permite un desarrollo ágil y efectivo, superando significativamente los aspectos negativos en términos de rendimiento, especialmente en un entorno académico y de desarrollo rápido. Python se presenta, por tanto, como una herramienta ideal para cumplir con los objetivos planteados.

5.1.2. Entorno de desarrollo: Visual Studio Code

Para el desarrollo de este proyecto se ha elegido *Visual Studio Code* (VS Code) como entorno de desarrollo integrado (IDE). A continuación, se exponen las razones que fundamentan esta elección, así como las ventajas y desventajas asociadas.

Ventajas de Visual Studio Code

- **Multiplataforma:** VS Code es compatible con los principales sistemas operativos: Windows, macOS y Linux, lo que facilita su uso en diversos entornos de desarrollo.
- **Ligereza y rapidez:** A diferencia de otros IDEs más pesados, VS Code se caracteriza por ser relativamente ligero, ofreciendo un rendimiento ágil sin consumir excesivos recursos del sistema.
- **Extensibilidad:** VS Code cuenta con una vasta colección de extensiones que permiten agregar funcionalidades adicionales, adaptándose a las necesidades específicas de cualquier proyecto. Entre las extensiones más populares se encuentran las de soporte para lenguajes de programación, herramientas de depuración y control de versiones.
- **Integración con Git:** VS Code incorpora de manera nativa la integración con Git, lo que facilita la gestión de versiones y el trabajo colaborativo mediante sistemas de control de versiones.
- **Interfaz de usuario amigable:** La interfaz de usuario de VS Code se distingue por ser intuitiva y personalizable, lo que mejora la experiencia del desarrollador y aumenta la productividad.
- **Depuración integrada:** VS Code ofrece herramientas de depuración integradas para múltiples lenguajes de programación, permitiendo identificar y corregir errores de manera eficiente.

Desventajas de Visual Studio Code

- **Consumo de recursos:** Aunque se trata de un IDE ligero en comparación con otros, VS Code puede llegar a consumir una cantidad considerable de recursos del sistema, especialmente cuando se utilizan múltiples extensiones.
- **Funcionalidades limitadas de serie:** Algunas funcionalidades avanzadas requieren la instalación de extensiones adicionales, lo que puede resultar inconveniente para aquellos usuarios que prefieren tener todas las herramientas integradas desde el inicio.

- **Curva de aprendizaje:** Para quienes no están familiarizados con VS Code, puede existir una curva de aprendizaje inicial, especialmente en lo que respecta a la configuración y uso de extensiones.

Conclusión

Visual Studio Code se selecciona como el IDE para este proyecto debido a su flexibilidad, rendimiento y amplia gama de extensiones que permiten adaptarlo a las necesidades específicas del desarrollo. Aunque presenta algunas desventajas, sus numerosas ventajas lo convierten en una herramienta poderosa y eficiente para el desarrollo de software.

5.2 Librerías sistemas P

Para simular modelos computacionales mediante el uso de plásmidos, resulta esencial poder simular primero los sistemas P con plásmidos. Con este propósito, se han desarrollado librerías especializadas que facilitan la operación y manipulación de sistemas P. Dichas librerías proporcionan las herramientas necesarias para inicializar sistemas P y seguir su evolución a lo largo del tiempo.

En las secciones siguientes, se detallan las clases que constituyen estas librerías. Para cada clase se describen sus parámetros de entrada, atributos y métodos públicos.

5.2.1. Clase Membrana: *Membrane*

La clase *Membrane* está diseñada para mantener la estructura y evolución del sistema P de manera coherente y eficiente. Al inicializar una instancia de esta clase, se definen el alfabeto, la identificación de la membrana, la identificación de la membrana padre, los objetos iniciales, los plásmidos, las reglas y las prioridades de las reglas. Esto proporciona una base sólida para la simulación del comportamiento dinámico del sistema.

```
def __init__(self, V, id:int, parent:int=None, objects:str='', plasmids
    =[], rules={}, p_rules=[]):
    """Membrane class constructor.
    Args:
        V (list): Membrane's alphabet (same as system's)
        id (int): Membrane's id
        parent (int, optional): Parent Membrane's id. Defaults to None.
        objects (str, optional): Membrane's objects. Defaults to ''.
        plasmids (list, optional): Membrane's plasmids. Default to [].
        rules (dict, optional): Membrane's rules | key: rule_id, value:
            list = tuple (lhs, rhs). Defaults to {}.
        p_rules (dict, optional): Rules priority in membrane. Defaults
            to [].
    """
```

Listing 5.1: "Membrane Class constructor"

Los atributos de esta clase son:

1. **alphabet:** Conjunto que representa el alfabeto de la membrana, el cual debe coincidir con el del sistema.

2. **id:** Número entero que indica la etiqueta de la membrana.

3. **parent:** Identificador de la membrana que contiene a la actual, es decir, la membrana padre.

4. **childs:** Conjunto de identificadores que representan las membranas hijas.

5. **rules:** Diccionario que contiene las reglas de la membrana, donde las claves son números enteros que identifican cada regla.

6. **p_rules:** Lista de pares que representan las prioridades en el sistema. En cada par, el primer elemento tiene prioridad sobre el segundo.

7. **plasmids:** Conjunto de plásmidos presentes inicialmente en la membrana.

8. **objects:** Diccionario que representa los objetos en la membrana, donde la clave es el objeto y el valor es la cantidad presente en la membrana.

9. **rhs_alphabet:** Conjunto auxiliar que representa los diferentes elementos que pueden aparecer en la parte derecha de una regla, como las membranas destino para el transporte de objetos o el símbolo de disolución de la membrana.

Los métodos públicos de la clase son:

1. **add_child(child_id):** Añade al conjunto de membranas hijas el identificador que se pasa como parámetro, indicando que la membrana con ese identificador es una membrana hija de la actual.

2. **remove_child(child_id):** Elimina del conjunto de membranas hijas el identificador que se pasa como parámetro, indicando que la membrana con ese identificador ya no es una membrana hija de la actual.

3. **add_objects(objects):** Añade los objetos de la cadena *objects* a la membrana.

Los métodos *add_child* y *remove_child* permiten gestionar la jerarquía de membranas dentro del sistema, facilitando la adición y eliminación de membranas hijas. El método *add_objects* permite actualizar el contenido de la membrana con nuevos objetos, reflejando así las transformaciones que pueden ocurrir durante la simulación.

En resumen, la clase *Membrane* constituye una parte fundamental de las librerías para sistemas P con plásmidos, proporcionando una estructura robusta para la simulación y análisis de estos sistemas biológicos y computacionales.

5.2.2. Clase Sistema P: *PSystem*

La clase `PSystem` ha sido diseñada para simular los sistemas P con la incorporación de plásmidos (capítulo 3). Esta clase se construye con varios parámetros iniciales que definen el alfabeto del sistema, la estructura base de las membranas, los objetos contenidos en cada membrana, los plásmidos presentes, las reglas de evolución y las prioridades de estas reglas. Esta flexibilidad en la configuración inicial permite adaptar la simulación a una amplia variedad de escenarios y estudios de caso.

```
def __init__(self, H=None, V:list=[], base_struct="[1]1", m_objects={0:''}, m_plasmids=None, m_rules={0:{}}, p_rules={0:[]}, i0=1):
    """PSystem class constructor.
    Args:
        H (dict, optional): Plasmids' alphabet and its rules. Defaults to {}.
        V (list, optional): System's alphabet. Defaults to [].
        base_struct (str, optional): Initial system's structure. Defaults to "[1]1".
        m_objects (dict, optional): Membranes' objects | key:int = memb_id, value:str = memb_objects. Defaults to {0:''}.
        m_plasmids (dict, optional): Membranes' plasmids. Defaults to None.
        m_rules (dict, optional): Membranes' rules | key:int = memb_id, value:dict = memb_rules. Defaults to {0:{}}.
        p_rules (dict, optional): Rules priority in each membrane | key:int = memb_id, value:list = memb_priority. Defaults to {0:[]}.
        i0 (int, optional): Output membrane. Defaults to 1.
    """
```

Listing 5.2: "PSystem Class constructor"

Los atributos de esta clase son:

1. **alphabet:** Conjunto que representa el alfabeto de la membrana, el cual debe coincidir con el del sistema.
2. **membranes:** Diccionario de las membranas que componen el sistema, las claves son las etiquetas de las membranas y los valores son objetos de la clase *Membrane*.
3. **out_region:** Representa la región de salida para al terminar de computar el sistema.
4. **plasmids:** Diccionario que indica los plásmidos que existen en el sistema. Las claves son los identificadores de los plásmidos y los valores son diccionarios de las reglas que almacenan en sus interior.
5. **m_rules:** Diccionario que almacena para cada membrana sus reglas. Las claves son las etiquetas de las membranas y los valores son diccionarios con las reglas.
6. **p_rules:** Diccionario que almacena para cada membrana las prioridades de sus reglas. Las claves son las etiquetas de las membranas y los valores listas de duplas que representan cada prioridad.

Los métodos públicos de esta clase son:

1. **steps(n, verbose):** Permite la evolución del sistema durante *n* pasos o hasta finalizar, lo que ocurra primero.
2. **while_evolve(verbose):** Facilita la evolución continua del sistema hasta que este alcance su finalización.
3. **get_feasible_rules():** Proporciona una lista que incluye las membranas y las reglas que pueden aplicarse sin conflictos, es decir, aquellas reglas ganadoras seleccionadas de manera no determinista cuando hay competencia por un objeto.
4. **evolve(feasible_rules, verbose):** Permite la evolución del sistema mediante la aplicación de las reglas contenidas en *feasible_rules*. El objeto *feasible_rules* está diseñado para ser el resultado del método *get_feasible_rules()*.
5. **accessible_plasmids(memb_id):** Devuelve un conjunto de plásmidos que pueden ingresar en la membrana con el identificador *memb_id*.
6. **print_system():** Muestra en la consola una representación gráfica del sistema.
7. **to_dict():** Convierte la estructura del sistema a un formato de diccionario para facilitar su manipulación y análisis.

Los métodos *steps* y *while_evolve* permiten gestionar la evolución del sistema de membranas, controlando la cantidad de pasos a realizar o permitiendo una evolución continua hasta alcanzar una condición de finalización. El método *get_feasible_rules* proporciona una forma de identificar las reglas aplicables sin conflictos en el sistema, mientras que *evolve* se encarga de aplicar estas reglas para actualizar el estado del sistema.

El método *accessible_plasmids* proporciona información sobre los plásmidos que pueden ingresar a una membrana específica, facilitando la gestión de la transferencia de plásmidos entre membranas. *print_system* permite visualizar el estado actual del sistema en la consola, proporcionando una representación gráfica del mismo. Finalmente, *to_dict* convierte la estructura del sistema a un formato de diccionario, lo cual facilita su manipulación y análisis posterior en diferentes contextos, como el almacenamiento, la transferencia de datos o la integración con otros sistemas.

5.2.3. Equivalencia entre la formulación matemática y la notación de las librerías para representar sistemas P

Para una mejor comprensión de cómo se utilizan las librerías, es conveniente plantear un ejemplo. A continuación, se presentará un sistema P que representa la instrucción copia vista en el capítulo 4.

En este punto, se hará una equivalencia entre cada uno de los parámetros de entrada de la clase *PSystem* y cada elemento de la tupla que define el sistema P de la instrucción copia.

Considerando que se trata de la primera instrucción, el objeto L_x que identifica la instrucción será L_1. Además, se asume que los registros i y j son las membranas con identificadores 2 y 3, respectivamente, y que no hay más membranas en el sistema.

1. **Alfabeto:**

   El alfabeto se representa mediante una lista de strings, que indican los posibles objetos que pueden existir en el sistema. Está diseñado para que los objetos sean caracteres en minúsculas, pero también se contempla la posibilidad de que sean un carácter en mayúscula seguido de un número entero, de modo que no se limite su tamaño.

V formal	V librerías
$\{ L_1, L_2, a, h, r, s, t, u \}$	['L1', 'L2', 'a', 'h', 'r', 's', 't', 'u']

2. **Plásmidos:**

   Los plásmidos se representan inicialmente como un diccionario que contiene todos los plásmidos del sistema, diferenciados por un identificador que siempre comienza con *'P_'* seguido de cualquier cadena. El valor de este diccionario es otro diccionario que contiene las reglas del plásmido, identificadas por el mismo identificador seguido de una barra baja y un número, a modo de identificador de la regla.

H formal	H librerías
$P_{cop} : \{a \rightarrow a, r_{out}\}$	{ "P_cop": { "P_cop$''_1$: ($''a''$,$''ar0''$)}}

3. **Estructura:**

   La estructura de un sistema en las librerías se define utilizando el carácter '[' para indicar la apertura de una membrana, seguido del identificador de la membrana. De manera similar, para indicar que la membrana se ha cerrado, se utiliza el carácter ']'.

$\mu formal$	base_struct librerías
$[_1[_2]_2[_3]_3]_1$	[1[2]2[3]3]1

4. **Plásmidos en membrana:**

   Ya definidos los plásmidos y su estructura, es necesario especificar en qué membrana se encuentra cada plásmido. Aunque inicialmente todos los plásmidos están en el entorno, las librerías están diseñadas para permitir la inicialización en un estado intermedio de la computación, por lo que los plásmidos podrían no estar en el entorno. Esto se define mediante un diccionario en el cual las claves son los identificadores de las membranas y los valores son listas de los plásmidos que se encuentran en esas membranas.

m_plasmids librerías
{ 0: ["P_cop"] , 1: [ ], 2: [ ], 3: [ ] }

Si una membrana no contiene plásmidos, no es necesario incluirla como una lista vacía en el diccionario; simplemente no se pone esa entrada y se considera como que no tiene plásmidos en ella.

5. **Reglas del sistema:**

   Las reglas del sistema se definen para cada membrana mediante un diccionario. En estos diccionarios, las claves son los identificadores de las reglas y los valores son las propias reglas. Cada membrana tiene asociado su propio diccionario de reglas, similar a cómo se gestionan los plásmidos en cada membrana.

   Las reglas de entorno serían del sistema de la instrucción copia serían:

R_0 formal	r_0 librerías
$\{P_{cop}[_1L_1t]_1 \rightarrow [_1P_{cop}L_1t]_1,$ $[_1P_{cop}L_2h]_1 \rightarrow P_{cop}[_1L_2t]_1\}$	$\{1 : (''P\_cop[L1t]1'','' [P\_cop, L1t]1''),$ $2 : (''[P\_cop, L2h]1'','' P\_cop[L2t]1'')\}$

   Las reglas de la membrana 1 serían:

R_1 formal	r_1 librerías
$\{r_1 : P_{cop}L_1t \rightarrow L_1t[_3P_{cop}]_3,$ $r_2 : L_1t[_3P_{cop}]_3 \rightarrow P_{cop}L_1u,$ $r_3 : L_1r \rightarrow L_1s[_2a]_2,$ $r_4 : L_1ss \rightarrow L_1s,$ $r_5 : L_1us \rightarrow L_2h,$ $r_6 : L_1u \rightarrow L_2h\}$	$\{1 : (''P\_cop, L1t'','' L1t[P\_cop]3''),$ $2 : (''L1t[P\_cop]3'','' P\_cop, L1u''),$ $3 : (''L1r'','' L1s[a]2''),$ $4 : (''L1ss'','' L1s''),$ $5 : (''L1us'','' L2h''),$ $6 : (''L1u,'' L2h'')\}$

   Dadas estas reglas las reglas del sistema en las librerías quedarían de la siguiente forma:

m_rules librerías
{ 0: r_0 , 1: r_1 }

6. **Prioridades de las reglas del sistema:**

   Las prioridades de las reglas se determinan para cada membrana mediante una lista de duplas. En cada dupla, los elementos son identificadores de reglas, donde el primer elemento tiene prioridad sobre el segundo.

prioridades formal	p$_r$*uleslibreras*
$\rho_1 : \{(r_1, r_2, r_3, r_4, > r_5), (r_1, r_2, r_3, r_4, r_5 > r_6)\}$	{ 1: [(1,5), (2,5), (3,5), (4,5), (1,6), (2,6), (3,6), (4,6), (5,6)] }

7. **Membrana de salida:**

   En este caso, no cambia. Si la membrana de salida (i_0) es 1, entonces $i0 = 1$

5.3 Compilador

Una vez desarrolladas las bibliotecas, es necesario transformar el código de máquina de registros en un sistema P con plásmidos, asegurando que refleje fielmente el comportamiento de la ejecución original. Para lograr esto, se ha diseñado un compilador que convierte el código almacenado en un fichero de texto en un sistema P con plásmidos equivalente.

El compilador opera en dos fases. En la primera fase, realiza una lectura preliminar del código para identificar los registros y las etiquetas de las instrucciones, estableciendo una relación entre los registros y los identificadores de membranas, así como entre las etiquetas y el orden de las instrucciones.

En la segunda fase, más exhaustiva y detallada, el compilador lee cada instrucción individualmente y, dependiendo de la instrucción específica, incorpora dinámicamente al sistema las reglas de los sistemas P equivalentes descritas en la sección 4.

Cada tipo de instrucción se trata de manera específica:

- La instrucción `suc` añade una regla que incrementa el valor del registro especificado.
- La instrucción `pre` decrece el registro, verificando si su valor es positivo.
- Las instrucciones de control de flujo como `goto` permiten cambiar el flujo de ejecución del sistema P según la etiqueta especificada.
- Las instrucciones aritméticas y lógicas como `sum`, `mul`, `div` y `asi` se transforman en un conjunto de reglas. Cuando es necesario, se añaden plásmidos al entorno del sistema para gestionar operaciones más complejas.

Un aspecto crítico del compilador es su capacidad para añadir plásmidos al entorno del sistema. Cada vez que una instrucción requiere plásmidos, el compilador no solo añade el plásmido necesario, sino que también incluye las reglas adecuadas para su manipulación, asegurando que el sistema P pueda gestionar estos elementos de manera efectiva.

El resultado de este proceso es un sistema P con plásmidos que refleja con precisión el comportamiento del código de máquina de registros original. Este sistema P se compone de una estructura base, objetos, plásmidos y reglas de evolución que permiten simular la ejecución del código de manera fiel.

En resumen, el compilador transforma detallada y exhaustivamente el código de máquina de registros a un sistema P con plásmidos, garantizando que cada instrucción se traduzca correctamente y que cualquier necesidad adicional, como la incorporación de plásmidos, se maneje de manera adecuada. Esto permite que el sistema P resultante sea una representación precisa y funcional del código original.

CAPÍTULO 6
Casos de prueba

En este punto, se presentan varios casos que ponen a prueba el simulador de modelos computacionales mediante computación con plásmidos.

Para ofrecer una representación visual de la simulación del modelo, se ha diseñado una interfaz intuitiva. Esta interfaz permite cargar fácilmente un fichero de texto con el código que se desea ejecutar. Con solo dos botones, se pueden aplicar las reglas de cada iteración de forma manual o bien optar por iterar automáticamente hasta completar la computación.

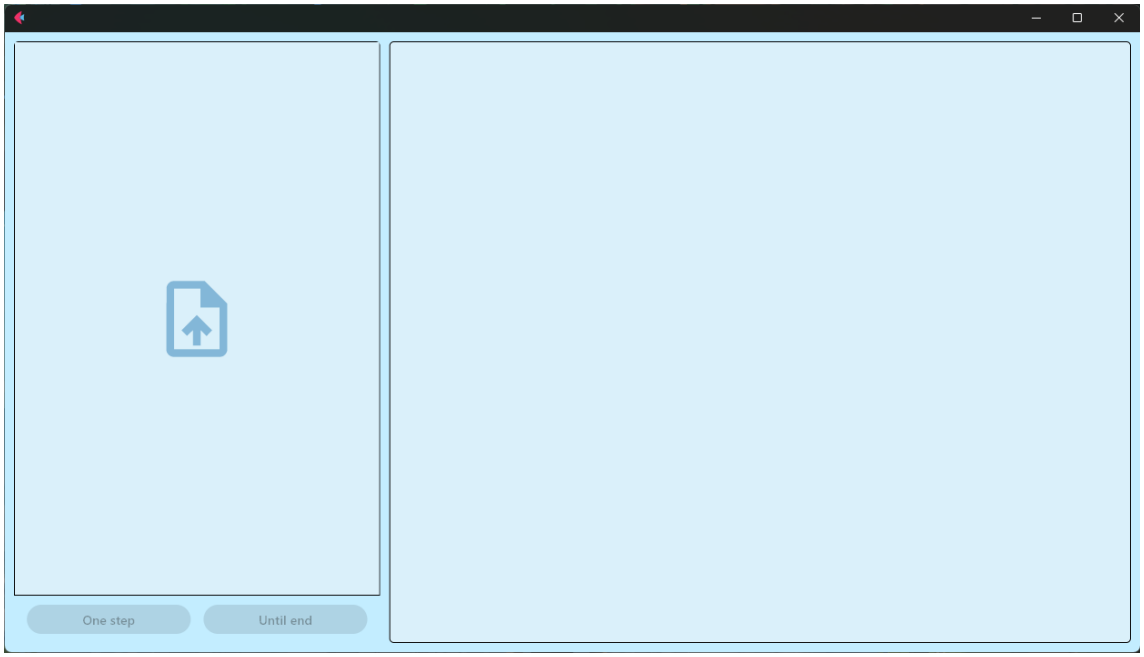

Figura 6.1: Interfaz inicial de la aplicación

Una vez que se sube el fichero, este se compila y genera el sistema P con plásmidos equivalente. En la parte derecha de la interfaz, aparece el sistema inicial, mientras que en la parte izquierda se puede ver el código cargado.

A continuación, se presentan tres casos de prueba: el factorial, el número de divisores y el cálculo del logaritmo en base 2.

6.1 Caso de Prueba: Factorial

En esta sección se examina el cálculo del factorial utilizando un código en máquina de registros. El código es el siguiente:

```
1            asi(5,j)
2            cop(j,q)
3            cer(p)
4            ig(q,p,cero,nocero)
5  cero:     suc(p)
6            goto(fin)
7  nocero:   asi(1,p)
8  bucle:    cop(p,m)
9            cer(p)
10           mul(q,m,p)
11           cer(m)
12           pre(q,fin)
13           pre(q,fin)
14           suc(q)
15           goto(bucle)
16 fin:      cop(p,i)
```

Listing 6.1: Código RM factorial de 5

Al compilar este código, se genera un sistema P con plásmidos que refleja la lógica del cálculo del factorial. La estructura inicial del sistema P, creada después de la compilación, se muestra en la siguiente figura:

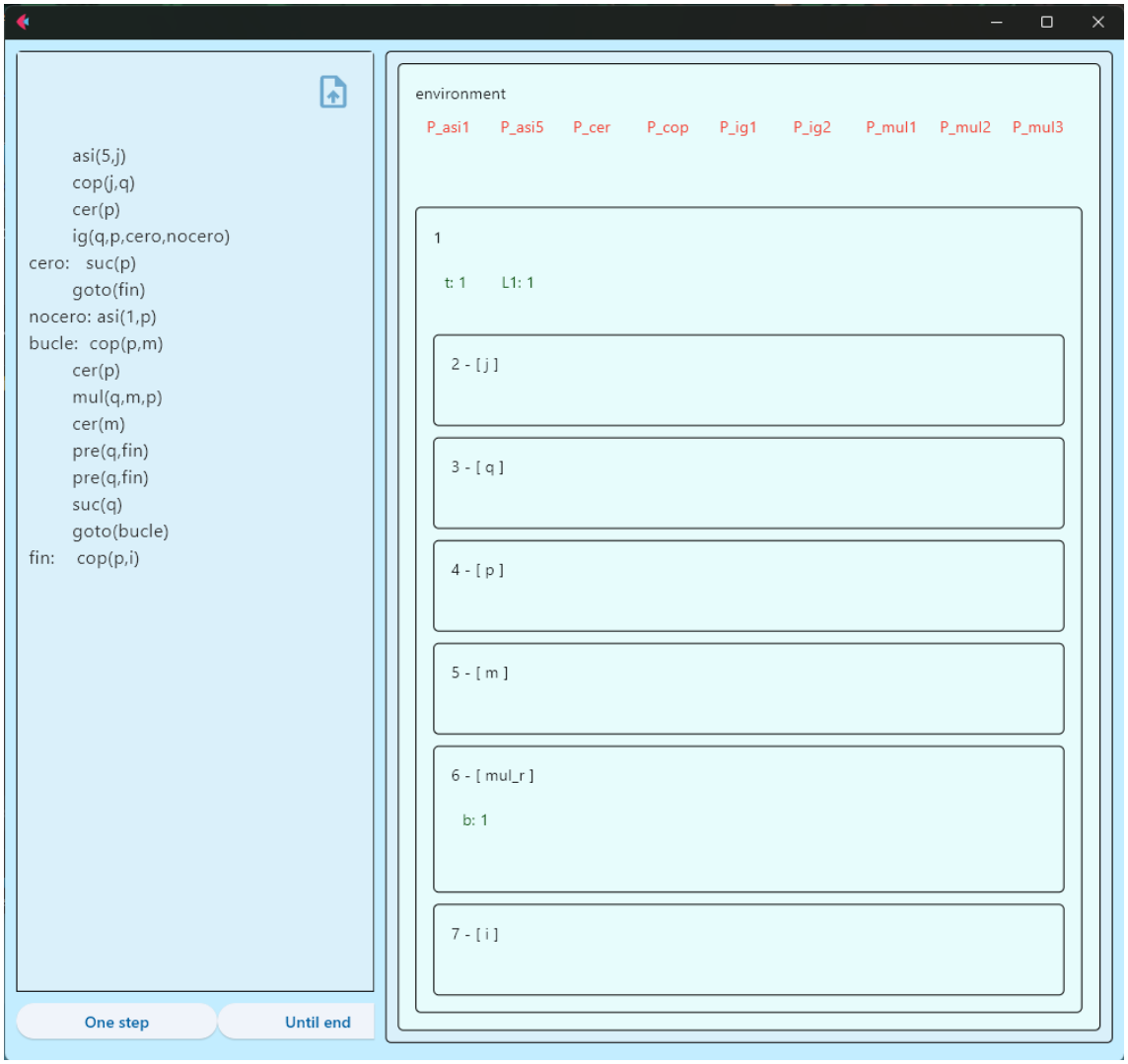

Figura 6.2: Sistema inicial del sistema P que computa el factorial de 5

Durante la ejecución del sistema P, las reglas correspondientes a las instrucciones del código de máquina de registros se aplican iterativamente. Una vez finalizada la computación, el sistema P alcanza su estado final, que se muestra en la siguiente figura:

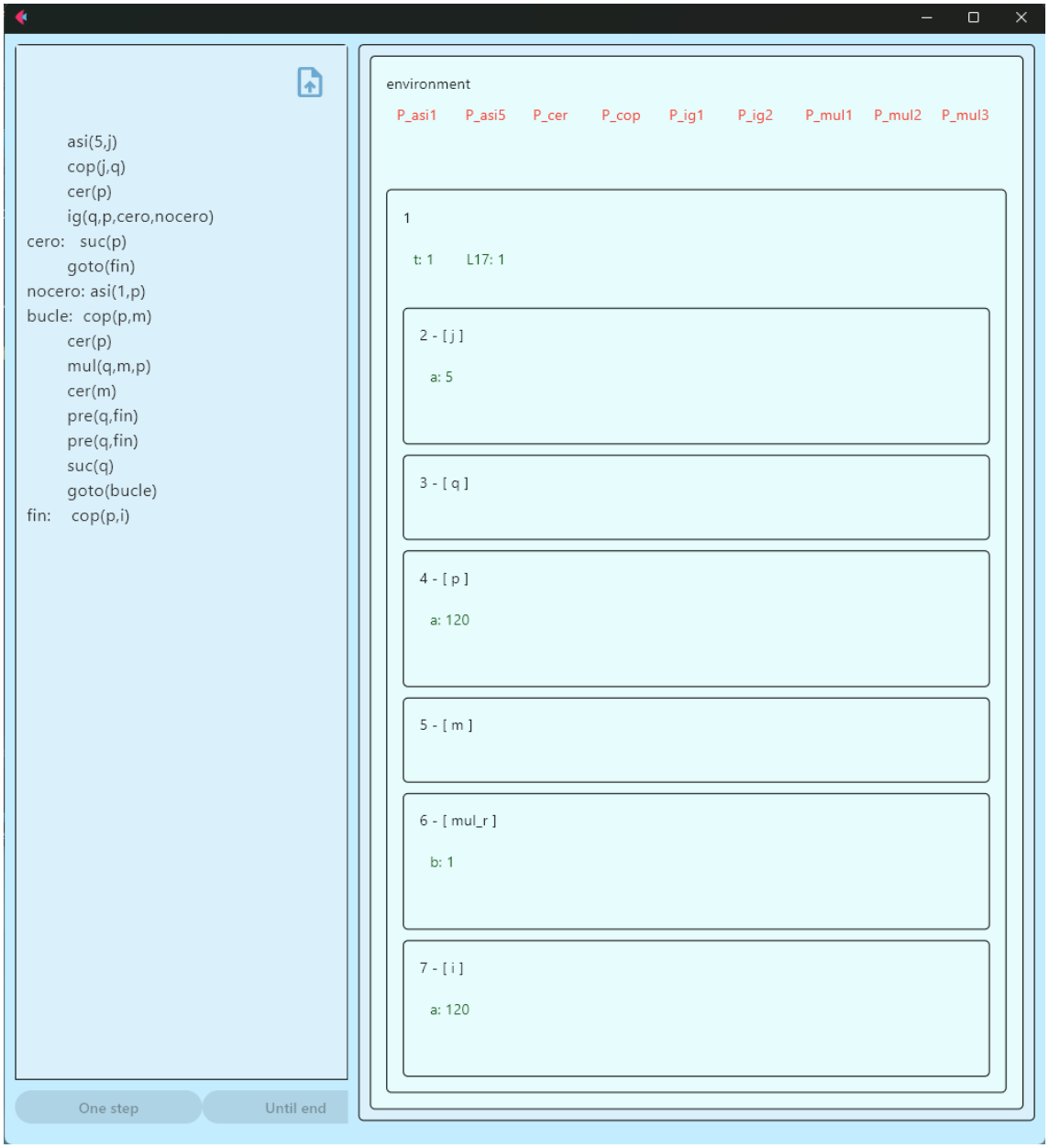

Figura 6.3: Sistema final del sistema P que computa el factorial de 5

El estado final del sistema P refleja el resultado del cálculo del factorial de 5, demostrando así la equivalencia y precisión del modelo de computación con plásmidos respecto a la máquina de registros original.

6.2 Caso de prueba: Número de divisores

Para este caso de prueba se estudia el cálculo del número de divisores de 25 utilizando un código en máquina de registros. El código utilizado es el siguiente:

```
1          asi(25,j)
2          asi(1,q)
3          cer(m)
4  test:   ig(j,q,fin,division)
5  division:cer(p)
```

```
6           div(j,q,p)
7           cop(p,aux)
8           cer(p)
9           mul(aux,q,p)
10          cer(aux)
11          ig(j,p,divisor,nodivisor)
12  divisor:suc(q)
13          suc(m)
14          goto(test)
15  nodivisor:suc(q)
16    goto(test)
17  fin:    suc(m)
18    cop(m,i)
```

Listing 6.2: Código RM número de divisores de 25

Al compilar este código, se genera un sistema P con plásmidos que refleja la lógica del cálculo del número de divisores de 25. La estructura inicial del sistema P, que se crea después de la compilación, es la siguiente:

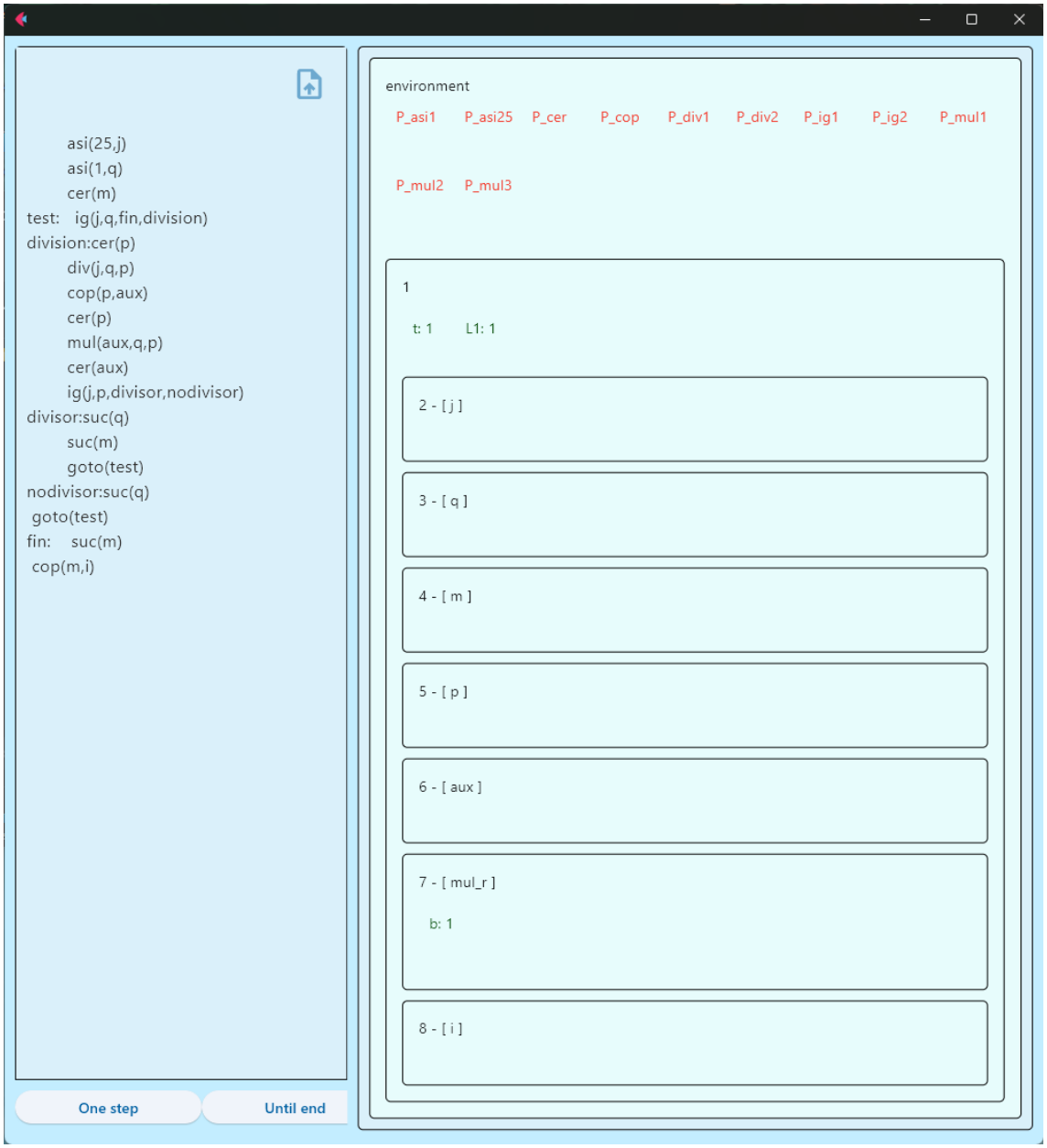

Figura 6.4: Sistema inicial del sistema P que computa el número de divisores de 25

Durante la ejecución del sistema P con plásmidos, las reglas correspondientes a las instrucciones del código de máquina de registros se aplican iterativamente. Una vez finalizada la computación, el sistema P alcanza su estado final, que se muestra en la siguiente figura:

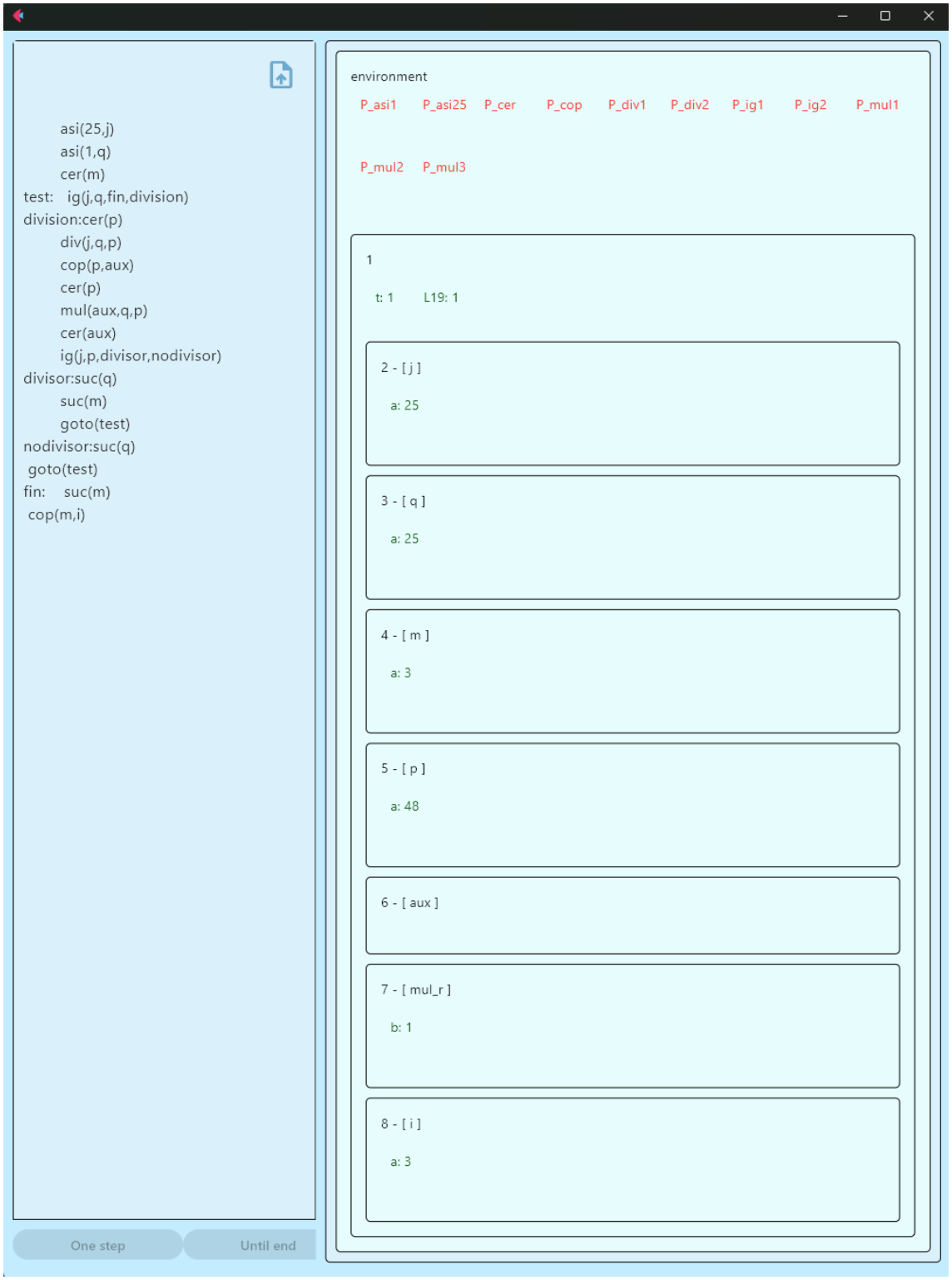

Figura 6.5: Sistema final del sistema P que computa el número de divisores de 25

El estado final del sistema P refleja el resultado del cálculo del número de divisores de 25, que se almacena en la membrana correspondiente al registro i.

6.3 Caso de prueba: Logaritmo en base 2

En esta sección se examina el cálculo $\log_2 8$ utilizando el siguiente código en máquina de registros:

```
1          asi(8,j)
2          cer(q)
3          ig(j,q,error,sig)
4  sig:    asi(1,q)
5          ig(j,q,uno,sig2)
6  uno:    cer(p)
7          goto(fin)
8  sig2:   cer(q)
9          asi(2,q)
10         asi(2,m)
11         asi(1,p)
12         mei(j,q,fin,bucle)
13 bucle:  cop(q,aux)
14         cer(q)
15         mul(aux,m,q)
16         cer(aux)
17         suc(p)
18         ig(j,q,fin,menor)
19 menor:  mei(j,q,resta1,bucle)
20 resta1: pre(p,error)
21         suc(p)
22 fin:    cop(p,i)
23 error:
```

Listing 6.3: Código RM Logaritmo en base 2 de 8

Al compilar este código, se genera un sistema P con plásmidos que refleja la lógica del cálculo del logaritmo en base 2. La estructura inicial del sistema P, que se crea después de la compilación, es la siguiente:

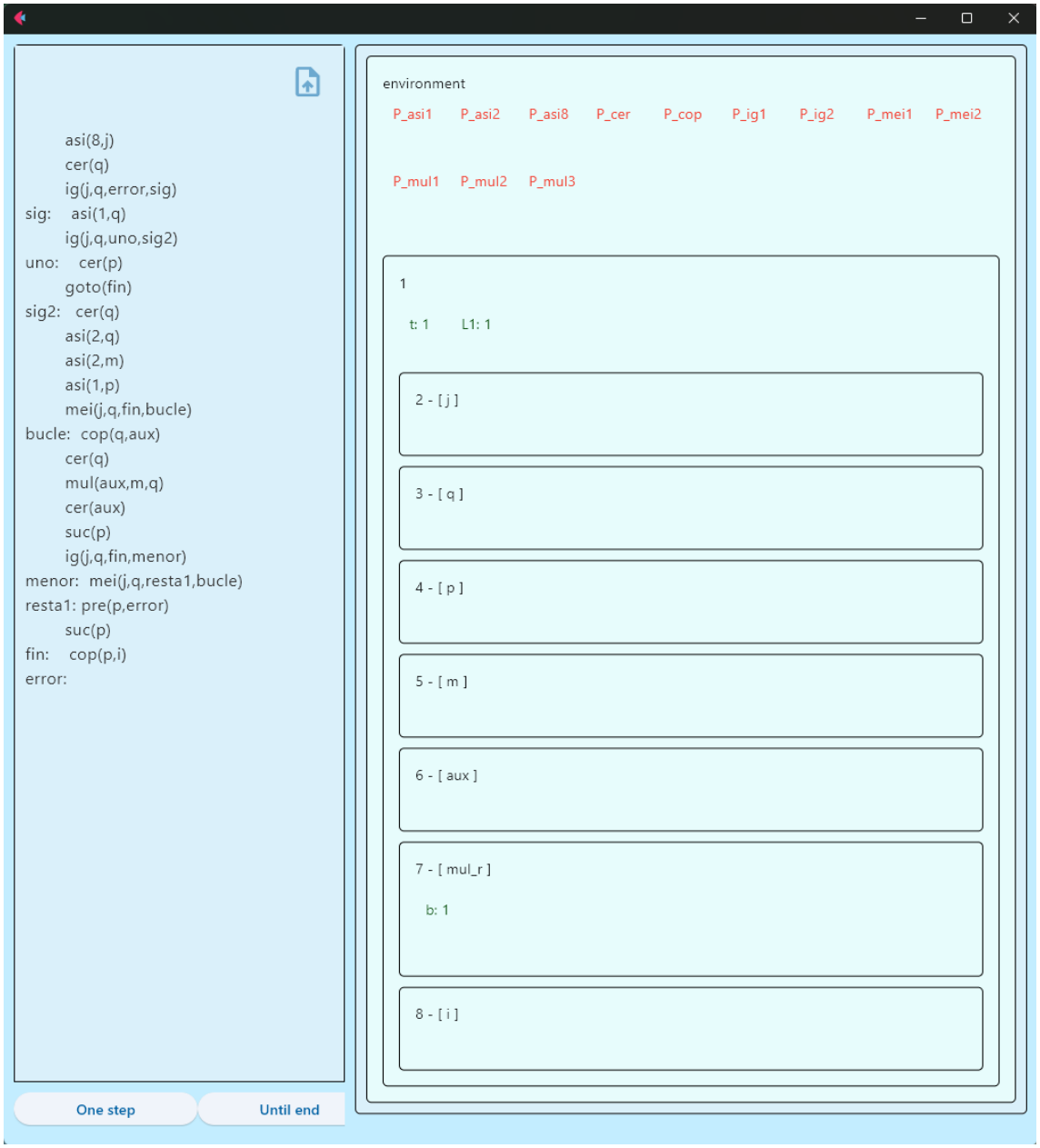

Figura 6.6: Sistema inicial del sistema P que computa $\log_2 8$

Durante la ejecución del sistema P con plásmidos, las reglas correspondientes a las instrucciones del código de máquina de registros se aplican iterativamente. Una vez finalizada la computación, el sistema P alcanza su estado final, que se muestra en la siguiente figura:

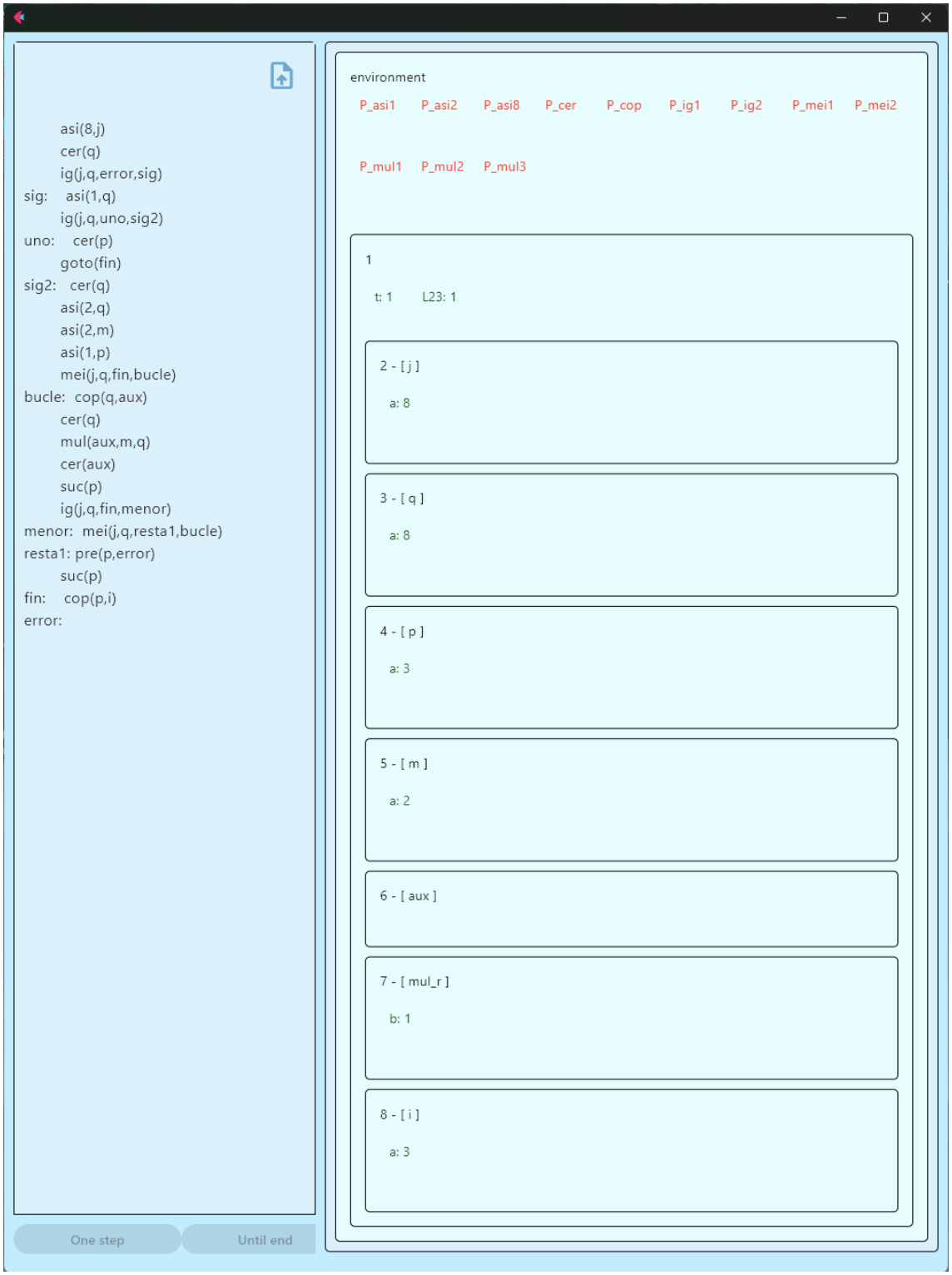

Figura 6.7: Sistema final del sistema P que computa el logaritmo de 8

El estado final del sistema P refleja el resultado del cálculo de $\log_2 8$, almacenado en la membrana correspondiente al registro i.

CAPÍTULO 7
Conclusiones

En este Trabajo de Fin de Grado se ha abordado la simulación de modelos computacionales mediante la computación con plásmidos, con el objetivo de demostrar que este enfoque constituye un modelo de computación completo. A lo largo del desarrollo del proyecto, se han planteado y alcanzado varios objetivos específicos, los cuales se detallan a continuación.

En primer lugar, se ha desarrollado una serie de librerías que permiten la simulación de la computación con plásmidos. Estas librerías han sido diseñadas para ser flexibles y extensibles, facilitando la creación y manipulación de sistemas P con plásmidos. La implementación de estas librerías ha sido fundamental para el éxito del proyecto, ya que proporcionan las herramientas necesarias para simular y analizar el comportamiento de los sistemas P.

El segundo objetivo consistía en demostrar que la computación con plásmidos es un modelo de computación completo. Para ello, se ha llevado a cabo la simulación de máquinas de registros mediante el uso de plásmidos, mostrando que estas pueden emular el comportamiento de una máquina de Turing. Los resultados obtenidos en los casos de prueba, como el cálculo del factorial, el número de divisores y el logaritmo en base 2, confirman que la computación con plásmidos puede replicar las operaciones básicas de las máquinas de registros y, por ende, de las máquinas de Turing.

El tercer objetivo se centraba en desarrollar y validar algoritmos específicos para la simulación de máquinas de registros utilizando plásmidos. Se han diseñado procedimientos detallados que permiten replicar las operaciones básicas de las máquinas de registros mediante el uso de plásmidos. Estos algoritmos han sido implementados y validados con éxito, demostrando su eficacia y precisión en la simulación de diversos casos de prueba.

Finalmente, se ha contribuido al cuerpo de conocimiento existente sobre modelos de computación no convencionales, proporcionando una base teórica y práctica que puede ser utilizada en investigaciones futuras sobre computación biológica y molecular. Este trabajo abre nuevas posibilidades en el ámbito de la biocomputación, aprovechando las capacidades únicas de los plásmidos para realizar operaciones complejas de manera eficiente y a escala molecular.

En conclusión, los objetivos planteados al inicio del proyecto se han cumplido satisfactoriamente. La computación con plásmidos se ha demostrado como un modelo de computación completo, capaz de emular el comportamiento de las

máquinas de Turing. Las librerías desarrolladas y los algoritmos implementados proporcionan una base sólida para futuras investigaciones en este campo, abriendo nuevas oportunidades para el desarrollo de aplicaciones en biocomputación. Este trabajo no solo amplía el entendimiento actual de los modelos de computación, sino que también sienta las bases para futuros proyectos y avances en esta área emergente.

7.1 Posibles Trabajos Futuros

El presente Trabajo de Fin de Grado ha demostrado la viabilidad y eficacia de la computación con plásmidos como un modelo de computación completo. Sin embargo, este campo de estudio es vasto y ofrece múltiples oportunidades para futuras investigaciones y desarrollos. A continuación, se presentan algunas líneas de trabajo que podrían ser exploradas en el futuro para ampliar y profundizar en los hallazgos obtenidos.

7.1.1. Optimización de Algoritmos

Uno de los aspectos más prometedores para futuros trabajos es la optimización de los algoritmos desarrollados. Aunque los algoritmos actuales han demostrado ser funcionales, existe un margen considerable para mejorar su eficiencia. La optimización podría centrarse en reducir el número de pasos necesarios para completar una computación, así como en minimizar el uso de recursos, como el número de plásmidos y objetos necesarios.

7.1.2. Ampliación del Conjunto de Instrucciones

El conjunto de instrucciones de las máquinas de registros utilizado en este trabajo es básico pero suficiente para demostrar la equivalencia con la computación con plásmidos. Sin embargo, en aplicaciones prácticas, podría ser beneficioso ampliar este conjunto de instrucciones para incluir operaciones más complejas y específicas. Esto permitiría una mayor flexibilidad y aplicabilidad del modelo en diferentes contextos.

7.1.3. Implementación de Nuevas Variantes de Sistemas P

Los sistemas P con plásmidos representan solo una de las muchas variantes posibles de sistemas P. Futuras investigaciones podrían explorar la implementación y simulación de otras variantes, como los sistemas P con membranas móviles o los sistemas P con comunicación entre membranas. Estas variantes podrían ofrecer nuevas perspectivas y aplicaciones en el campo de la biocomputación.

7.1.4. Aplicaciones en Biología Computacional

La computación con plásmidos tiene un gran potencial en el ámbito de la biología computacional. Futuras investigaciones podrían centrarse en la aplicación de este modelo para simular procesos biológicos complejos, como la replicación del ADN, la dinámica de poblaciones celulares o la interacción entre diferentes tipos de células. Estos estudios podrían proporcionar una comprensión más profunda de los procesos biológicos y abrir nuevas vías para el desarrollo de terapias y tratamientos médicos.

7.1.5. Integración con Tecnologías de Inteligencia Artificial

La integración de la computación con plásmidos con tecnologías de inteligencia artificial (IA) es otra área prometedora para futuras investigaciones. La IA podría utilizarse para optimizar la selección y aplicación de reglas en los sistemas P, así como para predecir el comportamiento de estos sistemas en diferentes condiciones. Esta integración podría mejorar significativamente la eficiencia y aplicabilidad del modelo.

7.1.6. Estudios Comparativos con Otros Modelos de Computación

Finalmente, sería interesante realizar estudios comparativos entre la computación con plásmidos y otros modelos de computación no convencionales, como la computación cuántica o la computación basada en ADN. Estos estudios podrían proporcionar una visión más completa de las ventajas y limitaciones de cada modelo, así como identificar posibles sinergias y áreas de colaboración.

7.1.7. Conclusiones de los Posibles Trabajos Futuros

En resumen, la computación con plásmidos ofrece un campo de estudio rico y variado, con numerosas oportunidades para futuras investigaciones. Los trabajos futuros podrían centrarse en la optimización de algoritmos, la ampliación del conjunto de instrucciones, la implementación de nuevas variantes de sistemas P, las aplicaciones en biología computacional, la integración con tecnologías de inteligencia artificial y los estudios comparativos con otros modelos de computación. Estas líneas de trabajo no solo ampliarían el conocimiento actual, sino que también podrían abrir nuevas vías para el desarrollo de aplicaciones prácticas en diversos campos.

Bibliografía

[1] George Păun. Computing with Membranes. *Institute of Mathematics of the Romanian Academy, P.O. Box 1-764, 70700 Bucharest, Romania*, January 1999.

[2] Gil Adrover, Mario. TFG Diseño e implementación de un simulador de sistemas P con plásmidos. *Universitat Politècnica de València, Valencia, España*, Curso académico 2022-2023.

[3] Pérez Pérez, David Computación celular con membranas: un estudio de modelos de cómputo bioinspirados. *Universidad Politécnica de Madrid, Madrid, España*, 2016.

[4] Gheorghe Păun, Grzegorz Rozenberg Arto Salomaa. *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2005.

[5] Gabriel Ciobanu, Gheorghe Păun Mario J. Pérez-Jiménez. *Applications of Membrane Computing*. Springer, 2006.

[6] Mario de Jesús Pérez Jiménez, Fernando Sancho Caparrini. *Máquinas moleculares basadas en ADN*. Universidad de Sevilla, 2003.

[7] Información sobre el plásmido. Consultado en https://www.genome.gov/es/genetics-glossary/Plasmido.

APÉNDICE A
Sistemas P con plásmidos generados en los casos de prueba

Después de compilar los códigos de máquina de los registros, se generan los sistemas P siguientes. Para mejorar la legibilidad, se presentarán en formato JSON, dado que las bibliotecas facilitan su conversión a diccionarios Python. Los sistemas equivalentes de los casos de prueba vistos en el capítulo 6 se detallan en las secciones siguientes.

A.1 Factorial

```
{'environment':
    {'childs':
        {1: {'childs': {2: {'objects': {}, 'rules': {}},
                        3: {'objects': {}, 'rules': {}},
                        4: {'objects': {}, 'rules': {}},
                        5: {'objects': {}, 'rules': {}},
                        6: {'objects': {'b': 1},
                            'rules': {}},
                        7: {'objects': {}, 'rules': {}}},
             'objects': {'L1': 1, 't': 1},
             'rules': {1: ('P_asi5,L1t[]2', 'L1[P_asi5,r]2'),
                       2: ('L1[P_asi5,s]2', 'P_asi5,L2h[]2'),
                       3: ('P_cop,L2t[]2', 'L2t[P_cop]2'),
                       4: ('L2t[P_cop]2', 'P_cop,L2u[]2'),
                       5: ('L2r', 'L2s[a]3'),
                       6: ('L2ss', 'L2s'),
                       7: ('L2us', 'L3h'),
                       8: ('L2u', 'L3h'),
                       9: ('P_cer,L3t[]4', 'L3[P_cer]4'),
                       10: ('L3[P_cer]4', 'P_cer,L4h[]4'),
                       11: ('P_ig1,L4t[]3', 'L4t[P_ig1]3'),
                       12: ('L4t[P_ig1]3', 'P_ig1,L4s[]3'),
                       13: ('P_ig2,L4s[]4', 'L4s[P_ig2]4'),
                       14: ('L4s[P_ig2]4', 'P_ig2,L4b[]4'),
                       15: ('L4bxy', 'L4b'),
                       16: ('L4bxx', 'L4bx'),
                       17: ('L4byy', 'L4by'),
                       18: ('L4bx', 'L7P3'),
                       19: ('L4by', 'L7P3'),
```

```
                    20: ('L4b', 'L5P3'),
                    21: ('L5t', 'L6t[a]4'),
                    23: ('L6', 'L16'),
                    24: ('P_asi1,L7t[]4',
                         'L7[P_asi1,r]4'),
                    25: ('L7[P_asi1,s]4',
                         'P_asi1,L8h[]4'),
                    26: ('P_cop,L8t[]4', 'L8t[P_cop]4'),
                    27: ('L8t[P_cop]4', 'P_cop,L8u[]4'),
                    28: ('L8r', 'L8s[a]5'),
                    29: ('L8ss', 'L8s'),
                    30: ('L8us', 'L9h'),
                    31: ('L8u', 'L9h'),
                    32: ('P_cer,L9t[]4', 'L9[P_cer]4'),
                    33: ('L9[P_cer]4', 'P_cer,L10h[]4'),
                    34: ('P_mul1,L10t[]3',
                         'L10p[P_mul1]3'),
                    35: ('L10p[P_mul1]3',
                         'P_mul1,L10q[]3'),
                    36: ('P_mul2,L10q[a]6',
                         'L10r[P_mul2,a]6'),
                    37: ('L10r[P_mul2]6',
                         'P_mul2,L10s[]6'),
                    38: ('P_mul3,L10s[]5',
                         'L10o[P_mul3]5'),
                    39: ('L10o[P_mul3]5',
                         'P_mul3,L10q[]5'),
                    40: ('L10x', 'L10u[a]6'),
                    41: ('L10y', 'L10u[a]4'),
                    42: ('L10uu', 'L10u'),
                    43: ('L10qu', 'L11P3'),
                    44: ('P_mul2,L10q[]6',
                         'P_mul2,L10qu[]6'),
                    45: ('P_cer,L11t[]5', 'L11[P_cer]5'),
                    46: ('L11[P_cer]5', 'P_cer,L12h[]5'),
                    47: ('L12t[a]3', 'L13t'),
                    48: ('L12t', 'L16t'),
                    49: ('L13t[a]3', 'L14t'),
                    50: ('L13t', 'L16t'),
                    51: ('L14t', 'L15t[a]3'),
                    53: ('L15', 'L8'),
                    54: ('P_cop,L16t[]4', 'L16t[P_cop]4'),
                    55: ('L16t[P_cop]4', 'P_cop,L16u[]4'),
                    56: ('L16r', 'L16s[a]7'),
                    57: ('L16ss', 'L16s'),
                    58: ('L16us', 'L17h'),
                    59: ('L16u', 'L17h')}}},
    'objects': {},
    'plasmids': {'P_asi1',
                 'P_asi5',
                 'P_cer',
                 'P_cop',
                 'P_ig1',
                 'P_ig2',
                 'P_mul1',
                 'P_mul2',
                 'P_mul3'},
    'rules': {1: ('P_asi5[L1t]1', '[P_asi5,L1t]1'),
              2: ('[P_asi5,L2h]1', 'P_asi5[L2t]1'),
```

```
            3: ('P_cop[L2t]1', '[P_cop,L2t]1'),
            4: ('[P_cop,L3h]1', 'P_cop[L3t]1'),
            5: ('P_cer[L3t]1', '[P_cer,L3t]1'),
            6: ('[P_cer,L4h]1', 'P_cer[L4t]1'),
            7: ('P_ig1[L4t]1', '[P_ig1,L4P1]1'),
            8: ('P_ig2[L4P1]1', '[P_ig2,L4t]1'),
            9: ('[P_ig1,L5P3]1', 'P_ig1[L5h]1'),
            10: ('[P_ig1,L7P3]1', 'P_ig1[L7h]1'),
            11: ('[P_ig2,L5h]1', 'P_ig2[L5t]1'),
            12: ('[P_ig2,L7h]1', 'P_ig2[L7t]1'),
            13: ('P_asi1[L7t]1', '[P_asi1,L7t]1'),
            14: ('[P_asi1,L8h]1', 'P_asi1[L8t]1'),
            15: ('P_cop[L8t]1', '[P_cop,L8t]1'),
            16: ('[P_cop,L9h]1', 'P_cop[L9t]1'),
            17: ('P_cer[L9t]1', '[P_cer,L9t]1'),
            18: ('[P_cer,L10h]1', 'P_cer[L10t]1'),
            19: ('P_mul1[L10t]1', '[P_mul1,L10P1]1'),
            20: ('P_mul2[L10P1]1', '[P_mul2,L10P2]1'),
            21: ('P_mul3[L10P2]1', '[P_mul3,L10t]1'),
            22: ('[P_mul1,L11P3]1', 'P_mul1[L11P4]1'),
            23: ('[P_mul2,L11P4]1', 'P_mul2[L11h]1'),
            24: ('[P_mul3,L11h]1', 'P_mul3[L11t]1'),
            25: ('P_cer[L11t]1', '[P_cer,L11t]1'),
            26: ('[P_cer,L12h]1', 'P_cer[L12t]1'),
            27: ('P_cop[L16t]1', '[P_cop,L16t]1'),
            28: ('[P_cop,L17h]1', 'P_cop[L17t]1')}}}
```

Listing A.1: Sistema P con plásmidos compilado. Factorial de 5

A.2 Número de divisores

```
{'environment':
    {'childs':
        {1: {'childs': {2: {'objects': {}, 'rules': {}},
                        3: {'objects': {}, 'rules': {}},
                        4: {'objects': {}, 'rules': {}},
                        5: {'objects': {}, 'rules': {}},
                        6: {'objects': {}, 'rules': {}},
                        7: {'objects': {'b': 1},
                            'rules': {}},
                        8: {'objects': {}, 'rules': {}}},
             'objects': {'L1': 1, 't': 1},
             'rules': {1: ('P_asi25,L1t[]2',
                           'L1[P_asi25,r]2'),
                       2: ('L1[P_asi25,s]2',
                           'P_asi25,L2h[]2'),
                       3: ('P_asi1,L2t[]3', 'L2[P_asi1,r]3'),
                       4: ('L2[P_asi1,s]3', 'P_asi1,L3h[]3'),
                       5: ('P_cer,L3t[]4', 'L3[P_cer]4'),
                       6: ('L3[P_cer]4', 'P_cer,L4h[]4'),
                       7: ('P_ig1,L4t[]2', 'L4t[P_ig1]2'),
                       8: ('L4t[P_ig1]2', 'P_ig1,L4s[]2'),
                       9: ('P_ig2,L4s[]3', 'L4s[P_ig2]3'),
                       10: ('L4s[P_ig2]3', 'P_ig2,L4b[]3'),
                       11: ('L4bxy', 'L4b'),
                       12: ('L4bxx', 'L4bx'),
```

```
13: ('L4byy', 'L4by'),
14: ('L4bx', 'L5P3'),
15: ('L4by', 'L5P3'),
16: ('L4b', 'L17P3'),
17: ('P_cer,L5t[]5', 'L5[P_cer]5'),
18: ('L5[P_cer]5', 'P_cer,L6h[]5'),
19: ('P_div1,L6t[a]2',
     'L6t[P_div1,a]2'),
20: ('L6t[P_div1]2', 'P_div1,L6s[]2'),
21: ('P_div2,L6s[a]3',
     'L6sb[P_div2,a]3'),
22: ('L6s[P_div2]3', 'P_div2,L6r[]3'),
23: ('L6rbxy', 'L6sa5'),
24: ('L6xy', 'L6b'),
25: ('L6sx', 'L20'),
26: ('L6b', 'L6'),
27: ('L6y', 'L6'),
28: ('L6r', 'L7P2'),
29: ('P_div1,L6t', 'P_div1,L7P2'),
30: ('P_cop,L7t[]5', 'L7t[P_cop]5'),
31: ('L7t[P_cop]5', 'P_cop,L7u[]5'),
32: ('L7r', 'L7s[a]6'),
33: ('L7ss', 'L7s'),
34: ('L7us', 'L8h'),
35: ('L7u', 'L8h'),
36: ('P_cer,L8t[]5', 'L8[P_cer]5'),
37: ('L8[P_cer]5', 'P_cer,L9h[]5'),
38: ('P_mul1,L9t[]6', 'L9p[P_mul1]6'),
39: ('L9p[P_mul1]6', 'P_mul1,L9q[]6'),
40: ('P_mul2,L9q[a]7',
     'L9r[P_mul2,a]7'),
41: ('L9r[P_mul2]7', 'P_mul2,L9s[]7'),
42: ('P_mul3,L9s[]3', 'L9o[P_mul3]3'),
43: ('L9o[P_mul3]3', 'P_mul3,L9q[]3'),
44: ('L9x', 'L9u[a]7'),
45: ('L9y', 'L9u[a]5'),
46: ('L9uu', 'L9u'),
47: ('L9qu', 'L10P3'),
48: ('P_mul2,L9q[]7',
     'P_mul2,L9qu[]7'),
49: ('P_cer,L10t[]6', 'L10[P_cer]6'),
50: ('L10[P_cer]6', 'P_cer,L11h[]6'),
51: ('P_ig1,L11t[]2', 'L11t[P_ig1]2'),
52: ('L11t[P_ig1]2', 'P_ig1,L11s[]2'),
53: ('P_ig2,L11s[]5', 'L11s[P_ig2]5'),
54: ('L11s[P_ig2]5', 'P_ig2,L11b[]5'),
55: ('L11bxy', 'L11b'),
56: ('L11bxx', 'L11bx'),
57: ('L11byy', 'L11by'),
58: ('L11bx', 'L15P3'),
59: ('L11by', 'L15P3'),
60: ('L11b', 'L12P3'),
61: ('L12t', 'L13t[a]3'),
63: ('L13t', 'L14t[a]4'),
65: ('L14', 'L4'),
66: ('L15t', 'L16t[a]3'),
68: ('L16', 'L4'),
69: ('L17t', 'L18t[a]4'),
71: ('P_cop,L18t[]4', 'L18t[P_cop]4'),
```

```
                    72: ('L18t[P_cop]4', 'P_cop,L18u[]4'),
                    73: ('L18r', 'L18s[a]8'),
                    74: ('L18ss', 'L18s'),
                    75: ('L18us', 'L19h'),
                    76: ('L18u', 'L19h')}}},
    'objects': {},
    'plasmids': {'P_asi1',
                 'P_asi25',
                 'P_cer',
                 'P_cop',
                 'P_div1',
                 'P_div2',
                 'P_ig1',
                 'P_ig2',
                 'P_mul1',
                 'P_mul2',
                 'P_mul3'},
    'rules': {1: ('P_asi25[L1t]1', '[P_asi25,L1t]1'),
              2: ('[P_asi25,L2h]1', 'P_asi25[L2t]1'),
              3: ('P_asi1[L2t]1', '[P_asi1,L2t]1'),
              4: ('[P_asi1,L3h]1', 'P_asi1[L3t]1'),
              5: ('P_cer[L3t]1', '[P_cer,L3t]1'),
              6: ('[P_cer,L4h]1', 'P_cer[L4t]1'),
              7: ('P_ig1[L4t]1', '[P_ig1,L4P1]1'),
              8: ('P_ig2[L4P1]1', '[P_ig2,L4t]1'),
              9: ('[P_ig1,L17P3]1', 'P_ig1[L17h]1'),
              10: ('[P_ig1,L5P3]1', 'P_ig1[L5h]1'),
              11: ('[P_ig2,L17h]1', 'P_ig2[L17t]1'),
              12: ('[P_ig2,L5h]1', 'P_ig2[L5t]1'),
              13: ('P_cer[L5t]1', '[P_cer,L5t]1'),
              14: ('[P_cer,L6h]1', 'P_cer[L6t]1'),
              15: ('P_div1[L6t]1', '[P_div1,L6P1]1'),
              16: ('P_div2[L6P1]1', '[P_div2,L6t]1'),
              17: ('[P_div1,L7P2]1', 'P_div1[L7h]1'),
              18: ('[P_div2,L7h]1', 'P_div2[L7t]1'),
              19: ('P_cop[L7t]1', '[P_cop,L7t]1'),
              20: ('[P_cop,L8h]1', 'P_cop[L8t]1'),
              21: ('P_cer[L8t]1', '[P_cer,L8t]1'),
              22: ('[P_cer,L9h]1', 'P_cer[L9t]1'),
              23: ('P_mul1[L9t]1', '[P_mul1,L9P1]1'),
              24: ('P_mul2[L9P1]1', '[P_mul2,L9P2]1'),
              25: ('P_mul3[L9P2]1', '[P_mul3,L9t]1'),
              26: ('[P_mul1,L10P3]1', 'P_mul1[L10P4]1'),
              27: ('[P_mul2,L10P4]1', 'P_mul2[L10h]1'),
              28: ('[P_mul3,L10h]1', 'P_mul3[L10t]1'),
              29: ('P_cer[L10t]1', '[P_cer,L10t]1'),
              30: ('[P_cer,L11h]1', 'P_cer[L11t]1'),
              31: ('P_ig1[L11t]1', '[P_ig1,L11P1]1'),
              32: ('P_ig2[L11P1]1', '[P_ig2,L11t]1'),
              33: ('[P_ig1,L12P3]1', 'P_ig1[L12h]1'),
              34: ('[P_ig1,L15P3]1', 'P_ig1[L15h]1'),
              35: ('[P_ig2,L12h]1', 'P_ig2[L12t]1'),
              36: ('[P_ig2,L15h]1', 'P_ig2[L15t]1'),
              37: ('P_cop[L18t]1', '[P_cop,L18t]1'),
              38: ('[P_cop,L19h]1', 'P_cop[L19t]1')}}}
```

Listing A.2: Sistema P con plásmidos compilado. Número de divisores de 25

A.3 Logaritmo en base 2

```
{'environment':
    {'childs':
        {1: {'childs': {2: {'objects': {}, 'rules': {}},
                        3: {'objects': {}, 'rules': {}},
                        4: {'objects': {}, 'rules': {}},
                        5: {'objects': {}, 'rules': {}},
                        6: {'objects': {}, 'rules': {}},
                        7: {'objects': {'b': 1},
                            'rules': {}},
                        8: {'objects': {}, 'rules': {}}},
             'objects': {'L1': 1, 't': 1},
             'rules': {1: ('P_asi8,L1t[]2', 'L1[P_asi8,r]2'),
                       2: ('L1[P_asi8,s]2', 'P_asi8,L2h[]2'),
                       3: ('P_cer,L2t[]3', 'L2[P_cer]3'),
                       4: ('L2[P_cer]3', 'P_cer,L3h[]3'),
                       5: ('P_ig1,L3t[]2', 'L3t[P_ig1]2'),
                       6: ('L3t[P_ig1]2', 'P_ig1,L3s[]2'),
                       7: ('P_ig2,L3s[]3', 'L3s[P_ig2]3'),
                       8: ('L3s[P_ig2]3', 'P_ig2,L3b[]3'),
                       9: ('L3bxy', 'L3b'),
                       10: ('L3bxx', 'L3bx'),
                       11: ('L3byy', 'L3by'),
                       12: ('L3bx', 'L4P3'),
                       13: ('L3by', 'L4P3'),
                       14: ('L3b', 'L25P3'),
                       15: ('P_asi1,L4t[]3',
                            'L4[P_asi1,r]3'),
                       16: ('L4[P_asi1,s]3',
                            'P_asi1,L5h[]3'),
                       17: ('P_ig1,L5t[]2', 'L5t[P_ig1]2'),
                       18: ('L5t[P_ig1]2', 'P_ig1,L5s[]2'),
                       19: ('P_ig2,L5s[]3', 'L5s[P_ig2]3'),
                       20: ('L5s[P_ig2]3', 'P_ig2,L5b[]3'),
                       21: ('L5bxy', 'L5b'),
                       22: ('L5bxx', 'L5bx'),
                       23: ('L5byy', 'L5by'),
                       24: ('L5bx', 'L8P3'),
                       25: ('L5by', 'L8P3'),
                       26: ('L5b', 'L6P3'),
                       27: ('P_cer,L6t[]4', 'L6[P_cer]4'),
                       28: ('L6[P_cer]4', 'P_cer,L7h[]4'),
                       29: ('L7', 'L22'),
                       30: ('P_cer,L8t[]3', 'L8[P_cer]3'),
                       31: ('L8[P_cer]3', 'P_cer,L9h[]3'),
                       32: ('P_asi2,L9t[]3',
                            'L9[P_asi2,r]3'),
                       33: ('L9[P_asi2,s]3',
                            'P_asi2,L10h[]3'),
                       34: ('P_asi2,L10t[]5',
                            'L10[P_asi2,r]5'),
                       35: ('L10[P_asi2,s]5',
                            'P_asi2,L11h[]5'),
                       36: ('P_asi1,L11t[]4',
                            'L11[P_asi1,r]4'),
                       37: ('L11[P_asi1,s]4',
                            'P_asi1,L12h[]4'),
```

```
                38: ('P_mei1,L12t[]2',
                     'L12t[P_mei1]2'),
                39: ('L12t[P_mei1]2',
                     'P_mei1,L12s[]2'),
                40: ('P_mei2,L12s[]3',
                     'L12s[P_mei2]3'),
                41: ('L12s[P_mei2]3',
                     'P_mei2,L12b[]3'),
                42: ('L12bxy', 'L12b'),
                43: ('L12bxx', 'L12bx'),
                44: ('L12byy', 'L12by'),
                45: ('L12bx', 'L13P3'),
                46: ('L12by', 'L22P3'),
                47: ('L12b', 'L22P3'),
                48: ('P_cop,L13t[]3', 'L13t[P_cop]3'),
                49: ('L13t[P_cop]3', 'P_cop,L13u[]3'),
                50: ('L13r', 'L13s[a]6'),
                51: ('L13ss', 'L13s'),
                52: ('L13us', 'L14h'),
                53: ('L13u', 'L14h'),
                54: ('P_cer,L14t[]3', 'L14[P_cer]3'),
                55: ('L14[P_cer]3', 'P_cer,L15h[]3'),
                56: ('P_mul1,L15t[]6',
                     'L15p[P_mul1]6'),
                57: ('L15p[P_mul1]6',
                     'P_mul1,L15q[]6'),
                58: ('P_mul2,L15q[a]7',
                     'L15r[P_mul2,a]7'),
                59: ('L15r[P_mul2]7',
                     'P_mul2,L15s[]7'),
                60: ('P_mul3,L15s[]5',
                     'L15o[P_mul3]5'),
                61: ('L15o[P_mul3]5',
                     'P_mul3,L15q[]5'),
                62: ('L15x', 'L15u[a]7'),
                63: ('L15y', 'L15u[a]3'),
                64: ('L15uu', 'L15u'),
                65: ('L15qu', 'L16P3'),
                66: ('P_mul2,L15q[]7',
                     'P_mul2,L15qu[]7'),
                67: ('P_cer,L16t[]6', 'L16[P_cer]6'),
                68: ('L16[P_cer]6', 'P_cer,L17h[]6'),
                69: ('L17t', 'L18t[a]4'),
                71: ('P_ig1,L18t[]2', 'L18t[P_ig1]2'),
                72: ('L18t[P_ig1]2', 'P_ig1,L18s[]2'),
                73: ('P_ig2,L18s[]3', 'L18s[P_ig2]3'),
                74: ('L18s[P_ig2]3', 'P_ig2,L18b[]3'),
                75: ('L18bxy', 'L18b'),
                76: ('L18bxx', 'L18bx'),
                77: ('L18byy', 'L18by'),
                78: ('L18bx', 'L19P3'),
                79: ('L18by', 'L19P3'),
                80: ('L18b', 'L22P3'),
                81: ('P_mei1,L19t[]2',
                     'L19t[P_mei1]2'),
                82: ('L19t[P_mei1]2',
                     'P_mei1,L19s[]2'),
                83: ('P_mei2,L19s[]3',
                     'L19s[P_mei2]3'),
```

```
                    84: ('L19s[P_mei2]3',
                         'P_mei2,L19b[]3'),
                    85: ('L19bxy', 'L19b'),
                    86: ('L19bxx', 'L19bx'),
                    87: ('L19byy', 'L19by'),
                    88: ('L19bx', 'L13P3'),
                    89: ('L19by', 'L20P3'),
                    90: ('L19b', 'L20P3'),
                    91: ('L20t[a]4', 'L21t'),
                    92: ('L20t', 'L25t'),
                    93: ('L21t', 'L22t[a]4'),
                    95: ('P_cop,L22t[]4', 'L22t[P_cop]4'),
                    96: ('L22t[P_cop]4', 'P_cop,L22u[]4'),
                    97: ('L22r', 'L22s[a]8'),
                    98: ('L22ss', 'L22s'),
                    99: ('L22us', 'L23h'),
                    100: ('L22u', 'L23h')}}},
 'objects': {},
 'plasmids': {'P_asi1',
              'P_asi2',
              'P_asi8',
              'P_cer',
              'P_cop',
              'P_ig1',
              'P_ig2',
              'P_mei1',
              'P_mei2',
              'P_mul1',
              'P_mul2',
              'P_mul3'},
 'rules': {1: ('P_asi8[L1t]1', '[P_asi8,L1t]1'),
           2: ('[P_asi8,L2h]1', 'P_asi8[L2t]1'),
           3: ('P_cer[L2t]1', '[P_cer,L2t]1'),
           4: ('[P_cer,L3h]1', 'P_cer[L3t]1'),
           5: ('P_ig1[L3t]1', '[P_ig1,L3P1]1'),
           6: ('P_ig2[L3P1]1', '[P_ig2,L3t]1'),
           7: ('[P_ig1,L25P3]1', 'P_ig1[L25h]1'),
           8: ('[P_ig1,L4P3]1', 'P_ig1[L4h]1'),
           9: ('[P_ig2,L25h]1', 'P_ig2[L25t]1'),
           10: ('[P_ig2,L4h]1', 'P_ig2[L4t]1'),
           11: ('P_asi1[L4t]1', '[P_asi1,L4t]1'),
           12: ('[P_asi1,L5h]1', 'P_asi1[L5t]1'),
           13: ('P_ig1[L5t]1', '[P_ig1,L5P1]1'),
           14: ('P_ig2[L5P1]1', '[P_ig2,L5t]1'),
           15: ('[P_ig1,L6P3]1', 'P_ig1[L6h]1'),
           16: ('[P_ig1,L8P3]1', 'P_ig1[L8h]1'),
           17: ('[P_ig2,L6h]1', 'P_ig2[L6t]1'),
           18: ('[P_ig2,L8h]1', 'P_ig2[L8t]1'),
           19: ('P_cer[L6t]1', '[P_cer,L6t]1'),
           20: ('[P_cer,L7h]1', 'P_cer[L7t]1'),
           21: ('P_cer[L8t]1', '[P_cer,L8t]1'),
           22: ('[P_cer,L9h]1', 'P_cer[L9t]1'),
           23: ('P_asi2[L9t]1', '[P_asi2,L9t]1'),
           24: ('[P_asi2,L10h]1', 'P_asi2[L10t]1'),
           25: ('P_asi2[L10t]1', '[P_asi2,L10t]1'),
           26: ('[P_asi2,L11h]1', 'P_asi2[L11t]1'),
           27: ('P_asi1[L11t]1', '[P_asi1,L11t]1'),
           28: ('[P_asi1,L12h]1', 'P_asi1[L12t]1'),
           29: ('P_mei1[L12t]1', '[P_mei1,L12P1]1'),
```

```
175             30: ('P_mei2[L12P1]1', '[P_mei2,L12t]1'),
176             31: ('[P_mei1,L22P3]1', 'P_mei1[L22h]1'),
177             32: ('[P_mei1,L13P3]1', 'P_mei1[L13h]1'),
178             33: ('[P_mei2,L22h]1', 'P_mei2[L22t]1'),
179             34: ('[P_mei2,L13h]1', 'P_mei2[L13t]1'),
180             35: ('P_cop[L13t]1', '[P_cop,L13t]1'),
181             36: ('[P_cop,L14h]1', 'P_cop[L14t]1'),
182             37: ('P_cer[L14t]1', '[P_cer,L14t]1'),
183             38: ('[P_cer,L15h]1', 'P_cer[L15t]1'),
184             39: ('P_mul1[L15t]1', '[P_mul1,L15P1]1'),
185             40: ('P_mul2[L15P1]1', '[P_mul2,L15P2]1'),
186             41: ('P_mul3[L15P2]1', '[P_mul3,L15t]1'),
187             42: ('[P_mul1,L16P3]1', 'P_mul1[L16P4]1'),
188             43: ('[P_mul2,L16P4]1', 'P_mul2[L16h]1'),
189             44: ('[P_mul3,L16h]1', 'P_mul3[L16t]1'),
190             45: ('P_cer[L16t]1', '[P_cer,L16t]1'),
191             46: ('[P_cer,L17h]1', 'P_cer[L17t]1'),
192             47: ('P_ig1[L18t]1', '[P_ig1,L18P1]1'),
193             48: ('P_ig2[L18P1]1', '[P_ig2,L18t]1'),
194             49: ('[P_ig1,L22P3]1', 'P_ig1[L22h]1'),
195             50: ('[P_ig1,L19P3]1', 'P_ig1[L19h]1'),
196             51: ('[P_ig2,L22h]1', 'P_ig2[L22t]1'),
197             52: ('[P_ig2,L19h]1', 'P_ig2[L19t]1'),
198             53: ('P_mei1[L19t]1', '[P_mei1,L19P1]1'),
199             54: ('P_mei2[L19P1]1', '[P_mei2,L19t]1'),
200             55: ('[P_mei1,L20P3]1', 'P_mei1[L20h]1'),
201             56: ('[P_mei1,L13P3]1', 'P_mei1[L13h]1'),
202             57: ('[P_mei2,L20h]1', 'P_mei2[L20t]1'),
203             58: ('[P_mei2,L13h]1', 'P_mei2[L13t]1'),
204             59: ('P_cop[L22t]1', '[P_cop,L22t]1'),
205             60: ('[P_cop,L23h]1', 'P_cop[L23t]1')}}}
```

Listing A.3: Sistema P con plásmidos compilado. $log_2 8$

APÉNDICE B
Objetivos de Desarrollo Sostenible (ODS)

En este apéndice se identifican y analizan los Objetivos de Desarrollo Sostenible (ODS) que se ven impactados directa o indirectamente por el desarrollo y resultados de este proyecto. A continuación, se presenta una evaluación detallada del grado de relación del trabajo con cada uno de los ODS.

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. Fin de la pobreza.				×
ODS 2. Hambre cero.				×
ODS 3. Salud y bienestar.	×			
ODS 4. Educación de calidad.		×		
ODS 5. Igualdad de género.				×
ODS 6. Agua limpia y saneamiento.				×
ODS 7. Energía asequible y no contaminante.	×			
ODS 8. Trabajo decente y crecimiento económico.				×
ODS 9. Industria, innovación e infraestructuras.	×			
ODS 10. Reducción de las desigualdades.				×
ODS 11. Ciudades y comunidades sostenibles.				×
ODS 12. Producción y consumo responsables.			×	
ODS 13. Acción por el clima.			×	
ODS 14. Vida submarina.				×
ODS 15. Vida de ecosistemas terrestres.				×
ODS 16. Paz, justicia e instituciones sólidas.				×
ODS 17. Alianzas para lograr objetivos.				×

- **ODS 3: Salud y bienestar**

  El ODS 3 busca garantizar una vida sana y promover el bienestar para todos en todas las edades. Este trabajo contribuye significativamente a este objetivo al explorar la computación con plásmidos, una técnica que puede

tener aplicaciones directas en la biomedicina y la biotecnología. La capacidad de los plásmidos para realizar operaciones complejas a nivel molecular puede ser aprovechada en el desarrollo de nuevas terapias y medicamentos, mejorando así la salud y el bienestar de las personas.

- **ODS 4: Educación de calidad**

  El ODS 4 se centra en asegurar una educación inclusiva, equitativa y de calidad, y promover oportunidades de aprendizaje durante toda la vida para todos. Este proyecto tiene un impacto medio en este objetivo, ya que fomenta la investigación y el desarrollo en el ámbito de la computación no convencional. Además, proporciona una base teórica y práctica que puede ser utilizada en la educación superior y en programas de formación avanzada, contribuyendo así a la mejora de la calidad educativa.

- **ODS 7: Energía asequible y no contaminante**

  El ODS 7 busca garantizar el acceso a una energía asequible, fiable, sostenible y moderna para todos. La computación con plásmidos, al ser una técnica que opera a nivel molecular, tiene el potencial de ser altamente eficiente en términos energéticos. La implementación de sistemas de computación biológica puede reducir significativamente el consumo de energía en comparación con los sistemas de computación tradicionales, contribuyendo así a la sostenibilidad energética.

- **ODS 9: Industria, innovación e infraestructuras**

  El ODS 9 promueve la construcción de infraestructuras resilientes, la industrialización inclusiva y sostenible, y la innovación. Este trabajo tiene un impacto alto en este objetivo, ya que introduce un modelo de computación innovador que puede transformar la industria de la biotecnología y la informática. La computación con plásmidos abre nuevas posibilidades para el desarrollo de tecnologías avanzadas y la creación de infraestructuras más eficientes y sostenibles.

- **ODS 1, 2, 5, 6, 8, 10, 11, 12, 13, 14, 15, 16 y 17**

  Estos ODS tienen un impacto bajo o no proceden en el contexto de este trabajo. Aunque la computación con plásmidos puede tener aplicaciones indirectas en algunos de estos objetivos, como la mejora de la eficiencia en la producción y el consumo responsables (ODS 12) o la contribución a la acción por el clima (ODS 13), el impacto directo de este proyecto en estos objetivos es limitado.

En resumen, el Trabajo de Fin de Grado "Simulación de modelos computacionales mediante computación con plásmidos" contribuye de manera significativa a varios Objetivos de Desarrollo Sostenible, especialmente en los ámbitos de la salud y el bienestar, la educación de calidad, la energía asequible y no contaminante, y la industria, innovación e infraestructuras. Este proyecto no solo amplía el conocimiento en el campo de la computación no convencional, sino que también sienta las bases para futuras investigaciones y desarrollos que pueden tener un impacto positivo en la sociedad y el medio ambiente.