UNIVERSIDAD DE SEVILLA
Dpto. de Ciencias de la Computación
e Inteligencia Artificial

Complejidad y Universalidad en Modelos de Computación Celular

Memoria presentada por
Álvaro Romero Jiménez
para optar al grado de Doctor
por la Universidad de Sevilla

Álvaro

Álvaro Romero Jiménez

V.º B.º El Director de la Tesis

D. Mario de J. Pérez Jiménez

Sevilla, Marzo de 2003

A mi madre

Agradecimientos

Una Tesis Doctoral no es el resultado de un trabajo individual, sino que en su elaboración participan de diversas maneras muchas personas.

Desde luego, entre ellas nadie más fundamental que Mario, mi director de Tesis. Él ha estado ahí desde mi llegada al Departamento, guiándome para que no me desviara del camino correcto y trabajando igual o más que todos nosotros juntos. Le estaré siempre agradecido.

No puedo dejar tampoco de mencionar a Fernando, por sus atinados consejos. Además, él estuvo involucrado, o más bien debería decir que fue el causante de que en este Departamento se investiguen modelos de Computación Natural.

Al Grupo de Computación Natural y al Departamento de Ciencias de la Computación de la Universidad de Sevilla les agradezco su apoyo constante. En particular, Carmen no solo me apoya, sino que me aguanta todos los días. Gracias.

A mi madre, por ser una Madre (así, con mayúsculas). A mi familia, por estar ahí. A mi familia política, por ser como es. A Macarena, porque con ella he aprendido a vivir.

Y, finalmente, a mis amigos, por ser mis amigos.

Índice general

Introducción

Hoy en día los ordenadores resultan indispensables. A pesar de su «simplicidad», pues básicamente no son más que una colección de interruptores, cada uno de los cuales solo admite dos estados posibles (apagado o encendido), son capaces de realizar tareas realmente complejas y de resolver problemas bastante complicados.

En 1984, la revista TIME publicó un artículo de portada sobre programas de ordenador. En él el editor de cierta revista de ordenadores se cita diciendo:

> *Pon la clase correcta de programa en un ordenador, y hará lo que quieras que haga. Puede haber límites en lo que puedas hacer con las propias máquinas, pero no hay límites en lo que puedes hacer con los programas.*

Desgraciadamente, esta aseveración es completamente errónea, tal y como nos enseña la *Teoría de la Computación* [56]. Esta disciplina de estudio se desarrolla a partir de la década de los 30, y su germen se encuentra en lo que se conoce como el *programa finitista* de David Hilbert, cuyo objetivo era probar la consistencia de la Aritmética y en el que proponía usar la finitud de las pruebas matemáticas para establecer que no se podían derivar contradicciones, así como en el *Entscheidungsproblem*, sobre la decidibilidad de la lógica de primer orden.

Sin embargo, por un lado Kurt Gödel, con sus teoremas de incompletitud, demuestra la imposibilidad de llevar a cabo el programa finitista de Hilbert. Por otro lado, a principios de la década de los treinta el propio Kurt Gödel, Alonzo Church, Stephen Kleene y Alan Turing formalizan los conceptos de algoritmo y de función computable, e introducen los primeros modelos teóricos de computación. En 1936, A. Church y A. Turing, de manera independiente, dan una respuesta negativa al *Entscheidungsproblem*, mostrando la *existencia de problemas que no pueden ser resueltos mediante procesos mecánicos* en los modelos diseñados, lo que proporciona una limitación importante al

método de formalización del concepto de algoritmo.

En la década de los cincuenta, aparecen las primeras máquinas de cálculo programables de propósito general (es decir, los ordenadores electrónicos) y, con ellas, surgen los que podríamos denominar informalmente *modelos de computación práctica*, propiciando el desarrollo de la *Teoría de la Complejidad* [55], que básicamente estudia la cantidad de recursos computacionales (usualmente tiempo y/o espacio) necesarios para resolver un problema abstracto de forma mecánica, con el objetivo de clasificar dichos problemas en función de su «dificultad». Así, es interesante hacer notar la *existencia de problemas que para su resolución necesitan una cantidad de recursos que excede las capacidades de cualquier ordenador*, independientemente de cuál sea el procedimiento utilizado para resolverlo.

Surge entonces la necesidad de encontrar nuevos modelos que de alguna forma nos permitan solucionar instancias de dichos problemas *intratables* de tamaño «relativamente grande». La noción de *Computación Paralela* [13] supuso un notable avance, ya que para algunos problemas «difíciles» existen algoritmos paralelos que permiten mejorar en varios órdenes de magnitud el tiempo empleado en resolverlos. Así, la miniaturización de las componentes físicas de las máquinas se convierte en un objetivo primordial. Hacia finales de la década de los cincuenta, Richard Feynman [12] introduce el concepto teórico de computación a nivel molecular y lo postula como una innovación revolucionaria en la carrera por la miniaturización.

Otro camino seguido, en cierto modo relacionado con el anterior, es la *Computación Natural.* Esta se inspira en el funcionamiento de los organismos vivos y tiene como objetivo fundamental la simulación e implementación de los procesos dinámicos que se dan en la Naturaleza y que son susceptibles de ser considerados como procedimientos de cálculo.

Dentro de este campo de investigación se enmarcan los siguientes modelos de computación:

- La *Computación Evolutiva* [57], que usa modelos computacionales de procesos evolutivos como elementos claves en el diseño e implementación de sistemas de resolución de problemas por ordenador.

- Las *Redes Neuronales Artificiales* [10], originalmente inspiradas en el funcionamiento del cerebro humano, que son redes de procesadores simples, conectados por canales de comunicación, que operan solo en sus datos locales y en los datos recibidos por dichos canales. Adicionalmen-

te, la mayor parte de estos modelos disponen de un procedimiento de «entrenamiento» de alguna clase.

Es interesante hacer notar que a partir de estos dos modelos se han implementado algoritmos prácticos para la resolución de determinados tipos de problemas en los ordenadores electrónicos convencionales.

- La *Computación Molecular basada en ADN* [11], que se sirve de la gran densidad de información que son capaces de almacenar las moléculas de ADN, así como del alto grado de paralelismo con que se realizan operaciones entre ellas.

  Esta disciplina tiene su origen teórico en el modelo de sistemas de recombinación (*splicing*) propuesto por Tom Head [14] en 1987, aunque es a partir de 1994 cuando conoce un gran desarrollo. En efecto, a finales de ese año Leonard Adleman [1] resuelve una instancia concreta del problema del camino hamiltoniano (que, como es bien sabido, es un problema **NP**–completo) manipulando moléculas de ADN en el laboratorio, mostrando de esta forma la posibilidad práctica de codificar y manejar información mediante secuencias de ADN a fin de resolver problemas matemáticos *especialmente difíciles.*

Tal y como L. Adleman demostró, la computación molecular basada en ADN permite resolver problemas en el laboratorio, operando con las moléculas de ADN *in vitro.* El objetivo es lograr hacerlo *in vivo*, utilizando alguna especie de «hardware biológico».

- Por último, la *Computación Celular con Membranas* [58], dentro de la cual se enmarca este trabajo, y que se fundamenta en la consideración de que los procesos que tienen lugar en el interior de una célula se pueden interpretar como procedimientos de cálculo.

  Introducida a finales de 1998 por Gheorghe Păun [32], es un modelo de computación no determinista, de tipo paralelo y distribuido. Esquemáticamente, podemos considerar un sistema de computación celular con membrana, o sistema P, como un dispositivo constituido por una estructura de membranas en la cual las regiones delimitadas por las membranas contienen objetos que se desplazan y evolucionan de acuerdo a determinadas reglas especificadas.

Con respecto a este nuevo modelo de computación no convencional hay que destacar que no existe por el momento ninguna implementación práctica, ni en medios electrónicos, tal y como ocurre con los Algoritmos Evolutivos y las Redes Neuronales Artificiales, ni en medios biológicos, ya sea in vitro, como ocurre con la Computación Molecular basada en ADN, ya sea in vivo, lo que parece todavía más complicado, dado el estado actual de nuestro conocimiento acerca del funcionamiento de una célula.

Contenido de la memoria

Esta memoria está estructurada en dos partes bien diferenciadas. En la primera de ellas se intenta establecer un marco formal general para el estudio de los sistemas de computación celular con membranas. Además, se estudia la potencia computacional de diversas variantes de estos sistemas a través de modelos clásicos, utilizando, por tanto, una vía muy diferente de los lenguajes formales, que son los considerados habitualmente. En la segunda parte iniciamos, a partir de ideas discutidas por miembros del *Grupo de Investigación en Computación Natural* de la Universidad de Sevilla [27] con el fundador de la disciplina, G. Păun, el desarrollo de una Teoría de la Complejidad Computacional para sistemas P, con el doble objetivo de determinar cuáles son las características más adecuadas de estos dispositivos a la hora de resolver problemas complejos, y de disponer de una herramienta que nos permita atacar el problema $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ por medios no convencionales.

Resumimos a continuación el contenido de cada uno de los capítulos de este trabajo.

El **capítulo 1** establece los conceptos preliminares que serán necesarios a lo largo de la memoria.

El **capítulo 2** presenta una introducción a los sistemas de computación celular con membranas, así como una descripción de los sistemas P de transición, que es el modelo considerado por G. Păun en el artículo fundacional de la disciplina, por lo que cronológicamente fue el primero en aparecer.

A continuación, se establece un esquema formal de definición de sistemas de computación celular, fijando los elementos que determinan la sintaxis y los que determinan la semántica de estos modelos.

El **capítulo 3** comienza con la formalización del concepto de estructura de membranas, marco general donde se realizan las computaciones en los sistemas P. Posteriormente se definen con detalle las propiedades que deben

cumplir los elementos de un sistema de computación celular para obtener un sistema P de transición con salida externa, modelo básico con el que trabajamos, y del cual se estudian diversas variantes en el resto de capítulos de la primera parte de esta memoria.

El **capítulo 4** está dedicado al estudio de los sistemas P de transición con salida externa como dispositivos de aceptación de lenguajes. Además, dada una máquina de Turing determinista, se diseña un sistema de este tipo que la simula y se realiza una verificación formal de dicha simulación, de donde se concluye la completitud computacional de estos sistemas.

En el **capítulo 5** se consideran los sistemas P de transición con salida externa como dispositivos de generación de lenguajes y, basándonos en los sistemas del capítulo anterior, se establece la completitud computacional de estos sistemas. Para ello, dada una máquina de Turing se construye un sistema de este tipo que genera, de manera no determinista, todas las posibles cadenas de entrada de la máquina, pasando a simular a continuación el funcionamiento de esta sobre dichas cadenas.

En el **capítulo 6** se estudian los sistemas P de transición con salida externa como dispositivos de cálculo de funciones y se justifica su capacidad para calcular cualquier función recursiva. Para ello se describe cómo realizar composiciones, iteraciones y minimizaciones con estos sistemas.

En el **capítulo 7** se presentan los sistemas P de transición con salida externa como dispositivos de generación de conjuntos de tuplas de números naturales. A partir de los sistemas considerados en el capítulo anterior, mostramos cómo con estos sistemas es posible generar cualquier conjunto diofántico, que, en virtud del teorema **MRDP** (de Matiyasevich, Robinson, Davis y Putnam), coinciden con los conjuntos recursivamente enumerables.

La segunda parte de la memoria está dedicada a desarrollar una *Teoría de la Complejidad* en modelos de computación celular. En el **capítulo 8** se introducen tres clases distintas de complejidad, dependiendo de si los problemas son resueltos por familias de sistemas sin membrana de entrada, por sistemas con membrana de entrada, o por familias de sistemas con membrana de entrada.

En el **capítulo 9** se estudia la relación existente entre los sistemas P de transición con salida externa como dispositivos de aceptación de lenguajes y la conjetura $\mathbf{P} \neq \mathbf{NP}$ de la Teoría clásica de la Complejidad. Se justifica que la clase de problemas resoluble en tiempo polinomial por estos sistemas coincide con la clase **P**, lo que permite establecer una caracterización de la

conjetura antes citada en términos de irresolubilidad en tiempo polinomial de problemas por los sistemas descritos.

El **capítulo 10** introduce un nuevo tipo de sistemas de computación celular, con un nivel adicional de paralelismo en la forma de reglas que permiten la división de membranas: los sistemas P con membranas activas. Para ello se detallan las propiedades que deben cumplir los elementos de un sistema de computación celular para que den lugar a un sistema de este tipo.

Finalmente, en el **capítulo 11** se diseñan sendas familias de sistemas P con membranas activas que resuelven el problema **SAT** y el problema **VALIDITY** en tiempo *lineal*, respectivamente. De donde se deduce que la clase de problemas resolubles en tiempo polinomial por familias de los sistemas considerados contiene las clases **NP** y **coNP**.

Aportaciones

A continuación citamos algunas de las aportaciones de esta memoria que consideramos más relevantes.

1. Desarrollo de un esquema formal de sistemas de computación celular con membranas que sirve de marco general para el estudio de la mayoría de variantes estudiadas hasta ahora (sección 2.2 del capítulo 2).

2. Presentación de una formalización de los sistemas P de transición con salida externa, que constituye una versión mejorada de la mostrada en [54] (capítulo 3).

3. Demostración de la completitud computacional de diversas variantes de estos últimos sistemas a través de modelos clásicos y, por tanto, sin hacer uso de la teoría de lenguajes formales. En concreto,

   - Construcción de sistemas P de transición con salida externa, considerados como dispositivos de aceptación, que simulan cualquier máquina de Turing determinista (capítulo 4).
   - Construcción de sistemas P con salida externa, como dispositivos de generación de lenguajes, que generan cualquier lenguaje recursivamente enumerable (capítulo 5).

- Definición de operaciones entre sistemas P de transición con salida externa como dispositivos de cálculo de funciones: composición, iteración y minimización. Estas operaciones permiten probar que dichos sistemas calculan cualquier función recursiva parcial. El diseño de operaciones entre sistemas de computación celular representa una novedad en este campo (capítulo 6).
- Construcción de sistemas P con salida externa, como dispositivos de generación de conjuntos, que generan cualquier conjunto diofántico (capítulo 7).

4. Desarrollo de una metodología para la verificación de la completitud computacional de las dos primeras variantes de sistemas P de transición con salida externa comentadas en el punto anterior (sección 4.3 del capítulo 4 y sección 5.3 del capítulo 5).

5. Primera aproximación al desarrollo de una *Teoría de la Complejidad Computacional* en sistemas de computación celular (capítulo 8).

6. Demostración de las propiedades básicas de las clases de complejidad introducidas y de las relaciones existentes entre ellas (sección 8.2 del capítulo 8).

7. Descripción de la clase **P** a través de problemas de decisión resolubles en tiempo polinomial por familias de sistemas P de transición con salida externa (capítulo 9).

8. Caracterización de la conjetura $\mathbf{P} \neq \mathbf{NP}$ en términos de irresolubilidad en tiempo polinomial de problemas en modelos de computación celular con membranas (sección 9.3 del capítulo 9).

9. Presentación de una formalización de los sistemas P con membranas activas (capítulo 10).

10. Diseño y verificación de familias de sistemas P con membranas activas, como dispositivos de aceptación, que resuelven los problemas **SAT** y **VALIDITY** en tiempo lineal (capítulo 11).

Capítulo 1

Preliminares

En este capítulo se incluyen las nociones básicas, junto con sus propiedades más relevantes, que utilizaremos a lo largo de la memoria. La mayoría de los conceptos y resultados introducidos pueden encontrarse en [53].

Comenzamos presentando, en la sección 1.1, unos rudimentos sobre alfabetos y lenguajes [21]; en la sección 1.2 comentamos brevemente los multiconjuntos, una extensión de los conjuntos que permite tener elementos repetidos; a continuación, en la sección 1.3, exponemos los fundamentos que necesitaremos acerca de grafos y árboles [26]; posteriormente, incluimos en la sección 1.4 las clases de funciones (primitiva) recursivas, cuyas propiedades pueden encontrarse demostradas en [8]; la importancia de los conjuntos diofánticos se explica en la sección 1.5, y, con más detalle, en [25]; por último, se muestran las máquinas de Turing, en la sección 1.6, y las clases de complejidad asociadas, en la sección 1.7, siguiendo [28].

1.1. Alfabetos y lenguajes

Un *alfabeto* es un conjunto finito no vacío. Un *símbolo* es un elemento de un alfabeto. Una *cadena* sobre un alfabeto es una sucesión finita de símbolos de dicho alfabeto. La cadena vacía, denotada por λ, es la cadena sin ningún símbolo.

La *longitud* de una cadena w sobre un alfabeto Γ, denotada por $|w|$, es el número de símbolos que aparecen en w. El número de ocurrencias de un símbolo $a \in \Gamma$ en la cadena w se denota por $|w|_a$.

La *concatenación* de dos cadenas $w_1 = a_1 a_2 \ldots a_m$ y $w_2 = b_1 b_2 \ldots b_n$ es la

cadena $w_1w_2 = a_1a_2 \ldots a_mb_1b_2 \ldots b_n$ obtenida «pegando» la cadena w_2 a la derecha de la cadena w_1. Dado un alfabeto Γ, denotamos por Γ^* el conjunto de todas las cadenas sobre Γ y por Γ^+ el conjunto de todas las cadenas no vacías sobre Γ; es decir, Γ^* es el monoide libre generado por Γ bajo la operación de concatenación y $\Gamma^+ = \Gamma^* \setminus \{\lambda\}$.

Si $w = w_1w_2$, con $w_1, w_2 \in \Gamma^*$, entonces w_1 se denomina un *prefijo* de w y w_2 se denomina un *sufijo* de w. Si $w = w_1w_2w_3$, con $w_1, w_2, w_3 \in \Gamma^*$, entonces w_2 se denomina una *subcadena* de w.

Un *lenguaje (formal)*, L, sobre un alfabeto Γ es un subconjunto de Γ^*. Si L no contiene la cadena vacía entonces se dice que es *λ-libre*. La concatenación de dos lenguajes, L_1 y L_2, es el lenguaje L_1L_2 que contiene todas las cadenas $w = w_1w_2$, donde $w_1 \in L_1$ y $w_2 \in L_2$.

El conjunto de símbolos que aparecen en una cadena w se denota por $alph(w)$. Para un lenguaje $L \subseteq \Gamma^*$, denotamos $alph(L) = \bigcup_{w \in L} alph(w)$. El *vector de Parikh* asociado a la cadena w sobre el alfabeto $\Gamma = \{a_1, \ldots, a_n\}$ es $\Psi_\Gamma(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$ (obsérvese que la ordenación de los símbolos de Γ es relevante). La *aplicación de Parikh* asociada a Γ se define entonces por $\Psi_\Gamma(L) = \{\Psi_\Gamma(w) : w \in L\}$, para cada $L \subseteq \Gamma^*$.

Una *gramática no restringida* es una tupla $G = (N, T, S, P)$ tal que: N es un alfabeto finito no vacío de *símbolos no terminales*; T es un alfabeto finito no vacío, disjunto con N, de *símbolos terminales*; S es un símbolo no terminal llamado *axioma*; P es un conjunto finito de *reglas de producción* de la forma $\alpha \to \beta$, donde $\alpha \in (N \cup T)^*N(N \cup T)^*$ y $\beta \in (N \cup T)^*$.

Diremos que la cadena y *deriva directamente* de la cadena x respecto a la gramática G, y lo notamos por $x \Rightarrow_G y$, si existe una regla de producción $u \to v \in P$ y existen cadenas $w_1, w_2 \in (N \cup T)^*$ tales que $x = w_1uw_2$ e $y = w_1vw_2$. Una *derivación* de la cadena y a partir de la cadena x es una sucesión finita de derivaciones directas, $x \Rightarrow_G z_1, \ldots, z_n \Rightarrow_G y$. Si existe una derivación de la cadena x en la cadena y, entonces lo notaremos por $x \Rightarrow_G^* y$.

El *lenguaje generado* por G es $L(G) = \{y \in T^* : S \Rightarrow_G^* y\}$. La clase de lenguajes generados por las gramáticas no restringidas es la clase RE de *lenguajes recursivamente enumerables.*

Atendiendo a la forma de sus reglas, las gramáticas $G = (N, T, S, P)$ se clasifican como sigue: G es *sensible al contexto* si cada regla de producción $\alpha \to \beta \in P$ cumple que $\alpha = u_1Au_2$ y $\beta = u_1xu_2$, con $u_1, u_2 \in (N \cup T)^*$, $A \in N$ y $x \in (N \cup T)^+$ (se permite la producción $S \to \lambda$, siempre que S no aparezca en el consecuente de las reglas en P); G es *libre de contexto* si

cada regla de producción $\alpha \to \beta \in P$ cumple que $\alpha \in N$; G es *lineal* si cada regla de producción $\alpha \to \beta \in P$ cumple que $\alpha \in N$ y $\beta \in T^* \cup T^*NT^*$; G es *regular* si cada regla de producción $\alpha \to \beta \in P$ cumple que $\alpha \in N$ y $\beta \in T \cup TN \cup \{\lambda\}$.

Denotamos por FIN la familia de lenguajes finitos, y por REG, LIN, CF y CS las familias de lenguajes generados por gramáticas regulares, lineales, libres de contexto y sensibles al contexto, respectivamente. La *jerarquía de Chomsky* es la siguiente sucesión de inclusiones estrictas:

$$FIN \subset REG \subset LIN \subset CF \subset CS \subset RE$$

1.2. Multiconjuntos

Un *multiconjunto* sobre un conjunto, A, es una aplicación $m : A \to \mathbb{N}$, siendo $\mathbb{N}$ el conjunto de los números naturales. Un subconjunto $B \subseteq A$ se puede identificar con el multiconjunto sobre A dado por la aplicación $m(a) = 1$, si $a \in B$, y $m(a) = 0$, si $a \notin B$. Denotamos por $\mathbf{M}(A)$ el conjunto de todos los multiconjuntos sobre A. Obsérvese que si A es un conjunto finito no vacío, entonces $\mathbf{M}(A)$ es un conjunto infinito numerable.

El *soporte* de $m \in \mathbf{M}(A)$ es el conjunto $supp(m) = \{a \in A : m(a) > 0\}$. Un multiconjunto se dice que es finito si su soporte es un conjunto finito. Análogamente, un multiconjunto se dice que es vacío, y se denota por $m = \emptyset$, si su soporte es el conjunto vacío.

El *tamaño* de un multiconjunto finito, m, se define por $|m| = \sum_{a \in A} m(a)$ (obsérvese que esta suma es finita, ya que solo contiene un número finito de términos no nulos).

Podemos definir los siguientes predicados y operaciones entre multiconjuntos, donde $m_1, m_2 \in \mathbf{M}(A)$ y $n \in \mathbb{N}$:

- *Inclusión:* $m_1 \leq m_2 \Leftrightarrow \forall a\big(a \in A \to m_1(a) \leq m_2(a)\big)$.
- *Inclusión estricta:* $m_1 < m_2 \Leftrightarrow m_1 \leq m_2 \wedge m_1 \neq m_2$.
- *Unión:* $(m_1 + m_2)(a) = m_1(a) + m_2(a)$, para cada $a \in A$.
- *Diferencia:* $(m_1 - m_2)(a) = \text{máx}\{m_1(a) - m_2(a), 0\}$, para cada $a \in A$.
- *Amplificación:* $(n \cdot m_1)(a) = n \cdot m_1(a)$, para cada $a \in A$.

Es fácil observar que cualquier multiconjunto finito, m, con soporte $supp(m) = \{a_1, \ldots, a_n\}$, puede representarse como una cadena w sobre A con $|w|_{a_i} = m(a_i)$, para cada $i = 1, \ldots, n$. Claramente, todas las cadenas obtenidas a partir de w permutando sus símbolos también representan a m. El multiconjunto vacío, $\emptyset$, se representa por la cadena vacía, λ. Esta será la representación que usaremos a lo largo de este trabajo.

1.3. Grafos y árboles

Un *grafo no dirigido* es un par formado por un conjunto V, cuyos elementos se llaman *vértices* o *nodos*, y un conjunto E de pares no ordenados $\{u, v\}$ tales que $u, v \in V$ y $u \neq v$, cuyos elementos se llaman *aristas* (ambos conjuntos finitos, a menos que se declare lo contrario). Dado un grafo no dirigido G, notaremos por $V(G)$ el conjunto de sus vértices y por $E(G)$ el conjunto de sus aristas.

Los *extremos* de una arista son los vértices que une. Una arista e es *incidente* con un vértice v si v es un extremo de e. Dos vértices u y v son *adyacentes* si existe una arista cuyos extremos son u y v. Dos aristas son *adyacentes* si tienen un extremo común.

Un *subgrafo* de un grafo G es un grafo H cuyos conjuntos de vértices y aristas son subconjuntos de $V(G)$ y $E(G)$, respectivamente, de forma tal que para cada arista e en $E(H)$, los extremos de e (tal como ocurren en G) están en $V(H)$. El *subgrafo de G inducido por un conjunto de vértices* $V \subseteq V(G)$ es el subgrafo cuyo conjunto de vértices es V y cuyo conjunto de aristas es $E = \{\{u, v\} \in E(G) : u, v \in V\}$. El *subgrafo de G inducido por un conjunto de aristas* $E \subseteq E(G)$ es el subgrafo cuyo conjunto de vértices es $V = \{u \in V(G) : \exists e \in E \ (u \text{ es un extremo de } e)\}$ y cuyo conjunto de aristas es E.

Un *camino*, en un grafo G, de longitud k que conecta un vértice u con un vértice v es una sucesión de vértices $\{x_i\}_{i \leq k}$ tales que $u = x_0$, $v = x_k$ y $\{x_i, x_{i+1}\} \in E(G)$, para todo $i = 0, \ldots, k-1$. Un camino es *simple* si todas los aristas y todos los vértices que lo forman (excepto posiblemente el primer vértice y el último) son distintos. Un *ciclo* es un camino simple de longitud al menos uno que empieza y termina en el mismo vértice.

Un grafo G se dice *conexo* si para cualquier par de vértices de G existe un camino en G que los contiene. Un *árbol* es un grafo conexo sin ciclos. Un *bosque* es un conjunto de árboles. Un *árbol enraizado* es un árbol con un

vértice distinguido (la *raíz* del árbol).

Nótese que en todo árbol para cualesquiera dos vértices, u y v, existe un único camino que conecta u con v.

Un *hijo* de un vértice v en un árbol enraizado T es un vértice que es inmediato sucesor de v en el único camino que conecta la raíz con ese vértice. Notaremos por $ch_T(v)$ el conjunto de hijos de v. Un *vértice interno* en un árbol enraizado es un vértice con hijos. Una *hoja* es un vértice que no tiene hijos.

El *padre* de un vértice v, distinto de la raíz, en un árbol enraizado T es un vértice que es inmediato predecesor de v en el único camino que conecta la raíz con v. Notaremos por ft_T la función que lleva cualquier vértice distinto de la raíz en su padre y no está definida para la raíz. Dos vértices con el mismo padre se dicen *hermanos.*

Un *descendiente* de un vértice v en un árbol enraizado T es el propio v o cualquier vértice que es un sucesor de v en un camino que conecta la raíz de T con ese vértice. El *subárbol principal* en un vértice v es el subgrafo inducido por el conjunto de descendientes de v, y que tiene v como raíz.

La *profundidad* de un vértice en un árbol enraizado es la longitud del único camino que conecta la raíz del árbol con ese vértice. La *altura* del árbol es la profundidad máxima que alcanzan sus vértices. El *nivel de profundidad* de un vértice es la diferencia entre la altura del árbol y la profundidad del vértice.

Un árbol enraizado se puede representar por su conjunto de vértices junto con su función padre.

1.4. Funciones recursivas

En Matemáticas existen básicamente dos orientaciones para caracterizar una clase, $\mathcal{F}$, de funciones:

1. *Orientación descriptiva:* se determina $\mathcal{F}$ como la clase de funciones que satisfacen ciertas propiedades. Esta orientación es la que usualmente se utiliza para definir conjuntos (cuando se definen por alguna propiedad característica).

2. *Orientación generativa:* se muestra la clase de funciones ofreciendo dos elementos:

   - Un conjunto de funciones básicas.
   - Una serie de procedimientos de definición de funciones, a partir de otras.

   La clase $\mathcal{F}$ se define entonces como la menor clase de funciones que contiene las funciones básicas y es cerrada bajo los procedimientos de definición.

Dada una clase $\mathcal{F}$ definida de la segunda forma podemos usar el método de *demostración por inducción* para probar que todas las funciones de $\mathcal{F}$ cumplen una determinada propiedad. En efecto, se verifica el siguiente resultado:

Supongamos que Φ es una propiedad acerca de funciones verificando lo siguiente:

- *Toda función básica de la clase $\mathcal{F}$ satisface la propiedad Φ.*
- *Si una función se obtiene aplicando a funciones que satisfacen la propiedad Φ algún método de definición permitido para la clase, entonces la nueva función también satisface Φ.*

Entonces toda función de la clase $\mathcal{F}$ verifica la propiedad Φ.

Sean entonces la *función constante igual a cero*, $\mathcal{O} : \mathbb{N} \to \mathbb{N}$, dada por $\mathcal{O}(x) = 0$, para todo $x \in \mathbb{N}$; la *función sucesor*, $\mathcal{S} : \mathbb{N} \to \mathbb{N}$, dada por $\mathcal{S}(x) = x + 1$, para todo $x \in \mathbb{N}$; las *funciones proyecciones*, $\Pi_i^n : \mathbb{N}^n \to \mathbb{N}$, con $1 \leq i \leq n$, dadas por $\Pi_i^n(x_1, \ldots, x_n) = x_i$, para toda $(x_1, \ldots, x_n) \in \mathbb{N}^n$. El conjunto de estas funciones es el conjunto de *funciones básicas.*

Definimos la *composición* de las funciones parciales $f : \mathbb{N}^m \dashrightarrow \mathbb{N}^n$ y $g_1 : \mathbb{N}^r \dashrightarrow \mathbb{N}^{s_1}, \ldots, g_t : \mathbb{N}^r \dashrightarrow \mathbb{N}^{s_t}$, con $s_1 + \cdots + s_t = m$, como la función $C(f; g_1, \ldots, g_t) : \mathbb{N}^r \dashrightarrow \mathbb{N}^n$ dada por

$$C(f; g_1, \ldots, g_t)(x_1, \ldots, x_r) = f(g_1(x_1, \ldots, x_r), \ldots, g_t(x_1, \ldots, x_r))$$

para todo $(x_1, \ldots, x_r) \in \mathbb{N}^r$.

Obsérvese que si $f, g_1, \ldots, g_t$ son funciones totales, entonces la composición de dichas funciones también es total.

Diremos que la función parcial $f : \mathbb{N}^{n+1} - \to \mathbb{N}$ se obtiene por *recursión primitiva* a partir de $g : \mathbb{N}^n - \to \mathbb{N}$ y $h : \mathbb{N}^{n+2} - \to \mathbb{N}$ si está caracterizada por las condiciones siguientes: para todo $(x_1, \ldots, x_n) \in \mathbb{N}^n$ y todo $y \in \mathbb{N}$,

$$\begin{aligned} f(x_1, \ldots, x_n, 0) &= g(x_1, \ldots, x_n) \\ f(x_1, \ldots, x_n, y+1) &= h(x_1, \ldots, x_n, y+1, f(x_1, \ldots, x_n, y)) \end{aligned}$$

Se puede mostrar que si g y h son dos funciones con las aridades descritas, entonces existe una única función de aridad $n+1$ que satisface las dos condiciones anteriores. Además, obsérvese que si g y h son funciones totales, entonces la función f también es total.

La clase $\mathcal{PR}$ de *funciones primitivas recursivas* es la menor clase de funciones que contiene las funciones básicas y es cerrada bajo composición y recursión primitiva.

La operación de *minimización no acotada* o *μ-recursión* aplicada a la función parcial $f : \mathbb{N}^{n+1} - \to \mathbb{N}$ nos proporciona la función $Min(f) : \mathbb{N}^n - \to \mathbb{N}$ dada por

$$Min(f)(x_1, \ldots, x_n) = \begin{cases} y_{x_1,\ldots,x_n}, & \text{si } y_{x_1,\ldots,x_n} \text{ existe} \\ \text{no definida}, & \text{en otro caso} \end{cases}$$

para todo $(x_1, \ldots, x_n) \in \mathbb{N}^n$, donde

$$\begin{aligned} y_{x_1,\ldots,x_n} = \text{mín}\{y \in \mathbb{N} : \forall z < y\big(&f \text{ está definida sobre } (x_1, \ldots, x_n, z)\big) \wedge \\ &f(x_1, \ldots, x_n, y) = 0\} \end{aligned}$$

Diremos que una función total $f : \mathbb{N}^{n+1} \to \mathbb{N}$ es *regular* si para toda tupla $(x_1, \ldots, x_n) \in \mathbb{N}^n$ existe $y \in \mathbb{N}$ tal que $f(x_1, \ldots, x_n, y) = 0$. Obsérvese que la función minimización de una función regular es una función total.

La clase $\mathcal{P}$ de *funciones recursivas parciales* es la menor clase de funciones que contiene las funciones básicas y es cerrada bajo composición, recursión primitiva y minimización no acotada. La clase $\mathcal{R}$ de *funciones recursivas totales* es la menor clase de funciones que contiene las funciones básicas y es cerrada bajo composición, recursión primitiva y minimización no acotada, esta última aplicada solamente a funciones regulares.

La operación de *iteración* aplicada a la función parcial $f : \mathbb{N}^n - \to \mathbb{N}^n$ nos proporciona la función $It(f) : \mathbb{N}^{n+1} - \to \mathbb{N}^n$ caracterizada por las condiciones siguientes: para todo $(x_1, \ldots, x_n) \in \mathbb{N}^n$ y todo $y \in \mathbb{N}$,

$$It(f)(x_1, \ldots, x_n, 0) = (x_1, \ldots, x_n)$$
$$It(f)(x_1, \ldots, x_n, y+1) = It(f)(f(x_1, \ldots, x_n), y)$$

No es difícil probar que en las tres clases de funciones introducidas arriba podemos sustituir la operación de recursión primitiva por la operación de iteración (ver [8]).

Finalmente, diremos que un conjunto es *recursivo* si su función característica es una función recursiva. Por otra parte, diremos que es *recursivamente enumerable* si coincide con el dominio de alguna función recursiva. Obsérvese que todo conjunto recursivo es recursivamente enumerable.

1.5. Conjuntos diofánticos

En la sección inaugural del *2º Congreso Internacional de Matemáticos*, celebrado en París en 1900, David Hilbert planteó una lista de 23 problemas, con la intención de resaltar los más importantes problemas matemáticos no resueltos que el siglo XX iba a heredar del siglo XIX. En dicha lista aparecía un único problema de decisión, el Problema Décimo:

Determinación de la Resolubilidad de una Ecuación Diofántica

> Dada una ecuación diofántica con cualquier número de incógnitas y con coeficientes enteros: *idear un proceso conforme al cual pueda determinarse en un número finito de operaciones si la ecuación es resoluble en números enteros.*

El propio enunciado del Problema Décimo indica claramente que para su resolución es necesario relacionar la teoría de ecuaciones diofánticas con la teoría de la computación. Así, un *conjunto diofántico* es un conjunto de tuplas de números naturales, $A \subseteq \mathbb{N}^m$, tal que existe un polinomio $P(a_1, \ldots, a_m, x_1, \ldots, x_n)$ con coeficientes enteros verificando que A coincide con el conjunto

$$\{(a_1, \ldots, a_m) \in \mathbb{N}^m : \exists (x_1, \ldots, x_n) \in \mathbb{N}^n \big(P(a_1, \ldots, a_m, x_1, \ldots, x_n) = 0\big)\}$$

Es relativamente fácil probar que todo conjunto diofántico es recursivamente enumerable.

Culminando los trabajos previos de Julia Robinson, Martin Davis y Hillary Putnam, Yuri Matiyasevich demuestra en 1970 la inclusión contraria.

Teorema (MRDP). *Todo conjunto recursivamente enumerable es un conjunto diofántico.*

Este resultado, junto con la existencia de conjuntos recursivamente enumerables que no son recursivos, proporciona una respuesta negativa al Problema Décimo de Hilbert.

1.6. Máquinas de Turing

Una *máquina de Turing determinista*, como *dispositivo de aceptación*, es una tupla

$$TM = (Q, q_0, q_N, q_Y, \Gamma, \Sigma, B, \triangleright, \delta)$$

donde

- $Q \cap \Gamma = \emptyset$.
- $Q = \{q_N, q_Y, q_0, \ldots, q_n\}$ es un conjunto finito de *estados*.
- $q_0 \in Q$ es el *estado inicial*; $q_N, q_Y \in Q$ son los *estados finales*: el primero es el *estado de rechazo*, el segundo es el *estado de aceptación*.
- $\Gamma = \{B, \triangleright, a_1, \ldots, a_m\}$ es el *alfabeto de trabajo*.
- $\Sigma = \{a_1, \ldots, a_p\}$, con $p \leq m$, es el *alfabeto de entrada*. Obsérvese que se verifica que $\Sigma \subsetneq \Gamma$.
- $B \in \Gamma \setminus \Sigma$ es el *símbolo blanco*; $\triangleright \in \Gamma \setminus \Sigma$ es el *símbolo más a la izquierda*. Denotamos $a_B = B$ y $a_0 = \triangleright$.
- $\delta : \big(Q \setminus \{q_N, q_Y\}\big) \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$ es la *función de transición*.

Podemos imaginar que TM consta de una sola *cinta* dividida en *celdas*, infinita hacia la derecha, pero con una primera celda. Cada celda contendrá o el símbolo blanco, indicando que la celda está vacía, u otro símbolo del alfabeto de trabajo. El símbolo $\triangleright$ jugará un papel especial en el sentido de

que siempre marcará la celda más a la izquierda y no aparecerá en ningún otro lugar.

Los símbolos se leen y escriben en las celdas de la cinta a través de una *cabeza* lectora/escritora, que en cada momento analiza una y solo una de las celdas. La cabeza se puede mover a lo largo de la cinta a la izquierda y a la derecha, o permanecer quieta (salvo en la primera casilla, donde el desplazamiento a la izquierda no está permitido).

La siguiente acción de la máquina está totalmente determinada, a través de la función de transición, δ, por su estado actual y por el símbolo que la cabeza analiza en ese instante. Denotemos, para $q_i \in Q \setminus \{q_N, q_Y\}$ y $a_j \in \Gamma$, $\delta(q_i, a_j) = (q_{Q(i,j)}, a_{A(i,j)}, D(i,j))$, que se interpreta como sigue: el nuevo estado de la máquina será $q_{Q(i,j)}$, en la celda que ha sido analizada se escribirá el símbolo $a_{A(i,j)}$, y la cabeza lectora pasa a analizar la celda de la izquierda si $D(i,j) = -1$, la misma celda si $D(i,j) = 0$, o la celda de la derecha si $D(i,j) = 1$. Entonces, la función de transición debe satisfacer las siguientes condiciones:

$$\big(A(i,0) = 0 \wedge D(i,0) = 1\big) \wedge \forall j \big(j \neq 0 \rightarrow A(i,j) \neq 0\big)$$

Es decir, el símbolo $\triangleright$ siempre dirige la cabeza hacia la derecha, y nunca es borrado de la celda más a la izquierda ni escrito en una celda distinta de esta. La función de transición, δ, viene a ser el «programa» de la máquina.

Una *descripción instantánea* de una máquina de Turing es una cadena uqv, donde $u \in \Gamma^*$, $q \in Q$ y $v \in \Gamma^*(\Gamma \setminus \{B\}) \cup \{B\}$, que se interpreta como sigue: el estado de la máquina de Turing es q, la cinta contiene la cadena uv y la cabeza está analizando el primer símbolo de v.

Decimos que uqv es una descripción instantánea de *parada* si $q = q_Y$ (descripción instantánea de *aceptación*) o $q = q_N$ (descripción instantánea de *rechazo*). Dada $x \in \Sigma^*$, la descripción instantánea *inicial* sobre x, denotada por I_x, es $q_0 \triangleright x$.

Podemos definir, sobre el conjunto de las descripciones instantáneas de TM, la relación $\vdash_{TM}$ como sigue:

$$\begin{array}{ll}
uaqbv \vdash_{TM} upacv \Leftrightarrow \delta(q,b) = (p,c,-1) & uaqb \vdash_{TM} upac \Leftrightarrow \delta(q,b) = (p,c,-1) \\
uaqbv \vdash_{TM} uapcb \Leftrightarrow \delta(q,b) = (p,c,0) & uaqb \vdash_{TM} uapc \Leftrightarrow \delta(q,b) = (p,c,0) \\
uaqbv \vdash_{TM} uacpv \Leftrightarrow \delta(q,b) = (p,c,1) & uaqb \vdash_{TM} uacpB \Leftrightarrow \delta(q,b) = (p,c,1)
\end{array}$$

donde $p, q \in Q$ y $a, b, c \in \Gamma$.

Sea $\vdash_{TM}^*$ la clausura reflexiva y transitiva de $\vdash_{TM}$. Dado $x \in \Sigma^*$, decimos que TM *para* sobre x, denotado por $TM \downarrow x$, si existe una descripción instantánea de parada, I_x^f, tal que $I_x \vdash_{TM}^* I_x^f$ (se prueba fácilmente que, en tal caso, la descripción instantánea I_x^f es única).

Sea L un lenguaje sobre el alfabeto Σ. Decimos que la máquina TM *decide* el lenguaje L si para cualquier cadena $x \in \Sigma^*$ se tiene que $TM \downarrow x$ y, además,

$$x \in L \Leftrightarrow I_x^f \text{ es una descripción instantánea de aceptación}$$

Decimos que la máquina TM *acepta* el lenguaje L si para cualquier cadena $x \in \Sigma^*$ se tiene que

$$x \in L \Leftrightarrow M \downarrow x \wedge I_x^f \text{es una descripción instantánea de aceptación}$$

La clase de lenguajes aceptados por máquinas de Turing deterministas coincide con la clase de lenguajes recursivamente enumerables.

Podemos considerar también máquinas de Turing deterministas como *dispositivos de cálculo* de la siguiente manera: sustituimos el estado de aceptación, q_Y, y el estado de rechazo, q_N, por un único estado de parada, q_h. Ahora una descripción instantánea, uqv, es de parada si $q = q_h$. Entonces decimos que la máquina TM *calcula* la función $f : \Sigma^* \to \Gamma^*$ (o que la función f es *computable* por la máquina TM) si para cualquier cadena $x \in \Sigma^*$ se tiene que TM para sobre x y, además, en la descripción instantánea de parada, I_x^f, la cinta contiene exáctamente la cadena $f(x)$.

1.7. Clases de complejidad

Sea TM una máquina de Turing determinista, bien como dispositivo de aceptación, bien como dispositivo de cálculo. Dada una cadena x sobre el alfabeto de entrada Σ de la máquina, definimos el *tiempo empleado por* TM *sobre* x como el número de pasos realizados por la máquina de Turing hasta llegar a la configuración de parada I_x^f, si TM para sobre x, o como ∞, en caso contrario.

De forma análoga a las máquinas de Turing introducidas en la sección anterior, es posible considerar *máquinas de Turing con k cintas*, en lugar de con una sola. Sin embargo, aunque pueden facilitar el diseño de una máquina que

decida o acepte un lenguaje, o que calcule una función, este tipo de máquinas no aporta ninguna característica adicional, ya que pueden ser simuladas por máquinas de Turing con una sola cinta, con un aumento en tiempo empleado por estas únicamente cuadrático con respecto a las originales.

Otra posible variante, esta vez sí innovadora, viene dada por las *máquinas de Turing no deterministas*. En estas máquinas la función de transición proporciona más de una posibilidad de respuesta; es decir, consideramos que $\delta : \big(Q \setminus \{q_N, q_Y\}\big) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{-1, 0, 1\})$. La definición de la relación $\vdash_{TM}$ se adecúa entonces convenientemente, de donde resulta que para una cadena x sobre el alfabeto de entrada Σ el conjunto de descripciones instantáneas de parada I_x^f tales que $I_x \vdash_{TM}^* I_x^f$ ya no tiene por qué ser unitario.

Dados un lenguaje L y una máquina de Turing TM no determinista, decimos que TM *decide* el lenguaje L si para cualquier $x \in \Sigma^*$ se verifica que $x \in L$ si y solo si existe, al menos, *una* descripción instantánea de aceptación, I_x^f, tal que $I_x \vdash_{TM}^* I_x^f$. Por otra parte, definimos el *tiempo empleado por* TM *sobre* x como el mínimo número de pasos realizados por una computación de TM con entrada x que sea de aceptación, o cero si ninguna computación de TM con entrada x es de aceptación.

Sea $f : \mathbb{N} \to \mathbb{N}$ una función computable por una máquina de Turing. Decimos que una máquina de Turing TM decide un lenguaje L en tiempo $O(f)$ si TM decide L y, además, para cualquier cadena x sobre el alfabeto de entrada Σ, el tiempo empleado por TM sobre x está acotado por $f(|x|)$.

Un *problema de decisión*, X, es un par (I_X, θ_X) tal que I_X es un lenguaje sobre un alfabeto finito, cuyos elementos se denominan *instancias* del problema, y θ_X es una función total booleana sobre I_X. Decimos que una máquina de Turing *resuelve* el problema X en tiempo $O(f)$ si decide el lenguaje $\{x \in I_X : \theta(x) = 1\}$ en tiempo $O(f)$.

La clase de complejidad $\mathbf{TIME}(f)$ es la clase de todos los lenguajes que son decidibles por una máquina de Turing determinista en tiempo $O(f)$. Análogamente, la clase de complejidad $\mathbf{NTIME}(f)$ es la clase de todos los lenguajes que son decidibles por una máquina de Turing no determinista en tiempo $O(f)$. Es evidente que $\mathbf{TIME}(f) \subseteq \mathbf{NTIME}(f)$.

Las dos clases de complejidad en tiempo más importantes son las clases

$$\mathbf{P} = \bigcup_{k>0} \mathbf{TIME}(n^k) \qquad \text{y} \qquad \mathbf{NP} = \bigcup_{k>0} \mathbf{NTIME}(n^k)$$

Por otra parte, la clase **coNP** es la clase de aquellos lenguajes tales que

su complementario pertenece a la clase **NP**. En general, dada una clase de complejidad, $\mathcal{C}$, se define la clase complementaria, co$\mathcal{C}$, como la clase que contiene los lenguajes complementarios de los lenguajes en $\mathcal{C}$.

Es claro que $\mathbf{P} \subseteq \mathbf{NP}$. El problema más importante hoy en día en Teoría de la Complejidad es determinar si dicha inclusión es o no estricta. Todos los indicios apuntan a que existen problemas resolubles en tiempo polinomial por una máquina de Turing no determinista, pero que no lo son por una máquina de Turing determinista; es decir, que las dos clases anteriores son distintas.

Decimos que un lenguaje $L_1 \subseteq \Sigma_1^*$ es *reducible* en tiempo polinomial a un lenguaje $L_2 \subseteq \Sigma_1^*$ si existe una función total, $R : \Sigma_1^* \to \Sigma_2^*$, computable en tiempo polinomial por una máquina de Turing determinista, tal que para todo $x \in \Sigma_1^*$ se tiene que $x \in L_1$ si y solo si $R(x) \in L_2$. La función R se llama una *reducción en tiempo polinomial* de L_1 en L_2. Se verifica que las clases **P**, **NP** y **coNP** son cerradas bajo reducibilidad en tiempo polinomial.

Sean $\mathcal{C}$ una clase de complejidad y $L \in \mathcal{C}$. Decimos que L es $\mathcal{C}$–completo si cualquier lenguaje $L' \in \mathcal{C}$ se puede reducir en tiempo polinomial a L. Como ejemplos significativos se tiene que el problema **SAT** es un problema **NP**–completo y que el problema **VALIDITY** es un problema **coNP**–completo.

Se verifica el siguiente resultado: si dos clases de complejidad $\mathcal{C}$ y $\mathcal{C}'$ son ambas cerradas bajo reducibilidad en tiempo polinomial y existe un lenguaje L que es completo tanto para $\mathcal{C}$ como para $\mathcal{C}'$, entonces $\mathcal{C} = \mathcal{C}'$. De esta forma, si encontráramos un lenguaje **NP**–completo que pertenezca a **P** o del que pudiéramos probar que no pertenece a **P**, entonces habríamos resuelto el problema $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

Parte I

Completitud computacional

Capítulo 2

Sistemas de computación celular con membranas

En este capítulo introducimos el concepto de sistema de computación celular con membranas. Para ello, en la sección 2.1 comenzamos exponiendo brevemente cuál es la idea que trata de capturar este modelo, para posteriormente describir, de manera informal, los sistemas P de transición presentados por G. Păun en el artículo fundacional de la disciplina [32], así como distintas variantes de estos que han ido surgiendo desde su creación. En la sección 2.2 mostramos una manera viable de formalizar la sintaxis y la semántica de estos sistemas, con el objetivo de que dicha formalización abarque el mayor número de variantes posible, muchas de las cuales pueden encontrarse en [35].

2.1. De la célula a los sistemas P

En los últimos años el campo de investigación denominado con el nombre genérico de *Computación Natural* ha conocido un enorme desarrollo. Los trabajos dentro de este campo estudian la manera de imitar el modo en el que la Naturaleza «calcula», aprehendiendo nuevos paradigmas y modelos de computación a partir de ella. Son varias las áreas de la Computación Natural que están ya bien establecidas: las *redes neuronales*, los *algoritmos genéticos*, la *computación molecular basada en ADN* y la *computación celular con membranas*.

La presente memoria se emarca dentro de esta última disciplina, que se inspira en el funcionamiento de la célula como organismo vivo capaz de

procesar y generar información. En efecto, las células están constituidas por diferentes vesículas, delimitadas por membranas. Dentro de estas vesículas tienen lugar reacciones químicas que provocan no solo la transformación de los elementos contenidos en ellas, sino también un flujo de dichos elementos entre las diferentes estructuras que integran la célula. Estos procesos a nivel celular pueden ser interpretados como procedimientos de cálculo.

A la hora de diseñar un sistema formal que abstraiga este esquema de funcionamiento existen dos caminos a seguir: podemos describir lo más detalladamente posible los procesos que tienen lugar, con la idea de que el sistema nos proporcione un mayor conocimiento acerca de las células; o bien podemos extraer las principales características que definen una célula, con la esperanza de obtener un nuevo modelo de computación, sencillo pero potente, que nos ayude a resolver problemas que resultan especialmente complicados y/o complejos en otros modelos más clásicos. Este último camino fue el seguido en octubre de 1998 por G. Păun al introducir los *sistemas P de transición* [32], creando así una nueva especialidad dentro de la Computación Natural: la *computación celular con membranas.*

La noción de sistema P deriva directamente de la componente fundamental de una célula: la membrana. Nótese que todas las subestructuras internas que componen una célula (incluso la propia célula) están delimitadas por membranas. No obstante, estas membranas no generan compartimentos estancos, sino que permiten el paso de compuestos químicos a través de ellas, la mayoría de las veces de forma selectiva. Es dentro de estos compartimentos donde se producen las reacciones químicas en la célula.

Básicamente, como se ilustra en la figura 2.1, un sistema P consta de un conjunto de *membranas*, normalmente organizado jerárquicamente en una *estructura de membranas.* Existe una *membrana piel* que engloba todas las demás, separando al sistema del *entorno externo.* Además, a las membranas que no contienen otras membranas en su interior se les denomina *membranas elementales.* Las *regiones* delimitadas por las membranas (es decir, el espacio acotado por una membrana y las membranas inmediatamente interiores, si existe alguna) pueden contener ciertos *objetos*, que en general se permite que aparezcan repetidos. Mediante la aplicación de determinadas *reglas de evolución* asociadas a las membranas, estos objetos pueden transformarse en otros, e incluso pueden pasar de una región a otra adyacente, atravesando la membrana que las separa.

Obsérvese que un sistema de computación celular con membranas no está

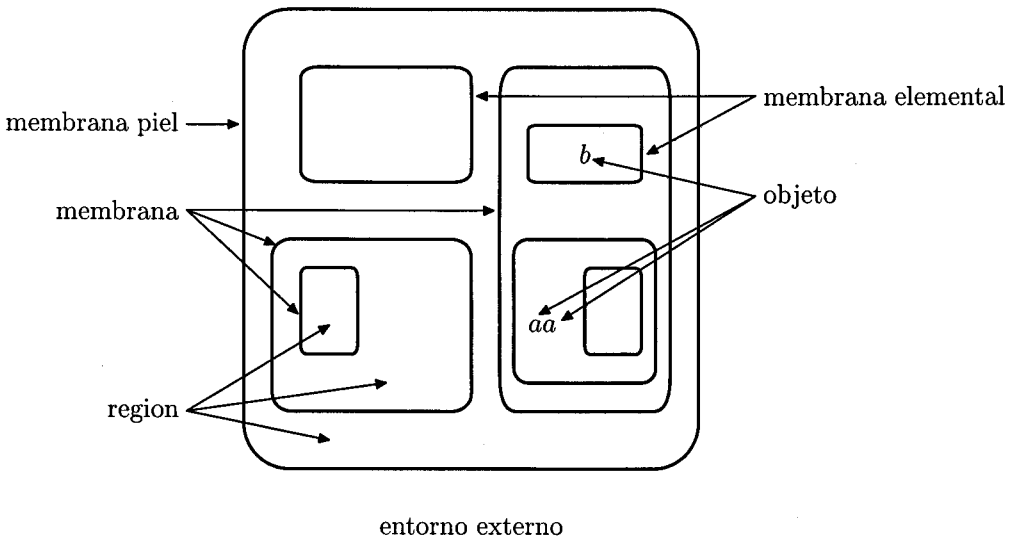

Figura 2.1: *Sistema de computación celular con membranas*

descrito a través de un lenguaje de programación; es decir, no proporciona una serie de operaciones básicas susceptibles de ser secuenciadas sobre un dato de entrada para obtener un resultado final. En su lugar, un sistema P es un dispositivo cuya ejecución, al modo de las máquinas de Turing, modifica el contenido de las distintas componentes que lo integran hasta llegar, en su caso, a un estado de parada, en el que el sistema deja de funcionar. En este sentido, dicha ejecución podría decirse que es independiente del usuario puesto que, una vez construido el sistema, no es necesario, en principio, dirigirlo.

Sistemas P de transición

Veamos a continuación una descripción, detallada aunque informal, de un modelo de computación celular con membranas, tal y como fue introducido por G. Păun en [32]: los sistemas P de transición.

Comenzamos estableciendo el marco dentro del cual se realizarán las computaciones del modelo: una estructura de membranas, μ, es un elemento del conjunto MS, definido por recursión como sigue:

1. $[\,] \in MS$.

2. Si $\mu_1, \ldots, \mu_k \in MS$, entonces $[\,\mu_1 \ldots \mu_k\,] \in MS$.

Cada par de corchetes coincidentes que aparezca en μ se denomina membrana. El par de corchetes externo se denomina membrana piel mientras que los pares de corchetes que no contienen otros en su interior se denominan membranas elementales. El grado de μ es el número de membranas que contiene.

Dadas dos membranas mb_1 y mb_2 de μ tales que la segunda está «contenida» en la primera, diremos que dichas membranas son adyacentes si no existe ninguna membrana contenida en mb_1 y tal que contiene a mb_2. En ese caso también diremos que mb_1 es la membrana padre de mb_2 y que mb_2 es una membrana hija de mb_1.

A continuación especificamos la sintaxis del modelo: un sistema P de transición de grado $p \geq 1$ es una tupla

$$\Pi = \big(\Gamma, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, (R_1, \rho_1), \ldots, (R_p, \rho_p), om\big)$$

donde:

- Γ es un alfabeto finito.
- μ_Π es una estructura de membranas.

  Las membranas de μ_Π están etiquetadas de forma unívoca usando los números naturales desde 1 hasta p (esto significa, desde luego, que μ_Π contiene exactamente p membranas, que a partir de ahora identificamos con su etiqueta).
- $\mathcal{M}_i$ es un multiconjunto finito sobre Γ asociado a la membrana i del sistema, para cada $i = 1, \ldots, p$.
- R_i es un conjunto finito de reglas de evolución de transición asociado a la membrana i del sistema, para cada $i = 1, \ldots, p$.

  Una regla de evolución de transición es un par (u, v), habitualmente representado $u \to v$, donde u es una cadena sobre Γ y $v = v'$ o $v = v'\delta$, siendo v' una cadena sobre $\Gamma \times \big(\{here, out\} \cup \{in_i : i = 1, \ldots, p\}\big)$.
- ρ_i es un orden parcial estricto sobre R_i, para cada $i = 1, \ldots, p$, que establece una relación de prioridad entre las reglas de R_i.
- om es un número natural entre 1 y p que indica la membrana de salida del sistema.

Para describir la semántica del modelo se introduce previamente el concepto de configuración del sistema, de donde seguirá, una vez establecida la forma en que se aplican las reglas de transición, la noción de computación del sistema.

Una configuración de Π es una tupla $(\mu, M_{i_1}, \ldots, M_{i_q})$ tal que

- μ es la estructura de membranas que se obtiene a partir de μ_Π al eliminar las membranas distintas de las membranas i_1 a i_q. La membrana piel no se puede eliminar, por lo que coincide en ambas estructuras.

- M_{i_j} es un multiconjunto finito sobre Γ, para cada $j = 1, \ldots, q$.

La configuración inicial de Π es la tupla $(\mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p)$.

La aplicación de una regla a una membrana presente en una configuración se realiza como sigue: dada una regla $u \to v$ asociada a una membrana i, los objetos en u se eliminan de la membrana i (esta debe contener, por tanto, suficientes objetos en la membrana para que la regla se pueda aplicar); entonces, para cada $(ob, out) \in v$ un objeto $ob \in \Gamma$ se incluye en la membrana padre de la membrana i (o abandona el sistema si la membrana i es la membrana piel); para cada $(ob, here) \in v$ un objeto $ob \in \Gamma$ se añade a la membrana i; para cada $(ob, in_j) \in v$ un objeto $ob \in \Gamma$ se introduce en la membrana j (teniendo en cuenta que si la membrana j no es una membrana hija de la membrana i, entonces la regla no se puede aplicar); finalmente, si $\delta \in v$, entonces la membrana i se disuelve; es decir, se elimina de la estructura de membranas, pasando sus objetos a la membrana padre y desapareciendo las reglas de evolución y relaciones de prioridad asociadas a ella (la membrana piel no se puede disolver).

Por otra parte, interpretamos las relaciones de prioridad entre las reglas en un sentido fuerte: dichas relaciones prohíben la aplicación de una regla si se puede aplicar otra de mayor prioridad. Otra posibilidad sería interpretarlas en un sentido débil, de forma que las relaciones de prioridad solamente indiquen un orden de aplicación de las reglas.

Dadas dos configuraciones, C y C', de Π, decimos que C' se obtiene a partir de C en un paso de transición, y lo escribimos $C \Rightarrow_\Pi C'$, si la segunda es el resultado de aplicar a la primera, en paralelo (de forma simultánea), y para todas las membranas al mismo tiempo, las reglas de evolución contenidas en un determinado (multi)conjunto de reglas asociadas a las membranas que aparecen en la estructura de membranas de C. En dicho (multi)conjunto se

indica el número de veces que se aplica cada una de las reglas y debe ser maximal, en el sentido de que, tras la aplicación de las reglas, en ninguna membrana permanezcan objetos sin evolucionar que puedan activar alguna de las reglas asociada a la membrana. Puesto que para una configuración usualmente existen más de un (multi)conjunto de reglas aplicable, los pasos de transición se realizan de manera no determinista.

Una computación $\mathcal{C}$ de Π es una sucesión (finita o infinita) de configuraciones, $\{C^i\}_{i<r}$, tal que

- C^0 es la configuración inicial de Π.
- $C^i \Rightarrow_\Pi C^{i+1}$, para todo $i < r-1$.
- O bien $r \in \mathbb{N}^+$ (es decir, es un número natural distinto de cero) y no existe ninguna regla que se pueda aplicar en ninguna de las membranas de C^{r-1} (en cuyo caso se dice que $\mathcal{C}$ es de parada, ha realizado $r-1$ pasos y C^{r-1} es su configuración de parada), o bien $r = \infty$ (en cuyo caso se dice que $\mathcal{C}$ no es de parada).

Diremos que una computación $\mathcal{C}$ es exitosa si es una computación de parada y, además, la membrana *om* aparece en C^{r-1} como membrana elemental.

Dada una computación exitosa de Π, podemos definir la salida de dicha computación como el tamaño del multiconjunto asociado a la membrana de salida del sistema en su configuración de parada. Entonces, un sistema P de transición es una máquina que genera el conjunto de números naturales, $N(\Pi)$, formado por las salidas de todas las computaciones exitosas de dicho sistema.

Variantes de sistemas P

Tras la introducción de los sistemas P de transición por G. Păun se ha producido un gran auge en el estudio de los sistemas de computación celular con membranas, habiendo aparecido multitud de variantes que buscan, unas veces, mayor eficiencia en la solución de problemas complejos y, otras veces, una mayor aproximación al modelo biológico real en el que se inspiran.

Dentro de las principales diferencias que podemos encontrar entre las distintas variantes podemos destacar las siguientes:

- Uso de cadenas de un determinado alfabeto como objetos básicos del modelo, en lugar de considerar objetos atómicos sin estructura interna.

- Uso de reglas que permiten crear o duplicar membranas.
- Uso de reglas que únicamente permiten la comunicación entre membranas, pero no la evolución de los objetos.
- Consideración de redes, en lugar de estructuras, de membranas.

Además, siempre podemos considerar, a partir de un modelo de computación celular con membranas, otros modelos que difieran del original en los siguientes aspectos:

- Existencia o no de una membrana de entrada en la que se introducen determinados objetos al inicio de una computación.
- La salida de una computación se recoge en una membrana (elemental o no) específica o en el entorno externo del sistema.

Finalmente, a la hora de fijar qué consideramos como salida de un sistema P encontramos múltiples posibilidades. Así, podemos establecer que dicho sistema genera conjuntos de números naturales, que genera o acepta lenguajes, que simula funciones, o cualquier otro comportamiento que resulte adecuado a nuestros propósitos.

2.2. Esquema formal de sistemas P

En esta sección presentamos un marco general de sistemas de computación celular con membranas que intenta capturar la mayoría de las variantes estudiadas de sistemas P. Para ello establecemos qué elementos determinan la sintaxis y qué elementos determinan la semántica de estos sistemas.

Definición 2.1. *La sintaxis de un sistema de computación celular con membranas viene precisada por una tupla*

$$MCS_{Sin} = (AL, MG, LB, ER, AE, MB)$$

donde:

- *AL es un conjunto finito de* alfabetos *finitos no vacíos.*
- *MG es un grafo finito de* membranas.

- *LB es un conjunto finito de* etiquetas.
- *ER es una función total de dominio el conjunto de etiquetas LB (la aplicación de* reglas de evolución *asociadas a las etiquetas).*
- *AE es una función total de dominio las membranas de MG (la aplicación de* elementos asociados *inicialmente a las membranas).*
- *MB es un conjunto de membranas distinguidas de MG.*

Respecto a la definición anterior hacemos notar a continuación las siguientes consideraciones:

1. El conjunto *AL* abarca todos los alfabetos y elementos señalados de dichos alfabetos que se usan en el sistema.

   En general, en un sistema P existe un alfabeto, que podemos llamar alfabeto de *trabajo*, a partir del cual se construyen los objetos usados en las computaciones del sistema. No obstante, en muchas variantes se usa un alfabeto de *salida* (por ejemplo, [39]), que determina el conjunto de objetos tenidos en cuenta a la hora de calcular la salida del sistema, mientras que en otras se incluyen en el sistema objetos construidos a partir de un alfabeto de *entrada* ([45], [51]).

   También podemos encontrar alfabetos de *catalizadores* [32], de *portadores* [24] y otros.

2. *MG* es el marco inicial dentro del cual se realizan las computaciones del sistema P. Normalmente es una *estructura de membranas* (es decir, un conjunto jerarquizado de membranas), pero en ocasiones consideramos un marco más general, como en los sistemas de membranas como tejidos ([35], sección 6.2) o los sistemas P neuronales ([35], sección 6.3), en los que las membranas se hallan interconectadas unas con otras.

3. *LB* representa el conjunto de símbolos usados para etiquetar las membranas del sistema P. Estas etiquetas se asimilan habitualmente a los nodos del grafo de membranas. No obstante, existen variantes, tales como los sistemas con membranas activas [34] o los sistemas con creación de membranas [18], que, a lo largo del tiempo, pueden modificar considerablemente la estructura de membranas inicial; así, en estos sistemas podemos encontrarnos con más de una membrana etiquetada de la misma forma.

4. ER es una aplicación que asocia a cada posible etiqueta de las membranas un conjunto finito de reglas de evolución, de forma que todas las membranas con idéntica etiqueta apliquen las mismas reglas. En algunas variantes [32] también debemos explicitar un orden parcial sobre el conjunto de reglas, que establece las prioridades de ejecución entre ellas.

5. AE es una aplicación que proporciona los elementos que están asociados inicialmente a cada una de las membranas del sistema. Entre los distintos elementos que encontramos en las diversas variantes podemos destacar: la etiqueta de la membrana, el contenido, ya sean símbolos, cadenas o incluso objetos bidimensionales y objetos dibujos [17]; la carga [34] y la permeabilidad [33].

6. MB es un conjunto de membranas relevantes del sistema. Así, es natural considerar dos grupos entre las distintas variantes de sistemas P: aquellos en los que al inicio de una computación se introducen una serie de objetos en una determinada *membrana de entrada* y aquellos en los que se trabaja solamente con los contenidos iniciales de las membranas.

   Por otra parte, en función de dónde se recoge la salida de las computaciones de los sistemas, podemos clasificar estos en otros dos grupos, dependiendo de si aquella se recoge en una determinada *membrana de salida*, o si, por el contrario, se recoge en el entorno externo del sistema.

Definición 2.2. *La semántica de un sistema de computación celular con membranas viene precisada por una tupla*

$$MCS_{Sem} = (Conf, IConf, TStep, Output)$$

donde:

- $Conf$ *es el conjunto de* configuraciones *del sistema.*
- $IConf \subseteq Conf$ *es el conjunto de* configuraciones iniciales *del sistema.*
- $TStep$ *es una función parcial de* $Conf$ *en* $\mathcal{P}(Conf)$ *(la función* paso de transición*).*
- *Output es una función parcial sobre el conjunto de computaciones del sistema (la aplicación* salida de las computaciones*).*

Respecto a la definición anterior hacemos notar a continuación las siguientes consideraciones:

1. Una configuración caracteriza completamente el estado en el que se encuentra el sistema P en un momento concreto de una computación. De esta forma, los elementos de $Conf$ deben ser tales que a partir de ellos podamos conseguir la información relevante que determine completamente dicho estado.

   Entre los elementos más habituales que debe contener una configuración destacamos los siguientes: el grafo de membranas obtenido por evolución del grafo de membranas inicial, MG (a veces no solo basta dicho grafo, sino que también hemos de considerar el entorno externo asociado a él, como en [45] y [51]); las etiquetas de LB asociadas a cada membrana de dicho grafo; los objetos contenidos en ellas; y otros elementos, según la variante de sistemas P considerada (como la carga en [34] o la permeabilidad en [33]). Es decir, las configuraciones deben contener todas las variables dinámicas del sistema.

2. $IConf$ está formado por aquellas configuraciones particulares que caracterizan el estado en el que se encuentra inicialmente el sistema P; es decir, antes de comenzar propiamente una computación. Lo usual es encontrarnos con una única configuración inicial en los sistemas P sin membrana de entrada, y con una configuración inicial por cada dato de entrada en los sistemas P con membrana de entrada.

3. Dada una configuración de un sistema de computación celular con membranas, Π, aplicando las reglas de evolución de dicho sistema adecuadamente según la variante considerada (es decir, usando la función $TStep$ concreta para esa variante), obtenemos, en general de forma no determinista, una nueva configuración. Si $C' \in TStep(C)$, entonces diremos que la configuración C' se ha obtenido en un paso de transición a partir de la configuración C, y lo notaremos por $C \Rightarrow_{\Pi} C'$.

   Una *configuración de parada* es una configuración para la cual $TStep$ no está definida. Es decir, es una configuración en la que no se puede aplicar ninguna regla de evolución, por lo que no es posible realizar un paso de transición a partir de ella.

4. A partir de una configuración inicial, un sistema de computación celular con membranas, Π, evoluciona de acuerdo a sus reglas de evolución.

De esta forma, se obtienen sucesiones de configuraciones del sistema que conforman el conjunto de sus computaciones. Formalmente, una *computación*, $\mathcal{C}$, de un sistema P es una sucesión de configuraciones, $\{C^i\}_{i<r}$, donde:

- C^0 es una configuración inicial del sistema.
- $C^i \Rightarrow_\Pi C^{i+1}$, para todo $i < r$.
- O bien $r \in \mathbb{N}^+$ y C^{r-1} es una configuración de parada, en cuyo caso se dice que $\mathcal{C}$ es una computación de *parada* y que ha realizado $r-1$ pasos, o bien $r = \infty$, en cuyo caso se dice que $\mathcal{C}$ no es de parada.

Dada una configuración inicial, un sistema P puede evolucionar, dependiendo de cómo se apliquen sus reglas de evolución, de muchas maneras distintas, o bien solo de una. Así, diremos que un sistema de computación celular con membranas es *determinista* si para cualquier configuración inicial del sistema, C, existe una única computación $\mathcal{C} = \{C^i\}_{i<r}$ tal que $C^0 = C$. En caso contrario diremos que el sistema es *no determinista.*

5. La salida de una computación viene dada por la aplicación *Output*. En general, solo producen una salida las computaciones de parada, aunque existen variantes en las que las computaciones infinitas también pueden generar un resultado ([29]).

Capítulo 3

Sistemas P con salida externa

Una primera clasificación de los sistemas P puede realizarse en función de dónde se recoge la salida de las computaciones del sistema. En este capítulo presentamos una formalización genérica de los sistemas P de transición con salida externa. Dichos sistemas se caracterizan por las siguientes propiedades: sus computaciones se realizan dentro del marco de una estructura de membranas; se trabaja con objetos símbolos; se aplican reglas de evolución de transición; la salida de las computaciones se recoge en el entorno externo de la estructura de membranas.

En la primera sección definimos qué entendemos por estructura de membranas y cómo asociar a estas estructuras un entorno externo. Las secciones 3.2 y 3.3 están dedicadas a fijar las propiedades que deben cumplir los elementos que determinan la sintaxis y la semántica de un sistema de computación celular para que obtengamos un sistema P de transición con salida externa. Para ello nos basamos en la formalización realizada en [48]. En la última sección mostramos algunas variantes sintácticas de estos sistemas.

3.1. Estructuras de membranas

Comenzamos estableciendo con precisión qué entendemos por estructura de membranas.

Definición 3.1. *Una* estructura de membranas *es un árbol enraizado en el que los nodos se denominan* membranas, *la raíz se denomina* piel *y las hojas se denominan* membranas elementales.

El grado *de una estructura de membranas es el número de membranas que contiene (es decir, el número de nodos del árbol).*

Por ejemplo, la estructura de membranas de la figura 2.1 en la página 19 se corresponde formalmente con el árbol enraizado que se muestra en la figura 3.1.

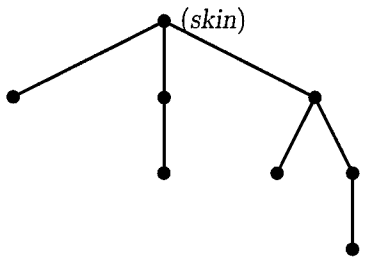

Figura 3.1: *Estructura de membranas*

La membrana piel de una estructura de membranas, a la que nos referiremos genéricamente usando la meta–etiqueta *skin*, aisla a la estructura de lo que se conoce como entorno externo de la estructura, al que nos referiremos con la meta–etiqueta *env*. En las variantes de sistemas P que vamos a considerar es en este entorno donde se recogerá la salida de las computaciones. Es por ello que debemos asociarlo de alguna manera a la estructura de membranas.

Definición 3.2. *Sea* $\mu = (V(\mu), E(\mu))$ *una estructura de membranas. La* estructura de membranas con entorno externo *asociada a* μ *es el árbol enraizado* $Ext(\mu)$ *donde*

- $V(Ext(\mu)) = V(\mu) \cup \{env\}$.
- $E(Ext(\mu)) = E(\mu) \cup \{\{env, skin\}\}$.
- *La raíz del árbol es el nodo env.*

El nuevo nodo se denomina entorno externo *de la estructura* μ.

Obsérvese que lo único que hacemos es añadir un nuevo nodo que representa al entorno externo y que, por tanto, solo es adyacente a la piel, mientras que la estructura de membranas original permanece inalterada. Así, la estructura de membranas con entorno externo asociada a la correspondiente a la figura 3.1 se muestra en la figura 3.2.

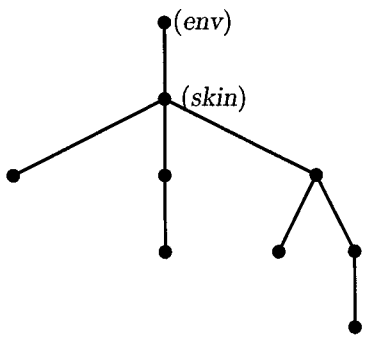

Figura 3.2: *Estructura de membranas con entorno externo*

3.2. Sintaxis de los sistemas P con salida externa

En esta sección se fijan las condiciones que deben cumplir los elementos de un sistema de computación con membranas para que den lugar a un sistema P de transición con salida externa.

En primer lugar especificamos el conjunto de reglas de evolución que serán aplicables en estos sistemas, y destacamos dos subconjuntos relevantes de dichas reglas.

Definición 3.3. *Sean μ una estructura de membranas y Γ un alfabeto finito no vacío. El conjunto de* reglas de evolución (de transición) *asociado a μ y Γ, que notaremos por $TER(\mu, \Gamma)$, es el conjunto de todas las posibles tuplas (u, v, δ), donde*

- *u es un multiconjunto sobre Γ.*
- *v es una aplicación de dominio $V(\mu) \cup \{here, out\}$ y rango contenido en $\mathbf{M}(\Gamma)$.*
- *$\delta \in \{y, n\}$.*

Notación. *Usualmente, dada una regla $(u, v, \delta) \in TER(\mu, \Gamma)$, identificaremos la aplicación v con la cadena sobre $\Gamma \times (\{here, out\} \cup \{in_l : l \in V(\mu)\})$ que contiene los siguientes símbolos:*

- *Para cada $l \in V(\mu)$ y cada $a \in v(l)$, un símbolo (a, in_l).*
- *Para cada $a \in v(here)$, un símbolo $(a, here)$ (o, más brevemente, solo escribiremos el símbolo a).*

- *Para cada* $a \in v(out)$, *un símbolo* (a, out).

Además, es usual notar la regla por $u \to v$, *en el caso de que* $\delta = n$, *y por* $u \to v\delta$, *en el caso de que* $\delta = y$.

Definición 3.4. *El conjunto de* reglas de evolución (de transición) no cooperativas, *que notamos por* $NCTER(\mu, \Gamma)$, *es el conjunto de aquellas tuplas* (u, v, δ) *de* $TER(\mu, \Gamma)$ *tales que* $|u| = 1$.

Dado un subconjunto $C \subseteq \Gamma$, *llamado conjunto de catalizadores, el conjunto de* reglas de evolución de transición catalíticas asociado a C, *que notamos por* $CTER(\mu, \Gamma, C)$, *es el conjunto de aquellas tuplas* (u, v, δ) *de* $TER(\mu, \Gamma)$ *tales que, o bien* $|u| = 1$, $C \cap u = \emptyset$ *y* $C \cap v = \emptyset$, *o bien* $|u| = 2$, $|C \cap u| = 1$ *y* $C \cap u = C \cap v$.

A continuación definimos qué entendemos por sistema P de transición con salida externa.

Definición 3.5. *Un* sistema P de transición con salida externa, Π, *es un sistema de computación celular con membranas tal que:*

- *AL consta de un alfabeto de trabajo,* Γ, *y un alfabeto de salida,* Λ, *de forma que* $\Lambda \subseteq \Gamma$. *Si el sistema tiene membrana de entrada, se incluye también un alfabeto de entrada,* $\Sigma \subseteq \Gamma$. *Además, existe un elemento distinguido* $\# \in \Gamma \setminus (\Sigma \cup \Lambda)$. *Por último, especificamos el conjunto* $C \subseteq \Gamma \setminus \Sigma$ *de catalizadores, aunque generalmente se omite cuando no consideramos reglas catalíticas.*

- *El marco de computación, MG, es una estructura de membranas,* μ_Π.

- *El conjunto de etiquetas, LB, que es posible asociar a las membranas, coincide con el conjunto de nodos de la estructura de membranas, de forma que hay una identificación unívoca entre nodo y etiqueta. Usualmente* $LB = \{1, \ldots, p\}$, *donde* p *es el grado de* μ_Π.

- *La aplicación ER asocia a cada etiqueta lb (y, por tanto, a cada membrana) un subconjunto finito* R_{lb} *(posiblemente vacío) de* $TER(\mu_\Pi, \Gamma)$ *y un orden parcial estricto* ρ_{lb} *(posiblemente vacío) sobre dicho subconjunto, que establece una prioridad de ejecución entre sus reglas. A menudo, dadas* $r_1, r_2 \in R_{lb}$, *en lugar de* $(r_1, r_2) \in \rho_{lb}$ *escribiremos* $r_1 > r_2$; *es decir,* r_1 *es una regla con mayor prioridad que* r_2.

- *La aplicación AE asocia a cada membrana, mb, de μ_Π un multiconjunto, $\mathcal{M}_{mb}$, sobre $\Gamma \setminus \Sigma$, que es el multiconjunto de objetos contenidos inicialmente en la membrana.*
- *MB contiene la membrana de entrada para los sistemas con entrada.*
- *La salida se recoge en el entorno externo de la estructura de membranas.*

Notación. *Notaremos un sistema P de transición con salida externa por una tupla*

$$\Pi = \big(\Sigma, \Gamma, \Lambda, C, \#, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, (R_1, \rho_1), \ldots, (R_p, \rho_p), im, ext\big)$$

teniendo en cuenta que el alfabeto de entrada Σ y la membrana de entrada im no se especifican en un sistema sin entrada, y lo mismo ocurre con el conjunto de catalizadores C cuando no usamos reglas catalíticas.

3.3. Semántica de los sistemas P con salida externa

A continuación vamos a detallar de qué forma evoluciona un sistema P de transición con salida externa atendiendo a los multiconjuntos de objetos contenidos en cada una de las membranas de su estructura de membranas, así como a las reglas de evolución asociadas a ellas.

Definición 3.6. *Sea Π un sistema P de transición con salida externa. Una configuración de Π es un par $C = \big(Ext(\mu), M\big)$ tal que verifica las siguientes condiciones:*

- *$\mu = (V(\mu), E(\mu))$ es una estructura de membranas.*
- *$Ext(\mu)$ es la estructura de membranas con entorno externo asociada a la estructura μ.*
- *El conjunto $V(\mu)$ de nodos de μ está incluido en el conjunto $V(\mu_\Pi)$ de nodos de μ_Π, y contiene la raíz de μ_Π.*
- *Las raíces de ambas estructuras de membranas coinciden.*
- *M es una aplicación de dominio $V(Ext(\mu))$ y rango contenido en $\mathbf{M}(\Gamma)$.*

Notación. *Notaremos por* $C = (\mu, M_{env}, M_{i_1}, \ldots, M_{i_q})$ *una configuración de* Π, *donde* $V(\mu) = \{i_1, \ldots, i_q\}$, $M_{env} = M(env)$ *es el multiconjunto asociado al entorno externo de* μ *y* $M_{i_j} = M(i_j)$ *es el multiconjunto asociado a la membrana* i_j *de* μ, *para cada* $j = 1, \ldots, q$.

Además, la estructura de membranas μ *se suele representar usando la notación de corchetes, tal y como indicamos a continuación:*

1. *Si* $V(\mu) = \{l\}$, *con* $1 \leq l \leq p$, *entonces* $\mu \equiv [_l\ ]_l$.
2. *Si* $l \in V(\mu)$, *con* $1 \leq l \leq p$, *es la raíz de* μ, *y* $\mu_1, \ldots, \mu_k$ *son los subárboles principales en cada hijo del nodo* l, *entonces* $\mu \equiv [_l\ \mu_1 \ldots \mu_k\ ]_l$, *donde* $\mu_1, \ldots, \mu_k$ *están representados usando la notación de corchetes.*

A la hora de definir las configuraciones que especifican el estado inicial de un sistema P (es decir, sus configuraciones iniciales) debemos tener en cuenta si dicho sistema posee o no membrana de entrada.

Definición 3.7. *Sea* Π *un sistema P de transición con salida externa.*

- *Caso 1:* Π *no posee membrana de entrada.*

  En este caso existe una única configuración inicial del sistema, a saber

  $$C^0 = (\mu_\Pi, \emptyset, \mathcal{M}_1, \ldots, \mathcal{M}_p)$$

- *Caso 2:* Π *posee membrana de entrada.*

  En este caso existe una configuración inicial para cada multiconjunto $m \in \mathbf{M}(\Sigma)$ *que se pueda introducir en dicha membrana, a saber*

  $$C^0(m) = (\mu_\Pi, \emptyset, \mathcal{M}_1, \ldots, \mathcal{M}_{im} + m, \ldots, \mathcal{M}_p)$$

Para definir la función *TStep* para un sistema P de transición con salida externa necesitamos, dada una configuración de dicho sistema, determinar qué reglas se pueden aplicar a dicha configuración en cada membrana.

Definición 3.8. *Diremos que la regla* $r = (u_r, v_r, \delta_r) \in TER(\mu_\Pi, \Gamma)$ *es* aplicable a la configuración C en la membrana i, *y lo notaremos* $r \in ApR(C, i)$, *si se cumple lo siguiente:*

- *La membrana existe en la configuración; es decir,* $i \in V(\mu)$.

- *La regla está asociada a la membrana; es decir,* $r \in R_i$.
- *La membrana tiene los objetos necesarios para aplicar la regla; es decir,* $u_r \leq M_i$.
- *Las membranas a las que la regla intenta enviar objetos existen en la configuración y son inmediatamente interiores a la membrana* i*; es decir,* $\forall j \in V(\mu_\Pi)\big(v_r(j) \neq \emptyset \rightarrow j \in V(\mu) \wedge j \in ch_\mu(i)\big)$.
- *La regla no intenta disolver la membrana piel; es decir, si* $i = skin$*, entonces* $\delta_r = n$.
- *No hay otras reglas aplicables a* C *en la membrana* i *con mayor prioridad; es decir,* $\neg\exists r'\big(r' \in ApR(C,i) \wedge \rho_i(r',r)\big)$.

Es posible que distintas reglas aplicables a una configuración en una membrana entren en conflicto de tal forma que, aunque por separado cada una de ellas sí sea aplicable, tomadas en conjunto no lo sean.

Así, dada una configuración de un sistema P de transición con salida externa, debemos calcular todos los multiconjuntos de reglas aplicables a dicha configuración. Estos son multiconjuntos no vacíos de reglas asociadas a las membranas de la configuración (la multiplicidad dentro del multiconjunto nos indica cuántas veces se debe aplicar cada regla), de tal forma que dichas reglas son aplicables de manera simultánea. Además, debe verificarse la condición de maximalidad, que establece que todos los objetos que pueden evolucionar lo hacen (es decir, una vez aplicado este multiconjunto de reglas, no se permite que queden en ninguna membrana los objetos suficientes para activar alguna de las reglas asociadas a la membrana).

En estas situaciones el sistema se comporta de manera no determinista, de forma que la función paso de transición nos proporcionará el conjunto de posibles configuraciones a las que el sistema pueda evolucionar a partir de una dada.

Definición 3.9. *Diremos que* $amr \in \mathbf{M}\big(TER(\mu_\Pi,\Gamma) \times V(\mu_\Pi)\big)$ *es un* multiconjunto de reglas aplicable a la configuración C*, y lo notaremos por* $amr \in ApM(C)$*, si verifica:*

- *Contiene al menos una regla; es decir,* $amr \neq \emptyset$.
- *Las reglas contenidas en el multiconjunto son aplicables a* C *en la membrana asociada; es decir,*

$$\forall r \in TER(\mu_\Pi, \Gamma)\ \forall i \in V(\mu_\Pi)\ \big(amr(r,i) \neq 0 \rightarrow r \in ApR(C,i)\big)$$

- *Todas las reglas se pueden aplicar a la vez; es decir,*

$$\forall i \in V(\mu_\Pi)\big(\sum_{r \in R_i} amr(r,i) \cdot u_r \leq M_i\big)$$

- *Las reglas se aplican de forma maximal; es decir,*

$$\neg\exists amr'\big(amr' \in ApM(C) \wedge amr < amr'\big)$$

La acción de un multiconjunto de reglas aplicable a una configuración da lugar a dos tipos de procesos claramente diferenciados: por una parte, los objetos contenidos en las membranas evolucionan y son desplazados entre ellas de acuerdo con las reglas del multiconjunto; por otra parte, la ejecución de reglas que supongan la disolución de membranas implica un cambio en la estructura de membranas dentro de la cual estamos trabajando.

Centrándonos en el primero de estos procesos, la aplicación de una regla $r = (u_r, v_r, \delta_r)$ a una configuración $C = (\mu, M_{env}, M_{i_1}, \ldots, M_{i_q})$ en una membrana i tiene los siguientes efectos sobre los objetos contenidos en la membrana: los objetos en u_r desaparecen de la membrana (se eliminan de M_i); los objetos en $v_r(here)$ se incluyen en la membrana (se añaden a M_i); los objetos en $v_r(out)$ se incluyen en la membrana padre (se añaden a $M_{ft_\mu(i)}$), o se expulsan al entorno externo (se añaden a M_{env}) si i es la membrana piel; los objetos en $v_r(j)$ se incluyen en la membrana j, hija de la membrana i (se añaden a M_j).

Puesto que estamos considerando un multiconjunto de reglas aplicable, obtenemos el resultado conjunto de aplicar todas las reglas de evolución contenidas en él.

Definición 3.10. *Sea* $C = \big(Ext(\mu), M\big)$ *una configuración de* Π *y* amr *un multiconjunto de reglas aplicable a* C. *Se define la aplicación* M'_{amr} *sobre* $V(Ext(\mu))$ *como sigue:*

- *Dado* $i \in V(\mu)$, $i \neq skin$,

$$M'_{amr}(i) = M(i) - \sum_{r \in R_i} amr(r,i) \cdot u_r + \sum_{r \in R_i} amr(r,i) \cdot v_r(here) + \\ + \sum_{r \in R_{ft_\mu(i)}} amr(r, ft_\mu(i)) \cdot v_r(i) + \sum_{j \in ch_\mu(i)} \sum_{r \in R_j} amr(r,j) \cdot v_r(out)$$

- *Dado* $i = skin$,

$$M'_{amr}(skin) = M(skin) - \sum_{r \in R_{skin}} amr(r, skin) \cdot u_r + \\ + \sum_{r \in R_{skin}} amr(r, skin) \cdot v_r(here) + \sum_{j \in ch_\mu(skin)} \sum_{r \in R_j} amr(r,j) \cdot v_r(out)$$

- *Dado* $i = env$,

$$M'_{amr}(env) = M(env) + \sum_{r \in R_{skin}} amr(r, skin) \cdot v_r(out)$$

Centrándonos ahora en el segundo de los procesos mencionados, para poder determinar la nueva estructura de membranas obtenida debemos primero caracterizar cuáles son las membranas de la estructura original que se van a disolver.

Definición 3.11. *Sea* $amr \in ApM(C)$ *un multiconjunto de reglas aplicable a* C. *El conjunto de* membranas disueltas *por la aplicación de* amr *se define como sigue:*

$$\Delta(amr, C) = \{i \in V(\mu_\Pi) : \exists r \in TER(\mu_\Pi, \Gamma)\big(amr(r,i) > 0 \wedge \delta_r = y\big)\}$$

Entonces se puede determinar la nueva estructura de membranas.

Definición 3.12. *Sea* $amr \in ApM(C)$ *un multiconjunto de reglas aplicable a* C. *Entonces la estructura de membranas que resulta de aplicar* amr *sobre* C, *que notamos por* μ_{amr}, *es el árbol enraizado dado por:*

- $V(\mu_{amr}) = V(\mu) \setminus \Delta(amr, C)$.
- *La raíz de* μ_{amr} *coincide con la raíz de* μ.

- *La función padre, que determina las aristas de μ_{amr}, viene dada como sigue: para todo $i, j \in V(\mu_{amr})$,*

$$i = ft_{\mu_{amr}}(j) \Leftrightarrow \exists i_0, \ldots, i_n \in V(\mu) \left(\begin{array}{l} i_1, \ldots, i_{n-1} \in \Delta(amr, C) \wedge \\ \wedge\, i_0 = i \wedge i_n = j \wedge \\ \wedge\, \forall l \big(0 \leq l < n \rightarrow i_l = ft_\mu(i_{l+1})\big) \end{array} \right)$$

Obsérvese que la función padre de μ_{amr} también puede calcularse como la restricción a $V(\mu_{amr})$ de la función ft'_μ definida sobre $V(\mu)$ por recursión como sigue:

$$ft'_\mu(j) = \begin{cases} \textit{no definida}, & \text{si } j = skin \\ ft_\mu(j), & \text{si } ft_\mu(j) \notin \Delta(amr, C) \\ ft'_\mu(ft_\mu(j)), & \text{en otro caso} \end{cases}$$

Es decir, μ_{amr} es el árbol enraizado obtenido de μ eliminando las membranas que se disuelven por la aplicación del multiconjunto amr y restaurando las conexiones en la forma correcta. Para ello téngase en cuenta que si queremos calcular el nuevo padre de una membrana que no se disuelve basta recorrer el único camino de membranas que la conecta en μ con la membrana piel hasta encontrar la primera que no se disuelva; esto es precisamente lo que hace la función ft'_μ.

Puesto que los objetos contenidos en una membrana que se disuelve pasan a la membrana padre, y en un mismo paso de transición se puede disolver más de una membrana, necesitamos determinar para cada membrana i que no se disuelva el conjunto de sus membranas donantes; es decir, el conjunto de aquellas membranas que, al disolverse, depositan los objetos que contenían en la membrana i.

Definición 3.13. *Sea $amr \in ApM(C)$ un multiconjunto de reglas aplicable a C. Para cada membrana $i \in V(\mu_{amr})$ se define el conjunto de* donantes de *i como sigue:*

$$Don(i, amr, C) = \{j \in V(\mu) : j \in \Delta(amr, C) \wedge ft'_\mu(j) = i\}$$

Podemos entonces calcular cómo se redistribuyen los objetos de las membranas disueltas.

Definición 3.14. *Sea* $C = \big(Ext(\mu), M\big)$ *una configuración de* Π *y amr un multiconjunto de reglas aplicable a* C. *Se define la aplicación* M_{amr} *sobre* $V(Ext(\mu_{amr}))$ *como sigue:*

- *Dado* $i \in V(\mu_{amr})$,

$$M_{amr}(i) = M'_{amr}(i) + \sum_{j \in Don(i,amr,C)} M'_{amr}(j)$$

- *Dado* $i = env$,

$$M_{amr}(env) = M'_{amr}(env)$$

Estamos ya en condiciones de establecer cuándo una configuración se ha obtenido en un paso de transición a partir de otra.

Definición 3.15. *La función paso de transición, TStep, para un sistema P de transición con salida externa se define como sigue: dadas dos configuraciones* $C_1 = \big(Ext(\mu_1), M_1\big)$ *y* $C_2 = \big(Ext(\mu_2), M_2\big)$ *del sistema,*

$$C_2 \in TStep(C_1) \Leftrightarrow \exists amr \in ApM(C)\big(\mu_2 = \mu_{1amr} \wedge M_2 = M_{1amr}\big)$$

Notación. *Sea* Π *un sistema P de transición con salida externa y sea* $\mathcal{C} = \{C^i\}_{i<r}$ *una computación de* Π. *Entonces notamos* $C^i = \big(Ext(\mu^i), M^i\big)$.

Además, notamos por $amr_{i,i+1} \in ApM(C^i)$ *al multiconjunto de reglas aplicable a la configuración* C^i *que se ha utilizado en la computación para obtener la configuración* C^{i+1}.

La idea de un sistema P de transición con salida externa consiste en no tener en cuenta lo que ocurre en el interior del sistema, sino solo fijarnos en lo que este expulsa al entorno externo. Surge entonces el problema de determinar cuándo una computación para, y aquí es donde entra en juego el objeto distinguido #.

Definición 3.16. *Diremos que* $r = (u_r, v_r, \delta_r)$ *es una* regla indicadora de parada *si* $\# \in v_r(out)$.

Definición 3.17. *Diremos que un sistema P de transición con salida externa* Π *es* válido *si dada una computación* $\mathcal{C} = \{C^i\}_{i<t}$ *del sistema se cumple lo siguiente:*

- *Si C es de parada, entonces:*
  - *Para todo $i < t-2$ y toda $r \in R_{skin}$, si r es una regla asociada en $amr_{i,i+1}$ a la membrana piel, entonces r no es indicadora de parada.*
  - *Existe $r \in R_{skin}$ tal que r es una regla asociada en $amr_{t-2,t-1}$ a la membrana piel y r es indicadora de parada.*
- *Si C no es de parada, entonces para todo $i < t-1$ y para toda $r \in R_{skin}$, si r es una regla asociada en $amr_{i,i+1}$ a la membrana piel, entonces r no es indicadora de parada.*

De esta forma, el hecho de que una computación haya parado o no viene determinado por la presencia o no de un objeto # en el entorno externo del sistema.

3.4. Variantes de sistemas P con salida externa

Una vez definidos los conjuntos de configuraciones y configuraciones iniciales de un sistema P de transición con salida externa, así como la función paso de transición de dicho sistema, podemos considerar distintas variantes «semánticas» sin más que fijar qué entendemos por salida de una computación; es decir, definiendo la función *Output* sobre el conjunto de computaciones del sistema.

No obstante, dada una variante de sistemas P de transición con salida externa, también podemos considerar diversas subvariantes «sintácticas» de la forma que se indica a continuación.

Definición 3.18. *Sea EPS una variante de sistemas P de transición con salida externa. Entonces un tal sistema Π pertenece a la clase $EPS(\alpha, \beta, \gamma)$, donde $\alpha \in \{Pri, nPri\}$, $\beta \in \{Coo, Cat, nCoo\}$ y $\gamma \in \{Dis, nDis\}$, si cumple las siguientes propiedades:*

- *Si $\alpha = nPri$, entonces $\rho_1 = \cdots = \rho_p = \emptyset$. Si $\alpha = Pri$, entonces $\rho_i \neq \emptyset$, para algún $i = 1, \ldots, p$.*

  Es decir, α indica si admitimos o no relaciones de prioridad entre las reglas de evolución que se usan en Π.

- *Si* $\beta = nCoo$, *entonces* $R_1 \cup \cdots \cup R_p \subseteq NCTER(\mu_\Pi, \Gamma)$. *Si* $\beta = Cat$, *entonces* $R_1 \cup \cdots \cup R_p \subseteq CTER(\mu_\Pi, \Gamma, C)$, *para algún conjunto* C *de catalizadores. Si* $\beta = Coo$, *entonces existe* i, *con* $1 \leq i \leq p$, *tal que* $R_i \not\subseteq NCTER(\mu_\Pi, \Gamma)$ *y* $R_i \not\subseteq CTER(\mu_\Pi, \Gamma, C)$, *para cualquier conjunto de catalizadores,* C, *que podamos considerar.*

  Es decir, β *establece el tipo de reglas de evolución que admitimos en el sistema: no cooperativas, catalíticas o sin restricciones.*

- *Si* $\gamma = nDis$, *entonces para toda regla* $(u, v, \delta) \in R_1 \cup \cdots \cup R_p$ *se tiene que* $\delta = n$. *Si* $\gamma = Dis$, *entonces existe* i, *con* $1 \leq i \leq p$, *y existe una regla* $(u, v, \delta) \in R_i$ *tal que* $\delta = y$.

  Es decir, γ *determina si permitimos que las reglas de evolución del sistema* P *puedan o no disolver las membranas.*

Capítulo 4

Sistemas P como dispositivos de aceptación

En este capítulo se estudian los sistemas P de transición con salida externa que aceptan lenguajes sobre un determinado alfabeto. Estos sistemas dispondrán de una membrana de entrada, de tal forma que al introducir en ella una cadena, debidamente codificada, obtendremos como respuesta si pertenece o no al lenguaje especificado.

En la primera sección definimos la función *Output* de los sistemas P de transición con salida externa para obtener la variante de sistemas P de aceptación de lenguajes. En la sección 4.2 mostramos cómo estos sistemas son capaces de simular el funcionamiento de las máquinas de Turing deterministas, presentando una construcción mejorada de la realizada en [52]. En la sección 4.3 establecemos una verificación de la corrección de la simulación anterior, siguiendo las ideas de [43].

4.1. Sistemas P de aceptación de lenguajes

Teniendo presente que tratamos con sistemas P de transición con salida externa, la sintaxis y semántica de los sistemas de computación celular con membranas que vamos a introducir a continuación quedó establecida en el capítulo anterior, a excepción de la función *Output*. En estos sistemas el alfabeto de salida constará solo de dos símbolos, Yes y No, y la salida de una computación será de aceptación o de rechazo, dependiendo de cuál de esos símbolos haya sido expulsado al entorno externo por el sistema P.

Definición 4.1. *Un* sistema P de aceptación de lenguajes, Π, *es un sistema de computación celular con membranas que verifica las siguientes propiedades:*

- Π *es un sistema P de transición con salida externa.*
- Π *es un sistema P con membrana de entrada.*
- *El alfabeto de salida de* Π *es* $\Lambda = \{Yes, No\}$.
- *La salida de una computación* $\mathcal{C} = \{C^i\}_{i<r}$ *viene dada por la siguiente función:*

$$Output(\mathcal{C}) = \begin{cases} Yes, & si\ \mathcal{C}\ es\ de\ parada,\ Yes \in M_{env}^{r-1}\ y \\ & No \notin M_{env}^{r-1} \\ No, & si\ \mathcal{C}\ es\ de\ parada,\ No \in M_{env}^{r-1}\ y \\ & Yes \notin M_{env}^{r-1} \\ no\ definida, & en\ cualquier\ otro\ caso \end{cases}$$

Si $\mathcal{C}$ *cumple alguna de las dos primeras condiciones anteriores, entonces diremos que es una* computación exitosa.

Obsérvese que hemos definido la salida únicamente en aquellas computaciones en las que tiene sentido: la computación es de parada y solo aparecen objetos Yes u objetos No en el entorno externo. Sin embargo, para que un sistema de este tipo sea propiamente un dispositivo de aceptación debemos exigir además que todas las computaciones de parada proporcionen una respuesta.

Definición 4.2. *Un sistema P de aceptación de lenguajes se dice que es* válido *si verifica las siguientes propiedades:*

- Π *es un sistema P de transición con salida externa válido.*
- *Si* $\mathcal{C} = \{C^i\}_{i<r}$ *es una computación de parada de* Π, *entonces es una computación exitosa.*

Notación. *Notaremos por* $\mathcal{LA}$ *la clase de los sistemas P de aceptación de lenguajes tales que dichos sistemas sean válidos.*

Seguidamente vamos a definir qué significa que un sistema P acepte un lenguaje sobre un determinado alfabeto.

Definición 4.3. *Sea L un lenguaje sobre un alfabeto Ω. Diremos que el sistema $\Pi \in \mathcal{LA}$* acepta *el lenguaje L si se verifican las siguientes propiedades:*

- *Existe una función total computable e inyectiva, $cod : \Omega^* \to \mathbf{M}(\Sigma)$, que codifica las cadenas sobre Ω mediante multiconjuntos sobre el alfabeto de entrada del sistema Π.*
- *Para toda cadena $w \in \Omega^*$ se tiene que:*
  - *Si $w \in L$, entonces existe* una *computación C de Π con entrada $cod(w)$ tal que C es de parada y $Output(C) = Yes$.*
  - *Si existe* una *computación C de Π con entrada $cod(w)$ tal que C es de parada y $Output(C) = Yes$, entonces $w \in L$.*

Análogamente a las máquinas de Turing, podemos definir también qué significa que un sistema P decida un lenguaje sobre un determinado alfabeto.

Definición 4.4. *Sea L un lenguaje sobre un alfabeto Ω. Diremos que el sistema $\Pi \in \mathcal{LA}$* decide *el lenguaje L si se verifican las siguientes propiedades:*

- *Toda computación de Π es de parada.*
- *Existe una función total computable e inyectiva, $cod : \Omega^* \to \mathbf{M}(\Sigma)$, que codifica las cadenas sobre Ω mediante multiconjuntos sobre el alfabeto de entrada del sistema Π.*
- *Para toda cadena $w \in \Omega^*$ se tiene que:*
  - *Si $w \in L$, entonces para* toda *computación C de Π con entrada $cod(w)$ se tiene que $Output(C) = Yes$.*
  - *Si $w \notin L$, entonces para* toda *computación C de Π con entrada $cod(w)$ se tiene que $Output(C) = No$.*

Obsérvese que se tienen las siguientes relaciones entre los lenguajes aceptados y los lenguajes decididos por un sistema de la clase $\mathcal{LA}$: si un sistema, Π, decide un lenguaje, L, entonces es fácil comprobar que también acepta dicho lenguaje; por otra parte, si un sistema, Π, *determinista*, acepta un lenguaje, L, y, además, todas las computaciones de dicho sistema son de parada, entonces el sistema Π decide el lenguaje L.

4.2. Completitud computacional a través de máquinas de Turing

En esta sección describimos cómo es posible simular una máquina de Turing determinista mediante un sistema P de aceptación de lenguajes, de tal manera que el lenguaje aceptado por el sistema coincide con el lenguaje aceptado por la máquina de Turing.

Para ello, dada una máquina de Turing determinista, TM, asociaremos a esta máquina un sistema P de aceptación de lenguajes, $\Pi_{TM} \in \mathcal{LA}$, capaz de reproducir completamente su funcionamiento. Así, las reglas de Π_{TM}, que serán reglas cooperativas entre las cuales existirán relaciones de prioridad, se elegirán cuidadosamente de forma tal que toda computación del sistema realice las siguientes operaciones:

- Se parte de una configuración inicial en la que se ha introducido un multiconjunto en la membrana de entrada.
- Se comprueba que dicho multiconjunto representa realmente una cadena, de acuerdo con una cierta función codificadora, sobre el alfabeto de entrada de la máquina de Turing. En caso contrario, la computación no será de parada, pues entrará en un bucle infinito.
- Se simulan los distintos pasos que la máquina de Turing efectúa al recibir como entrada la cadena codificada en la membrana de entrada. La simulación de cada uno de los pasos se realiza a través de varias fases:
  - Se lee el símbolo en la celda analizada por la cabeza lectora.
  - Se calcula el nuevo símbolo a escribir en la celda, se mueve la cabeza lectora y se cambia de estado.
  - Se borra el viejo símbolo y se escribe el nuevo (que puede coincidir con el anterior).
- Finalmente, tras la simulación de cada paso de la máquina de Turing se comprueba si se ha alcanzado un estado final. En ese caso, la computación del sistema será de parada, y expulsará al entorno externo, o bien un objeto Yes, si el estado al que se ha llegado es de aceptación, o bien un objeto No, si dicho estado es de rechazo.

Estas operaciones se llevarán a cabo en varias pequeñas etapas, cada una de ellas controlada por un grupo de reglas. Para evitar que reglas de distintas etapas se apliquen juntas, usaremos objetos de tipo s^- como *objetos prohibidores* y objetos de tipo s^+ como *objetos permitidores*; si s_j^- está presente en el sistema P, entonces las reglas asociadas a la etapa j **no se pueden** ejecutar; si, en cambio, s_j^+ es el objeto presente, entonces las reglas asociadas a la etapa j **deben** ser ejecutadas (si es posible). Por supuesto, en cualquier momento, para cada etapa solo el correspondiente objeto prohibidor o permitidor estará presente. También habrá siempre solamente un objeto permitidor al mismo tiempo.

Para indicar que queremos realizar una etapa de la computación, usaremos objetos de tipo s como *objetos promotores.* Cuando s_j aparezca en el sistema, transformará su correspondiente objeto prohibidor s_j^- en el permitidor s_j^+, dando lugar así a que las reglas para la etapa j se apliquen (si ello es posible). Entonces el objeto permitidor se transformará a sí mismo de nuevo en el objeto prohibidor, y en un objeto promotor adecuado.

Supongamos que el conjunto de estados de la máquina de Turing es $Q_{TM} = \{q_N, q_Y, q_0, \ldots, q_n\}$, el alfabeto de trabajo $\Gamma_{TM} = \{B, \triangleright, a_1, \ldots, a_m\}$, el alfabeto de entrada $\Sigma_{TM} = \{a_1, \ldots, a_p\}$, con $p \leq m$, y la función de transición $\delta_{TM}(q_i, a_j) = (q_{Q(i,j)}, a_{A(i,j)}, D(i,j))$. Recuérdese que denotamos $a_B = B$ (símbolo blanco) y $a_0 = \triangleright$ (símbolo situado en la celda más a la izquierda de la cinta).

Consideremos el sistema P de aceptación de lenguajes, Π_{TM}, definido como sigue:

$$\Pi_{TM} = \left(\Sigma, \Gamma, \Lambda, \#, \mu_{\Pi_{TM}}, \mathcal{M}_1, (R_1, \rho_1), im, ext\right)$$

donde

$$\begin{aligned}
\Sigma &= \{a_1, \ldots, a_p\} \\
\Gamma &= \{q_N, q_Y, h, h', s_E, \#\} \cup \{q_i : 0 \leq i \leq n\} \cup \\
&\quad \{a_i, a_i', a_i'', b_i : 0 \leq i \leq m\} \cup \{s_l, s_l^-, s_l^+ : l \in \{I_1, \ldots, I_5, 1, \ldots, 9\}\} \\
\Lambda &= \{Yes, No\} \\
\mu_{\Pi_{TM}} &= [_1]_1 \\
\mathcal{M}_1 &= q_0 a_0 s_{I_1} s_{I_1}^- \ldots s_{I_5}^- s_1^- \ldots s_9^- \\
im &= 1 \\
(R_1, \rho_1) &= (R^I, \rho^I) \cup (R^1, \rho^1) \cup (R^2, \rho^2) \cup (R^3, \rho^3) \cup (R^4, \rho^4)
\end{aligned}$$

y los conjuntos de reglas son

$(R^I, \rho^I) = (R^{I_1}, \rho^{I_1}) \cup (R^{I_2}, \rho^{I_2}) \cup (R^{I_3}, \rho^{I_3}) \cup (R^{I_4}, \rho^{I_4}) \cup (R^{I_5}, \rho^{I_5})$, con

$$(R^{I_1},\rho^{I_1}) \equiv \left\{ \begin{array}{l} s_E s_{I_1}^- \to (\#, out) > s_{I_1} s_{I_1}^- \to s_{I_1}^+ > s_{I_1}^- \to s_{I_1}^- > \\ \quad a_i s_{I_1}^+ \to a_i s_{I_1}^- s_{I_2} \quad (1 \le i \le p) > s_{I_1}^+ \to s_{I_1}^- s_1 \end{array} \right.$$

$$(R^{I_2},\rho^{I_2}) \equiv \left\{ \begin{array}{l} s_E s_{I_2}^- \to (\#, out) > s_{I_2} s_{I_2}^- \to s_{I_2}^+ > s_{I_2}^- \to s_{I_2}^- > \\ \quad \left\{ \begin{array}{l} a_i \to a_i a_i' \quad (1 \le i \le p) \\ s_{I_2}^+ \to s_{I_2}^- s_{I_3} \end{array} \right. \end{array} \right.$$

$$(R^{I_3},\rho^{I_3}) \equiv \left\{ \begin{array}{l} s_E s_{I_3}^- \to (\#, out) > s_{I_3} s_{I_3}^- \to s_{I_3}^+ > s_{I_3}^- \to s_{I_3}^- > \\ \quad a_i'^2 \to a_i'' \quad (1 \le i \le p) > a_i' s_{I_3}^+ \to a_i' s_{I_3}^- \quad (1 \le i \le p) > \\ \quad \left\{ \begin{array}{l} a_i'' \to a_i' \quad (1 \le i \le p) \\ s_{I_3}^+ \to s_{I_3}^- s_{I_4} \end{array} \right. \end{array} \right.$$

$$(R^{I_4},\rho^{I_4}) \equiv \left\{ \begin{array}{l} s_E s_{I_4}^- \to (\#, out) > s_{I_4} s_{I_4}^- \to s_{I_4}^+ > s_{I_4}^- \to s_{I_4}^- > \\ \quad a_i'^2 \to a_i'' \quad (1 \le i \le p) > a_i' a_j' s_{I_4}^+ \to a_i' a_j' s_{I_4}^- \quad (1 \le i,j \le p, i \ne j) > \\ \quad a_i' s_{I_4}^+ \to s_{I_4}^- s_{I_5} \quad (1 \le i \le p) > s_{I_4}^+ \to s_{I_4}^- \end{array} \right.$$

$$(R^{I_5},\rho^{I_5}) \equiv \left\{ \begin{array}{l} s_E s_{I_5}^- \to (\#, out) > s_{I_5} s_{I_5}^- \to s_{I_5}^+ > s_{I_5}^- \to s_{I_5}^- > \\ \quad a_i'' \to a_i' \quad (1 \le i \le p) > a_i' s_{I_5}^+ \to a_i' s_{I_5}^- s_{I_4} \quad (1 \le i \le p) > \\ \quad s_{I_5}^+ \to s_{I_5}^- s_1 \end{array} \right.$$

$(R^1, \rho^1) = (R^{1_1}, \rho^{1_1}) \cup (R^{1_2}, \rho^{1_2}) \cup (R^{1_3}, \rho^{1_3})$, con

$$(R^{1_1},\rho^{1_1}) \equiv \left\{ \begin{array}{l} s_E s_1^- \to (\#, out) > s_1 s_1^- \to s_1^+ > s_1^- \to s_1^- > \\ \quad \left\{ \begin{array}{l} h \to hh' \\ a_i \to a_i a_i' \quad (0 \le i \le m) \\ s_1^+ \to s_1^- s_2 \end{array} \right. \end{array} \right.$$

$$(R^{1_2},\rho^{1_2}) \equiv \begin{cases} s_E s_2^- \to (\#, out) > h' s_2 s_2^- \to s_2^+ > s_2 \to s_3 > \\ \quad s_2^- \to s_2^- > a_i'^2 \to a_i'' \quad (0{\le}i{\le}m) > \begin{cases} a_i' \to \lambda \quad (0{\le}i{\le}m) \\ a_i'' \to a_i' \quad (0{\le}i{\le}m) \\ s_2^+ \to s_2^- s_2 \end{cases} \end{cases}$$

$$(R^{1_3},\rho^{1_3}) \equiv \begin{cases} s_E s_3^- \to (\#, out) > s_3 s_3^- \to s_3^+ > s_3^- \to s_3^- > \\ \quad \begin{cases} a_i'^2 \to \lambda \quad (0{\le}i{\le}m) \\ s_3^+ \to s_3^- s_4 \end{cases} \end{cases}$$

$(R^2,\rho^2) = (R^{2_1},\rho^{2_1}) \cup (R^{2_2},\rho^{2_2})$, con

$$(R^{2_1},\rho^{2_1}) \equiv \quad s_E s_4^- \to (\#, out) > s_4 s_4^- \to s_4^+ > s_4^- \to s_4^- > \begin{cases} h \to hh' \\ s_4^+ \to s_4^- s_5 \end{cases}$$

$$(R^{2_2},\rho^{2_2}) \equiv \begin{cases} s_E s_5^- \to (\#, out) > s_5 s_5^- \to s_5^+ > s_5^- \to s_5^- > \\ \quad \begin{cases} \text{Reglas para la función de transición} \\ s_5^+ \to s_5^- s_6 \end{cases} \end{cases}$$

Las reglas para la función de transición, δ_{TM}, son las siguientes:

Caso 1: Estado q_r, símbolo $a_s \neq B$

Movimiento	*Reglas*	
izquierda	$q_r a_s' h \to q_{Q(r,s)} a_s' b_{A(r,s)}$,	si $A(r,s) \neq B$
	$q_r a_s' h \to q_{Q(r,s)} a_s'$,	si $A(r,s) = B$
igual	$q_r a_s' \to q_{Q(r,s)} a_s' b_{A(r,s)}$,	si $A(r,s) \neq B$
	$q_r a_s' \to q_{Q(r,s)} a_s'$,	si $A(r,s) = B$
derecha	$q_r a_s' \to q_{Q(r,s)} a_s' b_{A(r,s)} h$,	si $A(r,s) \neq B$
	$q_r a_s' \to q_{Q(r,s)} a_s' h$,	si $A(r,s) = B$

Caso 2: Estado q_r, símbolo $a_s = B$

Movimiento	*Reglas*	
izquierda	$q_r h \to q_{Q(r,B)} b_{A(r,B)},$	si $A(r,B) \neq B$
	$q_r h \to q_{Q(r,B)},$	si $A(r,B) = B$
igual	$q_r \to q_{Q(r,B)} b_{A(r,B)},$	si $A(r,B) \neq B$
	$q_r \to q_{Q(r,B)},$	si $A(r,B) = B$
derecha	$q_r \to q_{Q(r,B)} b_{A(r,B)} h,$	si $A(r,B) \neq B$
	$q_r \to q_{Q(r,B)} h,$	si $A(r,B) = B$

Para evitar conflictos, toda regla del caso 1 tiene una prioridad mayor que cada regla del caso 2.

$(R^3, \rho^3) = (R^{3_1}, \rho^{3_1}) \cup (R^{3_2}, \rho^{3_2})$, con

$$(R^{3_1},\rho^{3_1}) \equiv \begin{cases} s_E s_6^- \to (\#, out) > h' s_6 s_6^- \to s_6^+ > s_6 \to s_7 > \\ \quad s_6^- \to s_6^- > \begin{cases} a_i' \to a_i'^2 & (0 \leq i \leq m) \\ b_i \to b_i^2 & (0 \leq i \leq m) \\ s_6^+ \to s_6^- s_6 \end{cases} \end{cases}$$

$$(R^{3_2},\rho^{3_2}) \equiv \begin{cases} s_E s_7^- \to (\#, out) > s_7 s_7^- \to s_7^+ > s_7^- \to s_7^- > \\ \quad \begin{cases} a_i a_i' \to \lambda & (0 \leq i \leq m) \\ b_i \to a_i & (0 \leq i \leq m) \\ s_7^+ \to s_7^- s_8 \end{cases} \end{cases}$$

$(R^4, \rho^4) = (R^{4_1}, \rho^{4_1}) \cup (R^{4_2}, \rho^{4_2})$, con

$$(R^{4_1},\rho^{4_1}) \equiv \begin{cases} s_E s_8^- \to (\#, out) > s_8 s_8^- \to s_8^+ > s_8^- \to s_8^- > \\ \quad \begin{cases} q_Y \to q_Y s_9 \\ q_N \to q_N s_9 \\ q_i \to q_i s_1 & (0 \leq i \leq n) \\ s_8^+ \to s_8^- \end{cases} \end{cases}$$

$$(R^{4_2},\rho^{4_2}) \equiv \begin{cases} s_E s_9^- \to (\#, out) > s_9 s_9^- \to s_9^+ > s_9^- \to s_9^- > \\ \quad \begin{cases} q_Y \to (Yes, out) \\ q_N \to (No, out) \\ h \to \lambda \\ a_i \to \lambda \quad (0 \le i \le m) \\ s_9^+ \to s_9^- s_E \end{cases}{}^{14} \end{cases}$$

Para reproducir el funcionamiento de la máquina de Turing, es obvio que tenemos que fijar una representación para varios de sus elementos: cuál es el estado actual de la máquina de Turing, cuáles son los símbolos en las celdas de la cinta, y qué celda está siendo analizada por la cabeza.

Para representar los estados usaremos los objetos $q_N, q_Y, q_0, \ldots, q_n$.

La celda (numerada a partir de cero) analizada por la cabeza se representará por el objeto h, en base 1. El objeto h' se usará, cuando se necesite, como copia de trabajo del objeto h, y se utilizará también como contador en varias de las fases.

A lo largo de la simulación, los objetos $a_0, \ldots, a_m$ representarán, en base 2, qué celdas contienen los símbolos $a_0, \ldots, a_m$, respectivamente (nótese que, en cualquier momento, el número de celdas no vacías es finito); de esta forma, para determinar si la cadena $w = a_{i_1} \ldots a_{i_q} \in \Sigma^*_{TM}$ pertenece al lenguaje aceptado por la máquina de Turing, introducimos inicialmente en la membrana de entrada el multiconjunto $cod(w) = a^2_{i_1} \ldots a^{2^q}_{i_q} \in \mathbf{M}(\Sigma)$; los objetos $a'_0, \ldots, a'_m$ se usarán como copias de trabajo de los objetos anteriores y servirán también para guardar los símbolos leídos de las celdas; los objetos $a''_0, \ldots, a''_m$ se usarán en ciertos cálculos que involucran a los objetos $a'_0, \ldots, a'_m$; por último, los objetos $b_0, \ldots, b_m$ servirán para indicar los nuevos símbolos a escribir en las celdas.

A continuación, veamos con mayor detalle, de una manera informal, cómo trabaja este sistema P. En la siguiente sección estableceremos la verificación formal de la simulación.

Inicialmente, el sistema contiene el estado inicial q_0, el objeto a_0, que codifica el hecho de que la celda más a la izquierda contiene el símbolo $\triangleright$, los objetos inhibidores para todas las etapas, el multiconjunto $cod(w)$ introducido como entrada, y el objeto promotor de la primera etapa, s_{I_1}. Este objeto

induce la realización de la fase en la que se comprueba que el multiconjunto de entrada realmente codifica una cadena inicial de la máquina de Turing que se simula.

Fase I: *Comprobación del multiconjunto de entrada.*

Esta fase se lleva a cabo mediante las reglas del conjunto R^I.

En primer lugar, mediante las reglas de R^{I_1} descartamos que el multiconjunto de entrada sea vacío, el cual representa la cadena vacía. Esta es una cadena de entrada correcta, pero no es detectada por el resto de reglas de esta fase. Por ello, si la membrana del sistema contiene algún objeto a_i del alfabeto de entrada, entonces introducimos el objeto promotor s_{I_2} para seguir verificando si el multiconjunto de entrada es correcto. En caso contrario, introducimos el objeto promotor s_1 para continuar con la siguiente fase.

Si el multiconjunto de entrada no es vacío, entonces usando las reglas de R^{I_2} hacemos una copia de los objetos a_i del alfabeto de entrada mediante objetos a_i'. De esta forma obtenemos una copia de trabajo de dicho multiconjunto, para realizar los cálculos con ella y conservar el multiconjunto de entrada original en caso de que codifique una cadena correcta.

Las reglas de R^{I_3} comprueban que entre los objetos introducidos no hay ninguno que codifique que un símbolo ha de ir en la celda cero (recuérdese que esta celda debe siempre contener el símbolo $\triangleright$). Para ello dividimos el número de dichos objetos por dos, cambiando pares de objetos a_i' por un único objeto a_i'', y observamos los restos obtenidos. Debe cumplirse que todos sean cero, por lo que si no es así (es decir, si queda algún objeto a_i' en la membrana), entonces no introducimos ningún objeto promotor, en cuyo caso se aplican las reglas $s_j^- \to s_j^-$ una y otra vez y la computación no es de parada.

Finalmente, si los restos han sido cero, lo que queda es comprobar que el multiconjunto inicial efectivamente representa una cadena cuyos símbolos ocupan celdas consecutivas a partir de la segunda celda más a la izquierda. Esto lo realizan las reglas de R^{I_4} y R^{I_5}, una vez que los objetos a_i'' vuelven a convertirse en objetos a_i', de la siguiente forma: dividimos por dos el número de objetos a_i', transformando pares de

dichos objetos en objetos a_i''; si ha quedado más de un objeto a_i' (lo que indica más de un símbolo en la misma celda), o no ha quedado ninguno (lo que indica una celda vacía), entonces no introducimos ningún objeto promotor, con lo que la computación no es de parada; en otro caso, transformamos los objetos a_i'' en objetos a_i' y volvemos a realizar la misma operación, hasta que ya no queden objetos a_i'.

Si todos los controles han sido satisfactorios, introducimos el objeto promotor s_1, de manera que el sistema P comience a reproducir los pasos de computación realizados por la máquina de Turing.

La aparición del objeto s_1 en el sistema provoca el comienzo de la simulación de un paso de la computación de la máquina de Turing, iniciando la fase encargada de leer el símbolo que la cabeza de la máquina está analizando en ese momento. Esta fase acabará cuando el objeto s_4 aparezca dentro de la membrana.

Fase 1: *Lectura del símbolo.*

Esta fase se lleva a cabo mediante las reglas del conjunto R^1.

Para descodificar el símbolo leído por la cabeza a partir de los objetos dentro del sistema P, nos basamos en la siguiente propiedad:

> *El número x tiene un 1 como y-ésimo dígito en su representación en base 2 si y solo si el cociente de dividir x entre 2^y es impar (es decir, es congruente con 1 módulo 2).*

Recuérdese que la celda que está siendo analizada por la cabeza se representa mediante el número de objetos h y las celdas no vacías con símbolos $a_0, \ldots, a_m$ se representan, en base 2, mediante el número de objetos $a_0, \ldots, a_m$, respectivamente. Supongamos que esos números son y, para la celda analizada, y $x_0, \ldots, x_m$, para las celdas no vacías. Entonces, o solo uno, si la celda número y no está vacía, o ninguno, si la celda número y está vacía, de estos últimos tiene un 1 como y-ésimo dígito en base 2.

Así, calculamos el cociente de $x_0, \ldots, x_m$ cuando se dividen por 2^y y tomamos módulo 2. De esta forma tendremos el símbolo en la celda

número y si alguno de los resultados es 1 (y solo puede ser a lo sumo uno de ellos) o una celda número y vacía (y, por tanto, ningún símbolo) en caso contrario. Esto se realiza en tres pasos.

Primero, mediante las reglas de R^{1_1}, obtenemos copias de trabajo, h', $a'_0, \ldots, a'_m$, de los objetos $h, a_0, \ldots, a_m$, respectivamente, para hacer los cálculos sin cambiar el estatus de la (representación de la) máquina de Turing.

Las reglas de R^{1_2} son las encargadas de la división por 2^y. Esto se hace dividiendo por 2 cuantas veces sea necesario, deshaciéndonos de los restos obtenidos cada vez. Tenemos que dividir tantas veces como indique el número de objetos h'. Por tanto, para transformar el objeto prohibidor en el permitidor, de forma que podamos realizar la división, no solo necesitamos que el objeto promotor s_2 esté presente, sino también al menos algún h'. Si este no es el caso, entonces hemos terminado con la segunda etapa de esta fase y podemos continuar con la tercera. Así, s_2 se transforma a sí mismo en el objeto promotor para esta, s_3.

En otro caso, llevamos a cabo la división cambiando primero pares de a'_j por un solo a''_j. Entonces eliminamos los a'_j que quedan (los restos) y recuperamos a'_j a partir de a''_j. Finalmente, s_2^+ se cambia en s_2^- y en s_2, indicando de esta forma que queremos realizar otra división, si es posible (es decir, si quedan h').

La última etapa de esta fase es relativamente simple. Las reglas de R^{1_3} solo tienen que tomar módulo dos, y esto se hace eliminando tantos pares de a'_j como sea posible. Al final, si la celda número y contiene el símbolo a_k, $a_k \neq B$, entonces obtenemos un único a'_k en el sistema P, y ningún a'_j para todo $j \neq k$. En otro caso, si la celda y está vacía, entonces no obtenemos ningún a'_j, para cualquier $j = 0, \ldots, m$.

La presencia del objeto s_4 en el sistema da lugar al comienzo de la siguiente fase, que se encarga de calcular el nuevo símbolo que se va a escribir, mover la cabeza y cambiar el estado, de acuerdo con la función de transición, δ_{TM}. Esta fase finalizará cuando el objeto s_6 aparezca dentro de la membrana.

Fase 2: *Cálculo del nuevo símbolo, movimiento de la cabeza y cambio de estado.*

Esta fase se lleva a cabo mediante las reglas del conjunto R^2 y trata de simular la función de transición de la máquina de Turing.

En primer lugar, guardamos qué celda estaba analizando la cabeza antes de aplicar la función de transición, de forma que la siguiente fase pueda realizarse correctamente. Esto se lleva a cabo mediante las reglas de R^{2_1} simplemente haciendo una copia de los objetos h usando los objetos h'.

Entonces usamos las reglas de R^{2_2}, que se clasifican en dos tipos, dependiendo de si la celda número y está o no vacía. Las reglas para el último caso tienen mayor prioridad que las reglas para el primero, por lo que no hay ambigüedad; esto es, se aplicará una única regla asociada a la función de transición en cualquier caso.

Supongamos que el estado de la máquina de Turing es q_r y el símbolo en la celda número y es a_s. Por lo tanto tenemos un objeto q_r y un objeto a'_s en el sistema P. Entonces existe una sola regla que implica a estos dos objetos, la cual transforma q_r en $q_{Q(r,s)}$ (el nuevo estado), introduce un nuevo objeto $b_{A(r,s)}$ (el nuevo símbolo) o ninguno, si la celda tiene que borrarse, pero conservando el antiguo a'_s y, finalmente, si la cabeza tiene que moverse a la izquierda, entonces elimina un objeto h; si tiene que moverse a la derecha, añade un objeto h; y si tiene que permanecer en el mismo sitio, no hace nada.

Supongamos que el estado de la máquina de Turing es q_r y la celda número y está vacía. Por lo tanto tenemos un objeto q_r pero ningún objeto a'_s en el sistema P. Entonces existe una única regla que implica el primer objeto pero ninguno de los últimos, la cual transforma q_r en $q_{Q(r,s)}$ (el nuevo estado), introduce un nuevo objeto $b_{A(r,s)}$ (el nuevo símbolo) o ninguno, si la celda tiene que permanecer vacía y, finalmente, si la cabeza tiene que moverse a la izquierda, entonces elimina un objeto h; si tiene que moverse a la derecha, añade un objeto h; y si tiene que permanecer en el mismo sitio, no hace nada.

La presencia del objeto s_6 inicia la fase encargada de borrar el símbolo leído por la cabeza y de escribir en su lugar el nuevo símbolo calculado por

medio de las reglas asociadas a la función de transición. Esta fase terminará cuando el objeto s_8 aparezca dentro de la membrana.

Fase 3: *Borrado del viejo símbolo y escritura del nuevo.*

Esta fase se lleva a cabo mediante las reglas del conjunto R^3.

En esta situación, el sistema P consta de los siguientes tipos de objetos: objetos h' que representan la celda que la cabeza *analizó*; objetos h que representan la celda que la cabeza *va a analizar*, posiblemente diferente de la anterior; un objeto a'_s, o ningún objeto de este tipo, que representa el símbolo *leído* por la cabeza, o blanco, respectivamente; un objeto b_t, o ningún objeto de este tipo, que representa el nuevo símbolo *a escribir* en la celda antigua.

Como la representación de las celdas no vacías es en base 2, tenemos que hacer 2^y copias de los dos últimos objetos, a'_s y b_t. Obsérvese que no se conoce en principio qué objetos son, pero podemos sacar provecho del hecho de que existe a lo sumo solo uno de cada tipo. Por tanto, podemos poner una regla para cada a'_j teniendo la seguridad de que solo una de ellas, la que queremos, se ejecutará. Lo mismo se aplica a los objetos b_k.

La técnica para calcular las potencias 2^y de estos objetos es la misma que la usada para calcular las divisiones sucesivas en la fase 1, y se realiza mediante las reglas de R^{3_1}. Tenemos que duplicar el número de objetos a'_j y b_k tantas veces como indique el número de objetos h'. En consecuencia, para que el objeto prohibidor s_6^- pueda cambiar en el correspondiente permitidor es necesario no solo que el objeto promotor s_6 esté presente, sino también al menos un h'. Si este no es el caso, entonces hemos terminado con esta etapa y s_6 se transforma a sí mismo en s_7, para continuar con la siguiente etapa de esta fase. En otro caso, sustituimos cada a'_j y cada b_k por dos copias suyas y cambiamos s_6^+ por s_6^- y s_6 para indicar que tiene que hacerse otra multiplicación, si es posible.

Finalmente, las reglas de R^{3_2} se usan para borrar el viejo símbolo de la celda número y y escribir en ella el nuevo. Esto se hace cancelando cada copia de a'_j con un a_j y transformando cada copia de b_k en a_k.

La siguiente fase se ocupa del problema de comprobar si se ha alcanzado un estado final, realizando una de las siguientes acciones:

(a) Si no se ha alcanzado un estado final, entonces el proceso continúa de nuevo a partir de la fase 1, para simular otro paso de la computación de la máquina de Turing. Esto se hace introduciendo el objeto promotor s_1 dentro de la membrana.

(b) Si se ha alcanzado el estado q_Y o el estado q_N, entonces inserta el objeto promotor s_9 para devolver la salida y terminar.

Esta fase se inicia con la presencia del objeto promotor s_8 y termina, bien cuando aparece el objeto s_1 para volver a la fase 1, bien cuando se expulsa la salida al entorno externo y se eliminan todos los objetos del sistema, con lo que este para.

Fase 4: *Comprobación del estado.*

Esta fase se lleva a cabo mediante las reglas del conjunto R^4. Esta es la última fase de la simulación de la máquina de Turing. Detecta si se ha alcanzado o no un estado final. En el primer caso devuelve un objeto Yes o un objeto No al entorno externo, y para. En el segundo, continuamos la simulación a partir de la fase 1.

Las reglas de R^{4_1} se usan para distinguir entre todas las posibilidades: si q_Y o q_N está presente, lo que significa que hemos alcanzado el estado final de aceptación o el estado final de rechazo, entonces continuamos con la siguiente etapa de la fase, en lugar de con la fase 1, introduciendo el objeto promotor s_9; si cualquiera de los objetos $q_0, \ldots, q_n$ está presente, entonces la simulación de la máquina de Turing no ha concluido. Por tanto, introducimos el objeto promotor s_1 para comenzar de nuevo a partir de la fase 1. En todos los casos el objeto permitidor s_8^+ se transforma a sí mismo en el objeto prohibidor s_8^-.

Las reglas de R^{4_2} solo se aplican cuando se ha alcanzado un estado final. Su propósito es devolver la salida de la computación (Yes o No, dependiendo de si el estado es q_Y o q_N) al entorno externo y eliminar todos los objetos h y todos los objetos a_i presentes en el sistema. Además, introduce catorce objetos s_E, cada uno de los cuales se usará para

eliminar un objeto prohibidor y expulsar un objeto #. Así, ya no quedarán objetos en la membrana y, como consecuencia, la computación será de parada.

4.3. Verificación formal

En la sección anterior hemos construido, para cada máquina de Turing, un sistema P de aceptación de lenguajes que simula las computaciones de dicha máquina. Además, hemos descrito informalmente el funcionamiento de este sistema, mostrando cómo es capaz de reproducir un paso de computación de la máquina de Turing y, por consiguiente, también puede reproducir la computación en su totalidad.

En lo que sigue vamos a demostrar que si TM es una máquina de Turing, como dispositivo de aceptación de lenguajes, entonces el sistema $\Pi_{TM} \in \mathcal{LA}$ y, además, acepta exactamente el mismo lenguaje que TM. Para ello, verificaremos por separado que cada fase del funcionamiento del sistema es correcta, de donde se deducirá la corrección global de Π_{TM} en el sentido antes indicado.

Notación. *En esta sección consideraremos que $\mathcal{C} = \{C^i\}_{i<r}$ es una computación del sistema Π_{TM}.*

Notamos por $S^- = s^-_{I_1} \dots s^-_{I_5} s^-_1 \dots s^-_9$ el multiconjunto de objetos prohibidores y, si de dicho multiconjunto eliminamos el objeto s_j, entonces lo notamos por $S^-(j)$. Es decir, $S^-(j) = S^- \setminus \{s_j\}$.

Notamos por $R^- = \{s^-_l \to s^-_l : l = I_1, \dots, I_5, 1, \dots, 9\}$ el conjunto de reglas prohibidoras y, si de dicho conjunto eliminamos la regla $s^-_j \to s^-_j$, entonces lo notamos por $R^-(j)$. Es decir, $R^-(j) = R^- \setminus \{s^-_j \to s^-_j\}$.

4.3.1. Comprobación del multiconjunto de entrada

A continuación verificamos la fase I, en la cual se analiza si el multiconjunto introducido en la membrana de entrada representa adecuadamente, según la función *cod*, una cadena de entrada para la máquina de Turing. Si ese es el caso, entonces justificamos que esta fase termina y, además, lo hace en las condiciones necesarias para que se lleve a cabo la fase 1. En caso contrario, probamos que la computación $\mathcal{C}$ no es de parada.

En primer lugar, descartamos aquellos multiconjuntos que codifican que algún símbolo va en la celda cero, la celda más a la izquierda de la cinta de la máquina de Turing, ya que esta siempre contendrá el símbolo $\triangleright$.

Proposición 4.5. *Supongamos que*

$$M_1^0 = \mathcal{M}_1 + a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$$

con $w > 0$, $1 \leq u_1, \dots, u_w \leq p$, $k \geq 0$, $1 \leq j_1, \dots, j_k \leq p$ *y* $x_1, \dots, x_k \geq 1$. *Entonces,*

$$M_1^7 = q_0 a_0 S^- a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} a'_{u_1} \dots a'_{u_w} a''_{j_1}{}^{2^{x_1-1}} \dots a''_{j_k}{}^{2^{x_k-1}}$$

Además, para cada $i \geq 7$ *la configuración* C^i *no es de parada.*

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
0	$q_0 a_0 a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_1} S^-$	$s_{I_1} s_{I_1}^- \to s_{I_1}^+; R^-(I_1)$
1	$q_0 a_0 a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_1}^+ S^-(I_1)$	$a_l s_{I_1}^+ \to a_l s_{I_1}^- s_{I_2}$ (para algún $l \in \{u_1, \dots, u_w, j_1, \dots, j_k\}$); $R^-(I_1)$
2	$q_0 a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_2} S^-$	$s_{I_2} s_{I_2}^- \to s_{I_2}^+; R^-(I_2)$
3	$q_0 a_0 a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_2}^+ S^-(I_2)$	$a_l \to a_l a'_l$ (l=u_1,...,u_w,j_1,...,j_k); $s_{I_2}^+ \to s_{I_2}^- s_{I_3}; R^-(I_2)$
4	$q_0 a_0 a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3} S^-$ $a'_{u_1} \dots a'_{u_w} a'_{j_1}{}^{2^{x_1}} \dots a'_{j_k}{}^{2^{x_k}}$	$s_{I_3} s_{I_3}^- \to s_{I_3}^+; R^-(I_3)$
5	$q_0 a_0 a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3}^+ S^-(I_3)$ $a'_{u_1} \dots a'_{u_w} a'_{j_1}{}^{2^{x_1}} \dots a'_{j_k}{}^{2^{x_k}}$	$a_l'^2 \to a''_l$ (l=j_1,...,j_k); $R^-(I_3)$
6	$q_0 a_0 a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3}^+ S^-(I_3)$ $a'_{u_1} \dots a'_{u_w} a''_{j_1}{}^{2^{x_1-1}} \dots a''_{j_k}{}^{2^{x_k-1}}$	$a'_l s_{I_3}^+ \to a'_l s_{I_3}^-$ (para algún $l \in \{u_1, \dots, u_w\}$); $R^-(I_3)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
7	$q_0 a_0 a_{u_1} \dots a_{u_w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} S^-$ $a'_{u_1} \dots a'_{u_w} a''^{\,2^{x_1-1}}_{j_1} \dots a''^{\,2^{x_k-1}}_{j_k}$	

Para cada $i \geq 7$ se tiene que el conjunto de reglas en $amr_{i,i+1}$ para la membrana 1 es R^- y que $C^i = C^{i+1}$. Por tanto, para cada $i \geq 7$ la configuración C^i no es de parada. □

La aparición del objeto s_{I_4}, tal y como mostramos en el siguiente lema, induce que el número de objetos a'_l, para cada $l = 1, \dots, p$, se divida por dos, salvo que el sistema solamente contenga un tal objeto, pues entonces este se elimina.

Lema 4.6. *Supongamos que*

$$M_1^y = q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_{I_4} S^- a'_{j_0} a'^{\,2^{x_1}}_{j_1} \dots a'^{\,2^{x_k}}_{j_k}$$

con $w \geq 0$, $1 \leq u_1, \dots, u_w \leq p$, $v_1, \dots, v_w \geq 1$, $k \geq 1$, $1 \leq j_0, j_1, \dots, j_k \leq p$ *y* $x_1, \dots, x_k \geq 1$. *Entonces,*

$$M_1^{y+6} = q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_{I_4} S^- a'^{\,2^{x_1-1}}_{j_1} \dots a'^{\,2^{x_k-1}}_{j_k}$$

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_{I_4} S^-$ $a'_{j_0} a'^{\,2^{x_1}}_{j_1} \dots a'^{\,2^{x_k}}_{j_k}$	$s_{I_4} s^-_{I_4} \to s^+_{I_4}; R^-(I_4)$
$y+1$	$q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s^+_{I_4} S^-(I_4)$ $a'_{j_0} a'^{\,2^{x_1}}_{j_1} \dots a'^{\,2^{x_k}}_{j_k}$	$a'^2_l \to a''_l \quad (l=j_1,\dots,j_k);$ $R^-(I_4)$
$y+2$	$q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s^+_{I_4} S^-(I_4)$ $a'_{j_0} a''^{\,2^{x_1-1}}_{j_1} \dots a''^{\,2^{x_k-1}}_{j_k}$	$a'_{j_0} s^+_{I_4} \to s^-_{I_4} s_{I_5};$ $R^-(I_4)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$y+3$	$q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_{I_5} S^-$ $a''_{j_1}{}^{2^{x_1-1}} \dots a''_{j_k}{}^{2^{x_k-1}}$	$s_{I_5} s^-_{I_5} \to s^+_{I_5}; R^-(I_5)$
$y+4$	$q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s^+_{I_5} S^-(I_5)$ $a''_{j_1}{}^{2^{x_1-1}} \dots a''_{j_k}{}^{2^{x_k-1}}$	$a''_l \to a'_l$ $(l=j_1,\dots,j_k)$; $R^-(I_5)$
$y+5$	$q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s^+_{I_5} S^-(I_5)$ $a'_{j_1}{}^{2^{x_1-1}} \dots a'_{j_k}{}^{2^{x_k-1}}$	$a'_l s^+_{I_5} \to a'_l s^-_{I_5} s_{I_4}$ (para algún $l \in \{j_1,\dots,j_k\}$); $R^-(I_5)$
$y+6$	$q_0 a_0 a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_{I_4} S^-$ $a'_{j_1}{}^{2^{x_1-1}} \dots a'_{j_k}{}^{2^{x_k-1}}$	

□

A continuación, veamos que un multiconjunto de entrada debe incluir los símbolos de la cadena que codifica en celdas consecutivas a partir de la celda número uno, la segunda más a la izquierda.

Proposición 4.7. *Supongamos que*

$$M_1^0 = \mathcal{M}_1 + a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$$

con $w \geq 0$, $1 \leq u_1, \dots, u_w \leq p$, $k \geq 1$, $1 \leq j_1, \dots j_k \leq p$ *y* $x_1, \dots, x_k > w+1$. *Entonces,*

$$M_1^{7+6\cdot w+3} = q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} S^- a''_{j_1}{}^{2^{x_1-(w+2)}} \dots a''_{j_k}{}^{2^{x_k-(w+2)}}$$

Además, para cada $i \geq 7 + 6 \cdot w + 3$ *la configuración* C^i *no es de parada.*

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
0	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_1} S^-$	$s_{I_1} s^-_{I_1} \to s^+_{I_1}; R^-(I_1)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
1	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_1}^+ S^-(I_1)$	$a_l s_{I_1}^+ \to a_l s_{I_1}^- s_{I_2}$ (para algún $l \in \{u_1, \dots, u_w, j_1, \dots, j_k\}$); $R^-(I_1)$
2	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_2} S^-(I_1)$	$s_{I_2} s_{I_2}^- \to s_{I_2}^+; R^-(I_2)$
3	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_2}^+ S^-(I_2)$	$a_l \to a_l a_l'$ $(l{=}u_1,\dots,u_w,j_1,\dots,j_k)$; $s_{I_2}^+ \to s_{I_2}^- s_{I_3}; R^-(I_2)$
4	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3} S^-$ $a_{u_1}'^{\,2^1} \dots a_{u_w}'^{\,2^w} a_{j_1}'^{\,2^{x_1}} \dots a_{j_k}'^{\,2^{x_k}}$	$s_{I_3} s_{I_3}^- \to s_{I_3}^+; R^-(I_3)$
5	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3}^+ S^-(I_3)$ $a_{u_1}'^{\,2^1} \dots a_{u_w}'^{\,2^w} a_{j_1}'^{\,2^{x_1}} \dots a_{j_k}'^{\,2^{x_k}}$	$a_l'^2 \to a_l''$ $(l{=}u_1,\dots,u_w,j_1,\dots,j_k)$; $R^-(I_3)$
6	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3}^+ S^-(I_3)$ $a_{u_1}''^{\,2^0} \dots a_{u_w}''^{\,2^{w-1}} a_{j_1}''^{\,2^{x_1-1}} \dots a_{j_k}''^{\,2^{x_k-1}}$	$a_l'' \to a_l'$ $(l{=}u_1,\dots,u_w,j_1,\dots,j_k)$; $s_{I_3}^+ \to s_{I_3}^- s_{I_4}; R^-(I_3)$
7	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_4} S^-$ $a_{u_1}'^{\,2^0} \dots a_{u_w}'^{\,2^{w-1}} a_{j_1}'^{\,2^{x_1-1}} \dots a_{j_k}'^{\,2^{x_k-1}}$	

Aplicando w veces el lema 4.6, obtenemos que

$$M_1^{7+6\cdot w} = q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_4} S^- a_{j_1}'^{\,2^{x_1-(w+1)}} \dots a_{j_k}'^{\,2^{x_k-(w+1)}}$$

Entonces, haciendo $t = 7 + 6 \cdot w$,

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
t	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_4} S^-$ $a_{j_1}'^{\,2^{x_1-(w+1)}} \dots a_{j_k}'^{\,2^{x_k-(w+1)}}$	$s_{I_4} s_{I_4}^- \to s_{I_4}^+; R^-(I_4)$
$t+1$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_4}^+ S^-(I_4)$ $a_{j_1}'^{\,2^{x_1-(w+1)}} \dots a_{j_k}'^{\,2^{x_k-(w+1)}}$	$a_l'^2 \to a_l''$ $(l{=}j_1,\dots,j_k)$; $R^-(I_4)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$t+2$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_4}^+ S^-(I_4)$ $a''_{j_1}{}^{2^{x_1-(w+2)}} \dots a''_{j_k}{}^{2^{x_k-(w+2)}}$	$s_{I_4}^+ \to s_{I_4}^-; R^-(I_4)$
$t+3$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} S^-$ $a''_{j_1}{}^{2^{x_1-(w+2)}} \dots a''_{j_k}{}^{2^{x_k-(w+2)}}$	

Para cada $i \geq 7+6\cdot w+3$ se tiene que el conjunto de reglas en $amr_{i,i+1}$ para la membrana 1 es R^- y que $C^i = C^{i+1}$. Por tanto, para cada $i \geq 7+6\cdot w+3$ se tiene que la configuración C^i no es de parada. □

Finalmente, vamos a justificar que si el multiconjunto de entrada codifica más de un símbolo en la misma celda, entonces la computación C no es de parada.

Proposición 4.8. *Supongamos que*

$$M_1^0 = \mathcal{M}_1 + a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$$

con $w \geq 1$, $v \geq 2$, $1 \leq u_1, \dots, u_w^1, \dots, u_w^v$, $k \geq 0$, $1 \leq j_1, \dots j_k \leq p$ *y* $x_1, \dots, x_k > w$. *Entonces,*

$$M_1^{7+6\cdot(w-1)+3} = q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^v}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} S^- \; a'_{u_w^1}{}^{2^0} \dots a'_{u_w^v}{}^{2^0} a''_{j_1}{}^{2^{x_1-(w+1)}} \dots a''_{j_k}{}^{2^{x_k-(w+1)}}$$

Además, para cada $i \geq 7+6\cdot(w-1)+3$ *la configuración* C^i *no es de parada.*

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
0	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_1} S^-$	$s_{I_1} s_{I_1}^- \to s_{I_1}^+; R^-(I_1)$
1	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_1}^+ S^-(I_1)$	$a_l s_{I_1}^+ \to a_l s_{I_1}^- s_{I_2}$ (para algún $l \in \{u_1, \dots, u_w^1, \dots, u_w^v, j_1, \dots, j_k\}$); $R^-(I_1)$
2	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_2} S^-$	$s_{I_2} s_{I_2}^- \to s_{I_2}^+; R^-(I_2)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
3	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_2}^+ S^-(I_2)$	$a_l \to a_l a_l'$ (l=$u_1,\dots,u_w^1,\dots,u_w^v,j_1,\dots,j_k$); $s_{I_1}^+ \to s_{I_2}^- s_{I_3}; R^-(I_2)$
4	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3} S^-$ $a'_{u_1}{}^{2^1} \dots a'_{u_w^1}{}^{2^w} \dots a'_{u_w^v}{}^{2^w} a'_{j_1}{}^{2^{x_1}} \dots a'_{j_k}{}^{2^{x_k}}$	$s_{I_3} s_{I_3}^- \to s_{I_3}^+; R^-(I_3)$
5	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3}^+ S^-(I_3)$ $a'_{u_1}{}^{2^1} \dots a'_{u_w^1}{}^{2^w} \dots a'_{u_w^v}{}^{2^w} a'_{j_1}{}^{2^{x_1}} \dots a'_{j_k}{}^{2^{x_k}}$	$a_l'^2 \to a_l''$ (l=$u_1,\dots,u_w^1,\dots,u_w^v,j_1,\dots,j_k$); $R^-(I_3)$
6	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_3}^+ S^-(I_3)$ $a''_{u_1}{}^{2^0} \dots a''_{u_w^1}{}^{2^{w-1}} \dots a''_{u_w^v}{}^{2^{w-1}}$ $a''_{j_1}{}^{2^{x_1-1}} \dots a''_{j_k}{}^{2^{x_k-1}}$	$a_l'' \to a_l'$ (l=$u_1,\dots,u_w^1,\dots,u_w^v,j_1,\dots,j_k$); $s_{I_3}^+ \to s_{I_3}^- s_{I_4}; R^-(I_3)$
7	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_4} S^-$ $a'_{u_1}{}^{2^0} \dots a'_{u_w^1}{}^{2^{w-1}} \dots a'_{u_w^v}{}^{2^{w-1}}$ $a'_{j_1}{}^{2^{x_1-1}} \dots a'_{j_k}{}^{2^{x_k-1}}$	

Aplicando $w-1$ veces el lema 4.6, obtenemos que

$$M_1^{7+6\cdot(w-1)} = q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_{I_4} S^- \; a'_{u_w^1}{}^{2^0} \dots a'_{u_w^v}{}^{2^0} a'_{j_1}{}^{2^{x_1-w}} \dots a'_{j_k}{}^{2^{x_k-w}}$$

Entonces, haciendo $t = 7 + 6 \cdot (w-1)$,

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
t	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_{I_4} S^- a'_{u_w^1}{}^{2^0} \dots a'_{u_w^v}{}^{2^0}$ $a'_{j_1}{}^{2^{x_1-w}} \dots a'_{j_k}{}^{2^{x_k-w}}$	$s_{I_4} s_{I_4}^- \to s_{I_4}^+; R^-(I_4)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$t+1$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_{I_4}^+ S^-(I_4) {a'_{u_w^1}}^{2^0} \dots {a'_{u_w^v}}^{2^0}$ ${a'_{j_1}}^{2^{x_1}-w} \dots {a'_{j_k}}^{2^{x_k}-w}$	$a_l'^2 \to a_l''$ $(l=j_1,\dots,j_k)$; $R^-(I_4)$
$t+2$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_{I_4}^+ S^-(I_4) {a'_{u_w^1}}^{2^0} \dots {a'_{u_w^v}}^{2^0}$ ${a''_{j_1}}^{2^{x_1}-(w+1)} \dots {a''_{j_k}}^{2^{x_k}-(w+1)}$	$a'_f a'_g s_{I_4}^+ \to a'_f a'_g s_{I_4}^-$ (para algunos $f,g \in \{u_w^1,\dots,u_w^v\}$, con $f \neq g$); $R^-(I_4)$
$t+3$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w^1}^{2^w} \dots a_{u_w^v}^{2^w} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $S^- {a'_{u_w^1}}^{2^0} \dots {a'_{u_w^v}}^{2^0}$ ${a''_{j_1}}^{2^{x_1}-(w+1)} \dots {a''_{j_k}}^{2^{x_k}-(w+1)}$	

Para cada $i \geq 7 + 6 \cdot (w-1) + 3$ se tiene que el conjunto de reglas en $amr_{i,i+1}$ para la membrana 1 es R^- y que $C^i = C^{i+1}$. Por tanto, para cada $i \geq 7 + 6 \cdot (w-1) + 3$ la configuración C^i no es de parada. □

A continuación, justificamos que el multiconjunto vacío es admisible como multiconjunto de entrada para el sistema P asociado a la máquina de Turing.

Proposición 4.9. *Supongamos que* $M_1^0 = \mathcal{M}_1$. *Entonces,*

$$M_1^2 = q_0 a_0 s_1 S^-$$

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
0	$q_0 a_0 s_{I_1} S^-$	$s_{I_1} s_{I_1}^- \to s_{I_1}^+; R^-(I_1)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
1	$q_0 a_0 s_{I_1}^+ S^-(I_1)$	$s_{I_1}^+ \to s_{I_1}^- s_1; R^-(I_1)$
2	$q_0 a_0 s_1 S^-$	

□

Seguidamente, veamos que también se admiten como entrada del sistema aquellos multiconjuntos que codifican cadenas de tal forma que sus símbolos se incluyen consecutivamente a partir de la celda número uno y, además, solo se incluye un símbolo por celda.

Proposición 4.10. *Supongamos que*

$$M_1^0 = \mathcal{M}_1 + a_{u_1}^{2^1} \dots a_{u_w}^{2^w}$$

con $w \geq 1$, $1 \leq u_1, \dots, u_w \leq p$. *Entonces,*

$$M_1^{7+6\cdot(w-1)+4} = q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_1 S^-$$

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
0	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_1} S^-$	$s_{I_1} s_{I_1}^- \to s_{I_1}^+; R^-(I_1)$
1	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_1}^+ S^-(I_1)$	$a_l s_{I_1}^+ \to a_l s_{I_1}^- s_{I_2}$ (para algún $l \in \{u_1, \dots, u_w\}$); $R^-(I_1)$
2	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_2} S^-$	$s_{I_2} s_{I_2}^- \to s_{I_2}^+; R^-(I_2)$
3	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_2}^+ S^-(I_1)$	$a_l \to a_l a_l'$ $(l=u_1,\dots,u_w)$; $s_{I_2}^+ \to s_{I_2}^- s_{I_3}; R^-(I_2)$
4	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_3} S^-$ $a_{u_1}'^{2^1} \dots a_{u_w}'^{2^w}$	$s_{I_3} s_{I_3}^- \to s_{I_3}^+; R^-(I_3)$
5	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_3}^+ S^-(I_3)$ $a_{u_1}'^{2^1} \dots a_{u_w}'^{2^w}$	$a_l'^2 \to a_l''$ $(l=u_1,\dots,u_w)$; $R^-(I_3)$
6	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_3}^+ S^-(I_3)$ $a_{u_1}''^{2^0} \dots a_{u_w}''^{2^{w-1}}$	$a_l'' \to a_l'$ $(l=u_1,\dots,u_w)$; $s_{I_3}^+ \to s_{I_3}^- s_{I_4}; R^-(I_3)$
7	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_4} S^-$ $a_{u_1}'^{2^0} \dots a_{u_w}'^{2^{w-1}}$	

Aplicando $w-1$ veces el lema 4.6, obtenemos que

$$M_1^{7+6\cdot(w-1)} = q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_4} S^- {a'_{u_w}}^{2^0}$$

Entonces, haciendo $t = 7 + 6 \cdot (w-1)$,

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
t	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_4} S^- {a'_{u_w}}^{2^0}$	$s_{I_4} s_{I_4}^- \to s_{I_4}^+; R^-(I_4)$
$t+1$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_4}^+ S^-(I_4) {a'_{u_w}}^{2^0}$	$a'_{u_w} s_{I_4}^+ \to s_{I_4}^- s_{I_5}; R^-(I_4)$
$t+2$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_5} S^-$	$s_{I_5} s_{I_5}^- \to s_{I_5}^+; R^-(I_5)$
$t+3$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_{I_5}^+ S^-(I_5)$	$s_{I_5}^+ \to s_{I_5}^- s_1; R^-(I_5)$
$t+4$	$q_0 a_0 a_{u_1}^{2^1} \dots a_{u_w}^{2^w} s_1 S^-$	

□

4.3.2. Lectura del símbolo

A continuación vamos a verificar la fase 1, en la cual se obtiene el símbolo leído por la cabeza de la máquina de Turing.

No obstante, antes establecemos un lema en el que se justifica que la aparición de un objeto s_2 en la membrana del sistema induce, junto con la presencia de al menos un objeto h', la división por dos del número de objetos a'_l existentes en la misma.

Lema 4.11. *Supongamos que*

$$M_1^y = q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2 S^- {h'}^x {a'_{j_1}}^{2^{x_1}} \dots {a'_{j_k}}^{2^{x_k}}$$

con $0 \le u \le n$, $t \ge 0$, $w \ge 0$, $0 \le u_1, \dots, u_w \le m$, $0 \le v_1 < \dots < v_w$, $x > 0$, $k \ge 0$, $0 \le j_1, \dots, j_k \le m$, $0 \le x_1 < \dots < x_k$. *Entonces,*

$$M_1^{y+3} = q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2 S^- {h'}^{(x-1)} {a'_{j_2}}^{2^{x_2-1}} \dots {a'_{j_k}}^{2^{x_k-1}}, \quad si\ x_1 = 0$$

$$M_1^{y+3} = q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2 S^- {h'}^{(x-1)} {a'_{j_1}}^{2^{x_1-1}} \dots {a'_{j_k}}^{2^{x_k-1}}, \quad si\ x_1 > 0$$

Demostración.

Caso 1: $x_1 = 0$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2 S^-$ $h'^x a'_{j_1} a'_{j_2}{}^{2^{x_2}} \dots a'_{j_k}{}^{2^{x_k}}$	$h' s_2 s_2^- \to s_2^+; R^-(2)$
$y+1$	$q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2^+ S^-(2)$ $h'^{x-1} a'_{j_1} a'_{j_2}{}^{2^{x_2}} \dots a'_{j_k}{}^{2^{x_k}}$	$a_l'^2 \to a_l''$ (con $l = j_2, \dots, j_k$); $R^-(2)$
$y+2$	$q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2^+ S^-(2)$ $h'^{x-1} a'_{j_1} a''_{j_2}{}^{2^{x_2-1}} \dots a''_{j_k}{}^{2^{x_k-1}}$	$a'_{j_1} \to \lambda$; $a_l'' \to a_l'$ (con $l = j_2, \dots, j_k$); $s_2^+ \to s_2^- s_2; R^-(2)$
$y+3$	$q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2 S^-$ $h'^{x-1} a'_{j_2}{}^{2^{x_2-1}} \dots a'_{j_k}{}^{2^{x_k-1}}$	

Caso 2: $x_1 > 0$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2 S^-$ $h'^x a'_{j_1}{}^{2^{x_1}} \dots a'_{j_k}{}^{2^{x_k}}$	$h' s_2 s_2^- \to s_2^+; R^-(2)$
$y+1$	$q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2^+ S^-(2)$ $h'^{x-1} a'_{j_1}{}^{2^{x_1}} \dots a'_{j_k}{}^{2^{x_k}}$	$a_l'^2 \to a_l''$ (con $l = j_1, \dots, j_k$); $R^-(2)$
i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$y+2$	$q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2^+ S^-(2)$ $h'^{x-1} a''_{j_1}{}^{2^{x_1-1}} \dots a''_{j_k}{}^{2^{x_k-1}}$	$a_l'' \to a_l'$ (con $l = j_1, \dots, j_k$); $s_2^+ \to s_2^- s_2; R^-(2)$
$y+3$	$q_u h^t a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_2 S^-$ $h'^{x-1} a'_{j_1}{}^{2^{x_1-1}} \dots a'_{j_k}{}^{2^{x_k-1}}$	

□

Estamos, por tanto, en condiciones de probar el resultado requerido.

Proposición 4.12. *Supongamos que*

$$M_1^y = q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_1 S^-$$

con $0 \leq u \leq n$, $x \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $k \geq 0$, $0 \leq j_1, \dots, j_k \leq m$ *y* $0 \leq v_1 < \dots < v_w < x \leq x_1 < \dots x_k$. *Entonces,*

$$\begin{aligned} M_1^{y+2+3\cdot x+3} &= q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_4 S^- a'_{j_1}, && \text{si } x_1 = x \\ M_1^{y+2+3\cdot x+3} &= q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_4 S^-, && \text{si } x_1 > x \end{aligned}$$

Demostración.

Caso 1: $x_1 = x$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_1 S^-$	$s_1 s_1^- \to s_1^+; R^-(1)$
$y+1$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_1^+ S^-(1)$	$h \to h'; a_l \to a_l a'_l$ (con $l = u_1, \dots, u_w, j_1, \dots, j_k$); $s_1^+ \to s_1^- s_2; R^-(1)$
$y+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_2 S^-$ $h'^x a'^{2^{v_1}}_{u_1} \dots a'^{2^{v_w}}_{u_w} a'^{2^{x_1}}_{j_1} \dots a'^{2^{x_k}}_{j_k}$	

Aplicando x veces el lema 4.11, obtenemos que

$$M_1^{y+2+3x} = q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_2 S^- a'_{j_1} a'^{2^{x_2-x}}_{j_2} \dots a'^{2^{x_k-x}}_{j_k}$$

Entonces, haciendo $t = y + 2 + 3 \cdot x$,

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
t	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_2 S^- a'_{j_1} a'^{2^{x_2-x}}_{j_2} \dots a'^{2^{x_k-x}}_{j_k}$	$s_2 \to s_3; R^-(2)$
$t+1$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_3 S^- a'_{j_1} a'^{2^{x_2-x}}_{j_2} \dots a'^{2^{x_k-x}}_{j_k}$	$s_3 s_3^- \to s_3^+; R^-(3)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$t+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_3^+ S^-(3) a'_{j_1} a'^{2^{x_2-x}}_{j_2} \dots a'^{2^{x_k-x}}_{j_k}$	$a'^2_l \to \lambda$ (con $l = j_2, \dots, j_k$); $s_3^+ \to s_3^- s_4; R^-(3)$
$t+3$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_4 S^- a'_{j_1}$	

Caso 2: $x_1 > x$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_1 S^-$	$s_1 s_1^- \to s_1^+; R^-(1)$
$y+1$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_1^+ S^-(1)$	$h \to hh'$; $a_l \to a_l a'_l$ (con $l = u_1, \dots, u_w, j_1, \dots, j_k$); $s_1^+ \to s_1^- s_2; R^-(1)$
$y+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_2 S^-$ $h'^x a'^{2^{v_1}}_{u_1} \dots a'^{2^{v_w}}_{u_w} a'^{2^{x_1}}_{j_1} \dots a'^{2^{x_k}}_{j_k}$	

Aplicando x veces el lema 4.11, obtenemos que

$$M_1^{y+2+3x} = q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_2 S^- a'^{2^{x_1-x}}_{j_1} \dots a'^{2^{x_k-x}}_{j_k}$$

Entonces, haciendo $t = y + 2 + 3 \cdot x$,

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
t	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_2 S^- a'^{2^{x_1-x}}_{j_1} \dots a'^{2^{x_k-x}}_{j_k}$	$s_2 \to s_3; R^-(2)$
$t+1$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_3 S^- a'^{2^{x_1-x}}_{j_1} \dots a'^{2^{x_k-x}}_{j_k}$	$s_3 s_3^- \to s_3^+; R^-(3)$

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$t+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_3^+ S^-(3) a'^{2^{x_1-x}}_{j_1} \dots a'^{2^{x_k-x}}_{j_k}$	$a_l'^2 \to \lambda$ (con $l = j_2, \dots, j_k$); $s_3^+ \to s_3^- s_4; R^-(3)$
$t+3$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}$ $s_4 S^-$	

□

4.3.3. Cálculo del nuevo símbolo, movimiento de la cabeza y cambio del estado

A continuación establecemos la corrección de la fase 2, en la que se simula la aplicación de la función de transición de la máquina de Turing. Para ello, partimos de una configuración en la cual se codifica que el estado de la máquina de Turing es q_u, la celda analizada es la número x y el símbolo leído es a_j (o B), y justificamos que, en efecto, se llega a una configuración en la cual se codifica que el estado de la máquina de Turing es $q_{Q(u,j)}$, la nueva celda a analizar es la número $x+D(u,j)$ (preservándose el número de la celda que habíamos leído) y el símbolo a escribir es $a_{A(u,j)}$ (o el símbolo blanco).

Proposición 4.13. *Supongamos que*

$$M_1^y = q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_4 S^- a'_j$$

con $0 \leq u \leq n$, $x \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$ *y* $j \in \{u_1, \dots, u_w\}$. *Entonces,*

$$M_1^{y+3} = q_{Q(u,j)} h^{x+D(u,j)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x a'_j b_{A(u,j)}, \quad \text{si } A(u,j) \neq B$$
$$M_1^{y+3} = q_{Q(u,j)} h^{x+D(u,j)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x a'_j, \quad \text{si } A(u,j) = B$$

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_4 S^- a'_j$	$s_4 s_4^- \to s_4^+; R^-(4)$
$y+1$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_4^+ S^-(4) a'_j$	$h \to hh'$; $s_4^+ \to s_4^- s_5; R^-(4)$
$y+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_5 S^- h'^x a'_j$	$s_5 s_5^- \to s_5^+; R^-(5)$
$y+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_5^+ S^-(5) h'^x a'_j$	

Las distintas posibilidades que podemos encontrarnos para las reglas en $amr_{y+2,y+3}$ para la membrana 1 se resumen en la siguiente tabla:

	$A(u,j) \neq B$	$A(u,j) = B$
$D(u,j) = -1$	$q_u a'_j h \to q_{Q(u,j)} a'_j b_{A(u,j)}$; $s_5^+ \to s_5^- s_6; R^-(5)$	$q_u a'_j h \to q_{Q(u,j)} a'_j$; $s_5^+ \to s_5^- s_6; R^-(5)$
$D(u,j) = 0$	$q_u a'_j \to q_{Q(u,j)} a'_j b_{A(u,j)}$; $s_5^+ \to s_5^- s_6; R^-(5)$	$q_u a'_j \to q_{Q(u,j)} a'_j$; $s_5^+ \to s_5^- s_6; R^-(5)$
$D(u,j) = 1$	$q_u a'_j \to q_{Q(u,j)} a'_j b_{A(u,j)} h$; $s_5^+ \to s_5^- s_6; R^-(5)$	$q_u a'_j \to q_{Q(u,j)} a'_j h$; $s_5^+ \to s_5^- s_6; R^-(5)$

Por tanto,

$$M_1^{y+3} = q_{Q(u,j)} h^{x+D(u,j)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x a'_j b_{A(u,j)}, \quad \text{si } A(u,j) \neq B$$
$$M_1^{y+3} = q_{Q(u,j)} h^{x+D(u,j)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x a'_j, \quad \text{si } A(u,j) = B$$

□

Proposición 4.14. *Supongamos que*

$$M_1^y = q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_4 S^-$$

con $0 \leq u \leq n$, $x \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_w \leq m$ *y* $0 \leq v_1 < \dots < v_w$.

Entonces,

$$M_1^{y+3} = q_{Q(u,B)} h^{x+D(u,B)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x b_{A(u,B)}, \quad si\ A(u,B) \neq B$$
$$M_1^{y+3} = q_{Q(u,B)} h^{x+D(u,B)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x, \quad si\ A(u,B) = B$$

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_4 S^-$	$s_4 s_4^- \to s_4^+; R^-(4)$
$y+1$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_4^+ S^-(4)$	$h \to hh'$; $s_4^+ \to s_4^- s_5; R^-(4)$
$y+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_5 S^- h'^x$	$s_5 s_5^- \to s_5^+; R^-(5)$
$y+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_5^+ S^-(5) h'^x$	

Las distintas posibilidades que podemos encontrarnos para las reglas en $amr_{y+2,y+3}$ para la membrana 1 se resumen en la siguiente tabla:

	$A(u,B) \neq B$	$A(u,B) = B$
$D(u,B) = -1$	$q_u h \to q_{Q(u,B)} b_{A(u,B)}$; $s_5^+ \to s_5^- s_6; R^-(5)$	$q_u h \to q_{Q(u,B)}$; $s_5^+ \to s_5^- s_6; R^-(5)$
$D(u,B) = 0$	$q_u \to q_{Q(u,B)} b_{A(u,B)}$; $s_5^+ \to s_5^- s_6; R^-(5)$	$q_u \to q_{Q(u,B)}$; $s_5^+ \to s_5^- s_6; R^-(5)$
$D(u,B) = 1$	$q_u \to q_{Q(u,B)} b_{A(u,B)} h$; $s_5^+ \to s_5^- s_6; R^-(5)$	$q_u \to q_{Q(u,B)} h$; $s_5^+ \to s_5^- s_6; R^-(5)$

Por tanto,

$$M_1^{y+3} = q_{Q(u,B)} h^{x+D(u,B)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x b_{A(u,B)}, \quad \text{si } A(u,B) \neq B$$
$$M_1^{y+3} = q_{Q(u,B)} h^{x+D(u,B)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x, \quad \text{si } A(u,B) = B$$

□

4.3.4. Borrado del viejo símbolo y escritura del nuevo

Seguidamente pasamos a verificar la fase 2, en la que se borra el símbolo leído por la cabeza de la máquina de Turing y se escribe el símbolo indicado por la función de transición. Además, en lo que sigue adoptamos el convenio de que los objetos a'_B y b_B denotan objetos que no están presentes en la membrana del sistema y, por tanto, las reglas correspondientes a dichos objetos no se aplicarán.

En primer lugar, probamos un lema técnico que establece que, si la membrana del sistema Π_{TM} contiene simultáneamente un objeto s_6 y un objeto h', entonces el número de objetos a'_j y b_k se duplica.

Lema 4.15. *Supongamos que*

$$M_1^y = q_u h^v a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x {a'_j}^{2^{x_j}} {b_k}^{2^{x_k}}$$

con $0 \leq u \leq n$, $v \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$, $x > 0$, $j, k \in \{0, \dots, m, B\}$, $x_j, x_k \geq 0$. *Entonces,*

$$M_1^{y+2} = q_u h^v a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^{x-1} {a'_j}^{2^{x_j+1}} {b_k}^{2^{x_k+1}}$$

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_u h^v a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^-$ $h'^x {a'_j}^{2^{x_j}} {b_k}^{2^{x_k}}$	$h' s_6 s_6^- \to s_6^+; R^-(6)$
$y+1$	$q_u h^v a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6^+ S^-(6)$ $h'^{x-1} {a'_j}^{2^{x_j}} {b_k}^{2^{x_k}}$	$a'_j \to {a'_j}^2; b_k \to {b_k}^2;$ $s_6^+ \to s_6^- s_6; R^-(6)$
$y+2$	$q_u h^v a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^-$ $h'^{x-1} {a'_j}^{2^{x_j+1}} {b_k}^{2^{x_k+1}}$	

□

Finalmente establecemos la corrección de la fase 2.

Proposición 4.16. *Supongamos que*

$$M_1^y = q_u h^v a_{u_1}^{2^{v_1}} \dots a_j^{2^x} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x a'_j b_k$$

con $0 \leq u \leq n$, $v \geq 0$, $0 \leq u_1, \ldots, u_w \leq m$, $0 \leq v_1 < \cdots < x < \cdots < v_w$, $w \geq 0$, $j, k \in \{0, \ldots, m, B\}$. *Entonces,*

$$M_1^{y+2\cdot x+3} = q_u h^v a_{u_1}^{2^{v_1}} \ldots a_k^{2^x} \ldots a_{u_w}^{2^{v_w}} s_8 S^-$$

Demostración. Aplicando x veces el lema 4.15 obtenemos que

$$M_1^{y+2\cdot x} = q_u h^v a_{u_1}^{2^{v_1}} \ldots a_j^{2^x} \ldots a_{u_w}^{2^{v_w}} s_6 S^- {a'_j}^{2^x} b_k^{2^x}$$

Entonces, haciendo $t = y + 2 \cdot x$,

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
t	$q_u h^v a_{u_1}^{2^{v_1}} \ldots a_j^{2^x} \ldots a_{u_w}^{2^{v_w}} s_6 S^-$ ${a'_j}^{2^x} b_k^{2^x}$	$s_6 \to s_7; R^-(6)$
$t+1$	$q_u h^v a_{u_1}^{2^{v_1}} \ldots a_j^{2^x} \ldots a_{u_w}^{2^{v_w}} s_7 S^-$ ${a'_j}^{2^x} b_k^{2^x}$	$s_7 s_7^- \to s_7^+; R^-(7)$
$t+2$	$q_u h^v a_{u_1}^{2^{v_1}} \ldots a_j^{2^x} \ldots a_{u_w}^{2^{v_w}} s_7^+ S^-(7)$ ${a'_j}^{2^x} b_k^{2^x}$	$a_j a'_j \to \lambda; b_k \to a_k;$ $s_7^+ \to s_7^- s_8; R^-(7)$
$t+3$	$q_u h^v a_{u_1}^{2^{v_1}} \ldots a_k^{2^x} \ldots a_{u_w}^{2^{v_w}} s_8 S^-$	

□

4.3.5. Comprobación del estado

La fase 4 es la encargada de examinar si el estado alcanzado tras la fase anterior es o no un estado final de la máquina de Turing. Para establecer la corrección de esta fase distinguimos tres casos:

- Que el estado no sea final, en cuyo caso se llega a una configuración que permite activar de nuevo la fase 1.
- Que el estado sea de aceptación, en cuyo caso un objeto Yes se expulsa al entorno externo y se llega a una configuración de parada.
- Que el estado sea de rechazo, en cuyo caso un objeto No se expulsa al entorno externo y se llega a una configuración de parada.

El siguiente resultado demuestra la corrección del primer caso.

Proposición 4.17. *Supongamos que*

$$M_1^y = q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$$

con $0 \leq u \leq n$, $x \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$. *Entonces,*

$$M_1^{y+2} = q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_1 S^-$$

Demostración.

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$	$s_8 s_8^- \to s_8^+; R^-(8)$
$y+1$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8^+ S^-(8)$	$q_u \to q_u s_1$; $s_8^+ \to s_8^-; R^-(8)$
$y+2$	$q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_1 S^-$	

□

La verificación de los casos segundo y tercero se deduce de los resultados que se establecen a continuación.

Proposición 4.18. *Supongamos que*

$$M_1^y = q_Y h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$$

con $x \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$. *Entonces* C^{y+5} *es de parada y* $M_{env}^{y+5} = Yes\#$[14].

Demostración.

i	M_1^i	M_{env}^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_Y h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$		$s_8 s_8^- \to s_8^+; R^-(8)$
$y+1$	$q_Y h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8^+ S^-(8)$		$q_Y \to q_Y s_9$; $s_8^+ \to s_8^-; R^-(8)$

i	M_1^i	M_{env}^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$y+2$	$q_Y h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_9 S^-$		$s_9 s_9^- \to s_9^+; R^-(9)$
$y+3$	$q_Y h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_9^+ S^-(9)$		$q_Y \to (Yes, out); h \to \lambda;$ $a_l \to \lambda$ (con $l = u_1, \dots, u_w$); $s_9^+ \to s_9^- s_E^{14}; R^-(9)$
$y+4$	$s_E^{14} S^-$	Yes	$s_E s_l^- \to (\#, out)$ (con $l = I_1, \dots, I_5, 1, \dots, 9$)
$y+5$		$Yes\#^{14}$	

□

Proposición 4.19. *Supongamos que*

$$M_1^y = q_N h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$$

con $x \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$. *Entonces* C^{y+5} *es de parada y* $M_{env}^{y+5} = No\#^{14}$.

Demostración.

i	M_1^i	M_{env}^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$q_N h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$		$s_8 s_8^- \to s_8^+; R^-(8)$
$y+1$	$q_N h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8^+ S^-(8)$		$q_N \to q_N s_9;$ $s_8^+ \to s_8^-; R^-(8)$
$y+2$	$q_N h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_9 S^-$		$s_9 s_9^- \to s_9^+; R^-(9)$
$y+3$	$q_N h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_9^+ S^-(9)$		$q_N \to (No, out); h \to \lambda;$ $a_l \to \lambda$ (con $l = u_1, \dots, u_w$); $s_9^+ \to s_9^- s_E^{14}; R^-(9)$
$y+4$	$s_E^{14} S^-$	No	$s_E s_l^- \to (\#, out)$ (con $l = I_1, \dots, I_5, 1, \dots, 9$)

i	M_1^i	M_{env}^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$y+5$		$No\#^{14}$	

□

4.3.6. Simulación de una computación de TM

A continuación vamos a demostrar que el sistema Π_{TM} es capaz de reproducir cualquier computación de la máquina de Turing. Por tanto, lo primero que debemos probar es que Π_{TM} puede simular un paso de dicha computación.

Proposición 4.20. *Supongamos que*

$$M_1^y = q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_1 S^-$$

con $0 \leq u \leq n$, $x \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_2 \leq m$, $k \geq 0$, $0 \leq j_1, \dots, j_k \leq m$ *y* $0 \leq v_1 < \cdots v_w < x \leq x_1 < \cdots < x_k$. *Entonces,*

1. *Si* $x_1 = x$ *y* $A(u, j_1) \neq B$,

$$M_1^{y+5\cdot x+11} = q_{Q(u,j_1)} h^{x+D(u,j_1)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{A(u,j_1)}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_8 S^-$$

2. *Si* $x_1 = x$ *y* $A(u, j_1) = B$,

$$M_1^{y+5\cdot x+11} = q_{Q(u,j_1)} h^{x+D(u,j_1)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_2}^{2^{x_2}} \dots a_{j_k}^{2^{x_k}} s_8 S^-$$

3. *Si* $x_1 > x$ *y* $A(u, j_1) \neq B$,

$$M_1^{y+5\cdot x+11} = q_{Q(u,B)} h^{x+D(u,B)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{A(u,B)}^{2^{x}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_8 S^-$$

4. *Si* $x_1 > x$ *y* $A(u, j_1) \neq B$,

$$M_1^{y+5\cdot x+11} = q_{Q(u,B)} h^{x+D(u,B)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_8 S^-$$

Demostración. Basta tener en cuenta las proposiciones 4.12, 4.13, 4.14 y 4.16. □

Proposición 4.21. *Sea $w \in \Sigma_{TM}^*$ una cadena tal que la máquina TM* no para *cuando recibe dicha cadena como entrada. Supongamos que*

$$M_1^0 = \mathcal{M}_1 + cod(w)$$

Entonces $\mathcal{C}$ no es una computación de parada.

Demostración. Basta tener en cuenta las proposiciones 4.9, 4.10, 4.20 y 4.17. □

Proposición 4.22. *Sea $w \in \Sigma_{TM}^*$ una cadena tal que la máquina TM* para *cuando recibe dicha cadena como entrada y sea q el estado de la descripción instantánea de parada correspondiente. Supongamos que para la computación $\mathcal{C} = \{C^i\}_{i<r}$ se tiene que*

$$M_1^0 = \mathcal{M} + cod(w)$$

Entonces $r \in \mathbb{N}$ y, además,

$$\begin{aligned} M_1^{r-1} &= \lambda \text{ y } M_{env}^{r-1} = Yes\#^{14}, && si\ q = q_Y \\ M_1^{r-1} &= \lambda \text{ y } M_{env}^{r-1} = No\#^{14}, && si\ q = q_N \end{aligned}$$

Demostración. Basta tener en cuenta las proposiciones 4.9, 4.10, 4.20, 4.18 y 4.19. □

4.3.7. Aceptación de lenguajes

Para finalizar, vamos a demostrar que el sistema $\Pi_{TM} \in \mathcal{LA}$ (es decir, es un sistema válido de aceptación de lenguajes) y, además, acepta el mismo lenguaje que la máquina de Turing TM.

Teorema 4.23. *El sistema Π_{TM} es un sistema P de aceptación de lenguajes que es válido.*

Demostración. Según las proposiciones 4.5, 4.7, 4.8, 4.20, 4.21 y 4.22, las únicas computaciones de parada de Π_{TM} son aquellas tales que

$$M_1^0 = \mathcal{M}_1 + cod(w)$$

donde $w \in \Sigma_{TM}^*$ es una cadena tal que la máquina TM para al recibirla como entrada.

Entonces, analizando con detalle las demostraciones de las proposiciones 4.18 y 4.19, se deduce que Π_{TM} verifica las dos condiciones necesarias y suficientes para que sea un sistema válido. □

Teorema 4.24. *Sea L el lenguaje aceptado por TM. Entonces Π_{TM} acepta dicho lenguaje.*

Demostración. Basta tener en cuenta el teorema 4.23 y la proposición 4.22. □

Como consecuencia inmediata del resultado obtenido se deduce la completitud computacional de los sistemas P de aceptación de lenguajes.

Teorema 4.25. *Sea L un lenguaje recursivamente enumerable. Entonces existe $\Pi \in \mathcal{LA}(Pri, Coo, nDis)$ (es decir, con prioridad entre las reglas, las cuales son cooperativas, pero sin permitir disolución de membranas) tal que Π acepta L.*

Demostración. Dado L un lenguaje recursivamente enumerable, existe TM una máquina de Turing tal que acepta dicho lenguaje. De los teoremas 4.23 y 4.24, se deduce que $\Pi_{TM} \in \mathcal{LA}(Pri, Coo, nDis)$ y que Π_{TM} acepta L. □

Capítulo 5

Sistemas P como dispositivos generadores de lenguajes

En este capítulo se estudian sistemas P de transición con salida externa que generan lenguajes sobre un determinado alfabeto. Estos sistemas carecerán de membrana de entrada, de tal forma que cada computación de parada proporcionará una cadena del lenguaje a generar.

En la primera sección definimos la función *Output* de los sistemas P de transición con salida externa para obtener la variante de sistemas P de generación de lenguajes. En la sección 5.2 mostramos cómo estos sistemas son capaces de generar cualquier lenguaje recursivamente enumerable, basándonos en la simulación de máquinas de Turing deterministas por sistemas P de aceptación de lenguajes presentada en el capítulo 4. En la sección 5.3 realizamos una verificación de la corrección del funcionamiento de los sistemas construidos en la sección anterior.

5.1. Sistemas P de generación de lenguajes

Los sistemas que vamos a introducir a continuación son sistemas P de transición con salida externa. Además, para estos sistemas definimos la salida de una computación de parada como la cadena formada por todos los objetos del alfabeto de salida enviados al entorno externo del sistema, *en el orden en el que fueron expulsados.* Si se expulsaron varios objetos al mismo tiempo, entonces consideramos todas las permutaciones correspondientes como cadenas de salida. Es decir, la salida de una computación de parada será un conjunto de cadenas del alfabeto de salida.

Definición 5.1. *Un* sistema P de generación de lenguajes, Π, *es un sistema de computación celular con membranas que verifica las siguientes propiedades:*

- Π *es un sistema P de transición con salida externa.*
- Π *es un sistema P sin membrana de entrada.*
- *La salida de una computación* $\mathcal{C} = \{C^i\}_{i<r}$ *viene dada por la siguiente función:*

$$Output(\mathcal{C}) = \begin{cases} no\ definida, & si\ \mathcal{C}\ no\ es\ de\ parada \\ L(\mathcal{C}), & si\ \mathcal{C}\ es\ de\ parada \end{cases}$$

donde

$$L(\mathcal{C}) = \{w \in \Lambda^* : \exists w_1, \ldots, w_{r-1} \in \Lambda^* (w = w_1 \ldots w_{r-1} \wedge w_i \text{ representa el multiconjunto } (M^i_{env} - M^{i-1}_{env}) \restriction_\Lambda, \text{ para todo } i = 1, \ldots, r-1)\}$$

De esta forma, la salida de una computación de parada de Π *es un conjunto de cadenas sobre* Λ.

La definición anterior permite que en un paso de una computación se envíen al entorno externo del sistema varios objetos del alfabeto de salida, pero en ese caso dicha computación no proporciona una única cadena. Esta apreciación nos lleva a imponer una restricción adicional a los sistemas P de generación de lenguajes que consideraremos en el presente trabajo.

Definición 5.2. *Un sistema P de generación de lenguajes,* Π, *se dice que es* válido *si verifica las siguientes propiedades:*

- Π *es un sistema P de transición con salida externa válido; es decir, el sistema indica que una computación ha parado expulsando un objeto* # *al entorno externo.*
- *Si* $\mathcal{C} = \{C^i\}_{i<r}$ *es una computación de parada de* Π, *entonces* $|M^i_{env} - M^{i-1}_{env}| \leq 1$, *para todo* $i = 1, \ldots, r-1$. *Es decir, en cada paso de la computación se expulsa a lo sumo un objeto al entorno externo. Esto significa que la salida de* $\mathcal{C}$ *es una única cadena (eventualmente vacía).*

Notación. *Notaremos por $\mathcal{LG}$ la clase de los sistemas P de generación de lenguajes tales que dichos sistemas sean válidos.*

Ahora ya estamos en condiciones de definir el lenguaje generado por un sistema P del tipo antes considerado.

Definición 5.3. *Diremos que un sistema $\Pi \in \mathcal{LG}$ genera el lenguaje L sobre el alfabeto Λ si se verifican las siguientes condiciones:*

Para toda cadena $w \in \Lambda^*$ se tiene que:

- *Si $w \in L$, entonces existe una computación de parada, $\mathcal{C}$, de Π tal que $w = Output(\mathcal{C})$.*
- *Si existe una computación de parada, $\mathcal{C}$, de Π tal que $w = Output(\mathcal{C})$, entonces $w \in L$.*

Nota. *Obsérvese que si en la definición 5.2 eliminamos la segunda condición, entonces podemos considerar que los sistemas obtenidos generan lenguajes sustituyendo en la definición anterior la condición $w = Output(\mathcal{C})$ por $w \in Output(\mathcal{C})$. Las cadenas de los lenguajes generados por tales sistemas contendrían todas las permutaciones de los objetos expulsados al entorno externo en un mismo paso de una computación.*

5.2. Completitud computacional a través de lenguajes formales

En esta sección mostramos cómo los sistemas de computación celular con membranas introducidos en este capítulo son capaces de generar *cualquier* lenguaje recursivamente enumerable. Para ello nos basaremos en la simulación de máquinas de Turing deterministas por medio de sistemas P de aceptación de lenguajes que se ha descrito en el capítulo anterior.

Dado un lenguaje recursivamente enumerable, L, sobre un alfabeto Σ_L existe una máquina de Turing determinista, TM_L, que acepta dicho lenguaje. Entonces podemos asociar a este lenguaje un sistema P de generación de lenguajes, Π_L, no determinista, tal que toda computación suya realice las siguientes operaciones:

- Se genera, de manera no determinista, una cadena sobre el alfabeto Σ_L.
- Se simulan los distintos pasos que la máquina de Turing efectúa al recibir como entrada la cadena generada anteriormente. La simulación de cada paso se realiza a través de varias fases:
  - Se lee el símbolo en la celda analizada por la cabeza lectora.
  - Se calcula el nuevo símbolo a escribir en la celda, se mueve la cabeza lectora y se cambia de estado.
  - Se borra el viejo símbolo y se escribe el nuevo.
- Finalmente, tras la simulación de cada paso de transición de la máquina de Turing se comprueba si se ha alcanzado un estado final. Si se ha llegado al estado final de aceptación, entonces se termina con la fase siguiente. Si se ha llegado al estado final de rechazo, entonces se entra en un bucle infinito.
- En caso de que la máquina de Turing acepte la cadena generada en la primera fase, se devuelve dicha cadena como salida.

Obsérvese que las novedades de este sistema respecto al construido en la sección 4.2 del capítulo anterior radican en la introducción de la fase de generación de la cadena de entrada y la fase de devolución de dicha cadena, y en una ligera variación en la fase de comprobación del estado. La finalidad de la fase de generación reside en la necesidad de considerar, en un sistema sin entrada, todas las posibles cadenas sobre un alfabeto determinado. Las modificaciones en la fase de comprobación y la introducción de la fase de devolución de la cadena encuentran su explicación en la clase de sistemas que estamos considerando, ya que en estos cada computación de parada debe devolver una única cadena. Así, cuando la cadena generada no es aceptada por la máquina de Turing, obligamos a que la computación que simula el funcionamiento de la máquina sobre esa cadena no sea de parada. Por otra parte, cuando la cadena generada sí sea aceptada por la máquina de Turing, entonces la computación que simula el funcionamiento de la máquina sobre esa cadena es de parada y devuelve la propia cadena como salida.

Supongamos que el conjunto de estados de la máquina de Turing es $Q_{TM} = \{q_N, q_Y, q_0, \ldots, q_n\}$, el alfabeto de trabajo $\Gamma_{TM} = \{B, \triangleright, a_1, \ldots, a_m\}$, el alfabeto de entrada $\Sigma_{TM} = \{a_1, \ldots, a_p\}$, con $p \leq m$, y la función de transición $\delta_{TM}(q_i, a_j) = (q_{Q(i,j)}, a_{A(i,j)}, D(i,j))$. Recuérdese que denotamos $a_B = B$

(símbolo blanco) y $a_0 = \triangleright$ (símbolo situado en la celda de la cinta situada más a la izquierda).

Consideramos el sistema P de generación de lenguajes, Π_L, definido como sigue:

$$\Pi_L = \left(\Gamma, \Lambda, \#, \mu_{\Pi_L}, \mathcal{M}_1, (R_1, \rho_1), ext\right)$$

donde

$$\begin{aligned}
\Gamma = {} & \{q_N, q_Y, h, h', s, s_E, \#\} \cup \{q_i : 0 \leq i \leq n\} \cup \\
& \{a_i, a_i', a_i'', b_i : 0 \leq i \leq m\} \cup \{c_i, d_i : 1 \leq i \leq p\} \cup \\
& \{s_l, s_l^-, s_l^+ : l \in \{a_1, \ldots, a_p, I, 1, \ldots, 9, O\}\} \\
\Lambda = {} & \{a_1, \ldots, a_p\} \\
\mu_{\Pi_L} = {} & [_1]_1 \\
\mathcal{M}_1 = {} & q_0 h^2 a_0 s s_{a_1}^- \ldots s_{a_p}^- s_I^- s_1^- \ldots s_9^- s_O^- \\
(R_1, \rho_1) = {} & (R^I, \rho^I) \cup (R^1, \rho^1) \cup (R^2, \rho^2) \cup (R^3, \rho^3) \cup (R^4, \rho^4) \cup (R^O, \rho^O)
\end{aligned}$$

y los conjuntos de reglas son

$(R^I, \rho^I) = (R^{I_1}, \rho^{I_1}) \cup (R^{I_2}, \rho^{I_2})$, con

$$(R^{I_1},\rho^{I_1}) \equiv \begin{cases} s \to s_{a_i} \quad (1 \leq i \leq p) \\ s \to s_I \\ \begin{cases} s_E s_{a_i}^- \to (\#, out) \to s_{a_i} s_{a_i}^- \to s_{a_i}^+ > \\ \quad s_{a_i}^- \to s_{a_i}^- > \begin{cases} h \to h^2 a_i c_i & (1 \leq i \leq p) \\ s_{a_i}^+ \to s_{a_i}^- s \end{cases} \end{cases} \end{cases}$$

$$(R^{I_2},\rho^{I_2}) \equiv \begin{cases} s_E s_I^- \to (\#, out) > s_I s_I^- \to s_I^+ > s_I^- \to s_I^- > \\ \quad \begin{cases} h \to \lambda \\ s_I^+ \to s_I^- s_1 \end{cases} \end{cases}$$

$(R^1, \rho^1) = (R^{1_1}, \rho^{1_1}) \cup (R^{1_2}, \rho^{1_2}) \cup (R^{1_3}, \rho^{1_3})$, con

$$(R^{1_1},\rho^{1_1}) \equiv \begin{cases} s_E s_1^- \to (\#, out) > s_1 s_1^- \to s_1^+ > s_1^- \to s_1^- > \\ \quad \begin{cases} h \to hh' \\ a_i \to a_i a_i' \quad (0 \leq i \leq m) \\ s_1^+ \to s_1^- s_2 \end{cases} \end{cases}$$

$$(R^{1_2},\rho^{1_2}) \equiv \left\{ \begin{array}{l} s_E s_2^- \to (\#, out) > h' s_2 s_2^- \to s_2^+ > s_2 \to s_3 > \\ \quad s_2^- \to s_2^- > a_i'^2 \to a_i'' \quad (0 \leq i \leq m) > \left\{ \begin{array}{l} a_i' \to \lambda \quad (0 \leq i \leq m) \\ a_i'' \to a_i' \quad (0 \leq i \leq m) \\ s_2^+ \to s_2^- s_2 \end{array} \right. \end{array} \right.$$

$$(R^{1_3},\rho^{1_3}) \equiv \left\{ \begin{array}{l} s_E s_3^- \to (\#, out) > s_3 s_3^- \to s_3^+ > s_3^- \to s_3^- > \\ \quad \left\{ \begin{array}{l} a_i'^2 \to \lambda \quad (0 \leq i \leq m) \\ s_3^+ \to s_3^- s_4 \end{array} \right. \end{array} \right.$$

$(R^2, \rho^2) = (R^{2_1}, \rho^{2_1}) \cup (R^{2_2}, \rho^{2_2})$, con

$$(R^{2_1},\rho^{2_1}) \equiv \quad s_E s_4^- \to (\#, out) > s_4 s_4^- \to s_4^+ > s_4^- \to s_4^- > \left\{ \begin{array}{l} h \to hh' \\ s_4^+ \to s_4^- s_5 \end{array} \right.$$

$$(R^{2_2},\rho^{2_2}) \equiv \left\{ \begin{array}{l} s_E s_5^- \to (\#, out) > s_5 s_5^- \to s_5^+ > s_5^- \to s_5^- > \\ \quad \left\{ \begin{array}{l} \text{Reglas para la función de transición} \\ s_5^+ \to s_5^- s_6 \end{array} \right. \end{array} \right.$$

Las reglas para la función de transición, δ_{TM}, son las siguientes:

Caso 1: Estado q_r, símbolo $a_s \neq B$

Movimiento	*Reglas*	
izquierda	$q_r a_s' h \to q_{Q(r,s)} a_s' b_{A(r,s)}$,	si $A(r,s) \neq B$
	$q_r a_s' h \to q_{Q(r,s)} a_s'$,	si $A(r,s) = B$
igual	$q_r a_s' \to q_{Q(r,s)} a_s' b_{A(r,s)}$,	si $A(r,s) \neq B$
	$q_r a_s' \to q_{Q(r,s)} a_s'$,	si $A(r,s) = B$
derecha	$q_r a_s' \to q_{Q(r,s)} a_s' b_{A(r,s)} h$,	si $A(r,s) \neq B$
	$q_r a_s' \to q_{Q(r,s)} a_s' h$,	si $A(r,s) = B$

Caso 2: Estado q_r, símbolo $a_s = B$

Movimiento	*Reglas*
izquierda	$q_r h \to q_{Q(r,B)} b_{A(r,B)}$, si $A(r,B) \neq B$ $q_r h \to q_{Q(r,B)}$, si $A(r,B) = B$
igual	$q_r \to q_{Q(r,B)} b_{A(r,B)}$, si $A(r,B) \neq B$ $q_r \to q_{Q(r,B)}$, si $A(r,B) = B$
derecha	$q_r \to q_{Q(r,B)} b_{A(r,B)} h$, si $A(r,B) \neq B$ $q_r \to q_{Q(r,B)} h$, si $A(r,B) = B$

Para evitar conflictos, toda regla del caso 1 tiene una prioridad mayor que cada regla del caso 2.

$(R^3, \rho^3) = (R^{3_1}, \rho^{3_1}) \cup (R^{3_2}, \rho^{3_2})$, con

$$(R^{3_1},\rho^{3_1}) \equiv \begin{cases} s_E s_6^- \to (\#, out) > h' s_6 s_6^- \to s_6^+ > s_6 \to s_7 > \\ \quad s_6^- \to s_6^- > \begin{cases} a_i' \to a_i'^2 & (0 \leq i \leq m) \\ b_i \to b_i^2 & (0 \leq i \leq m) \\ s_6^+ \to s_6^- s_6 \end{cases} \end{cases}$$

$$(R^{3_2},\rho^{3_2}) \equiv \begin{cases} s_E s_7^- \to (\#, out) > s_7 s_7^- \to s_7^+ > s_7^- \to s_7^- > \\ \quad \begin{cases} a_i a_i' \to \lambda & (0 \leq i \leq m) \\ b_i \to a_i & (0 \leq i \leq m) \\ s_7^+ \to s_7^- s_8 \end{cases} \end{cases}$$

$(R^4, \rho^4) = (R^{4_1}, \rho^{4_1}) \cup (R^{4_2}, \rho^{4_2})$, con

$$(R^{4_1},\rho^{4_1}) \equiv \begin{cases} s_E s_8^- \to (\#, out) > s_8 s_8^- \to s_8^+ > s_8^- \to s_8^- > \\ \quad \begin{cases} q_Y \to q_Y s_9 \\ q_N \to q_N \\ q_i \to q_i s_1 & (0 \leq i \leq n) \\ s_8^+ \to s_8^- \end{cases} \end{cases}$$

$$(R^{4_2},\rho^{4_2}) \equiv \left\{ \begin{array}{l} s_E s_9^- \to (\#, out) > s_9 s_9^- \to s_9^+ > s_9^- \to s_9^- > \\ \quad \left\{ \begin{array}{l} q_Y \to \lambda \\ h \to \lambda \\ a_i \to \lambda \quad (0 \leq i \leq m) \\ s_9^+ \to s_9^- s_O \end{array} \right. \end{array} \right.$$

$(R^O, \rho^O) = (R^{O_1}, \rho^{O_1})$, con

$$(R^{O_1},\rho^{O_1}) \equiv \left\{ \begin{array}{l} s_E s_O^- \to (\#, out) > s_O s_O^- \to s_O^+ > s_O^- \to s_O^- > \\ \quad c_i^2 \to d_i \quad (1 \leq i \leq p) > \left\{ \begin{array}{l} c_i \to (a_i, out) \quad (1 \leq i \leq p) \\ d_i \to c_i \quad (1 \leq i \leq p) \end{array} \right\} > \\ \quad s_O^+ \to s_O^- s_E{}^{p+11} \end{array} \right.$$

Veamos con detalle únicamente las novedades que nos encontramos en el funcionamiento de este sistema, con relación al sistema descrito en el capítulo anterior.

Recordamos que usaremos objetos de tipo s^- como *objetos prohibidores* y objetos de tipo s^+ como *objetos permitidores*; si s_j^- está presente en el sistema P, entonces las reglas asociadas a la etapa j *no se pueden* ejecutar; si, en cambio, s_j^+ es el objeto presente, entonces las reglas asociadas a la etapa j *deben* ser ejecutadas (si es posible). Por supuesto, en cualquier momento, para cada etapa solo el correspondiente objeto prohibidor o permitidor estará presente. También habrá siempre solamente un objeto permitidor al mismo tiempo.

Debido a la presencia del objeto s en la configuración inicial del sistema, toda computación de $\Pi_{\mathcal{L}}$ comienza con la ejecución de una regla del tipo $s \to s_{a_j}$, para algún a_j (o la regla $s \to s_I$) y, simultáneamente, con la ejecución de reglas del tipo $s_i^- \to s_i^-$ que no alteran el contenido de la membrana piel.

Cada vez que el objeto s aparece en Π_L, se activa un mecanismo de elección, produciendo la selección de algunos objetos que codifican símbolos del dato de entrada para la máquina de Turing, junto con la celda que ocupan en la cinta de TM_L.

La presencia del objeto s_I en el sistema se producirá a través de la aplicación de la regla $s \to s_I$. Este objeto causará la eliminación de cualquier objeto h presente en la membrana (que a partir de ahora codificará, en base

1, la posición de la cabeza lectora de la máquina de Turing), indicando así que la cabeza está en la celda número cero, y permitirá la inserción del objeto s_1 en el sistema.

En ese momento, terminará la primera fase, que es la única no determinista, y está encargada de la generación de la cadena de entrada de la máquina de Turing, comenzando entonces la simulación de la computación de TM_L sobre ese dato de entrada.

Fase I: *Generación del dato de entrada.*

Esta fase se lleva a cabo mediante las reglas del conjunto R^I.

Obsérvese que en la configuración inicial aparecen dos copias de h; es decir, la primera celda a considerar para introducir el primer símbolo de la cadena de entrada es la celda número uno. Esto es así porque en esta fase usamos el objeto h para codificar en base 2 la celda. Además, la celda número cero está reservada para el símbolo $\triangleright$, que se codifica mediante el objeto a_0, el cual aparece en el sistema desde el comienzo de la simulación.

A causa de las reglas de prohibición del tipo $s^-_{a_j} \to s^-_{a_j}$, con $j = 1, \ldots, p$, ningún símbolo se puede escribir y la fase I no puede concluirse. No obstante, de manera no determinista el objeto s se transforma a sí mismo en un objeto promotor de tipo s_{a_j}, para algún $j = 1, \ldots, p$, o en el objeto promotor s_I. Por tanto, el correspondiente objeto prohibidor cambiará en el permitidor, haciendo así posible «escribir el símbolo a_j» o finalizar la fase de generación de la cadena.

En el primer caso copiamos tantos objetos a_j (para trabajar con ellos en las fases siguientes) y tantos objetos c_j (para mantenerlos hasta la fase final O) como objetos h estén presentes. También duplicamos el número de objetos h para considerar la siguiente celda en la cinta. Finalmente, el objeto permitidor $s^+_{a_j}$ se transforma en el prohibidor, $s^-_{a_j}$, y aparece un nuevo objeto s, para elegir la siguiente acción a realizar.

En el segundo caso, todos los objetos h se borran, comenzando la simulación de un paso de la máquina de Turing con la cabeza lectora empezando a partir de la celda número cero. Además, s^+_I se transforma a sí mismo en s^-_I y s_1, para inducir la ejecución de la fase 1.

El objetivo de la fase 4 consiste en comprobar si se ha alcanzado un estado final, realizando una de las siguientes acciones:

(a) Si no se ha alcanzado un estado final, entonces comienza de nuevo a partir de la fase 1, para simular otro paso de la computación de la máquina de Turing. Esto se hace introduciendo el objeto promotor s_1 dentro de la membrana.

(b) Si se ha alcanzado el estado q_N, entonces entra en un bucle infinito.

(c) Si se ha alcanzado el estado q_Y, entonces inserta el objeto promotor s_9 para continuar con la fase O final.

Esta fase se inicia con la presencia del objeto promotor s_8 y termina con la ejecución de la regla correspondiente al caso, de los tres anteriores, que ha tenido lugar y de la regla $s_8^+ \to s_8^-$.

Fase 4: *Comprobación del estado.*

Esta fase es la última de la simulación de la máquina de Turing y se lleva a cabo mediante las reglas del conjunto R^4. Detecta si se ha alcanzado o no un estado final. En el primer caso pasaremos a la fase O para devolver la cadena de entrada como salida externa si fue el estado de aceptación, o entramos en un bucle infinito si fue el estado de rechazo. En el segundo caso, continuamos la simulación a partir de la fase 1.

Las reglas de R^{41} se usan para distinguir las distintas posibilidades que pueden acontecer: si q_Y está presente en la membrana, lo que significa que hemos alcanzado el estado final de aceptación, entonces continuamos con la siguiente fase, en lugar de con la fase 1; si q_N está presente en la membrana, lo que significa que hemos alcanzado el estado final de rechazo, entonces no introducimos ningún objeto promotor. El efecto producido es que la simulación de la máquina de Turing finaliza propiamente, pero *la computación del sistema P no para.* Esto es así porque todas las reglas de prohibición (reglas del tipo $s_j^- \to s_j^-$), y ninguna otra, se pueden aplicar y, además, son aplicadas. Así, no hay salida del sistema en este caso; si cualquiera de los objetos $q_0, \ldots, q_n$ está presente en la membrana, entonces la simulación de la máquina de Turing no ha concluido. Por tanto, introducimos el objeto promotor s_1 para comenzar de nuevo a partir de la fase 1. En todos los casos el objeto permitidor s_8^+ se transforma en el objeto prohibidor s_8^-.

Las reglas de R^{4_2} solo se aplican cuando se ha alcanzado el estado final de aceptación. Dichas reglas permiten eliminar todos los objetos del sistema P excepto los objetos c_i, que codifican una copia de la cadena de entrada, el objeto promotor, s_O, de la última fase y los objetos prohibidores.

La presencia del objeto s_O en el sistema P significa que se ha alcanzado el estado de aceptación en la computación de TM_L que se está simulando. Entonces comienza la fase final en la cual el dato de entrada para la máquina de Turing será devuelto como cadena de salida de la computación de Π_L considerada.

Fase O: *Devolución del dato de entrada*

Esta fase se lleva a cabo mediante las reglas del conjunto R^O, las cuales se usan para descodificar la cadena de entrada y expulsarla, en el orden adecuado, al entorno externo.

Como la cadena está codificada en base 2, solamente tenemos que dividir por 2, repetidamente, el número de objetos c_j presentes. Cada vez, excepto la primera (porque no expulsamos el símbolo $\triangleright$, que siempre ocupa la celda número cero), uno y solo uno de estos números es impar, lo que indica el siguiente símbolo de la cadena de entrada a expulsar.

Primero aplicamos las reglas que sustituyen todos los posibles pares de objetos c_j con un solo d_j, dividiendo así por dos el número de objetos presentes. Entonces solo un objeto c_k permanece (excepto la primera vez), para algún k, el cual se expulsa fuera de la membrana. Los objetos d_j se transforman en los correspondientes c_j para repetir la misma operación una y otra vez hasta que no quede ningún objeto c_j en la membrana piel. Cuando esto ocurra, se introducen en el sistema $p+11$ objetos s_E, cada uno de los cuales se usará para eliminar un objeto prohibidor y expulsar un objeto #. Así, ya no quedarán objetos en la membrana y, como consecuencia, la computación será de parada.

5.3. Verificación formal

En la sección anterior, para cada lenguaje recursivamente enumerable y dada una máquina de Turing determinista que lo acepta, hemos diseñado un

sistema P de generación de lenguajes que lo genera. Además, hemos descrito informalmente el funcionamiento de dicho sistema.

En lo que sigue vamos a demostrar que, dado un lenguaje recursivamente enumerable, L, el sistema Π_L (que depende, además, de la máquina de Turing considerada para aceptar L) es un sistema de generación de lenguajes que es válido y, además, genera exactamente el lenguaje L. Para ello, vamos a verificar por separado que cada fase del funcionamiento del sistema es correcta, de donde se deducirá la corrección global de Π_L.

Notación. *En esta sección consideraremos que $\mathcal{C} = \{C^i\}_{i<r}$ es una computación del sistema $\Pi_{\mathcal{L}}$.*

Notamos por $S^- = s_{a_1}^- \dots s_{a_p}^- s_I^- s_1^- \dots s_9^- s_O^-$ el multiconjunto de objetos prohibidores y, si de dicho multiconjunto eliminamos el objeto s_j, entonces lo notamos por $S^-(j)$. Es decir, $S^-(j) = S^- \setminus \{s_j\}$.

Notamos por $R^- = \{s_l^- \to s_l^- : l = a_1, \dots, a_p, I, 1, \dots, 9, O\}$ el conjunto de reglas prohibidoras y, si de dicho conjunto eliminamos la regla $s_j^- \to s_j^-$, entonces lo notamos por $R^-(j)$. Es decir, $R^-(j) = R^- \setminus \{s_j^- \to s_j^-\}$.

5.3.1. Generación del dato de entrada

A continuación verificamos la fase I, en la cual se genera un multiconjunto que codifica una cadena de entrada para la máquina de Turing. Veremos que en esta fase el objeto s_I juega un papel fundamental, y demostraremos que toda computación en la que dicho objeto no aparezca en la membrana en algún momento, entra en un bucle infinito sin salir de esta fase. También probaremos que si este objeto aparece, entonces se ha generado correctamente una cadena de entrada y, además, que toda posible cadena de entrada se genera en alguna computación del sistema.

Empezamos presentando un lema que destaca la relevancia del objeto s_I en esta fase de generación.

Lema 5.4. *Sea $k \in \mathbb{N}$ y supongamos que $s_I \notin M_1^i$, para todo $i < 3 \cdot k - 1$. Entonces se tiene que $r > 3 \cdot k$ y existen $a_{i_1}, \dots, a_{i_k}, c_{i_1}, \dots, c_{i_k} \in \Gamma$ tales que $1 \leq i_1, \dots, i_k \leq p$ y*

$$M_1^{3 \cdot k} = q_0 a_0 s S^- h^{2^{k+1}} a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

Demostración. Por inducción sobre $k \in \mathbb{N}$.

Caso 1: $k = 0$.

Es evidente que $r > 0$ y, además,

$$M_1^0 = \mathcal{M}_1 = q_0 a_0 s S^- h^2$$

Caso 2: $k \to k+1$.

Supongamos que $s_I \notin M_1^i$, para todo $i < 3{\cdot}k{+}2$. Entonces, por hipótesis de inducción se tiene que $r > 3 \cdot k$ y existen $a_{i_1}, \ldots, a_{i_k}, c_{i_1}, \ldots, c_{i_k} \in \Gamma$ tales que $1 \leq i_1, \ldots, i_k \leq p$ y

$$M_1^{3\cdot k} = q_0 a_0 s S^- h^{2^{k+1}} a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$$

Puesto que $s_I \notin M_1^{3\cdot k+1}$, debe existir $a_{i_{k+1}} \in \Gamma$ tal que $1 \leq i_{k+1} \leq p$ y, además,

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$3 \cdot k$	$q_0 a_0 s S^- h^{2^{k+1}}$ $a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$	$s \to s_{a_{i_{k+1}}}; R^-$
$3 \cdot k + 1$	$q_0 a_0 s_{a_{i_{k+1}}} S^- h^{2^{k+1}}$ $a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$	$s_{a_{i_{k+1}}} s^-_{a_{i_{k+1}}} \to s^+_{a_{i_{k+1}}}$; $R^-(a_{i_{k+1}})$
$3 \cdot k + 2$	$q_0 a_0 s^+_{a_{i_{k+1}}} S^-(a_{i_{k+1}}) h^{2^{k+1}}$ $a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$	$h \to h^2 a_{i_{k+1}} c_{i_{k+1}}$; $s^+_{a_{i_{k+1}}} \to s^-_{a_{i_{k+1}}} s; R^-(a_{i_{k+1}})$
$3 \cdot k + 2$	$q_0 a_0 s S^- h^{2^{k+2}}$ $a_{i_1}^{2^1} \ldots a_{i_{k+1}}^{2^{k+1}} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k} c_{i_{k+1}}^{2^{k+1}}$	

Lo que prueba el resultado. □

A continuación justificamos cómo la presencia o no del objeto s_I determina el final de esta fase en el siguiente sentido: si dicho objeto no se encuentra en la membrana del sistema en ninguna configuración, entonces la fase no termina nunca; si, por el contrario, aparece en alguna configuración, entonces la fase finaliza y se ha generado un multiconjunto que codifica correctamente una cadena de entrada para la máquina de Turing.

Proposición 5.5. *Supongamos que $s_I \notin M_1^i$, para todo $i < r$. Entonces $\mathcal{C}$ es una computación que no es de parada.*

Demostración. Del lema 5.4, se deduce fácilmente que $r > 3 \cdot k$, para todo $k \in \mathbb{N}$. Por tanto, $r = \infty$. □

Proposición 5.6. *Supongamos que $s_I \in M_1^t$ y que $s_I \notin M_1^i$, para todo $i < t$. Entonces existe $k \in \mathbb{N}$ tal que $t = 3 \cdot k + 1$ y existen $a_{i_1}, \ldots, a_{i_k}, c_{i_1}, \ldots, c_{i_k} \in \Gamma$ tales que $1 \leq i_1, \ldots, i_k \leq p$ y*

$$M_1^{3 \cdot k+3} = q_0 a_0 s_1 S^- a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$$

Demostración. Sea $k = \max\{k' \in \mathbb{N} : 3 \cdot k' \leq t\}$. Por hipótesis, $s_I \notin M_1^i$, para todo $i < 3 \cdot k - 1$. Por tanto, aplicando el lema 5.4, existen $a_{i_1}, \ldots, a_{i_k}, c_{i_1}, \ldots, c_{i_k} \in \Gamma$ tales que $1 \leq i_1, \ldots, i_k \leq p$ y

$$M_1^{3 \cdot k} = q_0 a_0 s S^- h^{2^{k+1}} a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$$

Si $s \to s_{a_j} \in amr_{3 \cdot k, 3 \cdot k+1}$, para algún $a_j \in \Gamma$, con $1 \leq j \leq p$, entonces es fácil comprobar que $3 \cdot (k+1) \leq t$, lo que contradice la definición de k. Por tanto,

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$3 \cdot k$	$q_0 a_0 s S^- h^{2^{k+1}}$ $a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$	$s \to s_I; R^-$
$3 \cdot k + 1$	$q_0 a_0 s_I S^- h^{2^{k+1}}$ $a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$	$s_I s_I^- \to s_I^+; R^-(I)$
$3 \cdot k + 2$	$q_0 a_0 s_I^+ S^-(I) h^{2^{k+1}}$ $a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$	$h \to \lambda;$ $s_I^+ \to s_I^- s_1; R^-(I)$
$3 \cdot k + 3$	$q_0 a_0 s_1 S^- a_{i_1}^{2^1} \ldots a_{i_k}^{2^k} c_{i_1}^{2^1} \ldots c_{i_k}^{2^k}$	

□

Por último vamos a probar que para toda cadena del alfabeto de entrada de TM_L existe una computación de Π_L que genera un multiconjunto que codifica adecuadamente dicha cadena.

Lema 5.7. *Supongamos que*

$$M_1^{3\cdot k} = q_0 a_0 s S^- h^{2^{k+1}} a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

con $k \geq 0$ *y* $1 \leq i_1, \dots, i_k \leq p$. *Entonces, para todo* $a_{i_{k+1}} \in \Gamma$ *tal que* $1 \leq i_{k+1} \leq p$ *existe una computación* $\mathcal{C}' = \{C'^i\}_{i<r'}$ *que verifica*

$$M_1'^{3\cdot k+3} = q_0 a_0 s S^- h^{2^{k+2}} a_{i_1}^{2^1} \dots a_{i_k}^{2^k} a_{i_{k+1}}^{2^{k+1}} c_{i_1}^{2^1} \dots c_{i_k}^{2^k} c_{i_{k+1}}^{2^{k+1}}$$

Demostración. Basta hacer $C'^i = C^i$, para todo $i \leq 3 \cdot k$, y ahora considerar los siguientes multiconjuntos de reglas:

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$3 \cdot k$	$q_0 a_0 s S^- h^{2^{k+1}}$ $a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$	$s \to s_{a_{i_{k+1}}}; R^-$
$3 \cdot k + 1$	$q_0 a_0 s_{a_{i_{k+1}}} S^- h^{2^{k+1}}$ $a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$	$s_{a_{i_{k+1}}} s^-_{a_{i_{k+1}}} \to s^+_{a_{i_{k+1}}}; R^-(a_{i_{k+1}})$
$3 \cdot k + 2$	$q_0 a_0 s^+_{a_{i_{k+1}}} S^-(a_{i_{k+1}}) h^{2^{k+1}}$ $a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$	$h \to h^2 a_{i_{k+1}} c_{i_{k+1}}$; $s^+_{a_{i_{k+1}}} \to s^-_{a_{i_{k+1}}} s; R^-(a_{i_{k+1}})$
$3 \cdot k + 3$	$q_0 a_0 s S^- h^{2^{k+2}}$ $a_{i_1}^{2^1} \dots a_{i_k}^{2^k} a_{i_{k+1}}^{2^{k+1}} c_{i_1}^{2^1} \dots c_{i_k}^{2^k} c_{i_{k+1}}^{2^{k+1}}$	

□

Proposición 5.8. *Para toda cadena* $w = a_{i_1} \dots a_{i_k} \in \Sigma^*_{TM_{\mathcal{L}}}$ *existe una computación de* $\Pi_{\mathcal{L}}$ *tal que*

$$M_1^{3\cdot k+3} = q_0 a_0 s_1 S^- a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

Demostración. Aplicando k veces el lema 5.7 obtenemos una computación $\mathcal{C} = \{C^i\}_{i<r}$ de $\Pi_{\mathcal{L}}$ tal que

$$M_1^{3\cdot k} = q_0 a_0 s S^- h^{2^{k+1}} a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

Entonces podemos considerar la computación $\mathcal{C}' = \{C'^i\}_{i<r'}$ tal que $C'^i = C^i$, para todo $i \leq 3 \cdot k$, y, además, se producen los siguientes pasos:

i	M_1^i	Reglas en $amr_{i,i+1}$ para la membrana 1
$3 \cdot k$	$q_0 a_0 s S^- h^{2^{k+1}}$ $a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$	$s \to s_I; R^-$
$3 \cdot k + 1$	$q_0 a_0 s_I S^- h^{2^{k+1}}$ $a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$	$s_I s_I^- \to s_I^+; R^-(I)$
$3 \cdot k + 2$	$q_0 a_0 s_I^+ S^-(I) h^{2^{k+1}}$ $a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$	$h \to \lambda;$ $s_I^+ \to s_I^- s_1; R^-(I)$
$3 \cdot k + 3$	$q_0 a_0 s_1 S^-$ $a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$	

Con lo que se tiene el resultado requerido. □

5.3.2. Simulación de una computación de TM_L

Las demostraciones de las siguientes proposiciones, que establecen la corrección de la simulación por el sistema Π_L de la máquina TM_L, son análogas a las correspondientes proposiciones del capítulo anterior.

Proposición 5.9. *Supongamos que*

$$M_1^y = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_1 S^-$$

con $t \geq 0$, $1 \leq i_1, \dots, i_t \leq p$, $0 \leq u \leq n$, $x \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $k \geq 0$, $0 \leq j_1, \dots, j_k \leq m$ *y* $0 \leq v_1 < \dots < v_w < x \leq x_1 < \dots < x_k$. *Entonces,*

$$M_1^{y+2+3\cdot x+3} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_4 S^- a'_{j_1}, \quad \text{si } x_1 = x$$
$$M_1^{y+2+3\cdot x+3} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_4 S^-, \quad \text{si } x_1 > x$$

Proposición 5.10. *Supongamos que*

$$M_1^y = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_4 S^- a'_j$$

con $t \geq 0$, $1 \leq i_1, \dots, i_t \leq p$, $0 \leq u \leq n$, $x \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$ *y* $j \in \{u_1, \dots, u_w\}$.

Entonces,

$$M_1^{y+3} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_{Q(u,j)} h^{x+D(u,j)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x a'_j b_{A(u,j)}, \quad si\ A(u,j) \neq B$$
$$M_1^{y+3} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_{Q(u,j)} h^{x+D(u,j)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x a'_j, \quad si\ A(u,j) = B$$

Proposición 5.11. *Supongamos que*

$$M_1^y = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_4 S^-$$

con $t \geq 0$, $1 \leq i_1, \dots, i_t \leq p$, $0 \leq u \leq n$, $x \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_w \leq m$ *y* $0 \leq v_1 < \dots < v_w$. *Entonces,*

$$M_1^{y+3} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_{Q(u,B)} h^{x+D(u,B)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x b_{A(u,B)}, \quad si\ A(u,B) \neq B$$
$$M_1^{y+3} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_{Q(u,B)} h^{x+D(u,B)} a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x, \quad si\ A(u,B) = B$$

Proposición 5.12. *Supongamos que*

$$M_1^y = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^v a_{u_1}^{2^{v_1}} \dots a_j^{2^x} \dots a_{u_w}^{2^{v_w}} s_6 S^- h'^x a'_j b_k$$

con $t \geq 0$, $1 \leq i_1, \dots, i_t \leq p$, $0 \leq u \leq n$, $v \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < x < \dots < v_w$, $j, k \in \{0, \dots, m, B\}$. *Entonces,*

$$M_1^{y+2 \cdot x+3} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^v a_{u_1}^{2^{v_1}} \dots a_k^{2^x} \dots a_{u_w}^{2^{v_w}} s_8 S^-$$

Proposición 5.13. *Supongamos que*

$$M_1^y = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$$

con $t \geq 0$, $1 \leq i_1, \dots, i_t \leq p$, $0 \leq u \leq n$, $x \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$. *Entonces,*

$$M_1^{y+2} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_1 S^-$$

Proposición 5.14. *Supongamos que*

$$M_1^y = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_Y h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$$

con $t \geq 1$, $1 \leq i_1, \dots, i_t \leq p$, $x \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$.

Entonces,

$$M_1^{y+4} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} s_O S^-$$

Proposición 5.15. *Supongamos que*

$$M_1^y = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_N h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} s_8 S^-$$

con $t \geq 1$, $1 \leq i_1, \dots, i_t \leq p$, $x \geq 0$, $0 \leq u_1, \dots, u_w \leq m$, $0 \leq v_1 < \dots < v_w$. *Entonces*

$$M_1^{y+2} = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_N h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} S^-$$

Además, para cada $i \geq y+2$ *la configuración* C^i *no es de parada.*

Proposición 5.16. *Supongamos que*

$$M_1^y = c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_u h^x a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} s_1 S^-$$

con $t \geq 1$, $1 \leq i_1, \dots, i_t \leq p$, $0 \leq u\nu$, $x \geq 0$, $w \geq 0$, $0 \leq u_1, \dots, u_2 \leq m$, $k \geq 0$, $0 \leq j_1, \dots, j_k \leq m$ *y* $0 \leq v_1 < \dots v_w < x \leq x_1 < \dots < x_k$. *Entonces,*

$$\begin{aligned}
M_1^{y+5\cdot x+11} &= c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_{Q(u,j_1)} h^{x+D(u,j_1)} s_8 S^-, \quad \textit{si } x_1 = x \textit{ y } A(u,j_1) \neq B \\
&\quad a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{A(u,j_1)}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} \\
M_1^{y+5\cdot x+11} &= c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_{Q(u,j_1)} h^{x+D(u,j_1)} s_8 S^-, \quad \textit{si } x_1 = x \textit{ y } A(u,j_1) = B \\
&\quad a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_2}^{2^{x_2}} \dots a_{j_k}^{2^{x_k}} \\
M_1^{y+5\cdot x+11} &= c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_{Q(u,B)} h^{x+D(u,B)} s_8 S^-, \quad \textit{si } x_1 > x \textit{ y } A(u,j_1) \neq B \\
&\quad a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{A(u,B)}^{2^x} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}} \\
M_1^{y+5\cdot x+11} &= c_{i_1}^{2^1} \dots c_{i_t}^{2^t} q_{Q(u,B)} h^{x+D(u,B)} s_8 S^-, \quad \textit{si } x_1 > x \textit{ y } A(u,j_1) \neq B \\
&\quad a_{u_1}^{2^{v_1}} \dots a_{u_w}^{2^{v_w}} a_{j_1}^{2^{x_1}} \dots a_{j_k}^{2^{x_k}}
\end{aligned}$$

Proposición 5.17. *Supongamos que*

$$M_1^{3\cdot k} = q_0 a_0 s_1 S^- a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

Supongamos que TM_L *no para sobre la cadena* $w = a_{i_1}^{2^1} \dots a_{i_k}^{2^k}$, *o bien que* I_w^f *no es de aceptación. Entonces la computación* $\mathcal{C}$ *no es de parada.*

5.3.3. Devolución del dato de entrada

La fase final O es la encargada de, una vez comprobado que la máquina de Turing acepta la cadena de entrada generada, expulsar los símbolos de esta cadena uno por uno y en el orden adecuado. Mediante el lema y la proposición siguientes evidenciamos la corrección de esta fase final.

Lema 5.18. *Supongamos que*

$$M_1^y = s_O^+ S^-(O) c_{i_u}^{2^0} \dots c_{i_t}^{2^{t-u}}$$
$$M_{env}^y = a_{i_1} \dots a_{i_{u-1}}$$

con $u \leq t$ *y* $1 \leq i_u, \dots, i_t \leq p$. *Entonces,*

$$M_1^{y+2} = s_O^+ S^-(O) c_{i_{u+1}}^{2^0} \dots c_{i_t}^{2^{t-u-1}}$$
$$M_{env}^{y+2} = a_{i_1} \dots a_{i_u}$$

Demostración. En la computación se realizan los siguientes pasos:

i	M_1^i	M_{env}^i	Reglas en $amr_{i,i+1}$ para la membrana 1
y	$s_O^+ S^-(O) c_{i_u}^{2^0} \dots c_{i_t}^{2^{t-u}}$	$a_{i_1} \dots a_{i_{u-1}}$	$c_l^2 \to d_l$ $(l=i_{u+1},\dots,i_t)$; $R^-(O)$
$y+1$	$s_O^+ S^-(O) c_{i_u}^{2^0} d_{i_{u+1}}^{2^0} \dots d_{i_t}^{2^{t-u-1}}$	$a_{i_1} \dots a_{i_{u-1}}$	$c_{i_u} \to (a_{i_u}, out)$; $d_l \to c_l$ $(l=i_{u+1},\dots,i_t)$; $R^-(O)$
$y+2$	$s_O^+ S^-(O) c_{i_{u+1}}^{2^0} \dots c_{i_t}^{2^{t-u-1}}$	$a_{i_1} \dots a_{i_u}$	

□

Proposición 5.19. *Supongamos que*

$$M_1^{3\cdot k} = q_0 a_0 s_1 S^- a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

Supongamos también que TM_L *para sobre la cadena* $w = a_{i_1} \dots a_{i_k}$ *y que* I_w^f *es de aceptación (es decir,* TM_L *acepta* w*). Entonces* $r \in \mathbb{N}$ *y, además,*

$$M_1^{r-1} = \lambda \text{ y } M_{env}^{r-1} = a_{i_1} \dots a_{i_k} \#^{p+11}$$

Demostración. Por la proposición 5.14, existe $y < r$ tal que

$$M_1^y = s_O S^- c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

Entonces,

$$M_1^{y+1} = s_O^+ S^-(O) c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

Aplicando k veces el lema 5.18, resulta que

$$M_1^{y+1+2\cdot k} = s_O^+ S^-(O)$$
$$M_{env}^{y+1+2\cdot k} = a_{i_1} \dots a_{i_k}$$

Por tanto,

$$M_1^{y+1+2\cdot k+1} = S^- s_E^{p+11}$$
$$M_{env}^{y+1+2\cdot k+1} = a_{i_1} \dots a_{i_k}$$

y, en consecuencia,

$$M_1^{y+1+2\cdot k+2} = \lambda$$
$$M_{env}^{y+1+2\cdot k+2} = a_{i_1} \dots a_{i_k} \#^{p+11}$$

□

5.3.4. Generación de lenguajes

Finalmente, vamos a probar que Π_L es un sistema válido de generación de lenguajes y que, además, genera el lenguaje L.

Teorema 5.20. *El sistema Π_L es un sistema P de generación de lenguajes que es válido.*

Demostración. Según las proposiciones 5.5, 5.6 y 5.17 las únicas computaciones de parada de Π_L son aquellas tales que

$$M_1^{3\cdot k} = q_0 a_0 s_1 S^- a_{i_1}^{2^1} \dots a_{i_k}^{2^k} c_{i_1}^{2^1} \dots c_{i_k}^{2^k}$$

donde TM_L para sobre la cadena $w = a_{i_1} \dots a_{i_k}$ e I_w^f es de aceptación (es decir, TM_L acepta w).

Entonces, analizando con detalle la demostración de la proposición 5.19, se deduce que Π_L verifica las dos condiciones necesarias para que sea un sistema válido. □

Teorema 5.21. *Sea L un lenguaje recursivamente enumerable. Entonces el sistema Π_L genera el lenguaje L.*

Demostración. Basta tener en cuenta el teorema 5.20 y la proposición 5.8. □

Finalmente, establecemos la completitud computacional de los sistemas P de generación de lenguajes, que usan prioridad y cooperación.

Teorema 5.22. *Sea L un lenguaje recursivamente enumerable. Entonces existe $\Pi \in \mathcal{LG}(Pri, Coo, nDis)$ tal que Π genera L.*

Demostración. Basta tener en cuenta los teoremas 5.20 y 5.21. □

Capítulo 6

Sistemas P como dispositivos de cálculo

En este capítulo se estudian sistemas P de transición con salida externa que calculan funciones entre números naturales. Estos sistemas dispondrán de una membrana de entrada, de tal forma que al introducir en ella una tupla de números naturales obtendremos como respuesta otra tupla que coincide con el valor sobre la primera de la función que queremos calcular.

En la sección 6.1 definimos la función *Output* de los sistemas P de transición con salida externa para obtener la variante de sistemas P de cálculo de funciones. En la sección 6.2 definimos, siguiendo [51], las operaciones de composición, iteración y minimización sobre este tipo de sistemas, de donde deduciremos que son capaces de calcular cualquier función recursiva.

6.1. Sistemas P de cálculo de funciones

Fijado un orden entre los símbolos de un alfabeto, podemos representar tuplas de números naturales mediante multiconjuntos sobre ese alfabeto con solo atender a las multiplicidades de los símbolos en el multiconjunto. Así, en los sistemas de computación celular con membranas que vamos a introducir a continuación impondremos que tanto el alfabeto de entrada como el de salida estén ordenados. De esta forma, tiene sentido considerar que los multiconjuntos recibidos como entrada y obtenidos como salida representan tuplas de números naturales.

Definición 6.1. *Un* sistema P de cálculo de funciones, Π, *de orden* (m, n) *es un sistema de computación celular con membranas que verifica las siguientes propiedades:*

- Π *es un sistema P de transición con salida externa.*
- Π *es un sistema P con membrana de entrada.*
- *El alfabeto de entrada,* Σ, *de* Π *es un alfabeto ordenado de* m *elementos. Supondremos que* $\Sigma = \{a_1, \ldots, a_m\}$.
- *El alfabeto de salida,* Λ, *de* Π *es un alfabeto ordenado de* n *elementos. Supondremos que* $\Lambda = \{b_1, \ldots, b_n\}$.
- *La salida de una computación* $\mathcal{C} = \{C^i\}_{i<r}$ *viene dada por la siguiente función:*

$$Output(\mathcal{C}) = \begin{cases} no\ definida, & si\ \mathcal{C}\ no\ es\ de\ parada \\ \left(M_{env}^{r-1}(b_1), \ldots, M_{env}^{r-1}(b_n)\right), & si\ \mathcal{C}\ es\ de\ parada \end{cases}$$

De esta forma, la salida de una computación de parada de Π *es una tupla de* n *números naturales.*

De acuerdo con la definición anterior, en un sistema P de cálculo de funciones toda computación de parada devuelve una tupla de números naturales. Sin embargo, para un mismo dato de entrada pueden existir computaciones que sean de parada y otras que no sean de parada. Es más, la salida de dos computaciones sobre un mismo dato de entrada no tiene por qué ser la misma tupla. Esto no ocurre para las funciones: dada una tupla de números naturales, o bien la función no está definida sobre dicha tupla, o bien sí lo está y devuelve un único valor. Por tanto, hemos de exigir que los sistemas con los que vamos a trabajar capturen estas propiedades.

Definición 6.2. *Un sistema P de cálculo de funciones,* Π, *de orden* (m, n) *se dice que es* válido *si verifica las siguientes propiedades:*

- Π *es un sistema P de transición con salida externa válido.*
- *Dada una configuración inicial,* C, *del sistema se tiene que, o bien ninguna computación de* Π *con primera configuración* C *es de parada, o bien toda computación de* Π *con primera configuración* C *es de parada.*

- *Si C es una configuración inicial del sistema y $\mathcal{C}_1$ y $\mathcal{C}_2$ son dos computaciones de parada de Π con primera configuración C, entonces se tiene que $Output(\mathcal{C}_1) = Output(\mathcal{C}_2)$.*

Notación. *Notaremos por $\mathcal{FC}^{m,n}$ la clase de los sistemas P de cálculo de funciones de orden (m,n) tales que dichos sistemas sean válidos. La clase $\mathcal{FC}$ es la unión de todas las clases anteriores.*

Los sistemas de computación celular con membranas pertenecientes a la clase $\mathcal{FC}$ permiten calcular funciones parciales entre números naturales, de acuerdo con el criterio que exponemos a continuación.

Definición 6.3. *Diremos que un sistema $\Pi \in \mathcal{FC}^{m,n}$ calcula la función parcial $f : \mathbb{N}^m - \to \mathbb{N}^n$ si se verifican las siguientes propiedades:*

Para cada $(k_1, \ldots, k_m) \in \mathbb{N}^m$,

- *f está definida sobre $(k_1, \ldots, k_m)$ si y solo si existe una computación de Π con entrada el multiconjunto $a_1^{k_1} \ldots a_m^{k_m}$ tal que dicha computación es de parada.*

- *Si $\mathcal{C}$ es una computación de parada de Π con entrada el multiconjunto $a_1^{k_1} \ldots a_m^{k_m}$, entonces $Output(\mathcal{C}) = f(k_1, \ldots, k_m)$.*

En virtud de la definición 6.2, en la definición anterior la expresión «una computación» puede sustituirse por la expresión «cualquier computación».

6.2. Completitud computacional a través de funciones recursivas

En esta sección se trata de poner de manifiesto que usando sistemas P de cálculo de funciones somos capaces de reproducir el comportamiento de cualquier función recursiva. En efecto, vamos a diseñar sistemas tales que:

1. Calculan las funciones básicas: función idénticamente nula, función sucesor y funciones proyecciones.

2. Calculan la composición de funciones, a partir de sistemas que calculen las funciones a componer.

3. Calculan la iteración de funciones, a partir de un sistema que calcule la función a iterar.

4. Calculan la minimización de funciones, a partir de un sistema que calcule la función a minimizar.

De esta forma, los resultados de la sección 1.4 del capítulo 1 nos garantizan que es posible construir sistemas de computación celular con membranas que calculan cualquier función recursiva.

6.2.1. Funciones básicas.

Comenzamos describiendo sistemas P que permiten calcular las funciones básicas.

- La *función idénticamente nula*, $\mathcal{O} : \mathbb{N} \to \mathbb{N}$, definida por $\mathcal{O}(k) = 0$, para todo $k \in \mathbb{N}$, puede calcularse a través del sistema

$$\Pi^{zero} = \left(\Sigma, \Gamma, \Lambda, \#, \mu_{\Pi^{zero}}, \mathcal{M}_1, (R_1, \rho_1), im, ext\right)$$

donde

$$\begin{aligned} \Sigma = \{a\} \qquad \Gamma = \{a, b, \#\} \qquad \Lambda = \{b\} \\ \mu_{\Pi^{zero}} = [\,_1]_1 \qquad \mathcal{M}_1 = \# \qquad im = 1 \\ (R_1, \rho_1) = \# \to (\#, out) \end{aligned}$$

Es fácil comprobar que Π^{zero} es un sistema válido de cálculo de funciones, de orden $(1, 1)$ y que calcula la función idénticamente nula, $\mathcal{O}$.

- La *función sucesor*, $\mathcal{S} : \mathbb{N} \to \mathbb{N}$, definida por $\mathcal{S}(k) = k + 1$, para todo $k \in \mathbb{N}$, puede calcularse mediante el sistema

$$\Pi^{suc} = \left(\Sigma, \Gamma, \Lambda, \#, \mu_{\Pi^{suc}}, \mathcal{M}_1, (R_1, \rho_1), im, ext\right)$$

donde

$$\begin{aligned} \Sigma = \{a\} \qquad \Gamma = \{a, b, \#\} \qquad \Lambda = \{b\} \\ \mu_{\Pi^{suc}} = [\,_1]_1 \qquad \mathcal{M}_1 = \# \qquad im = 1 \\ (R_1, \rho_1) = \begin{cases} a \to (b, out) \\ \# \to (b, out)(\#, out) \end{cases} \end{aligned}$$

Es fácil comprobar que Π^{suc} es un sistema válido de cálculo de funciones, de orden $(1,1)$ y que calcula la función sucesor, $\mathcal{S}$.

- Las *funciones proyecciones*, $\Pi_j^n : \mathbb{N}^n \to \mathbb{N}$, con $n \geq 1$ y $1 \leq j \leq n$, definidas por $\Pi_j^n(k_1, \ldots, k_n) = k_j$, para cada $(k_1, \ldots, k_n) \in \mathbb{N}^n$, pueden calcularse a través de los sistemas

$$\Pi_{n,j}^{proj} = \left(\Sigma, \Gamma, \Lambda, \#, \mu_{\Pi_{n,j}^{proj}}, \mathcal{M}_1, (R_1, \rho_1), im, ext\right)$$

donde

$$\Sigma = \{a_1, \ldots, a_n\} \qquad \Gamma = \{a_1, \ldots, a_n, b, \#\} \qquad \Lambda = \{b\}$$
$$\mu_{\Pi_{n,j}^{proj}} = [_1]_1 \qquad \mathcal{M}_1 = \# \qquad im = 1$$
$$(R_1, \rho_1) = \begin{cases} a_j \to (b, out) \\ \# \to (\#, out) \end{cases}$$

Es fácil comprobar que $\Pi_{n,j}^{proj}$ es un sistema válido de cálculo de funciones, de orden $(n,1)$ y que calcula la función proyección, Π_j^n.

6.2.2. Composición.

A continuación vamos a diseñar un sistema P que calcula la composición de funciones, a partir de sistemas que calculan dichas funciones. Para ello sean $\Pi_f, \Pi_{g_1}, \ldots, \Pi_{g_t} \in \mathcal{FC}$ tales que calculan, respectivamente, la función $f : \mathbb{N}^m - \to \mathbb{N}^n$ y las funciones $g_1 : \mathbb{N}^r - \to \mathbb{N}^{s_1}, \ldots, g_t : \mathbb{N}^r - \to \mathbb{N}^{s_t}$ (con $s_1 + \cdots + s_t = m$).

Podemos suponer que

$$\Pi_f = \left(\Sigma_f, \Gamma_f, \Lambda_f, \#_f, \mu_{\Pi_f}, \mathcal{M}_1^f, \ldots, \mathcal{M}_{p_f}^f, (R_1^f, \rho_1^f), \ldots, (R_{p_f}^f, \rho_{p_f}^f), im_f, ext\right)$$
$$\Pi_{g_1} = \left(\Sigma_{g_1}, \Gamma_{g_1}, \Lambda_{g_1}, \#_{g_1}, \mu_{\Pi_{g_1}}, \mathcal{M}_1^{g_1}, \ldots, \mathcal{M}_{p_{g_1}}^{g_1}, (R_1^{g_1}, \rho_1^{g_1}), \ldots, (R_{p_{g_1}}^{g_1}, \rho_{p_{g_1}}^{g_1}), im_{g_1}, ext\right)$$
$$\vdots$$
$$\Pi_{g_t} = \left(\Sigma_{g_t}, \Gamma_{g_t}, \Lambda_{g_t}, \#_{g_t}, \mu_{\Pi_{g_t}}, \mathcal{M}_1^{g_t}, \ldots, \mathcal{M}_{p_{g_t}}^{g_t}, (R_1^{g_t}, \rho_1^{g_t}), \ldots, (R_{p_{g_t}}^{g_t}, \rho_{p_{g_t}}^{g_t}), im_{g_t}, ext\right)$$

Renombrando adecuadamente los elementos de los alfabetos (y, por consiguiente, también de las reglas) podemos suponer, además, que

- $\Sigma_{g_1} = \cdots = \Sigma_{g_t} = \{a_1, \ldots, a_r\}$.
- $\Lambda_{g_1} = \{b_1, \ldots, b_{s_1}\}, \ldots, \Lambda_{g_t} = \{b_{s_1+\cdots+s_{t-1}+1}, \ldots, b_m\}$.
- $\Sigma_f = \{c_1, \ldots, c_m\}$.
- $\Lambda_f = \{d_1, \ldots, d_n\}$.
- $\left(\Lambda_{g_1} \cup \cdots \cup \Lambda_{g_t}\right) \cap \Gamma_f = \emptyset$.
- El objeto $\#_{g_i}$ es distinto del objeto $\#_f$, para cada $i = 1, \ldots, t$.
- El objeto $\#_{g_i}$ es distinto del objeto $\#_{g_j}$, para cada $i \neq j$.

Consideremos el sistema P de cálculo de funciones

$$\Pi = \left(\Sigma, \Gamma, \Lambda, \#, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, (R_1, \rho_1), \ldots, (R_p, \rho_p), im, ext\right)$$

dado por

- $\Sigma = \{e_1, \ldots, e_r\}$. Podemos suponer, además, que se verifica que Σ es disjunto con $\bigcup_{i=1}^t \Gamma_{g_i} = \emptyset$.
- Existen elementos distinguidos $\oplus, \ominus, \odot \in \Gamma \setminus (\Gamma_f \cup \bigcup_{i=1}^t \Gamma_{g_i})$.
- $\Lambda = \{d_1, \ldots, d_n\}$.
- El objeto $\#$ es distinto del objeto $\#_f$ y de los objetos $\#_{g_i}$, para cada $i = 1, \ldots, t$.
- $\mu_\Pi = [_1 \mu_{\Pi_{g_1}} \ldots \mu_{\Pi_{g_t}} \mu_{\Pi_f}]_1$, donde las membranas de $\mu_{\Pi_{g_1}}, \ldots, \mu_{\Pi_{g_t}}, \mu_{\Pi_f}$ se han renombrado adecuadamente (y, por consiguiente, también se han adaptado las reglas de los correspondientes sistemas). Notaremos por $\sigma_{g_1}, \ldots, \sigma_{g_t}, \sigma_f$ las membranas piel de estos sistemas. Además, consideramos que $im_{g_1}, \ldots, im_{g_t}, im_f$ reflejan el nuevo etiquetado de las membranas de entrada de $\Pi_{g_1}, \ldots, \Pi_{g_t}, \Pi_f$, respectivamente.
- $p = p_{g_1} + \cdots + p_{g_t} + p_f + 1$.
- $\mathcal{M}_1 = \#\ominus$. Los restantes multiconjuntos son todos vacíos.
- $im = 1$.

- Las reglas de evolución y sus prioridades son las siguientes:

  - Reglas de evolución para la membrana 1:

$$e_i \to (e_i, in_{\sigma_{g_1}}) \ldots (e_i, in_{\sigma_{g_t}}) \quad (i=1,\ldots,r)$$
$$\ominus \to (\ominus, in_{\sigma_{g_1}}) \ldots (\ominus, in_{\sigma_{g_t}})$$
$$\#_{g_1} \ldots \#_{g_t} \# \to (\ominus, in_{\sigma_f}) > \# \to \# > b_i \to (b_i, in_{\sigma_f}) \quad (i=1,\ldots,m)$$
$$d_i \to (d_i, out) \quad (i=1,\ldots,n)$$
$$\#_f \to (\#, out)$$

  - Para toda función $fun \in \{g_1, \ldots, g_t, f\}$ y para toda membrana j de $\mu_{\Pi_{fun}}$, se incluyen las siguientes reglas:

$$\ominus \to \oplus(\ominus, in_{j_1}) \ldots (\ominus, in_{j_k})$$
$$\odot^u \oplus \to \mathcal{M}_j^{fun} > \oplus \to \oplus\odot$$

    Las reglas de evolución y prioridades asociadas a la membrana en Π_{fun}

    En donde $j_1, \ldots j_k$ son las membranas hijas de la membrana j y u es su nivel de profundidad dentro del árbol $\mu_{\Pi_{fun}}$. Además, si $j = im_{fun}$, entonces la regla $\oplus \to \oplus\odot$ tiene mayor prioridad que las reglas originales de Π_{fun} para esta membrana.

  - Sea $fun \in \{g_1, \ldots, g_t, f\}$ como arriba y sea $j_1, \ldots, j_q$ el camino de membranas que conecta σ_{fun} con la membrana de entrada, im_{fun}, en el árbol $\mu_{\Pi_{fun}}$. Entonces, para cada $k = 1, \ldots, q-1$ se incluyen en la membrana j_k las siguientes reglas:

$$e_i \to (e_i, in_{j_{k+1}}) \quad (i=1,\ldots,r), \qquad \text{para } fun = g_1, \ldots, g_t$$
$$b_i \to (b_i, in_{j_{k+1}}) \quad (i=1,\ldots,m), \qquad \text{para } fun = f$$

    También se incluyen en la membrana $j_q = im_{fun}$ las siguientes reglas:

$$e_i \to a_i \quad (i=1,\ldots,r), \qquad \text{para } fun = g_1, \ldots, g_t$$
$$b_i \to c_i \quad (i=1,\ldots,m), \qquad \text{para } fun = f$$

A continuación vamos a justificar que el sistema diseñado, que notamos por $C(\Pi_f; \Pi_{g_1}, \ldots, \Pi_{g_t})$, es un sistema P de cáculo de funciones que es válido y que, además, calcula la composición de f con $g_1, \ldots g_t$. Dicho sistema también preserva el uso o no de la disolución de membranas a partir de los sistemas P que calculan las funciones.

En efecto, el funcionamiento del sistema está estructurado, básicamente, en dos fases.

Fase 1: *Cálculo de las funciones $g_1, \ldots, g_t$ sobre el dato de entrada.*

Para realizar esta fase, es necesario llevar a cabo dos operaciones: la primera consiste en *transportar* los argumentos de entrada desde la membrana 1, la cual recuérdese es la membrana de entrada de Π, hasta cada una de las membranas de entrada de los sistemas $\Pi_{g_1}, \ldots, \Pi_{g_t}$. Esto se consigue fácilmente desplazando los objetos que representan los argumentos a través de todas las membranas que sea necesario.

La segunda operación es un poco más complicada: para que un sistema concreto, Π_{g_j}, calcule correctamente el valor de la función g_j sobre el dato de entrada, necesitamos que todas las membranas de este sistema empiecen a aplicar sus reglas *originales* al mismo tiempo (esto es, tenemos que conseguir una *sincronización local* de todas las membranas de cada Π_{g_j}). Esto se puede lograr usando contadores para cada una de estas membranas. En primer lugar, usamos el objeto $\ominus$ para activar los contadores, representados por objetos $\oplus$, en todas las membranas. Estos últimos objetos utilizan, a su vez, objetos $\odot$ para contar y, cuando se ha alcanzado una cierta cantidad, se permite a la membrana correspondiente aplicar las reglas de Π_{g_j}. A causa de la forma en que se ha implementado, estas cantidades coinciden con los niveles de profundidad de cada una de las membranas en la estructura $\mu_{\Pi_{g_j}}$.

También es importante observar que cuando el sistema Π_{g_j} comience a calcular el valor, los objetos que representan el dato de entrada deben haber alcanzado su correspondiente membrana de entrada. No obstante, como realizamos las dos operaciones anteriores simultáneamente, esto lo obtenemos de manera automática.

Finalmente, antes de permitir que se active el sistema Π_f, hay que asegurarse de que se hayan calculado todos los valores de $\Pi_{g_1}, \ldots, \Pi_{g_t}$ (esto es, debe existir una *sincronización global* en la piel de Π).

Veamos con mayor detalle las reglas involucradas en esta fase:

a) En el primer paso de una computación de Π en la que se calcula el valor de la función composición sobre la tupla $(k_1, \ldots, k_r)$, en la membrana 1 tenemos el multiconjunto $e_1^{k_1} \ldots e_r^{k_r} \# \ominus$ y las membranas restantes están vacías. Por lo tanto, solo se pueden aplicar las reglas que envían los objetos e_i y el objeto $\ominus$ dentro de las membranas piel correspondientes de $\mu_{\Pi_{g_1}}, \ldots, \mu_{\Pi_{g_t}}$, y la regla $\# \rightarrow \#$ en la membrana 1.

b) Ahora la membrana 1 espera los valores de las funciones $g_1, \ldots, g_t$ sobre la tupla $(k_1, \ldots, k_r)$ por medio de la regla $\# \rightarrow \#$. Respecto a las estructuras de membranas $\mu_{\Pi_{g_1}}$ a $\mu_{\Pi_{g_t}}$, la regla $\ominus \rightarrow \oplus(\ominus, in_{j_1}) \ldots (\ominus, in_{j_k})$ hace que el objeto $\ominus$ se propague a todas sus membranas, ya que cuando este alcanza una membrana particular, se transforma inmediatamente en un objeto contador $\oplus$ y también se envía a las membranas hijas. Así, de un paso de computación al siguiente, el objeto $\ominus$ alcanza las membranas con profundidad una unidad mayor. Mientras tanto, la regla $\oplus \rightarrow \oplus\odot$ hace que el objeto $\oplus$ genere objetos $\odot$. Una mirada atenta a la situación creada nos muestra que el objeto activador $\ominus$ ha alcanzado todas las membranas exactamente cuando el objeto contador $\oplus$ ha generado en cada membrana un número de objetos $\odot$ igual a sus niveles de profundidad en el árbol $\mu_{\Pi_{fun}}$ (siendo $fun \in \{g_1, \ldots, g_t\}$). En ese momento, la regla $\odot^u \oplus \rightarrow \mathcal{M}_j^{fun}$ introduce en la membrana j los objetos asociados a ella en Π_{fun}, y esto se hace para todas las membranas de cada Π_{fun} a la vez. A partir de ese instante, los valores de $g_1, \ldots, g_t$ sobre $(k_1, \ldots, k_r)$ se calculan exactamente de la misma manera que lo harían los sistemas $\Pi_{g_1}, \ldots, \Pi_{g_t}$.

c) Simultáneamente, los objetos e_i recorren el camino desde la membrana piel de cada $\mu_{\Pi_{g_j}}$ hasta la membrana de entrada de Π_{g_j}, por medio de las reglas $e_i \rightarrow (e_i, in_{j_{k+1}})$, y allí evolucionan en los correspondientes objetos a_i, por medio de las reglas $e_i \rightarrow a_i$. Téngase presente que los objetos e_i y el objeto $\ominus$ alcanzan la membrana de entrada de Π_{g_j} al mismo tiempo. De esta forma, cuando Π_{g_j} empieza a realizar su trabajo original, el dato de entrada está en el lugar adecuado.

Fase 2: *Cálculo de la función f*

La primera fase termina cuando la membrana 1 ha recogido, al menos, un objeto de cada $\#_{g_1}, \ldots, \#_{g_t}$. Es entonces cuando los valores calculados se tienen que enviar como datos de entrada al sistema Π_f. Para sincronizar el final de la primera fase con el principio de esta segunda, en la membrana 1 se aplica una y otra vez la regla $\# \rightarrow \#$ hasta que se pueda usar la regla $\#_{g_1} \ldots \#_{g_t} \# \rightarrow (\ominus, in_{\sigma_f})$.

Esta última regla envía un objeto $\ominus$ a la piel de μ_{Π_f}, con el objetivo de iniciar los contadores de sus membranas de forma que empiecen a aplicar sus reglas originales al mismo tiempo (*sincronización local* dentro de Π_f). Este proceso se realiza de manera similar al anterior. También en el siguiente paso de la computación los objetos b_i, que representan los valores obtenidos en la primera fase, se ponen dentro de la piel de μ_{Π_f} y, seguidamente, se mueven, por medio de las reglas $b_i \rightarrow (b_i, in_{j_{k+1}})$, a través de las membranas de μ_{Π_f}, desde la piel hasta la correspondiente membrana de entrada de Π_f.

Es fácil comprobar que, aunque existe un desfase de un paso de computación entre el momento en que el objeto $\ominus$ llega a una membrana y el momento en que llegan los objetos b_i, esto no representa problema alguno.

Seguidamente, se calcula el valor de la función f sobre los argumentos representados por los objetos c_i y, a lo largo de esta computación, se expulsan de μ_{Π_f} objetos d_i que representan el resultado. Estos objetos se recogen en la membrana 1 y se expelen inmediatamente de μ_Π. El proceso de cálculo termina cuando se recoge algún objeto $\#_f$ en la membrana 1, y todos ellos son enviados al entorno externo de μ_Π como objetos $\#$.

6.2.3. Iteración.

A continuación veamos cómo diseñamos, a partir de un sistema $\Pi_f \in \mathcal{FC}(nDis)$ que calcula la función $f : \mathbb{N}^m - \rightarrow \mathbb{N}^m$, un sistema P de cálculo de funciones que calcule la iteración de f.

Supongamos que

$$\Pi_f = (\Sigma_f, \Gamma_f, \Lambda_f, \#_f, \mu_{\Pi_f}, \mathcal{M}_1^f, \ldots, \mathcal{M}_{p_f}^f, (R_1^f, \rho_1^f), \ldots, (R_{p_f}^f, \rho_{p_f}^f), im_f, ext)$$

Renombrando adecuadamente los elementos de los alfabetos (y, por consiguiente, también de las reglas) podemos suponer, además, que

- $\Sigma_f = \{a_1, \ldots, a_m\}$.
- $\Lambda_f = \{b_1, \ldots, b_m\}$.

Consideremos el sistema P de cálculo de funciones

$$\Pi = \big(\Sigma, \Gamma, \Lambda, \#, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, (R_1, \rho_1), \ldots, (R_p, \rho_p), im, ext\big)$$

que verifica lo siguiente:

- $\Sigma = \{c_1, \ldots, c_{m+1}\}$. Podemos suponer, además, que se satisface la condición $\Sigma \cap \Gamma_f = \emptyset$.
- Existen elementos distinguidos $\oplus, \ominus, \odot, \otimes, \oslash \in \Gamma \setminus \Gamma_f$.
- $\Lambda = \{c_1, \ldots, c_m\}$.
- El objeto $\#$ es distinto del objeto $\#_f$.
- $\mu_\Pi = [_1 \mu_{\Pi_f}]_1$, donde las membranas de μ_{Π_f} se han renombrado adecuadamente (y, por consiguiente, también se han adaptado las reglas de Π_f). Notamos por σ_f la membrana piel de este sistema. Además, consideramos que im_f refleja el nuevo etiquetado de la membrana de entrada de Π_f.
- $p = p_f + 1$.
- $\mathcal{M}_1 = \#$. Los multiconjuntos restantes son todos vacíos.
- $im = 1$.
- Las reglas de evolución y sus prioridades son las siguientes:
  - Reglas de evolución para la membrana 1:

$$\begin{aligned}
&\#c_{m+1} \to (\ominus, in_{\sigma_f}) > \#c_i \to \#(c_i, out) \quad {\scriptstyle (i=1,\ldots,m)} > \\
&> \# \to (\#, out) > \#_f\#_f \to \#_f > \#_f \to (\oslash, in_{\sigma_f}) > \\
&> \otimes^u b_i \to \otimes^u c_i \quad {\scriptstyle (i=1,\ldots,m)} > \otimes^u \to \# > \\
&> c_i \to (c_i, in_{\sigma_f}) \quad {\scriptstyle (i=1,\ldots,m)}
\end{aligned}$$

donde u es el número de membranas de la estructura μ_{Π_f}.

- Para cada membrana j distinta de la membrana 1 se incluyen las siguientes reglas:

$$\ominus \to \oplus(\ominus, in_{j_1}) \ldots (\ominus, in_{j_k})$$
$$\odot^v \oplus \to \mathcal{M}_j^f > \oplus \to \oplus\odot$$
$$\oslash \to \otimes(\oslash, in_{j_1}) \ldots (\oslash, in_{j_k})$$
$$ob\otimes \to \otimes \quad (ob \in \Gamma_f) > \otimes \to (\otimes, out)$$

Las reglas de evolución y prioridades asociadas a la membrana en Π_f

En donde $j_1, \ldots, j_k$ son las membranas hijas de la membrana j y v es su nivel de profundidad dentro de μ_{Π_f}. Además, si $j = im_f$, entonces la regla $\oplus \to \oplus\odot$ tiene mayor prioridad que las reglas originales de esta membrana en Π_f.

- Sea $j_1, \ldots, j_q$ el camino de membranas que conecta σ_f con la membrana de entrada, im_f, en el árbol μ_{Π_f}. Entonces, para cada $k = 1, \ldots, q-1$, se incluyen en la membrana j_k las siguientes reglas:

$$c_i \to (c_i, in_{j_{k+1}}) \quad (i=1,\ldots,m)$$

También se incluyen en la membrana $j_q = im_f$ las siguientes reglas:

$$c_i \to a_i \quad (i=1,\ldots,m)$$

A continuación vamos a justificar que el sistema antes diseñado, que notamos por $It(\Pi_f)$, es un sistema P de cálculo de funciones que es válido y que, además, calcula la iteración de f.

En efecto, describimos seguidamente, de manera informal, cómo funciona dicho sistema.

El número de iteraciones de f a realizar viene dado por el $(m+1)$-ésimo argumento proporcionado a $It(f)$. Lo que hacemos entonces es reducir este argumento en uno y, a continuación, llevamos a cabo un proceso que consta de dos fases: la primera consiste en calcular una iteración de f; la segunda consiste en «reiniciar» el sistema Π_f a su estado inicial. Reiteramos este proceso hasta que el $(m+1)$-ésimo argumento se hace cero.

La condición para decidir si se tiene o no que realizar una iteración se comprueba en la membrana 1 examinando cuántos objetos c_{m+1}, que repre-

sentan el $(m+1)$-ésimo argumento, hay en dicha membrana. Si alguno de tales objetos está presente, se aplica la regla $\#c_{m+1} \rightarrow (\ominus, in_{\sigma_f})$ (seguida por las reglas $c_i \rightarrow (c_i, in_{\sigma_f})$), comenzando el cálculo de una nueva iteración de la función f.

Fase 1: *Cálculo de una iteración de f*

Esta fase comienza cuando se introduce un objeto $\ominus$ en la piel de μ_{Π_f}. Este objeto inicia contadores en las membranas de μ_{Π_f}, de forma análoga a como se realizó para la composición, a fin de asegurarnos que comenzarán a aplicar sus reglas *originales* al mismo tiempo (*sincronización local* dentro de Π_f). También, con un desfase de un paso de computación que no es relevante, el dato de entrada, representado por los objetos c_i, se transporta desde la piel de μ_{Π_f} hasta la membrana de entrada de Π_f. Aunque durante la ejecución de esta fase el resultado de una iteración se está enviando fuera de μ_{Π_f}, recogiéndose en la membrana 1 de Π, hay que observar que en esta membrana no se activa ninguna regla.

Fase 2: *Reinicio del sistema* Π_f

La primera fase termina cuando algún objeto $\#_f$ es recogido en la membrana 1 de Π. Antes de que podamos comenzar la simulación de otra iteración de f, es necesario borrar todos los objetos que quedan en las membranas de μ_{Π_f}. Esto es lo que se realiza en esta fase, que comienza reduciendo el número de objetos $\#_f$ presentes en la membrana 1 a solo uno. Entonces la regla $\#_f \rightarrow (\oslash, in_{\sigma_f})$ en dicha membrana introduce un objeto $\oslash$ en la piel de μ_{Π_f}.

Este objeto se propaga a todas las membranas de la misma manera que $\ominus$ lo hace en la fase anterior, y deja un objeto $\otimes$ en cada una de ellas. Estos últimos objetos actúan como borradores, eliminando todos los objetos de las membranas por medio de las reglas $ob\otimes \rightarrow \otimes$. Cuando una membrana se ha vaciado (es decir, cuando solo queda en ella un objeto $\otimes$), entonces se expulsa de ella el objeto $\otimes$.

Por consiguiente, esta fase termina cuando la membrana 1 recoge tantos objetos $\otimes$ como indica el grado de μ_{Π_f}. Solo entonces se pueden aplicar las reglas $\otimes^u b_i \rightarrow \otimes^u c_i$, las cuáles transforman el resultado de una iteración de f en datos de entrada de Π. Finalmente, se aplica la regla $\otimes^u \rightarrow \#$ para empezar el proceso de nuevo.

En el instante en que ningún objeto c_{m+1} esté presente en la membrana 1, hay que finalizar el proceso de simulación de iteraciones. Entonces hay que enviar los objetos $c_1, \ldots, c_m$ de esta membrana al entorno externo, seguido de un objeto #.

Obsérvese que durante la evaluación de la condición de parada no se puede aplicar ninguna regla en ninguna membrana distinta de la membrana piel, porque están vacías.

6.2.4. Minimización no acotada.

Finalmente, vamos a describir el diseño de un sistema P de cálculo de funciones que calcula, a partir de un sistema $\Pi_f \in \mathcal{FC}(nDis)$ que calcula una función $f : \mathbb{N}^{m+1} - \to \mathbb{N}$, la función obtenida por minimización no acotada a partir de f.

Supongamos que

$$\Pi_f = \left(\Sigma_f, \Gamma_f, \Lambda_f, \#_f, \mu_{\Pi_f}, \mathcal{M}_1^f, \ldots, \mathcal{M}_{p_f}^f, (R_1^f, \rho_1^f), \ldots, (R_{p_f}^f, \rho_{p_f}^f), im_f, ext\right)$$

Renombrando adecuadamente los elementos de los alfabetos (y, por consiguiente, también de las reglas) podemos suponer, además, que

- $\Sigma_f = \{a_1, \ldots, a_{m+1}\}$.
- $\Lambda_f = \{b\}$.

Consideremos el sistema P de cálculo de funciones

$$\Pi = \left(\Sigma, \Gamma, \Lambda, \#, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, (R_1, \rho_1), \ldots, (R_p, \rho_p), im, ext\right)$$

que verifica lo siguiente:

- $\Sigma = \{c_1, \ldots, c_m\}$. Podemos suponer, además, que se satisface la condición $\Sigma \cap \Gamma_f = \emptyset$.
- Existen elementos distinguidos $\oplus, \ominus, \odot, \otimes, \oslash \in \Gamma \setminus \Gamma_f$.
- $\Lambda = \{c_{m+1}\}$.
- El objeto # es distinto del objeto $\#_f$.

- $\mu_\Pi = [_1 \mu_{\Pi_f}]_1$, donde las membranas de μ_{Π_f} se han renombrado adecuadamente (y, por consiguiente, también se han adaptado las reglas de Π_f). Notamos por σ_f la membrana piel de este sistema. Además, consideramos que im_f refleja el nuevo etiquetado de la membrana de entrada de Π_f.
- $p = p_f + 1$.
- $\mathcal{M}_1 = \#$. Los multiconjuntos restantes son todos vacíos.
- $im = 1$.
- Las reglas de evolución y sus prioridades son las siguientes:
  - Reglas de evolución para la membrana 1:

    $$\#_f\#_f \to \#_f > \#_f b \to bc_{m+1}(\oslash, in_{\sigma_f}) > \otimes^u \to \# > \#b \to \# >$$
    $$\#c_i \to \#d_i \quad (i=1,\ldots,m+1) > \# \to (\ominus, in_{\sigma_f}) >$$
    $$d_i \to c_i(d_i, in_{\sigma_f}) \quad (i=1,\ldots,m+1) > \#_f c_{m+1} \to \#_f(c_{m+1}, out) >$$
    $$\#_f \to (\#, out)$$

    donde u es el número de membranas de la estructura μ_{Π_f}.
  - Para cada membrana j distinta de la membrana 1 se incluyen las siguientes reglas:

    $$\ominus \to \oplus(\ominus, in_{j_1}) \ldots (\ominus, in_{j_k})$$
    $$\odot^v \oplus \to \mathcal{M}_j^f > \oplus \to \oplus\odot$$
    $$\oslash \to \otimes(\oslash, in_{j_1}) \ldots (\oslash, in_{j_k})$$
    $$ob\otimes \to \otimes \quad (ob \in \Gamma_f) > \otimes \to (\otimes, out)$$

    Las reglas de evolución y prioridades asociadas a la membrana en Π_f

    En donde $j_1, \ldots, j_k$ son las membranas hijas de la membrana j y v es su nivel de profundidad dentro de μ_{Π_f}. Además, si $j = im_f$, entonces la regla $\oplus \to \oplus\odot$ tiene mayor prioridad que las reglas originales de esta membrana en Π_f.
  - Sea $j_1, \ldots, j_q$ el camino de membranas que conecta σ_f con la membrana de entrada, im_f, en el árbol μ_{Π_f}. Entonces, para ca-

da $k = 1, \ldots, q-1$, se incluyen en la membrana j_k las siguientes reglas:

$$d_i \rightarrow (d_i, in_{j_{k+1}}) \quad (i=1,\ldots,m+1)$$

También se incluyen en la membrana $j_q = im_f$ las siguientes reglas:

$$d_i \rightarrow a_i \quad (i=1,\ldots,m+1)$$

A continuación vamos a justificar que el sistema antes diseñado, que notamos por $Min(\Pi_f)$, es un sistema P de cálculo de funciones que es válido y que, además, calcula la minimización de f.

En efecto, describimos seguidamente, de manera informal, cómo funciona dicho sistema.

Dado un dato de entrada $\vec{x} \in \mathbb{N}^m$ tenemos que calcular los valores $f(\vec{x}, y)$ para $y = 0, 1, 2$ y así sucesivamente hasta encontrar el primero que sea cero, en cuyo caso devolvemos el valor correspondiente de y. El dato $\vec{x} = (x_1, \ldots, x_m)$ lo representamos mediante los objetos c_i, con $i = 1, \ldots, m$, mientras que el número y vendrá dado por la cantidad de objetos c_{m+1} presentes en el sistema.

Para realizar lo anterior, el sistema Π repite una y otra vez un proceso estructurado en dos fases: la primera consiste en calcular el valor de f aplicado al dato de entrada $\vec{x} = (x_1, \ldots, x_m)$ y a un determinado número y; en la segunda se comprueba el resultado obtenido. Si es cero, entonces hemos terminado y basta expulsar al entorno externo los objetos c_{m+1}. Si no es cero, entonces añadimos un nuevo objeto c_{m+1}, de forma que estos representen el número $y + 1$, y «reiniciamos» el sistema Π_f a su configuración inicial, volviendo a empezar con la primera fase.

Fase 1: *Cálculo de* $f(\vec{x}, y)$

Esta fase se activa con la presencia de un objeto # en la membrana 1 de Π. Lo primero que se hace es borrar, mediante la regla $\#b \rightarrow \#$, el resultado de $f(\vec{x}, y-1)$ que habríamos obtenido anteriormente. A continuación cambiamos los objetos c_i en objetos d_i, con el objetivo de poder enviarlos al sistema Π_f y, a la vez, conservarlos en la membrana de entrada de Π. Una vez hecho esto, enviamos un objeto $\ominus$ a la piel de μ_{Π_f} para, de forma análoga a como hemos visto en la composición y la iteración, realizar una *sincronización local* de sus membranas. También, con un desfase de un paso de computación que no es relevante, los

objetos d_i, que representan los argumentos a los que vamos a aplicar la función f, se transportan desde la piel de μ_{Π_f} hasta la membrana de entrada de Π_f. Además, nos quedamos con una copia en la membrana 1 usando objetos c_i.

A partir de ese instante no se podrá aplicar ninguna regla en la membrana 1 hasta que Π_f no termine de calcular el valor de la función f aplicado al par $(\vec{x}, y)$.

Fase 2: *Comprobación del resultado*

En esta fase lo primero que se hace es reducir a uno solo el número de objetos $\#_f$ recogidos en la membrana 1 de Π. Entonces, si el resultado de $f(\vec{x}, y)$ ha sido cero, las únicas reglas aplicables son la regla $\#_f c_{m+1} \to \#_f(c_{m+1}, out)$, que envía los objetos c_{m+1} al entorno externo, seguida de la regla $\#_f \to (\#, out)$, que termina la computación.

Si el resultado de $f(\vec{x}, y)$ ha sido distinto de cero, entonces en la membrana 1 de Π se ha recogido algún objeto b y, por tanto, será aplicable la regla $\#_f b \to b c_{m+1}(\oslash, in_{\sigma_f})$. Esta regla añade un nuevo objeto c_{m+1}, para que su multiplicidad represente el número $y + 1$. Además, dicha regla envía un objeto $\oslash$ a la membrana piel de μ_{Π_f} para reiniciar el sistema Π_f, exactamente igual a como hacíamos con la iteración. Entonces no se podrá aplicar ninguna regla en la membrana 1 de Π hasta que no aparezcan tantos objetos $\otimes$ como membranas contiene μ_{Π_f}. En ese momento, la regla $\otimes^u \to \#$ pone un objeto $\#$ para que vuelva a empezar la fase 1.

De todo lo descrito anteriormente se deduce la completitud computacional de los sistemas P de cálculo de funciones, que usan prioridad y cooperación.

Teorema 6.4. *Sea $f \in \mathcal{P}$ una función recursiva parcial. Entonces existe un sistema $\Pi_f \in \mathcal{FC}(Pri, Coo, nDis)$ que calcula la función f.*

Demostración. Para ello, basta tener presente que si f es una función recursiva, entonces existen funciones $g_1, \ldots, g_n$ tales que $g_n = f$ y para cada $j = 1, \ldots, n$ se tiene que, o bien g_j es una función básica, o bien g_j se obtiene de algunas de las funciones $g_1, \ldots, g_{j-1}$ mediante las operaciones de composición, iteración o minimización. □

Capítulo 7

Sistemas P como dispositivos generadores de conjuntos

En este capítulo se estudian sistemas P de transición con salida externa que generan conjuntos de tuplas de números naturales. En estos sistemas, que carecen de membrana de entrada, cada computación proporcionará una tupla del conjunto a generar.

En la sección 7.1 definimos la función *Output* de los sistemas P de transición con salida externa para obtener la variante de sistemas P de generación de conjuntos. En la sección 7.2 mostramos cómo construir sistemas P de cálculo de funciones que calculen cualquier polinomio con coeficientes enteros y, a partir de estos, cómo construir sistemas P de generación de conjuntos que generen cualquier conjunto diofántico.

7.1. Sistemas P de generación de conjuntos

Análogamente a los sistemas P de cálculo de funciones, el alfabeto de salida de los sistemas que vamos a introducir a continuación estará ordenado. Así, los multiconjuntos recogidos en el entorno externo del sistema en cada computación codificarán tuplas de números naturales.

Definición 7.1. *Un* sistema P de generación de conjuntos, Π, *de orden n es un sistema de computación con membranas que verifica las siguientes propiedades:*

- Π *es un sistema P de transición con salida externa.*
- Π *es un sistema P sin membrana de entrada.*
- *El alfabeto de salida,* Λ*, de* Π *es un alfabeto ordenado de* n *elementos. Supondremos que* $\Lambda = \{b_1, \ldots, b_n\}$.
- *La salida de una computación* $\mathcal{C} = \{C^i\}_{i<r}$ *viene dada por la siguiente función:*

$$Output(\mathcal{C}) = \begin{cases} \text{no definida}, & \text{si } \mathcal{C} \text{ no es de parada} \\ \left(M_{env}^{r-1}(b_1), \ldots, M_{env}^{r-1}(b_n)\right), & \text{si } \mathcal{C} \text{ es de parada} \end{cases}$$

De esta forma, la salida de una computación de parada de Π *es una tupla de* n *números naturales.*

Al contrario que con los sistemas de computación celular con membranas analizados en los capítulos anteriores, no exigimos condiciones de validez adicionales a los sistemas P de generación de conjuntos. Así, diremos que un tal sistema es *válido* si lo es como sistema P de transición con salida externa.

Notación. *Notaremos por* $\mathcal{SG}^n$ *la clase de los sistemas P de generación de conjuntos de orden* n *tales que dichos sistemas sean válidos. La clase* $\mathcal{SG}$ *es la unión de todas las clases anteriores.*

La definición de generación de conjuntos de tuplas de números naturales por sistemas pertenecientes a estas clases se realiza de manera natural.

Definición 7.2. *Diremos que un sistema* $\Pi \in \mathcal{SG}^n$ *genera el conjunto* $A \subseteq \mathbb{N}^n$ *si se verifican las siguientes propiedades:*

- *Para cada* $(k_1, \ldots, k_n) \in \mathbb{N}^n$*, si* $(k_1, \ldots, k_n) \in A$*, entonces existe una computación de parada* $\mathcal{C}$ *de* Π *tal que* $Output(\mathcal{C}) = (k_1, \ldots, k_n)$.
- *Si* $\mathcal{C}$ *es una computación de parada de* Π*, entonces* $Output(\mathcal{C}) \in A$.

7.2. Completitud computacional a través de conjuntos diofánticos

Recordemos que un conjunto de números naturales, $A \subseteq \mathbb{N}^m$, es diofántico si existe un polinomio $p(a_1, \ldots, a_m, x_1, \ldots, x_n)$ con coeficientes enteros tal que

el conjunto A coincide con el conjunto

$$\{(a_1,\ldots,a_m)\in\mathbb{N}^m : \exists(x_1,\ldots,x_n)\in\mathbb{N}^n\big(P(a_1,\ldots,a_m,x_1,\ldots,x_n)=0\big)\}$$

Además, en virtud del teorema **MRDP** (de Matiyasevich, Robinson, Davis, Putnam), la clase de conjuntos recursivamente enumerables coincide con la clase de conjuntos diofánticos.

En esta sección describimos los pasos a seguir para generar estos últimos conjuntos a través de sistemas de la clase $\mathcal{SG}$. Para ello, en primer lugar diseñamos ciertos sistemas P de cálculo de funciones que servirán de base para construir otros que calculan monomios y polinomios. A partir de estos obtendremos entonces los sistemas que generan conjuntos diofánticos arbitrarios.

7.2.1. Funciones básicas.

Comenzamos proporcionando unos sistemas P de cálculo de funciones que calculan ciertas funciones básicas.

- Las *funciones proyecciones*, $\Pi^n_j : \mathbb{N}^n \to \mathbb{N}$, con $n \geq 1$ y $1 \leq j \leq n$, definidas por $\Pi^n_j(k_1,\ldots,k_n) = k_j$, para cada $(k_1,\ldots,k_n)\in\mathbb{N}^n$, pueden calcularse mediante los sistemas

$$\Pi^{proj}_{n,j} = \big(\Sigma,\Gamma,\Lambda,\#,\mu_{\Pi^{proj}_{n,j}},\mathcal{M}_1,(R_1,\rho_1),im,ext\big)$$

donde

$$\Sigma=\{a_1,\ldots,a_n\} \qquad \Gamma=\{a_1,\ldots,a_n,b,\#\} \qquad \Lambda=\{b\}$$
$$\mu_{\Pi^{proj}_{n,j}} = [_1]_1 \qquad \mathcal{M}_1 = \# \qquad im = 1$$
$$(R_1,\rho_1) = \begin{cases} a_j \to (b,out) \\ \# \to (\#,out) \end{cases}$$

Es fácil ver que $\Pi^{proj}_{n,j}$ es un sistema válido de cálculo de funciones, de orden $(n,1)$ y que calcula la función proyección, Π^n_j.

- Las *funciones identidad*, $Id^n : \mathbb{N}^n \to \mathbb{N}^n$, con $n \geq 1$, definidas por $Id^n(k_1,\ldots,k_n) = (k_1,\ldots,k_n)$, para cada $(k_1,\ldots,k_n)\in\mathbb{N}^n$, pueden

calcularse a través de los sistemas

$$\Pi_n^{Id} = \left(\Sigma, \Gamma, \Lambda, \#, \mu_{\Pi_n^{Id}}, \mathcal{M}_1, (R_1, \rho_1), im, ext\right)$$

donde

$$\Sigma = \{a_1, \ldots, a_n\} \quad \Gamma = \{a_1, \ldots, a_n, b_1, \ldots, b_n, \#\} \quad \Lambda = \{b_1, \ldots, b_n\}$$
$$\mu_{\Pi_n^{Id}} = [\,_1]_1 \qquad \mathcal{M}_1 = \# \qquad im = 1$$
$$(R_1, \rho_1) = \begin{cases} a_i \to (b_i, out) & (1 \leq i \leq n) \\ \# \to (\#, out) \end{cases}$$

Es fácil ver que Π_n^{Id} es un sistema válido de cálculo de funciones, de orden (n, n) y que calcula la función identidad, Id^n.

- Las *funciones constante*, $C_c^n : \mathbb{N}^n \to \mathbb{N}$, con $n \geq 1$ y $c \in \mathbb{N}$, definidas por $C_c^n(k_1, \ldots, k_n) = c$, para cada $(k_1, \ldots, k_n) \in \mathbb{N}^n$, pueden calcularse mediante los sistemas

$$\Pi_{n,c}^{const} = \left(\Sigma, \Gamma, \Lambda, \#, \mu_{\Pi_{n,c}^{const}}, \mathcal{M}_1, (R_1, \rho_1), im, ext\right)$$

donde

$$\Sigma = \{a_1, \ldots, a_n\} \qquad \Gamma = \{a_1, \ldots, a_n, b, \#\} \qquad \Lambda = \{b\}$$
$$\mu_{\Pi_{n,c}^{const}} = [\,_1]_1 \qquad \mathcal{M}_1 = b^c\# \qquad im = 1$$
$$(R_1, \rho_1) = \begin{cases} b \to (b, out) \\ \# \to (\#, out) \end{cases}$$

Es fácil ver que $\Pi_{n,c}^{const}$ es un sistema válido de cálculo de funciones, de orden $(n, 1)$ y que calcula la función constante, C_c^n.

- La función $+' : \mathbb{N}^2 \to \mathbb{N}^2$, definida por $+'(k_1, k_2) = (k_1 + k_2, k_2)$, para cada $k_1, k_2 \in \mathbb{N}$, puede calcularse a través del sistema

$$\Pi_2^{sum'} = \left(\Sigma, \Gamma, \Lambda, \#, \mu_{\Pi_2^{sum'}}, \mathcal{M}_1, (R_1, \rho_1), im, ext\right)$$

donde

$$\Sigma = \{a_1, a_2\} \qquad \Gamma = \{a_1, a_2, b_1, b_2, \#\} \qquad \Lambda = \{b_1, b_2\}$$
$$\mu_{\Pi_2^{sum'}} = [_1]_1 \qquad \mathcal{M}_1 = \# \qquad im = 1$$
$$(R_1, \rho_1) = \begin{cases} a_1 \to (b_1, out) \\ a_2 \to (b_1, out)(b_2, out) \\ \# \to (\#, out) \end{cases}$$

Es fácil ver que $\Pi_2^{sum'}$ es un sistema válido de cálculo de funciones, de orden $(2, 2)$ y que calcula la función $+'$.

La iteración del sistema $\Pi_2^{sum'}$ es un sistema que calcula la función $It(+') : \mathbb{N}^3 \to \mathbb{N}^2$ dada por $It(+')(k_1, k_2, k_3) = (k_1 + k_2 \cdot k_3, k_2)$. Entonces, el sistema $\Pi_2^{prod'} = C(It(\Pi_2^{sum'}); \Pi_{2,0}^{const}, \Pi_{2,2}^{proj}, \Pi_{2,1}^{proj})$ calcula la función $*' : \mathbb{N}^2 \to \mathbb{N}^2$, definida por $*'(k_1, k_2) = (k_1 \cdot k_2, k_2)$, para cada $(k_1, k_2) \in \mathbb{N}^2$.

La iteración del sistema $\Pi_2^{prod'}$ es un sistema que calcula la función $It(*') : \mathbb{N}^3 \to \mathbb{N}^2$ dada por $It(*')(k_1, k_2, k_3) = (k_1 \cdot k_2^{k_3}, k_2)$. Entonces, el sistema $\Pi_2^{expt'} = C(It(\Pi_2^{prod'}); \Pi_{2,1}^{const}, \Pi_2^{Id})$ calcula la función $expt' : \mathbb{N}^2 \to \mathbb{N}^2$, definida por $expt'(k_1, k_2) = (k_1^{k_2}, k_1)$, para cada $(k_1, k_2) \in \mathbb{N}^2$.

- Las *funciones suma n-aria, producto n-ario y exponencial* pueden calcularse con los sistemas definidos por recursión sobre $n \geq 2$ como sigue:

$(n = 2)$

$$\Pi_2^{sum} = C(\Pi_{2,1}^{proj}; \Pi_2^{sum'})$$
$$\Pi_2^{prod} = C(\Pi_{2,1}^{proj}; \Pi_2^{prod'})$$
$$\Pi_2^{expt} = C(\Pi_{2,1}^{proj}; \Pi_2^{expt'})$$

$(n > 2)$

$$\Pi_n^{sum} = C(\Pi_{n-1}^{sum}; C(\Pi_2^{sum}; \Pi_{n,1}^{proj}, \Pi_{n,2}^{proj}), \Pi_{n,3}^{proj}, \ldots, \Pi_{n,n}^{proj})$$
$$\Pi_n^{prod} = C(\Pi_{n-1}^{prod}; C(\Pi_2^{prod}; \Pi_{n,1}^{proj}, \Pi_{n,2}^{proj}), \Pi_{n,3}^{proj}, \ldots, \Pi_{n,n}^{proj})$$

- La función $dif : \mathbb{N}^2 \to \mathbb{N}^2$, que se define para cada $k_1, k_2 \in \mathbb{N}$ por $dif(k_1, k_2) = (\text{máx}(k_1 - k_2, 0), |\,\text{mín}(k_1 - k_2, 0)|)$, puede calcularse mediante el sistema

$$\Pi_2^{dif} = \left(\Sigma, \Gamma, \Lambda, \#, \mu_{\Pi_2^{dif}}, \mathcal{M}_1, (R_1, \rho_1), im, ext\right)$$

donde

$$\Sigma = \{a_1, a_2\} \qquad \Gamma = \{a_1, a_2, b^+, b^-, \#\} \qquad \Lambda = \{b^+, b^-\}$$
$$\mu_{\Pi_2^{dif}} = [\,_1]_1 \qquad \mathcal{M}_1 = \# \qquad im = 1$$
$$(R_1, \rho_1) = a_1 a_2 \to \lambda > \begin{cases} a_1 \to (b^+, out) \\ a_2 \to (b^-, out) \\ \# \to (\#, out) \end{cases}$$

Es fácil ver que Π_2^{dif} es un sistema válido de cálculo de funciones, de orden $(2, 2)$ y que calcula la función dif.

7.2.2. Cálculo de un polinomio.

Dado un polinomio $P(\vec{a}, \vec{x}) \in \mathbb{Z}[\vec{a}, \vec{x}]$ (en donde $\vec{a} = (a_1, \ldots, a_m)$ y $\vec{x} = (x_1, \ldots, x_n)$), construimos un sistema P de cálculo de funciones que calcula dicho polinomio considerado como una función sobre $\vec{a}$ y $\vec{x}$.

- *Cálculo de un monomio:*

  Denotemos por $\Pi_{j,k}^{expt_i} = C(\Pi_2^{expt}; \Pi_{j,k}^{proj}, \Pi_{j,i}^{const})$, con $1 \leq k \leq j$, $i \geq 0$, un sistema que calcula la función $expt_i^{j,k} : \mathbb{N}^j \to \mathbb{N}$ definida como $expt_i^{j,k}(a_1, \ldots, a_j) = a_k^i$, para cada $(a_1, \ldots, a_j) \in \mathbb{N}^j$.

  Sea $m(\vec{a}, \vec{x}) = c\, a_1^{i_1} \ldots a_m^{i_m} x_1^{j_1} \ldots x_n^{j_n}$, con $c > 0$, un monomio de $P(\vec{a}, \vec{x})$. Entonces, el siguiente sistema

$$\Pi_{c,i_1,\ldots,i_m,j_1,\ldots,j_n}^{mon} = C(\Pi_{m+n+1}^{prod}; \Pi_{m+n,c}^{const}, \Pi_{m+n,1}^{expt_{i_1}}, \ldots, \Pi_{m+n,m}^{expt_{i_m}}, \Pi_{m+n,m+1}^{expt_{j_1}}, \ldots, \Pi_{m+n,m+n}^{expt_{j_n}})$$

  calcula el monomio $m(\vec{a}, \vec{x})$, considerado como función de $\mathbb{N}^{m+n}$ en $\mathbb{N}$.

- *Cálculo de un polinomio:*

  Supongamos que

$$P(\vec{a},\vec{x}) = \sum_{k=1}^{r_1} c_k a_1^{i_1^k} \cdots a_m^{i_m^k} x_1^{j_1^k} \cdots x_n^{j_n^k} - \sum_{k=r_1+1}^{r_1+r_2} c_k a_1^{i_1^k} \cdots a_m^{i_m^k} x_1^{j_1^k} \cdots x_n^{j_n^k}$$

  con $c_k > 0$ para cada $k = 1, \ldots, r_1 + r_2$. Entonces, el siguiente sistema

$$\Pi^{pol}_{P(\vec{a},\vec{x})} = C\big(\Pi_2^{dif}; \Pi^{pol+}_{P(\vec{a},\vec{x})}, \Pi^{pol-}_{P(\vec{a},\vec{x})}\big)$$

  donde

$$\Pi^{pol+}_{P(\vec{a},\vec{x})} = C\big(\Pi^{sum}_{r_1}; \Pi^{mon}_{c_1,\vec{i}^1,\vec{j}^1}, \ldots, \Pi^{mon}_{c_{r_1},\vec{i}^{r_1},\vec{j}^{r_1}}\big)$$
$$\Pi^{pol-}_{P(\vec{a},\vec{x})} = C\big(\Pi^{sum}_{r_2}; \Pi^{mon}_{c_{r_1+1},\vec{i}^{r_1+1},\vec{j}^{r_1+1}}, \ldots, \Pi^{mon}_{c_{r_1+r_2},\vec{i}^{r_1+r_2},\vec{j}^{r_1+r_2}}\big)$$

  calcula el polinomio $P(\vec{a},\vec{x})$, considerado como una función de $\mathbb{N}^{m+n}$ en $\mathbb{N}$ (en realidad, calcula $C(dif; P)$, como función de $\mathbb{N}^{m+n}$ en $\mathbb{N}^2$).

7.2.3. Generación de un conjunto diofántico.

Considerando que los alfabetos de entrada y de salida de $\Pi^{pol}_{P(\vec{a},\vec{x})}$ sean $\Sigma^{pol}_{P(\vec{a},\vec{x})} = \{d_1, \ldots, d_m, d_{m+1}, \ldots, d_{m+n}\}$ y $\Lambda^{pol}_{P(\vec{a},\vec{x})} = \{b^+, b^-\}$, respectivamente, definimos el siguiente sistema P de generación de conjuntos:

$$\Pi = \big(\Gamma, \Lambda, \#, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, (R_1, \rho_1), \ldots, (R_p, \rho_p), ext\big)$$

donde

- $\Lambda = \{e_1, \ldots, e_m\}$.
- Existen objetos distinguidos $\#', \#_1^{\vec{a}}, \ldots, \#_m^{\vec{a}}, \#_1^{\vec{x}}, \ldots, \#_n^{\vec{x}}$ pertenecientes a $\Gamma \setminus \{\#, \#^{pol}_{P(\vec{a},\vec{x})}\}$.
- $\mu_\Pi = [_1[_2]_2 \cdots [_{m+1}]_{m+1}[_{m+2}]_{m+2} \cdots [_{m+n+1}]_{m+n+1}\mu_{\Pi^{pol}_{P(\vec{a},\vec{x})}}]_1$, donde las membranas de $\mu_{\Pi^{pol}_{P(\vec{a},\vec{x})}}$ se han renombrado adecuadamente (y, por consiguiente, también se han adaptado las reglas de $\Pi^{pol}_{P(\vec{a},\vec{x})}$). Denotamos por σ_{pol} la membrana piel de esta última estructura de membranas.
- $p = p^{pol}_{P(\vec{a},\vec{x})} + m + n + 1$.

- $\mathcal{M}_2 = \#_1^{\vec{a}}, \ldots, \mathcal{M}_{m+1} = \#_m^{\vec{a}}, \mathcal{M}_{m+2} = \#_1^{\vec{x}}, \ldots, \mathcal{M}_{m+n+1} = \#_n^{\vec{x}}$ y $\mathcal{M}_1 = \#$. Todos los demás multiconjuntos son vacíos.

- Reglas de evolución y prioridades:

  - Reglas para la membrana 1:

$$\#_1^{\vec{a}} \ldots \#_m^{\vec{a}} \#_1^{\vec{x}} \ldots \#_n^{\vec{x}} \# \to \#' > \# \to \# > \left\{ \begin{array}{ll} d_i \to e_i(d_i, in_{\sigma_{pol}}) & (i=1,\ldots,m) \\ d_j \to (d_j, in_{\sigma_{pol}}) & (j=m+1,\ldots,m+n) \\ \#' \to (\#_{P(\vec{a},\vec{x})}^{pol}, in_{\sigma_{pol}})(\ominus, in_{\sigma_{pol}}) & \end{array} \right.$$

$$\left. \begin{array}{l} b^+ \to b^+ \\ b^- \to b^- \end{array} \right\} > \left\{ \begin{array}{ll} e_i \to (e_i, out) & (i=1,\ldots,m) \\ \#_{P(\vec{a},\vec{x})}^{pol} \to (\#, out) & \end{array} \right.$$

  - Reglas para la membrana i, con $2 \leq i \leq m+1$:

$$\#_{i-1}^{\vec{a}} \to (\#_{i-1}^{\vec{a}}, out)$$
$$\#_{i-1}^{\vec{a}} \to \#_{i-1}^{\vec{a}}(d_{i-1}, out)$$

  - Reglas para la membrana i, con $m+2 \leq i \leq m+n+1$:

$$\#_{i-(m+1)}^{\vec{x}} \to (\#_{i-(m+1)}^{\vec{x}}, out)$$
$$\#_{i-(m+1)}^{\vec{x}} \to \#_{i-(m+1)}^{\vec{x}}(d_{i-1}, out)$$

  - Las reglas para las membranas restantes son las mismas que en el sistema $\Pi_{P(\vec{a},\vec{x})}^{pol}$.

Entonces Π es un sistema P de generación de conjuntos que genera el conjunto diofántico $A \subseteq \mathbb{N}^m$ representado por el polinomio $P(\vec{a}, \vec{x}) \in \mathbb{Z}[\vec{a}, \vec{x}]$.

En efecto, el sistema Π funciona como sigue:

1. En primer lugar, se recogen en la membrana 1 una tupla $\vec{a}$ y una tupla $\vec{x}$, generadas de forma no determinista en las membranas $2, \ldots, m+1$ y en las membranas $m+2, \ldots, m+n+1$, respectivamente.

2. En segundo lugar, los objetos d_i que representan a las tuplas anteriores en la membrana 1 se envían a la membrana piel de $\mu_{\Pi_{P(\vec{a},\vec{x})}^{pol}}$ (usando

objetos e_i, nos quedamos con una copia de aquellos que representan a la tupla $\vec{a}$) iniciándose el cálculo de P sobre las tuplas $\vec{a}$ y $\vec{x}$.

3. Finalmente, si el resultado obtenido no es cero, entonces la computación entra en un bucle infinito: se aplica la regla $b^+ \to b^+$, o la regla $b^- \to b^-$, una y otra vez. Si el resultado obtenido es cero, entonces estas reglas no se pueden aplicar, enviándose entonces al entorno externo los objetos e_i conservados en la membrana 1 y objetos $\#$.

Se puede probar entonces el siguiente resultado, a partir de todo lo descrito anteriormente.

Teorema 7.3. *Sea $A \subseteq \mathbb{N}^n$ un conjunto recursivamente enumerable. Entonces existe $\Pi_A \in \mathcal{SG}(Pri, Coo, nDis)$ que genera A.*

De donde se deduce la completitud computacional de los sistemas P de generación de conjuntos, que usan prioridad y cooperación.

Parte II

Complejidad computacional

Capítulo 8

Una Teoría de la Complejidad en sistemas P

En este capítulo vamos a introducir clases de problemas de decisión que son resolubles, en un sentido concreto, por sistemas de computación celular en tiempo polinomial. Asímismo, probaremos que dichas clases son cerradas bajo reducibilidad en tiempo polinomial, propiedad que es natural exigir a una clase de complejidad, y estableceremos las relaciones entre ellas.

En la primera sección de este capítulo se introducen ciertas nociones previas, como son los conceptos de sistema reconocedor, de familias uniformes de sistemas y de sistema confluente. En la sección 8.2 definimos tres clases de complejidad en tiempo polinomial, para familias de sistemas P sin membrana de entrada, para sistemas P con membrana de entrada y para familias de sistemas P con membrana de entrada, respectivamente, siguiendo las ideas introducidas en [35] y desarrolladas en [43].

8.1. Definiciones previas

Comenzamos fijando qué entendemos, en general, por sistema reconocedor, en los cuales dividimos las computaciones de parada en computaciones de aceptación o de rechazo.

Definición 8.1. *Un* sistema P reconocedor *es un par* (Π, θ_Π) *tal que* Π *es un sistema P y* θ_Π *es una función total booleana sobre el conjunto de todas las computaciones de parada de* Π.

Sea $\mathcal{C}$ una computación de parada de Π. Si $\theta_\Pi(\mathcal{C}) = 1$, entonces diremos que $\mathcal{C}$ es una computación de aceptación. Si $\theta_\Pi(\mathcal{C}) = 0$, entonces diremos que $\mathcal{C}$ es una computación de rechazo.

Habitualmente identificamos el sistema P de decisión (Π, θ_Π) con el sistema Π, ya que la función θ_Π suele venir dada implícitamente por la semántica de este último.

Obsérvese que los sistemas P de aceptación de lenguajes (que sean sistemas válidos) introducidos en el capítulo 4 pueden considerarse como sistemas reconocedores, ya que las computaciones de aceptación serían aquellas computaciones de parada con salida igual a *Yes* y las computaciones de rechazo serían aquellas computaciones de parada con salida igual a *No*.

Dada ahora una familia de sistemas de computación celular, los conceptos de consistencia respecto a una clase de sistemas y de uniformidad por máquinas de Turing resultarán útiles más adelante.

Definición 8.2. *Sea $\mathbf{\Pi} = (\Pi(i))_{i\in I}$ una familia de sistemas de computación con membranas. Entonces,*

a) Diremos que la familia $\mathbf{\Pi}$ es consistente *respecto de una clase de sistemas reconocedores, $\mathcal{D}$, si para todo $i \in I$ se tiene que $\Pi(i) \in \mathcal{D}$.*

b) Diremos que la familia $\mathbf{\Pi}$ es uniforme, *por máquinas de Turing, si existe una tal máquina que, dado $i \in I$, construye el sistema $\Pi(i)$. Si además dicha construcción se realiza en tiempo polinomial en el tamaño de i, entonces diremos que la familia $\mathbf{\Pi}$ es* polinomialmente uniforme, *por máquinas de Turing.*

Los sistemas con entrada que vamos a considerar a la hora de definir las clases de complejidad seguirán la misma pauta que los sistemas P de transición con salida externa introducidos en el capítulo 3. Es decir, un tal sistema, Π, dispondrá de una colección de elementos de entrada, que notaremos por I_Π, de forma que al inicio de cada computación del sistema uno de esos elementos se habrá añadido al conjunto de objetos contenidos en la membrana de entrada.

Además, exigimos que los sistemas utilizados para determinar si un problema de decisión pertenece o no a una clase de complejidad sean confluentes, en el sentido que explicitamos a continuación.

Definición 8.3. *Diremos que un sistema de decisión,* (Π, θ_Π), *es* confluente, *si verifica las siguientes propiedades:*

1. *Toda computación de* Π *es de parada.*
2. *Si* $\mathcal{C}_1$ *y* $\mathcal{C}_2$ *son dos computaciones de* Π *con la misma configuración inicial, entonces* $\theta_\Pi(\mathcal{C}_1) = \theta_\Pi(\mathcal{C}_2)$.

Es decir, un sistema de decisión es confluente si, o bien todas sus computaciones con idéntica configuración inicial son de aceptación, o bien todas son de rechazo. Obsérvese que para que un sistema determinista sea confluente basta que todas sus computaciones sean de parada.

En lo que sigue supondremos que $\mathcal{D}$ denota una clase genérica de sistemas P reconocedores, tales que dichos sistemas son confluentes, $g : \mathbb{N}^+ \to \mathbb{N}^+$ una función total computable y $X = (I_X, \theta_X)$ un problema de decisión.

8.2. Clases de complejidad en sistemas P

Los primeros resultados sobre «resolución» de problemas **NP**-completos en tiempo polinomial (incluso lineal) por medio de sistemas de computación celular con membranas se han obtenido usando variantes de sistemas P que carecían de membrana de entrada. Por ello, en las pruebas de dichos resultados lo que realmente se consigue es construir, para *cada instancia* del problema, un sistema que la decide en tiempo polinomial. Esta apreciación nos lleva a la siguiente definición.

Definición 8.4. *Diremos que* $X \in MC_{\mathcal{D}}^{f}(g)$ *si existe una familia de sistemas de computación celular con membranas,* $\mathbf{\Pi} = \left(\Pi(w)\right)_{w \in I_X}$, *verificando las siguientes propiedades:*

1. *La familia* $\mathbf{\Pi}$ *es consistente, respecto de la clase* $\mathcal{D}$.
2. *Para cada* $w \in I_X$, *el sistema* $\Pi(w)$ *no tiene membrana de entrada.*
3. *La familia* $\mathbf{\Pi}$ *es polinomialmente uniforme, por máquinas de Turing.*
4. *La familia* $\mathbf{\Pi}$ *está acotada, respecto del problema* X *y la función* g; *es decir, para cada* $w \in I_X$ *se tiene que toda computación del sistema* $\Pi(w)$ *es de parada y, además, realiza a lo sumo* $g(|w|)$ *pasos.*

5. *La familia* $\mathbf{\Pi}$ *es adecuada, respecto del problema* X*; es decir, para cada* $w \in I_X$ *se tiene que si* $\theta_X(w) = 1$*, entonces toda computación del sistema* $\Pi(w)$ *es de aceptación.*

6. *La familia* $\mathbf{\Pi}$ *es completa, respecto del problema* X*; es decir, para cada* $w \in I_X$ *se tiene que si existe una computación del sistema* $\Pi(w)$ *que es de aceptación, entonces* $\theta_X(w) = 1$.

Si $X \in MC_{\mathcal{D}}^f(g)$, entonces diremos que el problema X es resoluble por una familia de sistemas de computación celular sin membrana de entrada, pertenecientes a la clase $\mathcal{D}$, en tiempo acotado por g.

Si en la definición anterior consideramos como funciones cota las funciones polinomiales, entonces obtenemos la clase de complejidad polinomial para familias de sistemas P sin entrada.

Definición 8.5. *La clase de problemas de decisión resolubles en tiempo polinomial por una familia de sistemas de computación celular sin membrana de entrada, pertenecientes a la clase* $\mathcal{D}$*, es la clase*

$$\mathbf{PMC}_{\mathcal{D}}^f = \bigcup_{k>0} MC_{\mathcal{D}}^f(n^k)$$

En primer lugar, vamos a justificar que esta clase es cerrada bajo reducibilidad en tiempo polinomial.

Proposición 8.6. *Sean* $X = (I_X, \theta_X)$ *e* $Y = (I_Y, \theta_Y)$ *dos problemas de decisión tales que* X *es reducible a* Y *en tiempo polinomial. Supongamos que* $Y \in \mathbf{PMC}_{\mathcal{D}}^f$*. Entonces* $X \in \mathbf{PMC}_{\mathcal{D}}^f$.

Demostración. Puesto que X es reducible a Y en tiempo polinomial, existe una función $h : I_X \to I_Y$, computable en tiempo polinomial, tal que $\theta_X(w) = 1$ si y solo si $\theta_Y(h(w)) = 1$, para todo $w \in I_X$. Por tanto, existe un número natural $k_h > 0$ tal que $|h(w)| \leq |w|^{k_h}$, para todo $w \in I_X$.

Como se tiene, por hipótesis, que $Y \in \mathbf{PMC}_{\mathcal{D}}^f$, entonces existen un número natural $k > 0$ y una familia de sistemas de computación celular con membranas, $\mathbf{\Pi}_Y = \left(\Pi_Y(w')\right)_{w' \in I_Y}$ tales que:

1. La familia $\mathbf{\Pi}_Y$ es consistente, respecto de la clase $\mathcal{D}$.

2. Para cada $w' \in I_Y$, el sistema $\Pi(w')$ no tiene membrana de entrada.

3. La familia $\mathbf{\Pi}_Y$ es polinomialmente uniforme, por máquinas de Turing.

4. La familia $\mathbf{\Pi}_Y$ está acotada, respecto del problema Y y la función n^k.

5. La familia $\mathbf{\Pi}_Y$ es adecuada y completa, respecto del problema Y.

Consideremos la familia $\mathbf{\Pi} = \big(\Pi(w)\big)_{w \in I_X}$, donde $\Pi(w) = \Pi_Y(h(w))$, para todo $w \in I_X$. Entonces se verifica que:

1. La familia $\mathbf{\Pi}$ es polinomialmente uniforme, por máquinas de Turing. En efecto, basta tener en cuenta que la familia $\mathbf{\Pi}_Y$ lo es y que la función h es computable en tiempo polinomial.

2. La familia $\mathbf{\Pi}$ está acotada, respecto del problema X y la función $n^{k_h \cdot k}$. En efecto, sea $w \in I_X$. Como $\Pi(w) = \Pi_Y(h(w))$, toda computación de $\Pi(w)$ es de parada y, además, realiza a lo sumo $|h(w)|^k \leq (|w|^{k_h})^k = |w|^{k_h \cdot k}$ pasos.

3. La familia $\mathbf{\Pi}$ es adecuada, respecto del problema X. En efecto, sea $w \in I_X$ tal que $\theta_X(w) = 1$. Entonces $\theta_Y(h(w)) = 1$. Luego toda computación de $\Pi_Y(h(w)) = \Pi(w)$ es de aceptación.

4. La familia $\mathbf{\Pi}$ es completa, respecto del problema X. En efecto, sea $w \in I_X$ tal que existe una computación de $\Pi(w) = \Pi_Y(h(w))$ que es de aceptación. Entonces, $\theta_Y(h(w)) = 1$. Luego $\theta_X(w) = 1$.

En consecuencia, $X \in \mathbf{PMC}^f_{\mathcal{D}}$. □

En la definición de la clase de complejidad que acabamos de introducir nos hemos visto en la necesidad de emplear familias de sistemas de computación celular ya que, al imponer que dichos sistemas carezcan de membrana de entrada, cada uno de ellos solamente decide una instancia del problema.

Por otra parte, si consideramos sistemas que sí tengan membrana de entrada, entonces es posible definir una clase de complejidad para sistemas P en cierta forma parecida a las clases de complejidad «clásicas» para máquinas de Turing.

Definición 8.7. *Diremos que $X \in MC^i_{\mathcal{D}}(g)$ si existe un sistema de computación celular con membranas, Π, verificando las siguientes propiedades:*

1. *El sistema* Π *pertenece a la clase* $\mathcal{D}$.

2. *El sistema* Π *tiene membrana de entrada.*

3. *Existe una función,* $cod : I_X \to I_\Pi$, *computable en tiempo polinomial, tal que:*

   - *El sistema* Π *está acotado, respecto del problema* X, *la función cod y la función* g; *es decir, para cada* $w \in I_X$, *toda computación del sistema* Π *con entrada* $cod(w)$ *es de parada y, además, realiza a lo sumo* $g(|w|)$ *pasos.*
   - *El sistema* Π *es adecuado, respecto del problema* X *y la función cod; es decir, para cada* $w \in I_X$ *se tiene que si* $\theta_X(w) = 1$, *entonces toda computación del sistema* Π *con entrada* $cod(w)$ *es de aceptación.*
   - *El sistema* Π *es completo, respecto del problema* X *y la función cod; es decir, para cada* $w \in I_X$ *se tiene que si existe una computación del sistema* Π *con entrada* $cod(w)$ *que es de aceptación, entonces* $\theta_X(w) = 1$.

Si $X \in MC^i_{\mathcal{D}}(g)$, entonces diremos que el problema X es resoluble por un sistema de computación celular con membrana de entrada, perteneciente a la clase $\mathcal{D}$, en tiempo acotado por g.

Fijando como cotas las funciones polinomiales obtenemos la clase de complejidad polinomial para sistemas P con entrada.

Definición 8.8. *La clase de problemas de decisión resolubles en tiempo polinomial por un sistema de computación celular con membrana de entrada, perteneciente a la clase* $\mathcal{D}$, *es la clase*

$$\mathbf{PMC}^i_{\mathcal{D}} = \bigcup_{k>0} MC^i_{\mathcal{D}}(n^k)$$

Veamos seguidamente que esta clase también es cerrada bajo reducibilidad en tiempo polinomial.

Proposición 8.9. *Sean* $X = (I_X, \theta_X)$ *e* $Y = (I_Y, \theta_Y)$ *dos problemas de decisión tales que* X *es reducible a* Y *en tiempo polinomial. Supongamos que* $Y \in \mathbf{PMC}^i_{\mathcal{D}}$. *Entonces* $X \in \mathbf{PMC}^i_{\mathcal{D}}$.

Demostración. Puesto que X es reducible a Y en tiempo polinomial, existe una función $h : I_X \to I_Y$, computable en tiempo polinomial, tal que $\theta_X(w) = 1$ si y solo si $\theta_Y(h(w)) = 1$, para todo $w \in I_X$. Por tanto, existe un número natural $k_h > 0$ tal que $|h(w)| \leq |w|^{k_h}$, para todo $w \in I_X$.

Como, por hipótesis, $Y \in \mathbf{PMC}^i_{\mathcal{D}}$, existen un número natural $k > 0$ y un sistema de computación celular con membranas, Π, tales que:

1. El sistema Π pertenece a la clase $\mathcal{D}$.

2. El sistema Π tiene membrana de entrada.

3. Existe una función, $cod_Y : I_Y \to I_\Pi$, computable en tiempo polinomial, tal que:

   - El sistema Π está acotado, respecto del problema Y, la función cod_Y y la función n^k.
   - El sistema Π es adecuado y completo, respecto del problema Y y la función cod_Y.

Consideremos la función $cod = C(cod_Y; h) : I_X \to I_\Pi$. Entonces cod es una función computable en tiempo polinomial. Además, se verifican las siguientes propiedades:

- El sistema Π está acotado, respecto del problema X, la función cod y la función $n^{k_h \cdot k}$. En efecto, sea $w \in I_X$. Entonces toda computación de Π con entrada $cod(w) = cod_Y(h(w))$ es de parada y, además, realiza a lo sumo $|h(w)|^k \leq (|w|^{k_h})^k = |w|^{k_h \cdot k}$ pasos.

- El sistema Π es adecuado, respecto del problema X y la función cod. En efecto, sea $w \in I_X$ tal que $\theta_X(w) = 1$. Entonces $\theta_Y(h(w)) = 1$. Luego toda computación de Π con entrada $cod_Y(h(w)) = cod(w)$ es de aceptación.

- El sistema Π es completo, respecto del problema X y la función cod. En efecto, sea $w \in I_X$ tal que existe una computación de Π con entrada $cod(w) = cod_Y(h(w))$ que es de aceptación. Entonces, $\theta_Y(h(w)) = 1$. Luego $\theta_X(w) = 1$.

En consecuencia, $X \in \mathbf{PMC}^i_{\mathcal{D}}$. □

Conviene observar que, si bien la definición de clase de complejidad para sistemas con entrada que acabamos de dar es bastante natural, en la práctica se suele diseñar, más que un solo sistema que resuelve un problema, una familia tal que cada elemento decide las instancias de «tamaño equivalente», en cierto sentido. Esto nos lleva a la definición de la tercera clase que incluimos en esta memoria.

Definición 8.10. *Diremos que* $X \in MC_{\mathcal{D}}^{fi}(g)$ *si existe una familia de sistemas de computación celular con membranas,* $\mathbf{\Pi} = \big(\Pi(n)\big)_{n\in\mathbb{N}^+}$, *verificando las siguientes propiedades:*

1. *La familia* $\mathbf{\Pi}$ *es consistente, respecto de la clase* $\mathcal{D}$.
2. *Para cada* $n \in \mathbb{N}^+$, *el sistema* $\Pi(n)$ *tiene membrana de entrada.*
3. *La familia* $\mathbf{\Pi}$ *es polinomialmente uniforme, por máquinas de Turing.*
4. *Existen dos funciones,* $cod : I_X \to \bigcup_{n\in\mathbb{N}^+} I_{\Pi(n)}$ *y* $s : I_X \to \mathbb{N}^+$, *computables en tiempo polinomial, tales que:*
   - *Para todo* $w \in I_X$, $cod(w) \in I_{\Pi(s(w))}$.
   - *La familia* $\mathbf{\Pi}$ *está acotada, respecto del problema* X, *la función cod, la función* s *y la función* g; *es decir, para cada* $w \in I_X$ *se tiene que toda computación del sistema* $\Pi(s(w))$ *con entrada* $cod(w)$ *es de parada y, además, realiza a lo sumo* $g(|w|)$ *pasos.*
   - *La familia* $\mathbf{\Pi}$ *es adecuada, respecto del problema* X, *la función cod y la función* s; *es decir, para cada* $w \in I_X$ *se tiene que si* $\theta_X(w) = 1$, *entonces toda computación del sistema* $\Pi(s(w))$ *con entrada* $cod(w)$ *es de aceptación.*
   - *La familia* $\mathbf{\Pi}$ *es completa, respecto del problema* X, *la función cod y la función* s; *es decir, para cada* $w \in I_X$ *se tiene que si existe una computación del sistema* $\Pi(s(w))$ *con entrada* $cod(w)$ *que es de aceptación, entonces* $\theta_X(w) = 1$.

Si $X \in MC_{\mathcal{D}}^{f}i(g)$, entonces diremos que el problema X es resoluble por una familia de sistemas de computación celular con membrana de entrada, pertenecientes a la clase $\mathcal{D}$, en tiempo acotado por g. Por otra parte, en las condiciones del teorema anterior diremos que el par (cod, s) es una codificación polinomial de X en $\mathbf{\Pi}$.

Como es usual, la clase de complejidad polinomial se obtiene usando como cotas las funciones polinomiales.

Definición 8.11. *La clase de problemas de decisión resolubles en tiempo polinomial por una familia de sistemas de computación celular con membrana de entrada, pertenecientes a la clase $\mathcal{D}$, es la clase*

$$\mathbf{PMC}_{\mathcal{D}}^{fi} = \bigcup_{k>0} MC_{\mathcal{D}}^{fi}(n^k)$$

Nuevamente justificamos que esta clase de complejidad es cerrada bajo reducibilidad en tiempo polinomial.

Proposición 8.12. *Sean $X = (I_X, \theta_X)$ e $Y = (I_Y, \theta_Y)$ dos problemas de decisión tales que X es reducible a Y en tiempo polinomial. Supongamos que $Y \in \mathbf{PMC}_{\mathcal{D}}^{fi}$. Entonces $X \in \mathbf{PMC}_{\mathcal{D}}^{fi}$.*

Demostración. Puesto que X es reducible a Y en tiempo polinomial, existe una función $h : I_X \to I_Y$, computable en tiempo polinomial, tal que $\theta_X(w) = 1$ si y solo si $\theta_Y(h(w)) = 1$, para todo $w \in I_X$. Por tanto, existe un número natural $k_h > 0$ tal que $|h(w)| \leq |w|^{k_h}$, para todo $w \in I_X$.

Como, por hipótesis, $Y \in \mathbf{PMC}_{\mathcal{D}}^{fi}$, existen un número natural $k > 0$ y una familia de sistemas de computación celular con membranas, $\mathbf{\Pi} = \big(\Pi(n)\big)_{n\in\mathbb{N}^+}$, tales que:

1. La familia $\mathbf{\Pi}$ es consistente, respecto de la clase $\mathcal{D}$.
2. Para cada $n \in \mathbb{N}^+$, el sistema $\Pi(n)$ tiene membrana de entrada.
3. La familia $\mathbf{\Pi}$ es polinomialmente uniforme, por máquinas de Turing.
4. Existen dos funciones, $cod_Y : I_Y \to \bigcup_{n\in\mathbb{N}^+} I_{\Pi(n)}$ y $s_Y : I_Y \to \mathbb{N}^+$, computables en tiempo polinomial, tales que:
   - Para todo $w' \in I_Y$, $cod_Y(w') \in I_{\Pi(s_Y(w'))}$.
   - La familia $\mathbf{\Pi}$ está acotada, respecto del problema Y, la función cod_Y, la función s_Y y la función n^k.
   - La familia $\mathbf{\Pi}$ es adecuada y completa, respecto del problema Y, la función cod_Y y la función s_Y.

Consideremos las funciones $cod = C(cod_Y; h) : I_X \to \bigcup_{n\in\mathbb{N}^+} I_{\Pi(n)}$ y $s = C(s_Y; h)$. Entonces cod y s son computables en tiempo polinomial. Además, se verifican las siguientes propiedades:

- Para todo $w \in I_X$, $cod(w) \in I_{\Pi(s(w))}$. En efecto, para todo $w \in I_X$ se tiene que $cod(w) = cod_Y(h(w)) \in I_{\Pi(s_Y(h(w)))} = I_{\Pi(s(w))}$.

- La familia $\mathbf{\Pi}$ está acotada, respecto del problema X, la función cod, la función s y la función $n^{k_h \cdot k}$. En efecto, sea $w \in I_X$. Entonces toda computación de $\Pi(s(w)) = \Pi(s_Y(h(w)))$ con entrada $cod(w) = cod_Y(h(w))$ es de parada y, además, realiza a lo sumo $|h(w)|^k \leq (|w|^{k_h})^k = |w|^{k_h \cdot k}$ pasos.

- La familia $\mathbf{\Pi}$ es adecuada, respecto del problema X, la función cod y la función s. En efecto, sea $w \in I_X$ tal que $\theta_X(w) = 1$. Entonces $\theta_Y(h(w)) = 1$. Luego toda computación de $\Pi(s_Y(h(w))) = \Pi(s(w))$ con entrada $cod_Y(h(w)) = cod(w)$ es de aceptación.

- La familia $\mathbf{\Pi}$ es completa, respecto del problema X, la función cod y la función s. En efecto, sea $w \in I_X$ tal que existe una computación de $\Pi(s(w)) = \Pi(s_Y(h(w)))$ con entrada $cod(w) = cod_Y(h(w))$ que es de aceptación. Entonces, $\theta_Y(h(w)) = 1$. Luego $\theta_X(w) = 1$.

En consecuencia, $X \in \mathbf{PMC}^{fi}_{\mathcal{D}}$. □

Es fácil ver que se tienen las siguientes relaciones de inclusión entre las clases de complejidad antes definidas.

Proposición 8.13. *Sean $\mathcal{D}$ una clase de sistemas P de decisión con membrana de entrada y $\mathcal{D}'$ la misma clase pero considerando los sistemas sin membrana de entrada.*

a) Si $X \in MC^{fi}_{\mathcal{D}}(g)$, entonces $X \in MC^{f}_{\mathcal{D}'}(g)$.

b) Si $X \in MC^{i}_{\mathcal{D}}(g)$, entonces $X \in MC^{fi}_{\mathcal{D}}(g)$.

Demostración.

a) Supongamos que $X \in MC^{fi}_{\mathcal{D}}(g)$. Entonces existe una familia de sistemas de computación celular con membranas, $\mathbf{\Pi} = \big(\Pi(n)\big)_{n\in\mathbb{N}^+}$, verificando lo siguiente:

1. La familia $\mathbf{\Pi}$ es consistente, respecto de la clase $\mathcal{D}$.
2. Para cada $n \in \mathbb{N}^+$, el sistema $\Pi(n)$ tiene membrana de entrada.
3. La familia $\mathbf{\Pi}$ es polinomialmente uniforme, por máquinas de Turing.
4. Existen dos funciones, $cod : I_X \to \bigcup_{n\in\mathbb{N}^+} I_{\Pi(n)}$ y $s : I_X \to \mathbb{N}^+$, computables en tiempo polinomial, tales que:
   - Para todo $w \in I_X$, $cod(w) \in I_{\Pi(s(w))}$.
   - La familia $\mathbf{\Pi}$ está acotada, respecto del problema X, la función cod, la función s y la función g.
   - La familia $\mathbf{\Pi}$ es adecuada y completa, respecto del problema X, la función cod y la función s.

Consideremos la familia $\mathbf{\Pi}' = \left(\Pi'(w)\right)_{w\in I_X}$ de forma que para cada $w \in I_X$ el sistema $\Pi'(w)$ se obtiene considerando el sistema $\Pi(s(w))$ como un sistema sin membrana de entrada, y en cuya configuración inicial se le ha añadido adicionalmente $cod(w)$ a la membrana de entrada. Entonces es fácil comprobar que dicha familia verifica las condiciones que permite garantizar que $X \in MC^f_{\mathcal{D}'}(g)$.

b) Supongamos que $X \in MC^i_{\mathcal{D}}(g)$. Entonces existe un sistema de computación celular con membranas, Π, verificando las siguientes propiedades:

1. El sistema Π pertenece a la clase $\mathcal{D}$.
2. El sistema Π tiene membrana de entrada.
3. El sistema Π se puede construir en tiempo polinomial, por una máquina de Turing.
4. Existe una función, $cod : I_X \to I_\Pi$, computable en tiempo polinomial, tal que:
   - El sistema Π está acotado, respecto del problema X, la función cod y la función g.
   - El sistema Π es adecuado y completo, respecto del problema X y la función cod.

Consideremos la familia $\mathbf{\Pi} = \left(\Pi(n)\right)_{n\in\mathbb{N}^+}$ de forma que para cada $n \in \mathbb{N}^+$, el sistema $\Pi(n)$ es igual al sistema Π. Entonces es fácil comprobar que dicha familia verifica las condiciones que permiten garantizar que $X \in MC^{fi}_{\mathcal{D}}(g)$.

□

En consecuencia, se tiene que $\mathbf{PMC}^{i}_{\mathcal{D}} \subseteq \mathbf{PMC}^{fi}_{\mathcal{D}} \subseteq \mathbf{PMC}^{f}_{\mathcal{D}}$.

Finalmente, obsérvese que, al considerar sistemas confluentes en las definiciones de las clases de complejidad anteriores, se deduce de manera inmediata que dichas clases son cerradas bajo complementario.

Por otra parte, si no exigimos que los sistemas de la clase $\mathcal{D}$ sean confluentes, y relajamos la noción de adecuación en las definiciones de las clases de complejidad (de forma que si $w \in I_X$ es una instancia del problema X tal que $\theta_X(w) = 1$, entonces *existe una* computación del sistema asociado a w que es de aceptación), obtendríamos otras clases de complejidad, en el sentido no determinista clásico.

Capítulo 9

Sistemas P y el problema P = NP

El objetivo de este capítulo es caracterizar la conjetura $\mathbf{P} \neq \mathbf{NP}$ mediante la irresolubilidad en tiempo polinomial de problemas **NP**–completos a través de la clase $\mathcal{LA}$ de sistemas P de aceptación de lenguajes. La caracterización que se presenta está basada en dos resultados básicos:

(a) Toda máquina de Turing determinista puede ser simulada en tiempo polinomial por una familia de sistemas P de aceptación de lenguajes.

(b) Si un problema de decisión es resoluble en tiempo polinomial por una familia de sistemas P de aceptación de lenguajes, entonces también lo es, en tiempo polinomial, por una máquina de Turing determinista.

Basándonos en el diseño de sistemas P de aceptación que simulan máquinas de Turing deterministas realizada en el capítulo 4, dada una tal máquina construimos en la sección 9.1 una familia de sistemas de aceptación cada uno de los cuales simula, en tiempo polinomial, dicha máquina de Turing sobre cadenas de entrada que tengan el mismo tamaño. En la sección 9.2, siguiendo las ideas desarrolladas por Claudio Zandron, Claudio Ferretti y Giancarlo Mauri en [60], probamos que todo problema de decisión resoluble en tiempo polinomial por familias de sistemas P de aceptación es resoluble en tiempo polinomial por máquinas de Turing deterministas. Finalmente, en la sección 9.3 caracterizamos, a partir de los resultados anteriores, la conjetura $\mathbf{P} \neq \mathbf{NP}$ en términos de irresolubilidad en tiempo polinomial de problemas de decisión en los sistemas considerados. Además, se presenta una descripción de la clase **P** a través de la clase de complejidad polinomial para familias de sistemas de aceptación de lenguajes.

9.1. Simulación de máquinas de Turing por sistemas P

En este capítulo consideramos las máquinas de Turing como dispositivos de decisión de lenguajes. Es decir, dicha máquina para sobre cualquier cadena sobre el alfabeto de entrada, siendo el estado de parada igual al estado de aceptación, en el caso en que la cadena pertenezca al lenguaje decidido, y siendo dicho estado igual al estado de rechazo, en el caso en que la cadena no pertenezca a ese lenguaje.

Comenzamos observando que es posible asociar a una máquina de Turing un problema de decisión, lo que nos permitirá establecer con precisión qué queremos decir cuando hablamos de que esa máquina es simulada por sistemas P.

Definición 9.1. *Sea TM una máquina de Turing, como dispositivo de decisión de lenguajes, y con alfabeto de entrada Σ_{TM}. El* problema de decisión asociado a TM *es el problema $X_{TM} = (I_{TM}, \theta_{TM})$, donde:*

- *El conjunto de instancias del problema, I_{TM}, es el lenguaje Σ_{TM}^*.*
- *Dada una cadena $w \in \Sigma_{TM}^*$, la función θ_{TM} sobre w vale 1 si y solo si TM acepta w.*

Entonces tenemos tres posibilidades para simular una máquina de Turing por sistemas P.

Definición 9.2. *Sea $\mathcal{D}$ una clase de sistemas P reconocedores sin membrana de entrada. Diremos que una máquina de Turing, TM, es simulada en tiempo polinomial por una familia de sistemas en la clase $\mathcal{D}$, si $X_{TM} \in \mathbf{PMC}_{\mathcal{D}}^{f}$.*

Definición 9.3. *Sea $\mathcal{D}$ una clase de sistemas P reconocedores con membrana de entrada. Diremos que una máquina de Turing, TM, es simulada en tiempo polinomial por un sistema en la clase $\mathcal{D}$, si $X_{TM} \in \mathbf{PMC}_{\mathcal{D}}^{i}$.*

Definición 9.4. *Sea $\mathcal{D}$ una clase de sistemas P reconocedores con membrana de entrada. Diremos que una máquina de Turing, TM, es simulada en tiempo polinomial por una familia de sistemas en la clase $\mathcal{D}$, si $X_{TM} \in \mathbf{PMC}_{\mathcal{D}}^{fi}$.*

Obsérvese que todo sistema P de transición con salida externa, como dispositivo de aceptación de lenguajes, puede ser considerado como un sistema

reconocedor, donde la función que determina si una computación de parada es de aceptación o de rechazo viene dada directamente por la propia semántica del sistema. Recordemos que notamos por $\mathcal{LA}$ la clase de sistemas de ese tipo tales que sean sistemas válidos (es decir, tales que la parada de una computación queda determinada por la expulsión al entorno externo de un objeto # y tales que ninguna computación de parada proporciona una salida ambigüa).

A continuación vamos a probar que toda máquina de Turing determinista puede ser simulada en tiempo polinomial por una familia de sistemas de la clase $\mathcal{LA}$.

Proposición 9.5. *Sea TM una máquina de Turing determinista que trabaja en tiempo polinomial. Entonces $X_{TM} \in \mathbf{PMC}^{fi}_{\mathcal{LA}}$.*

Demostración. En el capítulo 4 vimos cómo los sistemas P de aceptación de lenguajes eran capaces de reproducir el funcionamiento de las máquinas de Turing. Lamentablemente, la función que codifica, en dichos sistemas, las cadenas de entrada por multiconjuntos de objetos no es computable *en tiempo polinomial.* Esto nos obliga, para simular una máquina de Turing, TM, en tiempo polinomial, a considerar una familia de dichos sistemas, de manera que cada uno de ellos simule el comportamiento de TM sobre las cadenas de una determinada longitud. De esta forma podemos conseguir que la función de codificación sí sea computable en tiempo polinomial (es más, lo será en tiempo lineal), para que el sistema se encargue luego de transformar, también en tiempo polinomial (de nuevo, en realidad será en tiempo lineal), esta codificación a la codificación exponencial original.

Supongamos que el conjunto de estados de la máquina de Turing es $Q_{TM} = \{q_N, q_Y, q_0, \ldots, q_n\}$, el alfabeto de trabajo $\Gamma_{TM} = \{B, \triangleright, a_1, \ldots, a_m\}$, el alfabeto de entrada $\Sigma_{TM} = \{a_1, \ldots, a_p\}$, con $p \leq m$, y la función de transición $\delta_{TM}(q_i, a_j) = (q_{Q(i,j)}, a_{A(i,j)}, D(i,j))$. Recuérdese que denotamos $a_B = B$ (símbolo blanco) y $a_0 = \triangleright$ (símbolo situado en la celda de la cinta situada más a la izquierda).

Describimos a continuación la familia $\mathbf{\Pi}_{TM} = \left(\Pi_{TM}(k)\right)_{k \in \mathbb{N}^+}$ de sistemas P de aceptación de lenguajes que va a simular la máquina de Turing TM. Para cada $k \in \mathbb{N}^+$,

$$\Pi_{TM}(k) = \left(\Sigma(k), \Gamma(k), \Lambda, \#, \mu_{\Pi_{TM}(k)}, \mathcal{M}_1(k), (R_1(k), \rho_1(k)), im, ext\right)$$

donde

$$\begin{aligned}
\Sigma(k) &= \{\langle a_i, j\rangle : 1 \le i \le p, 1 \le j \le k\} \\
\Gamma(k) &= \{\langle a_i, j\rangle : 1 \le i \le p, 0 \le j \le k\} \cup \{q_N, q_Y, h, h', s_E, \#\} \cup \\
&\quad \{q_i : 0 \le i \le n\} \cup \{a_i, a_i', a_i'', b_i : 0 \le i \le m\} \cup \\
&\quad \{s_l, s_l^-, s_l^+ : l \in \{0, I_1, \ldots, I_5, 1, \ldots, 9\}\} \\
\Lambda &= \{Yes, No\} \\
\mu_{\Pi_{TM}(k)} &= [\,_1]_1 \\
\mathcal{M}_1(k) &= q_0 a_0 s_0 s_0^- s_{I_1}^- \ldots s_{I_5}^- s_1^- \ldots s_9^- \\
im &= 1 \\
(R_1(k), \rho_1(k)) &= (R^0(k), \rho^0(k)) \cup (R^I, \rho^I) \cup (R^1, \rho^1) \cup (R^2, \rho^2) \cup \\
&\quad (R^3, \rho^3) \cup (R^4, \rho^4)
\end{aligned}$$

y los conjuntos de reglas son

$(R^0(k), \rho^0(k)) = (R^{0_1}(k), \rho^{0_1}(k))$, con

$$(R^{0_1}(k),\rho^{0_1}(k)) \equiv \left\{ \begin{array}{l} s_E s_0^- \to (\#, out) > s_0 s_0^- \to s_0^+ > s_0^- \to s_0^- > \\ \quad \left\{ \begin{array}{ll} \langle a_i, j\rangle \to \langle a_i, j-1\rangle^2 & (1 \le i \le p, 1 \le j \le k) \\ \langle a_i, 0\rangle \to a_i & (1 \le i \le p) \end{array} \right\} > \\ \quad s_0^+ \to s_0^- s_{I_1} \end{array} \right.$$

$(R^I, \rho^I) = (R^{I_1}, \rho^{I_1}) \cup (R^{I_2}, \rho^{I_2}) \cup (R^{I_3}, \rho^{I_3}) \cup (R^{I_4}, \rho^{I_4}) \cup (R^{I_5}, \rho^{I_5})$, con

$$(R^{I_1},\rho^{I_1}) \equiv \left\{ \begin{array}{l} s_E s_{I_1}^- \to (\#, out) > s_{I_1} s_{I_1}^- \to s_{I_1}^+ > s_{I_1}^- \to s_{I_1}^- > \\ \quad a_i s_{I_1}^+ \to a_i s_{I_1}^- s_{I_2} \quad (1 \le i \le p) > s_{I_1}^+ \to s_{I_1}^- s_1 \end{array} \right.$$

$$(R^{I_2},\rho^{I_2}) \equiv \left\{ \begin{array}{l} s_E s_{I_2}^- \to (\#, out) > s_{I_2} s_{I_2}^- \to s_{I_2}^+ > s_{I_2}^- \to s_{I_2}^- > \\ \quad \left\{ \begin{array}{ll} a_i \to a_i a_i' & (1 \le i \le p) \\ s_{I_2}^+ \to s_{I_2}^- s_{I_3} \end{array} \right. \end{array} \right.$$

$$(R^{I_3},\rho^{I_3}) \equiv \left\{ \begin{array}{l} s_E s_{I_3}^- \to (\#, out) > s_{I_3} s_{I_3}^- \to s_{I_3}^+ > s_{I_3}^- \to s_{I_3}^- > \\ \quad a_i'^2 \to a_i'' \quad (1 \le i \le p) > a_i' s_{I_3}^+ \to a_i' s_{I_3}^- \quad (1 \le i \le p) > \\ \quad \left\{ \begin{array}{l} a_i'' \to a_i' \quad (1 \le i \le p) \\ s_{I_3}^+ \to s_{I_3}^- s_{I_4} \end{array} \right. \end{array} \right.$$

$$(R^{I_4},\rho^{I_4}) \equiv \left\{ \begin{array}{l} s_E s_{I_4}^- \to (\#, out) > s_{I_4} s_{I_4}^- \to s_{I_4}^+ > s_{I_4}^- \to s_{I_4}^- > \\ \quad a_i'^2 \to a_i'' \quad (1 \le i \le p) > a_i' a_j' s_{I_4}^+ \to a_i' a_j' s_{I_4}^- \quad (1 \le i,j \le p, i \ne j) > \\ \quad a_i' s_{I_4}^+ \to s_{I_4}^- s_{I_5} \quad (1 \le i \le p) > s_{I_4}^+ \to s_{I_4}^- \end{array} \right.$$

$$(R^{I_5},\rho^{I_5}) \equiv \left\{ \begin{array}{l} s_E s_{I_5}^- \to (\#, out) > s_{I_5} s_{I_5}^- \to s_{I_5}^+ > s_{I_5}^- \to s_{I_5}^- > \\ \quad a_i'' \to a_i' \quad (1 \le i \le p) > a_i' s_{I_5}^+ \to a_i' s_{I_5}^- s_{I_4} \quad (1 \le i \le p) > \\ \quad s_{I_5}^+ \to s_{I_5}^- s_1 \end{array} \right.$$

$(R^1,\rho^1) = (R^{1_1},\rho^{1_1}) \cup (R^{1_2},\rho^{1_2}) \cup (R^{1_3},\rho^{1_3})$, con

$$(R^{1_1},\rho^{1_1}) \equiv \left\{ \begin{array}{l} s_E s_1^- \to (\#, out) > s_1 s_1^- \to s_1^+ > s_1^- \to s_1^- > \\ \quad \left\{ \begin{array}{l} h \to hh' \\ a_i \to a_i a_i' \quad (0 \le i \le m) \\ s_1^+ \to s_1^- s_2 \end{array} \right. \end{array} \right.$$

$$(R^{1_2},\rho^{1_2}) \equiv \left\{ \begin{array}{l} s_E s_2^- \to (\#, out) > h' s_2 s_2^- \to s_2^+ > s_2 \to s_3 > \\ \quad s_2^- \to s_2^- > a_i'^2 \to a_i'' \quad (0 \le i \le m) > \left\{ \begin{array}{l} a_i' \to \lambda \quad (0 \le i \le m) \\ a_i'' \to a_i' \quad (0 \le i \le m) \\ s_2^+ \to s_2^- s_2 \end{array} \right. \end{array} \right.$$

$$(R^{1_3},\rho^{1_3}) \equiv \left\{ \begin{array}{l} s_E s_3^- \to (\#, out) > s_3 s_3^- \to s_3^+ > s_3^- \to s_3^- > \\ \quad \left\{ \begin{array}{l} a_i'^2 \to \lambda \quad (0 \le i \le m) \\ s_3^+ \to s_3^- s_4 \end{array} \right. \end{array} \right.$$

$(R^2, \rho^2) = (R^{2_1}, \rho^{2_1}) \cup (R^{2_2}, \rho^{2_2})$, con

$$(R^{2_1},\rho^{2_1}) \equiv \begin{cases} s_E s_4^- \to (\#, out) > s_4 s_4^- \to s_4^+ > s_4^- \to s_4^- > \\ \quad \begin{cases} h \to hh' \\ s_4^+ \to s_4^- s_5 \end{cases} \end{cases}$$

$$(R^{2_2},\rho^{2_2}) \equiv \begin{cases} s_E s_5^- \to (\#, out) > s_5 s_5^- \to s_5^+ > s_5^- \to s_5^- > \\ \quad \begin{cases} \text{Reglas para la función de transición} \\ s_5^+ \to s_5^- s_6 \end{cases} \end{cases}$$

Las reglas para la función de transición, δ_{TM}, son las siguientes:

Caso 1: Estado q_r, símbolo $a_s \neq B$

Movimiento	*Reglas*	
izquierda	$q_r a'_s h \to q_{Q(r,s)} a'_s b_{A(r,s)}$,	si $A(r,s) \neq B$
	$q_r a'_s h \to q_{Q(r,s)} a'_s$,	si $A(r,s) = B$
igual	$q_r a'_s \to q_{Q(r,s)} a'_s b_{A(r,s)}$,	si $A(r,s) \neq B$
	$q_r a'_s \to q_{Q(r,s)} a'_s$,	si $A(r,s) = B$
derecha	$q_r a'_s \to q_{Q(r,s)} a'_s b_{A(r,s)} h$,	si $A(r,s) \neq B$
	$q_r a'_s \to q_{Q(r,s)} a'_s h$,	si $A(r,s) = B$

Caso 2: Estado q_r, símbolo $a_s = B$

Movimiento	*Reglas*	
izquierda	$q_r h \to q_{Q(r,B)} b_{A(r,B)}$,	si $A(r,B) \neq B$
	$q_r h \to q_{Q(r,B)}$,	si $A(r,B) = B$
igual	$q_r \to q_{Q(r,B)} b_{A(r,B)}$,	si $A(r,B) \neq B$
	$q_r \to q_{Q(r,B)}$,	si $A(r,B) = B$
derecha	$q_r \to q_{Q(r,B)} b_{A(r,B)} h$,	si $A(r,B) \neq B$
	$q_r \to q_{Q(r,B)} h$,	si $A(r,B) = B$

Para evitar conflictos, toda regla del caso 1 tiene una prioridad mayor que cada regla del caso 2.

$(R^3, \rho^3) = (R^{3_1}, \rho^{3_1}) \cup (R^{3_2}, \rho^{3_2})$, con

$$(R^{3_1},\rho^{3_1}) \equiv \begin{cases} s_E s_6^- \to (\#, out) > h' s_6 s_6^- \to s_6^+ > s_6 \to s_7 > \\ \quad s_6^- \to s_6^- > \begin{cases} a_i' \to {a_i'}^2 \quad (0 \le i \le m) \\ b_i \to {b_i}^2 \quad (0 \le i \le m) \\ s_6^+ \to s_6^- s_6 \end{cases} \end{cases}$$

$$(R^{3_2},\rho^{3_2}) \equiv \begin{cases} s_E s_7^- \to (\#, out) > s_7 s_7^- \to s_7^+ > s_7^- \to s_7^- > \\ \quad \begin{cases} a_i a_i' \to \lambda \quad (0 \le i \le m) \\ b_i \to a_i \quad (0 \le i \le m) \\ s_7^+ \to s_7^- s_8 \end{cases} \end{cases}$$

$(R^4, \rho^4) = (R^{4_1}, \rho^{4_1}) \cup (R^{4_2}, \rho^{4_2})$, con

$$(R^{4_1},\rho^{4_1}) \equiv \begin{cases} s_E s_8^- \to (\#, out) > s_8 s_8^- \to s_8^+ > s_8^- \to s_8^- > \\ \quad \begin{cases} q_Y \to q_Y s_9 \\ q_N \to q_N s_9 \\ q_i \to q_i s_1 \quad (0 \le i \le n) \\ s_8^+ \to s_8^- \end{cases} \end{cases}$$

$$(R^{4_2},\rho^{4_2}) \equiv \begin{cases} s_E s_9^- \to (\#, out) > s_9 s_9^- \to s_9^+ > s_9^- \to s_9^- > \\ \quad \begin{cases} q_Y \to (Yes, out) \\ q_N \to (No, out) \\ h \to \lambda \\ a_i \to \lambda \quad (0 \le i \le m) \\ s_9^+ \to s_9^- {s_E}^{15} \end{cases} \end{cases}$$

Entonces se verifican las siguientes propiedades:

1. La familia $\mathbf{\Pi}_{TM}$ es consistente, respecto de la clase $\mathcal{LA}$.

   En efecto, obsérvese que las reglas de $R^0(k)$ transforman un multiconjunto de entrada, $\langle a_{i_1}, j_1 \rangle \ldots \langle a_{i_t}, j_t \rangle$, en el multiconjunto $a_{i_1}^{2^{j_1}} \ldots a_{i_t}^{2^{j_t}}$. Una vez efectuada esta operación, el sistema continuará trabajando exactamente igual que los sistemas diseñados en el capítulo 4. En con-

secuencia, teniendo en cuenta la verificación realizada en la sección 4.3 de dicho capítulo, concluimos que $\Pi_{TM}(k) \in \mathcal{LA}$, para todo $k \in \mathbb{N}^+$. Además, también se deduce que estos sistemas son *deterministas*.

2. Para cada $k \in \mathbb{N}^+$, el sistema $\Pi_{TM}(k)$ tiene membrana de entrada.

3. La familia $\mathbf{\Pi}_{TM}$ es polinomialmente uniforme, por máquinas de Turing.

   Obsérvese que la definición dada de los sistemas es uniforme. Basta probar, pues, que cada sistema de la familia $\mathbf{\Pi}_{TM}$ se puede construir en tiempo polinomial, respecto de $k \in \mathbb{N}^+$, y esto es en efecto así, ya que:

   - El tamaño del alfabeto de entrada es $p \cdot k$.
   - El tamaño del alfabeto de trabajo es $p \cdot k + p + n + 4 \cdot m + 56$.
   - El tamaño del alfabeto de salida es 2.
   - El grado del sistema es 1.
   - El tamaño del multiconjunto de objetos asociado inicialmente a la única membrana del sistema, $\mathcal{M}_1(k)$, es 18.
   - El número total de reglas del sistema es lineal en k.
   - La longitud (suma del número de objetos en el antecedente y en el consecuente) máxima de las reglas es 17.

   donde p, n y m son parámetros fijos que solo dependen de la máquina de Turing.

4. Consideremos la función $cod_{TM} : I_{TM} \to \bigcup_{k \in \mathbb{N}^+} I_{\Pi_{TM}(k)}$, dada por $cod(a_{i_1} \ldots a_{i_t}) = \langle a_{i_1}, 1\rangle \ldots \langle a_{i_t}, t\rangle$, y la función $s_{TM} : I_{TM} \to \mathbb{N}^+$, dada por $s(w) = |w|$.

   - Para todo $w \in I_{TM}$, $cod_{TM}(w) \in I_{\Pi_{TM}(s_{TM}(w))}$.
     En efecto, ya que si $s_{TM}(w) = |w| = t$, entonces $\Sigma(s_{TM}(w)) = \{\langle a_i, j\rangle : 1 \leq i \leq p, 1 \leq j \leq t\}$.
   - La familia $\mathbf{\Pi}_{TM}$ está acotada polinomialmente, respecto del problema X_{TM}, la función cod_{TM} y la función s_{TM}.
     Sea $u \in \mathbb{N}^+$ tal que la máquina de Turing TM, al recibir como entrada una cadena $w \in \Sigma^*_{TM}$, realiza un número de pasos del

orden $O(|w|^u)$. Calculemos el número de pasos que realiza el sistema $\Pi_{TM}(s_{TM}(w))$ cuando recibe como entrada el multiconjunto $cod_{TM}(w)$ (a partir de los resultados demostrados en la sección 4.3 del capítulo 4).

- Transformación de $cod_{TM}(w)$ en la codificación exponencial de la cadena w: $|w| + 3$ pasos.
- Comprobación de que la codificación exponencial es correcta: $7 + 6 \cdot (|w| - 1) + 4$ pasos.
- Lectura del símbolo en la celda x-ésima: $2 + 3 \cdot x + 3$ pasos.
- Cálculo del nuevo símbolo a escribir en la celda x-ésima, movimiento de la cabeza y cambio de estado: 3 pasos.
- Borrado del símbolo en la celda x-ésima y escritura del nuevo símbolo en dicha celda: $2 \cdot x + 3$ pasos.
- Simulación de un paso de la máquina de Turing, si la cabeza está leyendo la celda x-ésima: $5 \cdot x + 8$.
- Comprobación del estado: si no es estado final, 2 pasos; si es estado final, 5 pasos.

En consecuencia, el sistema $\Pi_{TM}(s_{TM}(w))$ acepta o rechaza el multiconjunto $cod_{TM}(w)$ en un número de pasos del orden $O(|w|^{2 \cdot u})$.

- La familia $\mathbf{\Pi}_{TM}$ es adecuada y completa, respecto del problema X_{TM}, la función cod_{TM} y la función s_{TM}.

  Esto se obtiene directamente como consecuencia del hecho de que estos sistemas simulan el comportamiento de la máquina TM.

En conclusión, hemos demostrado que $X_{TM} \in \mathbf{PMC}^{fi}_{\mathcal{LA}}$. □

Seguidamente, vamos a establecer una condición *suficiente* para que se verifique la relación $\mathbf{P} \neq \mathbf{NP}$, en términos de irresolubilidad en tiempo polinomial de problemas **NP**–completos en la clase de sistemas P de aceptación de lenguajes.

Proposición 9.6. *Supongamos que existe un problema* **NP***–completo irresoluble en tiempo polinomial por una familia de sistemas de la clase* $\mathcal{LA}$. *Entonces* $\mathbf{P} \neq \mathbf{NP}$.

Demostración. Sea X un problema **NP**–completo tal que $X \notin \mathbf{PMC}^{fi}_{\mathcal{LA}}$. Supongamos que $\mathbf{P} = \mathbf{NP}$. Entonces $X \in \mathbf{P}$. Luego existe una máquina de Turing determinista, TM, que resuelve el problema X en tiempo polinomial.

Por la proposición 9.5, el problema $X_{TM} \in \mathbf{PMC}^{fi}_{\mathcal{LA}}$. Por tanto, existe una familia $\mathbf{\Pi}_{TM} = \left(\Pi_{TM}(k)\right)_{k\in\mathbb{N}^+}$ de sistemas P de aceptación de lenguajes que simula la máquina TM en tiempo polinomial.

Entonces, si consideramos la función $cod_X : I_X \to \bigcup_{k\in\mathbb{N}^+} I_{\Pi_{TM}(k)}$, dada por $cod_X(w) = cod_{TM}(w)$, y la función $s_X : I_X \to \mathbb{N}^+$, dada por $s_X(w) = |w|$, se verifica que:

- La familia $\mathbf{\Pi}_{TM}$ es adecuada, respecto del problema X, la función cod_X y la función s_X.

  En efecto, sea $w \in I_X$ tal que $\theta_X(w) = 1$. Entonces TM acepta la cadena w. Luego $\theta_{TM}(w) = 1$. Por tanto, toda computación del sistema $\Pi_{TM}(s_{TM}(w)) = \Pi_{TM}(s_X(w))$ con entrada $cod_{TM}(w) = cod_X(w)$ es de aceptación.

- La familia $\mathbf{\Pi}_{TM}$ es completa, respecto del problema X, la función cod_X y la función s_X.

  En efecto, sea $w \in I_X$ tal que existe una computación del sistema $\Pi_{TM}(s_X(w)) = \Pi_{TM}(s_{TM}(w))$ con entrada $cod_X(w) = cod_{TM}(w)$ que es de aceptación. Entonces $\theta_{TM}(w) = 1$. Luego $\theta_X(w) = 1$.

En consecuencia, se tendría que $X \in \mathbf{PMC}^{fi}_{\mathcal{LA}}$. Lo que lleva a una contradicción. □

9.2. Simulación de sistemas P por máquinas de Turing

En esta sección vamos a probar que si un problema de decisión se puede resolver en tiempo polinomial por una familia de sistemas P de aceptación de lenguajes, entonces también se puede resolver en tiempo polinomial por una máquina de Turing determinista. Deduciremos entonces una condición *necesaria* para que se verifique la relación $\mathbf{P} \neq \mathbf{NP}$.

Para el diseño de la máquina de Turing nos hemos inspirado en un trabajo de C. Zandron, C. Ferretti y G. Mauri [60], con la diferencia de que dicho artículo se enmarca dentro del estudio de los sistemas P con membranas activas (que introduciremos en el capítulo siguiente) y, además, la descripción que realizan de la máquina de Turing está menos detallada que la que proporcionamos a continuación.

Proposición 9.7. *Sea $X \in \mathbf{PMC}^{fi}_{\mathcal{LA}}$. Entonces existe una máquina de Turing que resuelve el problema X en tiempo polinomial.*

Demostración. Por hipótesis, existe una familia de sistemas P de aceptación de lenguajes, $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbb{N}^+}$, verificando:

1. La familia $\mathbf{\Pi}$ es polinomialmente uniforme, por máquinas de Turing.
2. Existen una función $cod : I_X \to \bigcup_{n\in\mathbb{N}^+} I_{\Pi(n)}$ y una función $s : I_X \to \mathbb{N}^+$, computables en tiempo polinomial, tales que:
   - Para todo $w \in I_X$, $cod(w) \in I_{\Pi(s(w))}$.
   - La familia $\mathbf{\Pi}$ está polinomialmente acotada, respecto del problema X, la función cod y la función s.
   - La familia $\mathbf{\Pi}$ es adecuada y completa, respecto del problema X, la función cod y la función s.

Dado $n \in \mathbb{N}^+$, notemos por A_n el número de símbolos del alfabeto de entrada de $\Pi(n)$, por B_n el número de símbolos del alfabeto de trabajo, por C_n el número de símbolos del alfabeto de salida, por D_n el número de membranas, por E_n el tamaño máximo de los multiconjuntos asociados inicialmente a estas, por F_n el número total de reglas del sistema, y por G_n la longitud máxima de estas. Puesto que la familia $\mathbf{\Pi}$ es polinomialmente uniforme, por máquinas de Turing, estos datos son polinomiales en $n \in \mathbb{N}^+$.

Sea $m \in I_{\Pi(n)}$ un multiconjunto de entrada del sistema $\Pi(n)$. Dado una computación de este con entrada m, notamos por $H_n(m)$ el máximo número de dígitos, en base 2, de las multiplicidades de los objetos contenidos en los multiconjuntos asociados a las membranas del sistema y al entorno, en cualquier paso de dicha computación. Naturalmente, este dato depende de la computación escogida, pero lo que nos interesa, y lo probaremos al final de la demostración, es que *cualquier* computación del sistema $\Pi(s(w))$ con entrada $cod(w)$ verifica que $H_n(cod(w))$ es polinomial en el tamaño de la cadena w.

A continuación asociamos al sistema $\Pi(n)$ una máquina de Turing determinista, $TM(n)$, *con múltiples cintas*, tal que, dado un multiconjunto de entrada, $m \in I_{\Pi(n)}$, reproduce *una* determinada computación, fijada por el proceso de funcionamiento de la máquina, de $\Pi(n)$ sobre m.

El alfabeto de entrada de la máquina $TM(n)$ coincide con el del sistema $\Pi(n)$. Por otra parte, el alfabeto de trabajo contiene, además de los símbolos de este alfabeto: un símbolo por cada etiqueta asignada a las membranas

de dicho sistema; los símbolos 0 y 1, que permitirán operar con números representados en base 2; tres símbolos que indicarán si una membrana no se ha disuelto, ha de disolverse o se ha disuelto; y tres símbolos que indicarán si una regla está en espera, es aplicable o no es aplicable.

El proceso de simulación, como veremos, se puede estructurar en varias fases, cada una de las cuales lleva a cabo una tarea concreta. Estas fases realizan distintos bucles en los que se recorren las membranas de la estructura de membranas original del sistema y las reglas asociadas a ellas. Así, el conjunto de estados de la máquina de Turing contendrá estados relacionados con dichas membranas y reglas para cada una de las fases. Estos estados determinarán en qué momento de la simulación nos encontramos en cada paso del funcionamiento de la máquina.

Seguidamente, detallamos cuáles son las cintas de las que consta esta máquina.

- Una *cinta de entrada*, que guarda una cadena representando el multiconjunto de entrada recibido. Para fijar dicha representación suponemos que el alfabeto de entrada está ordenado y que, entonces, la cadena contiene, por orden, los símbolos contenidos en el multiconjunto, repetidos tantas veces como indique su multiplicidad.

- Por cada membrana del sistema tenemos:

  - Una *cinta de estructura*, que guarda en la segunda celda la etiqueta del padre de la membrana, y en la tercera uno de los tres símbolos que indican si la membrana no se ha disuelto, si la membrana ha de disolverse, o si la membrana se ha disuelto.

  - Para cada objeto del alfabeto de trabajo del sistema:

    - Una *cinta principal*, que guarda la multiplicidad del objeto, en base 2, en el multiconjunto contenido en la membrana.
    - Una *cinta auxiliar*, que guarda resultados temporales, también en base 2, de aplicar las reglas asociadas a la membrana.

    Tras la simulación de cada paso del sistema P, el contenido de estas dos cintas coincidirá.

  - Una *cinta de reglas*, en la que cada celda a partir de la segunda corresponde a una regla asociada a la membrana (suponemos que el conjunto de dichas reglas está ordenado), y guarda uno de los tres

símbolos que indican si dicha regla está en espera, si es aplicable, o si no es aplicable.

- Para cada objeto del alfabeto de salida tenemos:
  - Una *cinta de entorno*, que guarda la multiplicidad del objeto, en base 2, en el multiconjunto asociado al entorno externo.

En total son $D_n \cdot (2 \cdot B_n + 2) + C_n + 1$ cintas.

A continuación describimos los pasos realizados por la máquina para simular un paso del sistema. Téngase en cuenta que, efectuando un recorrido en anchura sobre la estructura de membranas *inicial* del sistema $\Pi(n)$ (que recordemos es un árbol enraizado), obtenemos un orden natural entre las membranas de dicha estructura. Pues bien, en los algoritmos que detallamos posteriormente consideramos que siempre se recorren *todas* las membranas de la estructura de membranas *original* y que, además, lo hacemos en el orden inducido por el recorrido en anchura de dicha estructura.

Por otra parte, hacemos notar que todas las operaciones aritméticas, sobre números representados en base 2, que consideraremos a lo largo de la simulación se pueden realizar en tiempo lineal en el máximo número de dígitos de dichos números.

I. Inicialización del sistema. En la primera fase del proceso de simulación seguido por la máquina de Turing se incluyen en las cintas correspondientes los símbolos necesarios para que quede reflejada la configuración inicial de la computación con entrada el multiconjunto m que se va a simular.

para cada membrana *mb* del sistema **hacer**
  si *mb* no es la membrana piel **entonces**
    – Escribir en la segunda celda de la cinta de estructura de *mb* la etiqueta correspondiente a la membrana padre de *mb*
  fin si
  – Marcar *mb* como membrana no disuelta en la tercera celda de la cinta de estructura de *mb*
  para cada símbolo *ob* del alfabeto de trabajo **hacer**
    – Escribir en la cinta principal de *mb* para *ob* la multiplicidad, en base 2, de *ob* en el multiconjunto asociado inicialmente a *mb*
  fin para
fin para

para cada símbolo *ob* del alfabeto de entrada **hacer**
- Leer la multiplicidad, en base 2, de *ob* en la cinta de entrada
- Añadir dicha multiplicidad a la cinta principal de la membrana de entrada para *ob*

fin para

El primer bucle principal de este algoritmo se puede efectuar en un número de pasos de orden $O(D_n \cdot B_n \cdot \log E_n)$ y el segundo bucle principal en un número de pasos de orden $O(A_n \cdot (\log |m| + \text{máx}(\log |m|, \log E_n)))$.

II. Determinar las reglas aplicables. Para simular un paso del sistema de computación celular, lo primero que debe hacer la máquina de Turing es determinar el conjunto de reglas que son aplicables (cada una de ellas de forma independiente) a la configuración considerada en las membranas a las que están asociadas.

para cada membrana *mb* del sistema **hacer**
  si *mb* no se ha disuelto **entonces**
    para cada regla *r* asociada a *mb* **hacer**
      - Marcar *r* como regla en espera
    fin para
    para cada regla *r* asociada a *mb* **hacer**
      si - *r* está en espera y
        - *mb* contiene el antecedente de *r* y
        - *r* solo envía objetos a membranas que no se han disuelto y de las cuales *mb* es la membrana padre y
        - *r* no intenta disolver la membrana piel
      entonces
        - Marcar *r* como regla aplicable
        para cada regla *r'* asociada a *mb* de menor prioridad que *r* **hacer**
          - Marcar *r'* como regla no aplicable
        fin para
      si no
        - Marcar *r* como regla no aplicable
      fin si
    fin para
  fin si
fin para

El número de pasos requerido para efectuar este proceso es de orden $O(D_n \cdot (F_n + F_n \cdot (B_n \cdot H_n(m) + G_n + F_n)))$.

III. Aplicar las reglas. Una vez determinadas las reglas aplicables, estas se aplican de forma maximal a las membranas a las que están asociadas. El hecho de que las reglas se consideren en un cierto orden (usando maximalidad local para cada regla, según dicho orden) determina un multiconjunto de reglas aplicable particular, fijando así la computación del sistema que simula la máquina de Turing. No obstante, de la definición de clases de complejidad que hemos realizado se deducirá que la computación elegida no es relevante para la prueba del resultado que pretendemos demostrar, debido a la confluencia del sistema.

para cada membrana *mb* del sistema **hacer**
  si *mb* no se ha disuelto **entonces**
    para cada regla *r* asociada a *mb* que sea aplicable **hacer**
      para cada objeto *ob* en el antecedente de *r* **hacer**
        - Calcular el cociente entero resultante de dividir la multiplicidad de *ob* en la cinta principal de *mb* por la multiplicidad de *ob* en el antecedente de *r*

      fin para
      - Calcular el mínimo de los valores obtenidos en el bucle anterior (dicho mínimo es el máximo número de veces que se puede aplicar la regla *r* a la membrana *mb*). Llamémosle índice de la regla *r*.

      para cada objeto *ob* en el antecedente de *r* **hacer**
        - Multiplicar la multiplicidad de *ob* en el antecedente de *r* por el índice de *r*
        - Borrar el resultado obtenido de la cinta principal de *mb* para el objeto *ob*

      fin para
      para cada objeto *ob* en el consecuente de *r* **hacer**
        - Multiplicar la multiplicidad de *ob* en el consecuente de *r* por el índice de *r*
        - Sumar el resultado obtenido a la cinta auxiliar para *ob* de la membrana que corresponda

      fin para

si r disuelve mb **entonces**
– Marcar mb como membrana a disolver en la tercera celda de la cinta de estructura de mb
fin si
fin para
fin si
fin para

El número de pasos requerido para efectuar este proceso es de orden $O(D_n \cdot F_n \cdot (G_n \cdot \text{máx}(H_n(m), \log G_n) + G_n \cdot H_n(m) + 2 \cdot G_n \cdot (\text{máx}(\log G_n, H_n(m)) + (\log G_n + H_n(m)))))$.

IV. Actualizar los multiconjuntos. Una vez aplicadas las reglas a las membranas, las cintas auxiliares de estas guardan los resultados obtenidos, por lo que dichos resultados deben ser trasladados a las cintas principales correspondientes.

para cada membrana mb del sistema **hacer**
si mb no se ha disuelto **entonces**
– Copiar el contenido de las cintas auxiliares de mb en las cintas principales correspondientes
fin si
fin para

El número de pasos requeridos para efectuar este proceso es de orden $O(D_n \cdot B_n \cdot H_n(m))$.

V. Disolver las membranas. Para terminar la simulación de un paso de la computación del sistema P es necesario disolver las membranas según indiquen las reglas que se han aplicado en la fase anterior y reestructurar el árbol de membranas, redefiniendo para ello la función padre de dicho árbol.

para cada membrana mb del sistema **hacer**
si mb no se ha disuelto
y
el padre de mb está marcado como membrana a disolver
entonces
– Hacer el padre de mb igual al padre del padre de mb
fin si
fin para

para cada membrana mb del sistema **hacer**
  si mb está marcada como membrana a disolver **entonces**
    – Copiar el contenido de las cintas principales de mb en las cintas principales del (posiblemente nuevo) padre de mb
    – Marcar mb como membrana disuelta en la tercera celda de la cinta de estructura de mb
  fin si
fin para

El número de pasos requeridos para efectuar este proceso es de orden $O(D_n + D_n \cdot B_n \cdot H_n(m))$.

VI. Comprobar si la simulación ha terminado. Finalmente, tras terminar la simulación de un paso de la computación del sistema $\Pi(n)$, la máquina de Turing debe comprobar si dicha computación ha llegado a una configuración de parada y, en su caso, si la computación es de aceptación o de rechazo.

si la cinta de entorno contiene el símbolo # **entonces**
  si la cinta de entorno contiene el símbolo Yes **entonces**
    Parar y aceptar el multiconjunto m
  si no
    Parar y rechazar el multiconjunto m
  fin si
si no
  Volver a simular un paso de la computación del sistema P
fin si

Este proceso se puede efectuar en un número de pasos de orden $O(1)$.

Es fácil comprobar que la familia $\left(TM(n)\right)_{n\in\mathbb{N}^+}$ se puede construir de manera uniforme y en tiempo polinomial a partir de $n \in \mathbb{N}^+$.

Consideremos finalmente la máquina de Turing determinista, $TM_{\mathbf{\Pi}}$, que trabaja como sigue:

Entrada: $w \in I_X$
– Calcular $s(w)$
– Construir $TM(s(w))$
– Calcular $cod(w)$
– Reproducir el funcionamiento de $TM(s(w))$ con dato de entrada $cod(w)$

Entonces se cumple lo siguiente:

1. La máquina TM_{Π} trabaja en tiempo polinomial sobre $|w|$.

   Como las funciones *cod* y s son polinomiales en $|w|$, los datos $A_{s(w)}$, $B_{s(w)}$, $C_{s(w)}$, $D_{s(w)}$, $E_{s(w)}$, $F_{s(w)}$, y $G_{s(w)}$ son polinomiales en $|w|$.

   Por otra parte, la familia $\mathbf{\Pi}$ está polinomialmente acotada respecto de X, *cod* y s, luego toda computación del sistema $\Pi(s(w))$ con entrada $cod(w)$ realiza un número de pasos polinomial en $|w|$. En consecuencia, el número de pasos, $P_{s(w),cod(w)}$, realizados por la computación simulada por la máquina $TM(s(w))$ sobre $cod(w)$ es polinomial en $|w|$.

   Por tanto, la máxima multiplicidad de los objetos contenidos en los multiconjuntos asociados a las membranas del sistema es de orden $O(E_{s(w)} \cdot G_{s(w)}^{P_{s(w),cod(w)}})$. De donde se deduce que $H_{s(w)}(cod(w))$ es de orden $O(P_{s(w),cod(w)} \cdot log_2(E_{s(w)} \cdot G_{s(w)}))$; es decir, polinomial en $|w|$.

   En consecuencia, el tiempo total empleado por TM_{Π} al recibir w como cadena de entrada es polinomial en $|w|$.

2. Supongamos que $\theta_X(w) = 1$. Entonces toda computación de $\Pi(s(w))$ con entrada $cod(w)$ es de aceptación. Por tanto, la computación simulada por $TM(s(w))$ también lo es. Luego la máquina TM_{Π} acepta la cadena w.

3. Supongamos que TM_{Π} acepta la cadena w. Entonces la computación de $\Pi(s(w))$ con entrada $cod(w)$ simulada por $TM(s(w))$ es de aceptación. Luego $\theta_X(w) = 1$.

En consecuencia, hemos probado que TM_{Π} resuelve el problema X en tiempo polinomial. □

Seguidamente vamos a establecer una condición necesaria para que se verifique la relación $\mathbf{P} \neq \mathbf{NP}$ en términos de irresolubilidad en tiempo polinomial de problemas **NP**–completos en la clase $\mathcal{LA}$.

Proposición 9.8. *Supongamos que* $\mathbf{P} \neq \mathbf{NP}$. *Entonces ningún problema* **NP**–*completo puede ser resuelto en tiempo polinomial por una familia de sistemas P de aceptación de lenguajes.*

Demostración. Supongamos que existe un problema **NP**–completo, X, tal que $X \in \mathbf{PMC}_{\mathcal{LA}}^{fi}$. Entonces, por la proposición 9.7 existe una máquina de

Turing que resuelve el problema X en tiempo polinomial. Por tanto, $X \in \mathbf{P}$. Luego $\mathbf{P} = \mathbf{NP}$, lo que lleva a una contradicción. □

9.3. Caracterización de la relación $\mathbf{P} \neq \mathbf{NP}$ a través de sistemas P

Estamos ya en condiciones de establecer caracterizaciones de la relación $\mathbf{P} \neq \mathbf{NP}$ mediante la irresolubilidad de problemas **NP**–completos por familias de sistemas P de aceptación de lenguajes.

Teorema 9.9. *Las proposiciones siguientes son equivalentes:*

1. $\mathbf{P} \neq \mathbf{NP}$.
2. $\exists X\big(X$ *es un problema de decisión* $\mathbf{NP}$*–completo* $\wedge\, X \notin \mathbf{PMC}^{fi}_{\mathcal{LA}}\big)$.
3. $\forall X\big(X$ *problema de decisión* $\mathbf{NP}$*–completo* $\rightarrow X \notin \mathbf{PMC}^{fi}_{\mathcal{LA}}\big)$.

Demostración. La implicación $1 \Rightarrow 3$ se deduce de la proposición 9.8.

La implicación $3 \Rightarrow 2$ es trivial, ya que la clase de los problemas **NP**–completos es no vacía.

La implicación $2 \Rightarrow 1$ se deduce de la proposición 9.6. □

Veamos finalmente que los resultados obtenidos en las secciones 9.1 y 9.2 de este capítulo nos proporcionan una descripción de la clase **P** a través de la clase de complejidad polinomial para familias de sistemas P de transición con salida externa, como dispositivos de aceptación de lenguajes.

Teorema 9.10. *Se verifica la siguiente igualdad entre clases de complejidad:*

$$\mathbf{P} = \mathbf{PMC}^{fi}_{\mathcal{LA}}$$

Demostración. Sea X un problema de decisión resoluble en tiempo polinomial por una máquina de Turing determinista. A partir de la proposición 9.5, se deduce que $X \in \mathbf{PMC}^{fi}_{\mathcal{LA}}$.

Por otra parte, sea X un problema de decisión resoluble en tiempo polinomial por una familia de sistemas P de aceptación de lenguajes. A partir de la proposición 9.7, se deduce que $X \in \mathbf{P}$. □

Capítulo 10

Sistemas P con membranas activas

Recordemos que los sistemas P de transición con salida externa presentan dos niveles de paralelismo: por un lado, las reglas asociadas a una membrana se aplican simultáneamente; por otro, esta operación se realiza al mismo tiempo en todas las membranas del sistema. Sin embargo, hemos mostrado en el capítulo anterior (teoremas 9.9 y 9.10) que esto no es suficiente para resolver problemas «complejos» en tiempo polinomial a través de familias de sistemas P de aceptación de lenguajes (salvo, naturalmente, que $\mathbf{P} = \mathbf{NP}$, lo que no parece probable).

En este capítulo se introduce una nueva variante de sistemas de computación celular, los sistemas P con membranas activas [34], que poseen un paralelismo mejorado, en cierto sentido.

Para la formalización de estos sistemas se sigue la línea desarrollada en [48] y [54]. Así, en las dos secciones del capítulo describimos formalmente los requisitos que deben cumplir los elementos de un sistema de computación celular con membranas para que determinen la sintaxis (sección 10.1) y la semántica (sección 10.2) de un sistema P con membranas activas.

10.1. Sintaxis de los sistemas P con membranas activas

En esta sección fijamos la sintaxis de los sistemas P con membranas activas. En primer lugar, detallamos el conjunto de reglas de evolución que serán consideradas en estos sistemas.

Definición 10.1. *Sean LB un conjunto finito de etiquetas y Γ un alfabeto finito no vacío. El conjunto* de reglas de evolución con membranas activas *asociado a LB y Γ, que notamos por $AMER(LB, \Gamma)$, es el conjunto de todas las posibles tuplas de los tipos siguientes:*

- *Reglas de tipo (a):* (a, l, c, o, m)

  En una membrana con etiqueta $l \in LB$ y carga $c \in \{+, -, 0\}$, un objeto $o \in \Gamma$ evoluciona a un multiconjunto de objetos $m \in \mathbf{M}(\Gamma)$.

  Usualmente notaremos por $[_l o \to m]_l^c$ este tipo de reglas.

- *Reglas de tipo (b):* $(b, l, c_1, c_2, o_1, o_2)$

  Un objeto $o_1 \in \Gamma$ se introduce en una membrana con etiqueta $l \in LB$ y carga $c_1 \in \{+, -, 0\}$, evolucionando al mismo tiempo a un objeto $o_2 \in \Gamma$ y cambiando la carga de la membrana a $c_2 \in \{+, -, 0\}$.

  Usualmente notaremos por $o_1[_l\ ]_l^{c_1} \to [_l o_2]_l^{c_2}$ este tipo de reglas.

- *Reglas de tipo (c):* $(c, l, c_1, c_2, o_1, o_2)$

  Un objeto $o_1 \in \Gamma$ es expulsado de una membrana con etiqueta $l \in LB$ y carga $c_1 \in \{+, -, 0\}$, evolucionando al mismo tiempo a un objeto $o_2 \in \Gamma$ y cambiando la carga de la membrana a $c_2 \in \{+, -, 0\}$.

  Usualmente notaremos por $[_l o_1]_l^{c_1} \to [_l\ ]_l^{c_2} o_2$ este tipo de reglas.

- *Reglas de tipo (d):* (d, l, c, o_1, o_2)

  Un objeto $o_1 \in \Gamma$ disuelve una membrana con etiqueta $l \in LB$ y carga $c \in \{+, -, 0\}$, evolucionando al mismo tiempo a un objeto $o_2 \in \Gamma$.

  Usualmente notaremos por $[_l o_1]_l^c \to o_2$ este tipo de reglas.

- *Reglas de tipo (e):* $(e, l, c_1, c_2, c_3, o_1, o_2, o_3)$

  Un objeto $o_1 \in \Gamma$ divide en dos una membrana con etiqueta $l \in LB$ y carga $c_1 \in \{+, -, 0\}$. En una de las membranas resultantes, el objeto evoluciona a $o_2 \in \Gamma$ y la carga cambia a $c_2 \in \{+, -, 0\}$. En la otra membrana resultante, el objeto evoluciona a $o_3 \in \Gamma$ y la carga cambia a $c_3 \in \{+, -, 0\}$.

  Usualmente notaremos por $[_l o_1]_l^{c_1} \to [_l o_2]_l^{c_2} [_l o_3]_l^{c_3}$ este tipo de reglas.

Estamos entonces en condiciones de definir qué entendemos por sistema P con membranas activas (que usa 2–división).

Definición 10.2. *Un* sistema P con membranas activas, Π, *es un sistema de computación celular con membranas,* (AL, MG, LB, ER, AE, MB), *tal que:*

- AL *consta de un alfabeto de trabajo,* Γ, *y un alfabeto de salida,* Λ, *de forma que* $\Lambda \subseteq \Gamma$. *Si el sistema tiene membrana de entrada, se incluye también un alfabeto de entrada,* $\Sigma \subseteq \Gamma$. *Además, existe un elemento distinguido* $\# \in \Gamma \setminus (\Sigma \cup \Lambda)$.
- MG, *es una estructura de membranas,* μ_Π.
- LB *es un conjunto finito de etiquetas.*
- *La aplicación* ER *asocia a cada etiqueta* $l \in LB$ *un subconjunto finito,* R_l, *de* $AMER(LB, \Gamma)$, *de forma que la etiqueta que aparece en todas las reglas de dicho conjunto coincide con* l.
- *La aplicación* AE *asocia a cada membrana,* mb, *de* μ_Π *una etiqueta* $l_{mb} \in LB$ *y un multiconjunto,* $\mathcal{M}_{mb} \in \mathbf{M}(\Gamma \setminus \Sigma)$, *de objetos contenidos inicialmente en la membrana.*
- MB *contiene la membrana de entrada,* im, *para los sistemas con entrada.*
- *La salida se recoge en el entorno externo de la estructura de membranas.*

Notación. *Notaremos un sistema* P *con membranas activas por una tupla*

$$\Pi = \left(\Sigma, \Gamma, \Lambda, \#, LB, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, R, im, ext\right)$$

donde el alfabeto de entrada Σ *y la membrana de entrada* im *no se especifican en un sistema sin entrada; la estructura* μ_Π *tiene grado* p *y* $\mathcal{M}_1, \ldots, \mathcal{M}_p$ *son los multiconjuntos asociados a sus membranas;* $R = \bigcup_{l \in LB} R_l$, *donde* R_l *es el conjunto de todas las reglas asociadas a la etiqueta* l.

10.2. Semántica de los sistemas P con membranas activas

A continuación vamos a detallar de qué forma evoluciona un sistema P con membranas activas atendiendo a los multiconjuntos de objetos contenidos en cada una de las membranas de su estructura de membranas, así como a las reglas de evolución asociadas a las etiquetas de estas.

Definición 10.3. *Sea* Π *un sistema P con membranas activas. Una configuración de* Π *es una tupla* $C = \big(Ext(\mu), Lb, Ch, M\big)$ *tal que verifica las siguientes condiciones:*

- $\mu = (V(\mu), E(\mu))$ *es una estructura de membranas.*
- *La raíz de* μ_Π *coincide con la raíz de* μ.
- $Ext(\mu)$ *es la estructura de membranas con entorno externo asociada a la estructura* μ.
- Lb *es una aplicación de dominio* $V(\mu)$ *y rango contenido en* LB, *que asocia una etiqueta a cada membrana de* μ.
- Ch *es una aplicación de dominio* $V(\mu)$ *y rango contenido en* $\{+, -, 0\}$, *que asocia una carga (positiva, negativa o neutra, respectivamente) a cada membrana de* μ.
- M *es una aplicación con dominio* $V(Ext(\mu))$ *y rango contenido en* $\mathbf{M}(\Gamma)$, *que asocia un multiconjunto de objetos al entorno externo y a cada membrana de* μ.

Notación. *Notaremos usualmente por* $C = (\mu, M_{env}, M_{i_1}, \ldots, M_{i_q})$ *una configuración de* Π, *donde* $V(\mu) = \{i_1, \ldots, i_q\}$, $M_{env} = M(env)$ *es el multiconjunto asociado al entorno externo de* μ *y* $M_{i_j} = M(i_j)$ *es el multiconjunto asociado a la membrana* i_j *de* μ, *para cada* $j = 1, \ldots, q$.

Además, la estructura de membranas μ *se suele representar usando la notación de corchetes, incluyendo la etiqueta y la carga de cada membrana, tal y como indicamos a continuación:*

1. *Si* $V(\mu) = \{i_1\}$, $Lb(i_1) = l$ *y* $Ch(i_1) = c$, *entonces* $\mu \equiv [_l\ ]_l^c$.
2. *Si* $i_1 \in V(\mu)$ *es la raíz de* μ, *con etiqueta* $Lb(i_1) = l$ *y carga* $Ch(i_1) = c$, *y* $\mu_1, \ldots, \mu_k$ *son los subárboles principales en cada hijo del nodo* i_1, *entonces* $\mu \equiv [_l \mu_1 \ldots \mu_k]_l^c$, *donde* $\mu_1, \ldots, \mu_k$ *están representados usando la notación de corchetes, etiqueta y carga.*

A la hora de definir las configuraciones que especifican el estado inicial de un sistema P (es decir, sus configuraciones iniciales) debemos tener en cuenta si dicho sistema posee o no membrana de entrada.

Definición 10.4. *Sea* Π *un sistema P con membranas activas.*

- *Caso 1:* Π *no posee membrana de entrada.*

  En este caso existe una única configuración inicial del sistema, a saber

  $$C^0 = (\mu_\Pi, \emptyset, \mathcal{M}_1, \ldots, \mathcal{M}_p)$$

  donde la carga inicial de cada membrana es 0.

- *Caso 2:* Π *posee membrana de entrada.*

  En este caso existe una configuración inicial para cada multiconjunto $m \in \mathbf{M}(\Sigma)$ *que se pueda introducir en dicha membrana, a saber*

  $$C^0(m) = (\mu_\Pi, \emptyset, \mathcal{M}_1, \ldots, \mathcal{M}_{im} + m, \ldots, \mathcal{M}_p)$$

  donde la carga inicial de cada membrana es 0.

Para definir la función paso de transición para un sistema P con membranas activas se necesita, dada una configuración de dicho sistema, determinar qué reglas se pueden aplicar a esa configuración en cada membrana.

Definición 10.5. *Diremos que la regla* $r \in AMER(LB, \Gamma)$ *es* aplicable a la configuración C en la membrana $i \in V(\mu)$, *y lo notaremos* $r \in ApR(C, i)$, *si cumple lo siguiente:*

- *Caso 1:* $r = (a, l^r, c^r, o^r, m^r)$
  - *La etiqueta asociada a la membrana i es* l^r*:* $Lb(i) = l^r$.
  - *La carga asociada a la membrana i es* c^r*:* $Ch(i) = c^r$.
  - *La membrana i contiene el objeto necesario para activar la regla:* $o^r \in M_i$.
- *Caso 2:* $r = (b, l^r, c_1^r, c_2^r, o_1^r, o_2^r)$
  - *La membrana i no es la membrana piel.*
  - *La etiqueta asociada a la membrana i es* l^r*:* $Lb(i) = l^r$.
  - *La carga asociada a la membrana i es* c_1^r*:* $Ch(i) = c_1^r$.
  - *La membrana padre de la membrana i contiene el objeto necesario para activar la regla:* $o_1^r \in M_{ft_\mu(i)}$.

- *Caso 3:* $r = (c, l^r, c_1^r, c_2^r, o_1^r, o_2^r)$
  - *La etiqueta asociada a la membrana i es l^r:* $Lb(i) = l^r$.
  - *La carga asociada a la membrana i es c_1^r:* $Ch(i) = c_1^r$.
  - *La membrana i contiene el objeto necesario para activar la regla:* $o_1^r \in M_i$.
- *Caso 4:* $r = (d, l^r, c^r, o_1^r, o_2^r)$
  - *La membrana i no es la membrana piel.*
  - *La etiqueta asociada a la membrana i es l^r:* $Lb(i) = l^r$.
  - *La carga asociada a la membrana i es c^r:* $Ch(i) = c^r$.
  - *La membrana i contiene el objeto necesario para activar la regla:* $o_1^r \in M_i$.
- *Caso 5:* $r = (e, l^r, c_1^r, c_2^r, c_3^r, o_1^r, o_2^r, o_3^r)$
  - *La membrana i no es la membrana piel.*
  - *La membrana i es una membrana elemental.*
  - *La etiqueta asociada a la membrana i es l^r:* $Lb(i) = l^r$.
  - *La carga asociada a la membrana i es c_1^r:* $Ch(i) = c_1^r$.
  - *La membrana i contiene el objeto necesario para activar la regla:* $o_1^r \in M_i$.

A continuación definimos qué colecciones de reglas de evolución se pueden aplicar a una configuración de manera simultánea. Para ello hay que tener en cuenta que, de acuerdo con la definición establecida en [34], las reglas de tipo (a) se usan tantas veces como sea posible. Sin embargo, cada membrana solo puede estar involucrada en una única regla, a lo sumo, de los tipos (b)–(e).

Definición 10.6. *Diremos que* $amr \in \mathbf{M}\big(AMER(LB, \Gamma) \times V(\mu)\big)$ *es un* multiconjunto de reglas aplicable a la configuración C, *y lo notaremos por* $amr \in ApM(C)$, *si verifica:*

- *Contiene al menos una regla:* $amr \neq \emptyset$.

- *Las reglas contenidas en el multiconjunto son aplicables a C en la membrana asociada:*

$$\forall r \in AMER(LB,\Gamma)\ \forall i \in V(\mu)\big(r \in ApR(C,i)\big)$$

- *Cada membrana tiene asociada en amr, a lo sumo, una regla de los tipos (b)-(e):*

$$\forall i \in V(\mu)\Big(\sum_{\substack{r\in R_{Lb(i)}\\ r \text{ de tipo (b)-(e)}}} amr(r,i) \leq 1\Big)$$

- *Todas las reglas se pueden aplicar a la vez:*

$$\forall i \in V(\mu)\ \Big(\sum_{\substack{r\in R_{Lb(i)}\\ r \text{ de tipo (a)}}} amr(r,i)\cdot o^r + \sum_{j\in ch_\mu(i)}\ \sum_{\substack{r\in R_{Lb(j)}\\ r \text{ de tipo (b)}}} amr(r,j)\cdot o_1^r + \sum_{\substack{r\in R_{Lb(i)}\\ r \text{ de tipo (c)-(e)}}} amr(r,i)\cdot o_1^r \leq M_i\Big)$$

- *Las reglas se aplican de forma maximal:*

$$\neg\exists amr'\big(amr' \in ApM(C) \wedge amr < amr'\big)$$

Podemos clasificar las reglas de evolución de un sistema P con membranas activas en dos grupos, atendiendo a cómo actúan: por un lado, la aplicación de una regla de tipo (a), (b) o (c) produce solamente la evolución o desplazamiento de un objeto; por otro, la aplicación de una regla de tipo (d) o (e) produce la modificación de la estructura de membranas original, aunque de forma diferente según de qué tipo sea.

Por ello, calcularemos el efecto de la aplicación de un multiconjunto de reglas en tres pasos: inicialmente consideraremos solo las reglas de los primeros tres tipos; después atenderemos a las reglas de duplicación de membranas; y, finalmente, disolveremos las membranas que correspondan.

Definición 10.7. *Sea $C = \big(Ext(\mu), Lb, Ch, M\big)$ una configuración de Π y amr un multiconjunto de reglas aplicable a C. Se definen las aplicaciones Lb'_{amr}, Ch'_{amr} sobre $V(\mu)$ y la aplicación M'_{amr} sobre $V(Ext(\mu))$ como sigue:*

- *Dado* $i \in V(\mu)$,

$$Lb'_{amr}(i) = Lb(i)$$

$$Ch'_{amr}(i) = \begin{cases} c_2^r, & \textit{si existe } r \in R_{Lb(i)} \textit{ de tipo (b) o (c)} \\ & \textit{tal que } amr(r,i) = 1 \\ Ch(i), & \textit{en otro caso} \end{cases}$$

- *Dado* $i \in V(\mu)$, $i \neq skin$,

$$\begin{aligned} M'_{amr}(i) = & M(i) - \sum_{\substack{r \in R_{Lb(i)} \\ r \textit{ de tipo (a)}}} amr(r,i) \cdot o^r - \sum_{\substack{r \in R_{Lb(i)} \\ r \textit{ de tipo (c)}}} amr(r,i) \cdot o_1^r - \\ & \sum_{j \in ch_\mu(i)} \sum_{\substack{r \in R_{Lb(j)} \\ r \textit{ de tipo (b)}}} amr(r,j) \cdot o_1^r + \sum_{\substack{r \in R_{Lb(i)} \\ r \textit{ de tipo (a)}}} amr(r,i) \cdot m^r + \\ & \sum_{\substack{r \in R_{Lb(i)} \\ r \textit{ de tipo (b)}}} amr(r,i) \cdot o_2^r + \sum_{j \in ch_\mu(i)} \sum_{\substack{r \in R_{Lb(j)} \\ r \textit{ de tipo (c)}}} amr(r,j) \cdot o_2^r \end{aligned}$$

- *Dado* $i = skin$,

$$\begin{aligned} M'_{amr}(i) = & M(i) - \sum_{\substack{r \in R_{Lb(i)} \\ r \textit{ de tipo (a)}}} amr(r,i) \cdot o^r - \sum_{\substack{r \in R_{Lb(i)} \\ r \textit{ de tipo (c)}}} amr(r,i) \cdot o_1^r - \\ & \sum_{j \in ch_\mu(i)} \sum_{\substack{r \in R_{Lb(j)} \\ r \textit{ de tipo (b)}}} amr(r,j) \cdot o_1^r + \sum_{\substack{r \in R_{Lb(i)} \\ r \textit{ de tipo (a)}}} amr(r,i) \cdot m^r + \\ & \sum_{j \in ch_\mu(i)} \sum_{\substack{r \in R_{Lb(j)} \\ r \textit{ de tipo (c)}}} amr(r,j) \cdot o_2^r \end{aligned}$$

- *Dado* $i = env$,

$$M'_{amr}(i) = M(i) + \sum_{\substack{r \in R_{Lb(skin)} \\ r \textit{ de tipo (c)}}} amr(r, skin) \cdot o_{2r}$$

En la definición anterior hemos seguido la notación genérica de una regla r que aparece en la definición 10.5.

Nos centramos a continuación en el segundo de los procesos mencionados;

es decir, en la división de membranas que resulta de aplicar reglas del tipo (e). Por ello, debemos determinar en primer lugar cuáles son las membranas que se van a dividir.

Definición 10.8. *Sea $amr \in ApM(C)$ un multiconjunto de reglas aplicable a C. El conjunto de* membranas que se dividen *por la aplicación de amr se define como sigue:*

$$Div(amr, C) = \{i \in V(\mu) : \exists r \in R_{Lb(i)}\big(r \text{ es de tipo (e)} \wedge amr(r, i) = 1\big)\}$$

Entonces la nueva estructura de membranas, y los elementos contenidos en ellas, se calcula como indicamos a continuación.

Definición 10.9. *Sea $amr \in ApM(C)$ un multiconjunto de reglas aplicable a C. Se definen la estructura de membranas μ''_{amr}, las aplicaciones Lb''_{amr} y Ch''_{amr} sobre $V(\mu''_{amr})$ y la aplicación M''_{amr} sobre $V(Ext(\mu''_{amr}))$ como sigue:*

- $V(\mu''_{amr}) = \big(V(\mu) \setminus Div(amr, C)\big) \cup \{i^1, i^2 : i \in Div(amr, C)\}$, *donde suponemos que $V(\mu) \cap \{i^1, i^2 : i \in Div(amr, C)\} = \emptyset$, y también que $\{i^1, i^2\} \cap \{j^1, j^2\} = \emptyset$, para todo $i, j \in Div(amr, C)$.*

- *La raíz de μ''_{amr} coincide con la raíz de μ.*

- *La función padre, que determina las aristas de μ''_{amr}, viene dada por*

$$ft_{\mu''_{amr}}(j) = \begin{cases} \text{no definida}, & \text{si } j = skin \\ ft_{\mu}(j), & \text{si } j \notin Div(amr, C) \\ ft_{\mu}(i), & \text{si } i \in Div(amr, C) \text{ y } j = i^1, i^2 \end{cases}$$

  para todo $j \in V(\mu''_{amr})$.

- *Dado $i \in V(\mu) \setminus Div(amr, C)$,*

$$Lb''_{amr}(i) = Lb'_{amr}(i), \quad Ch''_{amr}(i) = Ch'_{amr}(i), \quad M''_{amr}(i) = M'_{amr}(i)$$

- *Dado $i \in Div(amr, C)$, sea r de tipo (e) tal que $amr(r, i) = 1$.*

Entonces,

$$\begin{aligned} Lb''_{amr}(i^1) &= Lb'_{amr}(i) & \qquad Lb''_{amr}(i^2) &= Lb'_{amr}(i) \\ Ch''_{amr}(i^1) &= c_2^r & Ch''_{amr}(i^2) &= c_3^r \\ M''_{amr}(i^1) &= M'_{amr}(i) - o_1^r + o_2^r & M''_{amr}(i^2) &= M'_{amr}(i) - o_1^r + o_3^r \end{aligned}$$

- $M''_{amr}(env) = M'_{amr}(env)$.

Finalmente describimos el proceso de disolución de membranas. Para ello, en primer lugar se determina el conjunto de membranas a ser disuelta por la aplicación de un multiconjunto de reglas aplicable a una configuración.

Definición 10.10. *Sea $amr \in ApM(C)$ un multiconjunto de reglas aplicable a C. El conjunto de* membranas disueltas *por la aplicación de amr se define como sigue:*

$$Dis(amr, C) = \{i \in V(\mu) : \exists r \in R_{Lb(i)}\big(r \text{ es de tipo (d)} \wedge amr(r, i) = 1\big)\}$$

Seguidamente, se calcula la nueva estructura de membranas.

Definición 10.11. *Sea $amr \in ApM(C)$ un multiconjunto de reglas aplicable a C. Se define la estructura de membranas μ'''_{amr} como sigue:*

- $V(\mu'''_{amr}) = V(\mu''_{amr}) \setminus Dis(amr, C)$.
- *La raíz de μ'''_{amr} coincide con la raíz de μ''_{amr}.*
- *La función padre, que determina las aristas de μ'''_{amr}, viene dada como sigue: para todo $i, j \in V(\mu'''_{amr})$*

$$i = ft_{\mu'''_{amr}}(j) \Leftrightarrow \exists i_0, \ldots, i_n \in V(\mu''_{amr}) \left(\begin{array}{l} i_1, \ldots, i_{n-1} \in Dis(amr, C) \wedge \\ i_0 = i \wedge i_n = j \wedge \\ \forall l\big(0 \leq l < n \rightarrow i_l = ft_{\mu''_{amr}}(i_{l+1})\big) \end{array} \right)$$

Obsérvese que la función padre anterior puede también calcularse como la restricción a $V(\mu'''_{amr})$ de la función $\overline{ft}_{\mu''_{amr}}$ definida sobre $V(\mu''_{amr})$ por recursión como sigue:

$$\overline{ft}_{\mu''_{amr}}(j) = \begin{cases} no\ definida, & \text{si } j = skin \\ ft_{\mu''_{amr}}(j), & \text{si } ft_{\mu''_{amr}}(j) \notin Dis(amr, C) \\ \overline{ft}_{\mu''_{amr}}(ft_{\mu''_{amr}}(j)), & \text{en otro caso} \end{cases}$$

Puesto que los objetos contenidos en una membrana que se disuelve pasan a la membrana padre, y en un mismo paso de transición se pueden disolver más de una membrana, necesitamos determinar, para cada membrana i que no se disuelva, el conjunto de sus membranas donantes; es decir, el conjunto de aquellas membranas que, al disolverse, depositan los objetos que contenían en la membrana i.

Definición 10.12. *Sea $amr \in ApM(C)$ un multiconjunto de reglas aplicable a C. Para cada membrana $i \in V(\mu'''_{amr})$ se define el conjunto de* donantes de *i como sigue:*

$$Don(i, amr, C) = \{j \in V(\mu''_{amr}) : j \in Dis(amr, C) \wedge \overline{ft}_{\mu''_{amr}}(j) = i\}$$

Entonces, podemos determinar cómo se redistribuyen los objetos de las membranas disueltas.

Definición 10.13. *Sea $amr \in ApM(C)$ un multiconjunto de reglas aplicable a C. Se definen las aplicaciones Lb'''_{amr} y Ch'''_{amr} sobre $V(\mu'''_{amr})$ y la aplicación M'''_{amr} sobre $V(Ext(\mu'''_{amr}))$ como sigue:*

- *Dado $i \in V(\mu'''_{amr})$,*

$$\begin{aligned} Lb'''_{amr}(i) &= Lb''_{amr}(i) \\ Ch'''_{amr}(i) &= Ch''_{amr}(i) \\ M'''_{amr}(i) &= M''_{amr}(i) + \sum_{j \in Don(i,amr,C)} M''_{amr}(j) \end{aligned}$$

- *Dado $i = env$,*

$$M'''_{amr}(env) = M''_{amr}(env)$$

Estamos ya en condiciones de establecer cuándo una configuración se ha obtenido en un paso de transición a partir de otra.

Definición 10.14. *La función paso de transición, TStep, para un sistema P con membranas activas, Π, se define como sigue: dadas dos configuraciones* $C_1 = \big(Ext(\mu_1), Lb_1, Ch_1, M_1\big)$ *y* $C_2 = \big(Ext(\mu_2), Lb_2, Ch_2, M_2\big)$ *del sistema,*

$$C_2 \in TStep(C_1) \Leftrightarrow \exists amr \in ApM(C)\big(\mu_2 = \mu_1{}'''_{amr} \wedge Lb_2 = Lb_1{}'''_{amr} \wedge Ch_2 = Ch_1{}'''_{amr} \wedge M_2 = M_1{}'''_{amr}\big)$$

Si $C_2 \in TStep(C_1)$*, entonces lo notamos por* $C_1 \Rightarrow_\Pi C_2$*.*

Notación. *Sean Π un sistema P con membranas activas y* $\mathcal{C} = \{C^i\}_{i<r}$ *una computación de Π. Entonces notamos* $C^i = \big(Ext(\mu^i), Lb^i, Ch^i, M^i\big)$*.*

Además, notamos por $amr_{i,i+1} \in ApM(C^i)$ *al multiconjunto de reglas aplicable a la configuración* C^i *que se ha utilizado en la computación para obtener la configuración* C^{i+1}*.*

Puesto que la salida de los sistemas P con membranas activas se recoge en el entorno externo, usamos, al igual que en los sistemas P de transición con salida externa, el objeto # para determinar la parada de una computación.

Definición 10.15. *Diremos que* $r \in AMER(LB, \Gamma)$ *es una* regla indicadora de parada *si r es de tipo (c) y* $o_2^r = \#$*.*

Definición 10.16. *Diremos que un sistema P con membranas activas, Π, es válido, si dada una computación* $\mathcal{C} = \{C^i\}_{i<t}$ *del sistema se cumple lo siguiente:*

- *Si* $\mathcal{C}$ *es de parada, entonces:*
  - *Para todo* $i < t-2$ *y toda* $r \in R_{Lb^i(skin)}$*, si r es una regla asociada en* $amr_{i,i+1}$ *a la membrana piel, entonces r no es indicadora de parada.*
  - *Existe* $r \in R_{Lb^{t-2}(skin)}$ *tal que r está asociada en* $amr_{t-2,t-1}$ *a la membrana piel y r es indicadora de parada.*
- *Si* $\mathcal{C}$ *no es de parada, entonces:*
  - *Para todo* $i < t-1$ *y toda* $r \in R_{Lb^i(skin)}$*, si r está asociada en* $amr_{i,i+1}$ *a la membrana piel, entonces r no es indicadora de parada.*

De esta forma, el hecho de que una computación haya parado o no viene determinado por la presencia de un objeto # en el entorno externo del sistema.

Capítulo 11

Soluciones lineales de los problemas SAT y VALIDITY en sistemas con membranas activas

En este capítulo vamos a considerar sistemas P con membranas activas como dispositivos de aceptación de lenguajes. De esta forma, estos sistemas resultarán adecuados para resolver problemas de decisión y, en particular, resolver de manera *eficiente* problemas **NP**–completos. En concreto, mostraremos cómo los problemas de la satisfactibilidad de la lógica proposicional, **SAT**, y de la validez, **VALIDITY**, pueden resolverse en tiempo lineal por una familia de tales sistemas. De donde se deducirá que las clases **NP** y **coNP** están contenidas en la clase de problemas resolubles en tiempo polinomial por familias de sistemas P con membranas activas.

En la primera sección de este capítulo definimos la función *Output* de los sistemas P con membranas activas de tal forma que nos permita considerar dichos sistemas como dispositivos de aceptación de lenguajes. Así, definimos qué entendemos por que un tal sistema acepte o decida un lenguaje sobre un determinado alfabeto. En las secciones 11.2 y 11.3 se construyen familias de sistemas con membranas activas que resuelven en tiempo lineal los problemas **SAT** y **VALIDITY**, respectivamente. Además, se establece la verificación formal de la corrección del funcionamiento de los sistemas de dichas familias, en relación con los problemas de decisión para los que han sido diseñados.

11.1. Sistemas P con membranas activas como dispositivos de aceptación

En el capítulo anterior hemos descrito la sintaxis y la semántica de los sistemas P con membranas activas, excepto en lo que respecta a la función *Output* que define la salida de las computaciones. A continuación vamos a fijar dicha función de tal forma que determine una variante de estos sistemas que sea adecuada para aceptar y decidir lenguajes.

Definición 11.1. *Un sistema P con membranas activas* como dispositivo de aceptación de lenguajes, Π, *es un sistema de computación celular con membranas que verifica las siguientes propiedades:*

- Π *es un sistema P con membranas activas.*
- Π *es un sistema P con membrana de entrada.*
- *El alfabeto de salida de* Π *es* $\Lambda = \{Yes, No\}$.
- *La salida de una computación* $\mathcal{C} = \{C^i\}_{i<r}$ *viene dada por la siguiente función:*

$$Output(\mathcal{C}) = \begin{cases} Yes, & \text{si } \mathcal{C} \text{ es de parada, } Yes \in M_{env}^{r-1} \text{ y } No \notin M_{env}^{r-1} \\ No, & \text{si } \mathcal{C} \text{ es de parada, } No \in M_{env}^{r-1} \text{ y } Yes \notin M_{env}^{r-1} \\ \text{no definida}, & \text{en otro caso} \end{cases}$$

Si $\mathcal{C}$ *cumple alguna de las dos primeras condiciones anteriores, entonces diremos que es una* computación exitosa.

Obsérvese que hemos definido la salida únicamente en aquellas computaciones en las que tiene sentido: la computación es de parada y solo aparecen objetos Yes u objetos No en el entorno externo. Sin embargo, para que un sistema de este tipo sea propiamente un dispositivo de aceptación debemos exigir además que todas las computaciones de parada proporcionen una respuesta.

Definición 11.2. *Un sistema P con membranas activas de aceptación de lenguajes se dice que es* válido *si verifica las siguientes propiedades:*

- Π *es un sistema P con membranas activas válido.*
- *Si $\mathcal{C} = \{C^i\}_{i<r}$ es una computación de parada de Π, entonces es una computación exitosa.*

Notación. *Notaremos por $\mathcal{AM}$ la clase de los sistemas P con membranas activas de aceptación de lenguajes tales que dichos sistemas sean válidos.*

Estamos en condiciones de establecer qué significa que un sistema P con membranas activas acepte un lenguaje sobre un determinado alfabeto.

Definición 11.3. *Sea L un lenguaje sobre un alfabeto Ω. Diremos que el sistema $\Pi \in \mathcal{AM}$* acepta *el lenguaje L si se verifican las siguientes propiedades:*

- *Existe una aplicación total computable e inyectiva, $cod : \Omega^* \to \mathbf{M}(\Sigma)$, que codifica las cadenas sobre Ω mediante multiconjuntos sobre el alfabeto de entrada del sistema Π.*
- *Para toda cadena $w \in \Omega^*$ se tiene que:*
  - *Si $w \in L$, entonces existe* una *computación $\mathcal{C}$ de Π con entrada $cod(w)$ tal que $\mathcal{C}$ es de parada y $Output(\mathcal{C}) = Yes$.*
  - *Si existe* una *computación $\mathcal{C}$ de Π con entrada $cod(w)$ tal que $\mathcal{C}$ es de parada y $Output(\mathcal{C}) = Yes$, entonces $w \in L$.*

Análogamente, podemos definir también qué significa que un sistema P con membranas activas decida un lenguaje sobre un determinado alfabeto.

Definición 11.4. *Sea L un lenguaje sobre un alfabeto Ω. Diremos que el sistema $\Pi \in \mathcal{AM}$ decide el lenguaje L si se verifican las siguientes propiedades:*

- *Toda computación de Π es de parada.*
- *Existe una aplicación total computable e inyectiva, $cod : \Omega^* \to \mathbf{M}(\Sigma)$, que codifica las cadenas sobre Ω mediante multiconjuntos sobre el alfabeto de entrada del sistema Π.*

- *Para toda cadena* $w \in \Omega^*$ *se tiene que:*
  - *Si* $w \in L$, *entonces para* toda *computación* $\mathcal{C}$ *de* Π *con entrada* $cod(w)$ *se tiene que* $Output(\mathcal{C}) = Yes$.
  - *Si* $w \notin L$, *entonces para* toda *computación* $\mathcal{C}$ *de* Π *con entrada* $cod(w)$ *se tiene que* $Output(\mathcal{C}) = No$.

Obsérvese que se tienen las siguientes relaciones entre los lenguajes aceptados y los lenguajes decididos por un sistema con membranas activas: si un sistema, Π, decide un lenguaje, L, entonces es fácil comprobar que también acepta dicho lenguaje; por otra parte, si un sistema, Π, *determinista*, acepta un lenguaje, L, y, además, todas las computaciones de dicho sistema son de parada, entonces el sistema Π decide el lenguaje L.

11.2. Resolución del problema SAT

En esta sección presentamos una familia de sistemas de la clase $\mathcal{AM}$, considerados como sistemas reconocedores, que resuelve el problema **SAT** en tiempo *lineal*. Así, puesto que este problema es **NP**–completo, se deducirá que la clase **NP** está contenida en la clase $\mathbf{PMC}^{fi}_{\mathcal{AM}}$.

El problema de la satisfactibilidad de la lógica proposicional, **SAT**, es el siguiente:

> *Dada una fórmula del cálculo proposicional, en forma normal conjuntiva, determinar si existe una valoración de verdad de sus variables que haga cierta la fórmula.*

Consideramos ahora una familia de sistemas P con membranas activas de aceptación de lenguajes, $\mathbf{\Pi_{SAT}} = \big(\Pi(m,n)\big)_{m,n\in\mathbb{N}^+}$, de tal manera que el sistema $\Pi(m,n)$ se encargue de procesar todas aquellas fórmulas que posean m cláusulas y n variables. Para cada $m, n \in \mathbb{N}^+$, el sistema $\Pi(m,n)$ se define como sigue:

$$\Pi(m,n) = \big(\Sigma(m,n), \Gamma(m,n), \Lambda, \#, LB, \mu_{\Pi(m,n)}, \mathcal{M}_1, \mathcal{M}_2, R(m,n), im, ext\big)$$

donde

$$
\begin{aligned}
\Sigma(m,n) &= \{x_{i,j}, \overline{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\} \\
\Gamma(m,n) &= \Sigma(m,n) \cup \{Yes, No, \#\} \cup \{c_k : 1 \leq k \leq m+1\} \cup \\
&\quad \{d_k : 0 \leq k \leq n + 2 \cdot m + 5\} \cup \{r'_i : 1 \leq i \leq m\} \cup \\
&\quad \{r_i : 0 \leq i \leq m\} \\
\Lambda &= \{Yes, No\} \\
LB &= \{1, 2\} \\
\mu_{\Pi(m,n)} &= [_1\ [_2\ ]_2^0\ ]_1^0 \\
\mathcal{M}_1 &= \# \\
\mathcal{M}_2 &= d_0 \\
im &= 2 \\
R(m,n) &= R^1(m,n) \cup R^2(m,n) \cup R^3(m,n)
\end{aligned}
$$

y, además,

$R^1(m,n)$ es el conjunto de las siguientes reglas:

(1) $[_2 d_0]_2^0 \rightarrow [_2 d_1]_2^+ [_2 d_1]_2^-$

(2) $[_2 d_k]_2^+ \rightarrow [_2 d_{k+1}]_2^+ [_2 d_{k+1}]_2^-$ $(1 \leq k < n)$

$[_2 d_k]_2^- \rightarrow [_2 d_{k+1}]_2^+ [_2 d_{k+1}]_2^-$ $(1 \leq k < n)$

(3) $[_2 x_{i,1} \rightarrow r'_i]_2^+, [_2 x_{i,1} \rightarrow \lambda]_2^-$ $(1 \leq i \leq m)$

$[_2 \overline{x}_{i,1} \rightarrow r'_i]_2^-, [_2 \overline{x}_{i,1} \rightarrow \lambda]_2^+$ $(1 \leq i \leq m)$

(4) $[_2 x_{i,j} \rightarrow x_{i,j-1}]_2^+, [_2 x_{i,j} \rightarrow x_{i,j-1}]_2^-$ $(1 \leq i \leq m, 2 \leq j \leq n)$

$[_2 \overline{x}_{i,j} \rightarrow \overline{x}_{i,j-1}]_2^+, [_2 \overline{x}_{i,j} \rightarrow \overline{x}_{i,j-1}]_2^-$ $(1 \leq i \leq m, 2 \leq j \leq n)$

(5) $[_2 d_n]_2^+ \rightarrow [_2\ ]_2^0 d_{n+1}, [_2 d_n]_2^- \rightarrow [_2\ ]_2^0 d_{n+1}$

$R^2(m,n)$ es el conjunto de las siguientes reglas:

(6) $[_2 r'_i \rightarrow r_i]_2^0$ $(1 \leq i \leq m)$

(7) $[_1 d_{n+1} \rightarrow d_{n+2} c_1]_1^0$

(8) $c_1 [_2\ ]_2^0 \rightarrow [_2 c_1]_2^-$

(9) $[_2 r_1]_2^- \rightarrow [_2\ ]_2^+ r_1$

(10) $[_2 r_i \rightarrow r_{i-1}]_2^+$ $(1 \leq i \leq m)$

(11) $[_2 c_k \rightarrow c_{k+1}]_2^+$ $(1 \leq k \leq m)$

(12) $r_1[_2\ ]_2^+ \to [_2 r_0]_2^-$

(13) $[_2 c_{m+1}]_2^- \to [_2\ ]_2^+ c_{m+1}$

(14) $[_1 d_k \to d_{k+1}]_1^0 \quad (n+2 \leq k \leq n+2 \cdot m+3)$

$R^3(m,n)$ es el conjunto de las siguientes reglas:

(15) $[_1 c_{m+1}]_1^0 \to [_1\ ]_1^+ c_{m+1}$

(16) $[_1 d_{n+2 \cdot m+4} \to d_{n+2 \cdot m+5}]_1^0$

(17) $[_1 d_{n+2 \cdot m+5}]_1^+ \to [_1\ ]_1^- Yes, [_1 d_{n+2 \cdot m+5}]_1^0 \to [_1\ ]_1^- No$

(18) $[_1 \#]_1^- \to [_1\ ]_1^- \#$

Se trata de justificar que, dados $m, n \in \mathbb{N}^+$, el sistema $\Pi(m,n)$ decide la satisfactibilidad de todas las fórmulas con m cláusulas y n variables. Sea ϕ una tal fórmula, $\phi \equiv Cl_1 \wedge \cdots \wedge Cl_m$, en forma normal conjuntiva y tal que su conjunto de variables es $Var(\phi) = \{x_1, \ldots, x_n\}$.

Codificamos dicha fórmula por el multiconjunto sobre $\Sigma(m,n)$ siguiente,

$$\begin{aligned} cod(\phi) = &\{x_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq n \wedge x_j \in Cl_i\} \cup \\ &\{\overline{x}_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq n \wedge \neg x_j \in Cl_i\} \end{aligned}$$

Es decir, este multiconjunto contiene un objeto $x_{i,j}$ por cada literal x_j, y un objeto $\overline{x}_{i,j}$ por cada literal $\neg x_j$, que aparezca en la cláusula Cl_i de la fórmula.

Veamos que existe una *única* computación de $\Pi(m,n)$ con entrada el multiconjunto $cod(\phi)$, y que se puede estructurar en tres etapas: una etapa de *generación y aplicación* de valoraciones, una etapa de *comprobación* de los valores obtenidos, y una etapa de *producción de la salida* del sistema.

Para ello, comencemos llamando *membrana interna* de una configuración de $\Pi(m,n)$ a toda membrana de dicha configuración con etiqueta 2 y describamos con detalle cada una de las etapas anteriores.

Fase 1: En la etapa de generación y aplicación de valoraciones se generan las 2^n valoraciones relevantes para ϕ, a la vez que se determinan cuáles de las cláusulas de la fórmula van haciéndose verdaderas. Este proceso lo llevan a cabo las reglas del conjunto R^1.

En el primer paso de la computación, la única regla aplicable es la regla (1), que da lugar a que aparezcan dos membranas internas, polarizadas positiva y negativamente. Estas membranas contienen, además del

multiconjunto de entrada, un objeto d_1.

Téngase en cuenta que, a lo largo de esta fase, una membrana interna que contiene un objeto d_k, con $1 \leq k \leq n$, y tiene carga positiva (respectivamente, negativa) codifica el hecho de que ya se ha considerado una valoración parcial de todas las variables $x_1, \ldots, x_{k-1}$, mientras que debemos comprobar qué cláusulas se hacen verdaderas si asignamos a la variable x_k el valor 1 (respectivamente, el valor 0).

Ahora se aplican n veces consecutivas las reglas de (2), (3) y (4). Las reglas de los dos últimos conjuntos son las que llevan a cabo la aplicación de la valoración parcial codificada en la membrana: las de (3), si la carga de esta es positiva (respectivamente, negativa), transforman los objetos $x_{i,1}$ (respectivamente, $\overline{x}_{i,1}$) en objetos r'_i, para indicar que la cláusula Cl_i de ϕ es satisfecha por la valoración parcial considerada, mientras que elimina los objetos $\overline{x}_{i,1}$ (respectivamente, $x_{i,1}$), ya que estos no darán lugar a que dicha cláusula sea verdadera; por otra parte, las reglas de (4) hacen decrecer el segundo subíndice de los objetos $x_{i,j}$, con $2 \leq j \leq n$, de forma que los objetos $x_{i,1}$ vayan representando las sucesivas variables, $x_2, \ldots, x_n$, en los siguientes pasos de la computación.

Finalmente, las reglas de (2) vuelven a duplicar las membranas internas en otras dos polarizadas positiva y negativamente, transformando el objeto d_k que contienen en el objeto d_{k+1} para indicar que ya se ha considerado una valoración parcial para las variables $x_1, \ldots, x_k$ y que a la variable x_{k+1} se le da el valor 1 o el valor 0, respectivamente (según la polaridad de la membrana).

Cuando las membranas contengan el objeto d_n, esta primera fase debe terminar, ya que se habrá asignado un valor a todas las variables de ϕ. Entonces las reglas de (5) expulsan estos objetos de las membranas internas a la membrana piel, como objetos d_{n+1}, a la par que asignan carga neutra a dichas membranas internas. Así, en este momento nos encontramos con la siguiente configuración del sistema:

- La membrana piel, con etiqueta 1, tiene carga neutra y contiene un objeto # y 2^n objetos d_{n+1}.
- Existen 2^n membranas con etiqueta 2, cada una de ellas hija de la membrana piel, con carga neutra y conteniendo determinados ob-

jetos r'_i que representan las distintas cláusulas que son satisfechas por la valoración codificada por la membrana.

Fase 2: En la etapa de comprobación de los valores obtenidos se determina si alguna de las membranas internas contiene objetos representando que todas las cláusulas de ϕ y, por tanto, también dicha fórmula, son verdaderas. Este proceso lo llevan a cabo las reglas de R^2.

Las primeras reglas que se aplican en esta fase son las reglas de (6) y (7), con el resultado de que los objetos r'_i de las membranas internas se convierten en objetos r_i y los 2^n objetos d_{n+1} de la membrana piel se transforman en 2^n objetos d_{n+2} y 2^n objetos c_1. Ahora solo se puede aplicar la regla (8), que envía un objeto c_1 a cada una de las membranas internas, polarizándola negativamente, y la regla correspondiente de (14), que convierte los objetos d_{n+2} en objetos d_{n+3}.

En esta fase debemos comprobar si alguna de estas membranas contiene al menos un objeto r_i, para cada $i = 1, \ldots, m$, ya que estos objetos indican que la cláusula Cl_i es verdadera por la valoración codificada por la membrana. Pues bien, trataremos de conseguir que si una membrana interna contiene el objeto c_k, con $1 \leq k \leq m+1$, entonces dicha membrana contiene, así mismo, los objetos $r_1, \ldots, r_{k-1}$ y, por tanto, la valoración que codifica hace verdaderas las cláusulas $Cl_1, \ldots, Cl_{k-1}$.

Para ello, si el objeto r_1 se encuentra en la membrana, es posible aplicar la regla (9), que cambia la carga de la membrana de negativa a positiva y expulsa el objeto a la membrana piel. Esta contendrá entonces t objetos r_1, provenientes de sendas membranas internas que ahora tendrán carga positiva, mientras que el resto de membranas internas siguen con carga negativa. Estas últimas membranas ya no evolucionarán en el resto de la computación.

Mediante la regla (10) se rotan los objetos r_i de las t membranas anteriores para que el objeto r_1 vaya representando las sucesivas cláusulas en los siguientes pasos de la computación. Por otra parte, la regla (11) aumenta el subíndice de los objetos c_k de dichas membranas, para indicar que hemos encontrado verdadera una cláusula más.

Finalmente, la regla (12) envía los objetos r_1 que se encuentran en la membrana piel a cada una de las t membranas internas con carga

positiva, convirtiéndolo en el objeto r_0 y cambiando la carga de estas a negativa, para que podamos repetir de nuevo las operaciones anteriores.

Reiterándolas m veces, se tiene la seguridad de que se habrán comprobado todas las cláusulas. A través de las reglas de (14) controlamos el número de pasos realizados, ya que transforma en pasos sucesivos en la membrana piel el objeto d_{n+3} en el objeto $d_{n+2\cdot m+3}$. Cuando este último objeto aparezca en la membrana piel, si alguna de las membranas internas codifica una valoración que hace cierta la fórmula ϕ, entonces dicha membrana contendrá un objeto c_{m+1}. Entonces la regla (13) expulsa dichos objetos a la piel, polarizando positivamente las membranas internas correspondientes, mientras que la regla (14) transforma el objeto $d_{n+2\cdot m+3}$ en el objeto $d_{n+2\cdot m+4}$.

Así, en este momento nos encontramos con la siguiente configuración del sistema:

- La membrana piel, con etiqueta 1, tiene carga neutra y contiene un objeto #, 2^n objetos $d_{n+2\cdot m+3}$ y t objetos c_{m+1}.
- Existen t membranas con etiqueta 2, cada una de ellas hija de la membrana piel, con carga positiva y conteniendo solamente objetos r_0.
- Existen $2^n - t$ membranas con etiqueta 2, cada una de ellas hija de la membrana piel, con carga negativa y conteniendo un objeto c_k, con $1 \leq k \leq m$, y objetos r_i, con $i \neq 1$.

Fase 3: En la etapa de producción de la salida se expulsa al entorno externo un objeto Yes, si la fórmula ϕ es satisfactible, o un objeto No, en caso contrario. Por último, se expulsa al entorno externo un objeto # en el último paso de la computación. Este proceso lo llevan a cabo las reglas de R^3.

Para decidir si la fórmula ϕ es o no satisfactible, basta observar si la membrana piel contiene algún objeto c_{m+1}. Si esto es así, entonces se puede aplicar la regla (15), que expulsa uno de estos objetos al entorno externo para cambiar la carga de la membrana piel a positiva.

Tanto si la membrana piel contiene objetos c_{m+1} como si no, la regla (16) transforma los objetos $d_{n+2\cdot m+4}$ en objetos $d_{n+2\cdot m+5}$. Uno de estos

sale entonces al entorno externo, aplicando las reglas de (17), como un objeto Yes, si la carga de la piel es positiva, o como un objeto No, si es neutra. En ambos casos la membrana piel se polariza negativamente, para que se pueda aplicar la regla (18), que expulsa el objeto # al entorno externo. La computación finaliza en este momento, ya que no es posible aplicar ninguna regla más.

Vamos a demostrar a continuación que la familia $\mathbf{\Pi_{SAT}}$ resuelve el problema **SAT** en tiempo lineal. Para ello, sean ϕ una fórmula en forma normal conjuntiva, con m cláusulas y n variables, y $\mathcal{C} = \{C^i\}_{i<r}$ una computación de $\Pi(m,n)$ con entrada $cod(\phi)$.

Obsérvese que las únicas reglas de división de membranas que se podrían aplicar a las configuraciones de $\mathcal{C}$ son las de (1) y (2), y estas siempre dan lugar a membranas polarizadas, bien positiva, bien negativamente. Esto nos permite introducir el concepto de valoración codificada por una membrana interna (que probaremos posteriormente que se pueden identificar con valoraciones parciales de la fórmula ϕ).

Definición 11.5. *Sean C^i una configuración de $\mathcal{C}$ y mb una membrana interna de C^i. La* valoración codificada por mb en C^i *es la cadena sobre el alfabeto $\{0,1\}$ definida por recursión como sigue:*

- *Caso 1: $i = 0$.*

  En este caso, la valoración codificada por la membrana mb es la cadena vacía.

- *Caso 2: $i > 0$.*

  Si mb se ha obtenido por división de una membrana mb' de la configuración C^{i-1}, entonces la valoración codificada por mb en C^i es $\sigma 1$ (respectivamente, $\sigma 0$), si tiene carga positiva (respectivamente, carga negativa), donde σ es la valoración codificada por mb' en C^{i-1}.

  En caso contrario, la valoración codificada por mb en C^i es la valoración codificada por mb en C^{i-1}.

Veamos a continuación que en la fase de generación del sistema $\Pi(m,n)$ se generan 2^n membranas internas, cada una de ellas codificando una valoración distinta de longitud n (de donde se deduce que, globalmente, codifican todas las posibles cadenas sobre el alfabeto $\{0,1\}$ de longitud n). Además,

probaremos que en las dos fases subsiguientes no se produce ninguna división de membranas, por lo que en ellas las valoraciones codificadas por las membranas internas permanecen inalteradas en dichas fases.

Proposición 11.6.

1. *Para cada* $i = 0, \ldots, n$, *la configuración* C^i *verifica las siguientes propiedades:*
   - *La membrana piel tiene carga neutra y contiene un objeto* $\#$.
   - *En la estructura de membranas* μ^i *existen exactamente* 2^i *membranas internas, y cada una de ellas proviene de la división de una membrana interna de* μ^{i-1}.
   - *Si* $i = 0$, *entonces la única membrana interna de* μ^i *tiene carga neutra. Si* $i > 0$, *entonces todas las membranas internas están polarizadas, y su carga es positiva (respectivamente, negativa) si el último símbolo de la valoración que codifica es 1 (respectivamente, es 0).*
   - *Cada membrana interna contiene un único objeto* d_i.
     Además, el resto de objetos que contiene pertenecen al conjunto $\{x_{i,j}, \overline{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\} \cup \{r'_i : 1 \leq i \leq m\}$.
   - *Para cada cadena* w *sobre el alfabeto* $\{0,1\}$ *de longitud* i *existe una membrana interna de* μ^i *que codifica la valoración* w.
2. *La configuración* C^{n+1} *verifica las siguientes propiedades:*
   - *La membrana piel tiene carga neutra y contiene un objeto* $\#$ *y* 2^n *objetos* d_{n+1}.
   - *En la estructura de membranas* μ^{n+1} *existen exactamente* 2^n *membranas internas.*
   - *Todas las membranas internas tienen carga neutra.*
   - *Los objetos contenidos en cada membrana interna pertenecen al conjunto* $\{x_{i,j}, \overline{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\} \cup \{r'_i : 1 \leq i \leq m\}$.
   - *Para cada cadena* w *sobre el alfabeto* $\{0,1\}$ *de longitud* n *existe una membrana interna de* μ^{n+1} *que codifica la valoración* w.
3. *Para cada* $i > n+1$, *la configuración* C^i *verifica las siguientes propiedades:*

- *En la estructura de membranas μ^i existen exactamente 2^n membranas internas.*
- *Para cada cadena w sobre el alfabeto $\{0,1\}$ de longitud n existe una membrana interna de μ^i que codifica la valoración w.*

Demostración.

1. Por inducción sobre i.

   - Caso base: $i = 0$.

     La configuración inicial de la computación $\mathcal{C}$ es la configuración $C^0 = (\mu^0, M^0_{env}, M^0_1, M^0_2)$, donde $\mu^0 = [_1\,[_2\,]^0_2\,]^0_1$ y

     $$M^0_{env} = \emptyset \qquad M^0_1 = \# \qquad M^0_2 = d_0 + cod(\phi)$$

     Por definición, la valoración codificada por la única membrana interna de μ^0 es la cadena vacía.

   - Paso inductivo: $i \to i+1$.

     Sea $i < n$ y supongamos por hipótesis de inducción que

     - La membrana piel tiene carga neutra y contiene un objeto #.
     - En la estructura de membranas μ^i existen exactamente 2^i membranas internas, y cada una de ellas proviene de la división de una membrana interna de μ^{i-1}.
     - Si $i = 0$, entonces la única membrana interna de μ^i tiene carga neutra. Si $i > 0$, entonces todas las membranas internas están polarizadas, y su carga es positiva (respectivamente, negativa) si el último símbolo de la valoración que codifica es 1 (respectivamente, es 0).
     - Cada membrana interna contiene un único objeto d_i.
       Además, el resto de objetos que contiene pertenecen al conjunto $\{x_{i,j}, \overline{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\} \cup \{r'_i : 1 \leq i \leq m\}$.
     - Para cada cadena w sobre el alfabeto $\{0,1\}$ de longitud i existe una membrana interna de μ^i tal que la valoración codificada por dicha membrana coincide con la cadena w.

     Entonces, las únicas reglas aplicables a la configuración C^i son la (1), si $i = 0$, y las de (2), (3) y (4), si $i > 0$. Puesto que de ellas las únicas reglas que no son de tipo (a) son las de (1) y (2)

y, además, las membranas internas contienen el objeto necesario para activarlas, aquellas se aplicarán en cada una de estas. Por tanto,

- La membrana piel tiene carga neutra y contiene un objeto #.
- En la estructura de membranas μ^{i+1} existen exactamente 2^{i+1} membranas internas, ya que cada membrana interna de μ^i se divide en dos.
- Todas las membranas internas de μ^{i+1} están polarizadas, y su carga es positiva (respectivamente, negativa) si el último símbolo de la valoración que codifica es 1 (respectivamente, es 0), por definición.
- Cada membrana interna contiene un único objeto d_{i+1}, como resultado de aplicar las reglas de (1) y (2).
  Además, el resto de objetos que contiene pertenecen al conjunto $\{x_{i,j}, \overline{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\} \cup \{r'_i : 1 \leq i \leq m\}$, ya que este conjunto es cerrado bajo la aplicación de las reglas de (3) y (4).
- Como consecuencia de que cada membrana interna de μ^i se divide en dos, una con carga positiva y otra con carga negativa, se deduce que para cada cadena w sobre el alfabeto $\{0, 1\}$ de longitud $i + 1$ existe una membrana interna de μ^{i+1} que codifica la valoración w.

2. Las únicas reglas aplicables a la configuración C^n son las de (3), (4) y (5), de lo que se deducen las propiedades requeridas.

3. Teniendo en cuenta que no hay ninguna regla que introduzcan objetos d_k, con $1 \leq k < n$, en las membranas internas, y que estos son los únicos objetos que pueden activar una regla de división de membranas, se deduce que el conjunto de membranas internas no cambia a partir de la configuración C^{n+1}. De aquí se obtienen fácilmente las propiedades requeridas.

□

Del resultado anterior es claro que podemos identificar las valoraciones codificadas por las membranas internas presentes en la configuración C^{n+1} (o en las posteriores) con el conjunto de valoraciones relevantes para la fórmula

ϕ. En efecto, si $\sigma = \sigma(1)\ldots\sigma(n) \in \{0,1\}^*$ es una valoración codificada por una tal membrana, entonces consideramos que $\sigma(x_j) = \sigma(j)$, para cada $j = 1,\ldots,n$.

Finalicemos la corrección de la etapa de generación y aplicación de valoraciones.

Proposición 11.7. *Sean σ una valoración relevante para la fórmula ϕ, mb la membrana interna de C^{n+1} que codifica la valoración σ, y $mb_1,\ldots,mb_{n+1}$ una sucesión de membranas tales que:*

1. *mb_k es una membrana interna de μ^k, para cada $k = 1,\ldots,n$, y $mb_n = mb_{n+1} = mb$.*

2. *mb_{k+1} proviene de la división de mb_k, para cada $k = 1,\ldots,n-1$.*

3. *mb_k tiene carga positiva, si $\sigma(k) = 1$, y carga negativa, si $\sigma(k) = 0$, para cada $k = 1,\ldots,n$.*

Entonces, para cada $k = 0,\ldots,n$ se verifican las siguientes propiedades:

- *La membrana mb_{k+1} contiene un objeto $x_{i,j-k}$ por cada literal $x_j \in Cl_i$ y un objeto $\overline{x}_{i,j-k}$ por cada literal $\neg x_j \in Cl_i$, con $j = k+1,\ldots,n$.*

- *La membrana mb_1 no contiene ningún objeto r'_i, con $1 \leq i \leq m$. Además, si $k > 0$, entonces para cada $j = 1,\ldots,k$ la membrana mb_{k+1} contiene exactamente tantos objetos r'_i como literales x_j, si $\sigma(j) = 1$, o como literales $\neg x_j$, si $\sigma(j) = 0$, pertenezcan a las cláusulas Cl_i, para todo $i = 1,\ldots,m$.*

Demostración. Por inducción sobre k.

- Caso base: $k = 0$.

  La regla (1) es la única aplicable a la configuración inicial de $\mathcal{C}$. En esta existe una sola membrana inicial, que contiene el multiconjunto de objetos $d_0 + cod(\phi)$, y mb_1 proviene de la división de esta membrana. Por tanto, se tienen las propiedades requeridas.

- Paso inductivo: $k \to k+1$.

  Sea $k < n$ y supongamos por hipótesis de inducción que

- La membrana mb_{k+1} contiene un objeto $x_{i,j-k}$ por cada literal $x_j \in Cl_i$ y un objeto $\overline{x}_{i,j-k}$ por cada literal $\neg x_j \in Cl_i$, con $j = k+1, \ldots, n$ (en este contexto, podemos decir que un objeto $x_{i,j}$ (respectivamente, $\overline{x}_{i,j}$) de mb_{k+1} representa que $x_j \in Cl_i$ (respectivamente, $\neg x_j \in Cl_i$)).
- La membrana mb_1 no contiene ningún objeto r'_i, con $1 \leq i \leq m$. Además, si $k > 0$, entonces para cada $j = 1, \ldots, k$ la membrana mb_{k+1} contiene exactamente tantos objetos r'_i como literales x_j, si $\sigma(j) = 1$, o como literales $\neg x_j$, si $\sigma(j) = 0$, pertenezcan a las cláusulas Cl_i, para todo $i = 1, \ldots, m$.

Las únicas reglas aplicables a la configuración C^{k+1} en la membrana mb_{k+1}, además de las de división de membranas, son las reglas de (3) y (4). Por tanto,

- Si $x_j \in Cl_i$ (respectivamente, $\neg x_j$), con $j = k+2, \ldots, n$, entonces la membrana mb_{k+1} contiene un objeto $x_{i,j-k}$ (respectivamente, $\overline{x}_{i,j-k}$), que por la aplicación de las reglas de (4) se transforma en el objeto $x_{i,j-(k+1)}$ (respectivamente, $\overline{x}_{i,j-(k+1)}$) de mb_{k+2}.
- Los objetos r'_i contenidos en la membrana mb_{k+1} también los contiene la membrana mb_{k+2}.

  Si $\sigma(k+1) = 1$, entonces la membrana mb_{k+1} tiene carga positiva. Por tanto, las reglas de (4) transforman cada objeto $x_{i,1}$ de mb_{k+1}, que representan que $x_{k+1} \in Cl_i$, en un objeto r'_i de mb_{k+2}, y eliminan cada objeto $\overline{x}_{i,1}$ de mb_{k+1}, ya que representan que $\neg x_{k+1} \in Cl_i$.

  Si $\sigma(k+1) = 0$, entonces la membrana mb_{k+1} tiene carga negativa. Por tanto, las reglas de (4) transforman cada objeto $\overline{x}_{i,1}$ de mb_{k+1}, que representan que $\neg x_{k+1} \in Cl_i$, en un objeto r'_i de mb_{k+2}, y eliminan cada objeto $x_{i,1}$ de mb_{k+1}, ya que representan que $x_{k+1} \in Cl_i$.

□

Sea σ una valoración relevante para la fórmula ϕ. Por la proposición 11.6 existe una membrana interna en μ^{n+1} que codifica la valoración σ. Además, dicha membrana, que notaremos por $mb(\sigma)$, permanece en las configuraciones posteriores.

Estudiemos qué carga tiene y qué objetos contiene cada una de las membranas de las estructura μ^{n+1}. La proposición 11.6 nos asegura que todas las membranas tienen carga neutra y que la piel contiene el objeto # y 2^n objetos d_{n+1}. Por otra parte, de la proposición 11.7 se deduce que para cada valoración σ relevante para ϕ la membrana $mb(\sigma)$ contiene, para cada $i = 1, \ldots, m$, t_i^σ objetos r_i', donde t_i^σ es el número de literales de Cl_i verdaderos por σ.

Probemos a continuación la corrección de la etapa de comprobación de los valores obtenidos.

Proposición 11.8. *Para cada $k = 0, \ldots, m$, la configuración $C^{n+2\cdot k+3}$ verifica las siguientes propiedades:*

- *La membrana piel de $\mu^{n+2\cdot k+3}$ tiene carga neutra y contiene un objeto # y 2^n objetos $d_{n+2\cdot k+3}$.*
- *Dada σ una valoración relevante para ϕ, la membrana interna $mb(\sigma)$ de $\mu^{n+2\cdot k+3}$ tiene carga negativa. Además,*
  - *Si existe $j \leq k$ tal que $t_j^\sigma = 0$, entonces $mb(\sigma)$ contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq k \wedge t_j^\sigma = 0\}$.*
  - *En caso contrario, $mb(\sigma)$ contiene el objeto c_{k+1} y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_k^\sigma} + r_1^{t_{k+1}^\sigma} \ldots r_{m-k}^{t_m^\sigma}$.*

Por otra parte, la configuración $C^{n+2\cdot m+4}$ verifica las siguientes propiedades:

- *La membrana piel de $\mu^{n+2\cdot m+4}$ tiene carga neutra y contiene un objeto #, 2^n objetos $d_{n+2\cdot m+4}$ y t objetos c_{m+1}, donde t es el número de valoraciones σ relevantes para ϕ tales que $t_i^\sigma > 0$, para todo $i = 1, \ldots, m$.*
- *Dada σ una valoración relevante para ϕ, la membrana interna $mb(\sigma)$ de $\mu^{n+2\cdot m+4}$ cumple que:*
  - *Si existe $j \leq m$ tal que $t_j^\sigma = 0$, entonces la membrana $mb(\sigma)$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq k \wedge t_j^\sigma = 0\}$.*
  - *En caso contrario, la membrana $mb(\sigma)$ tiene carga positiva y contiene $t_1^\sigma + \cdots + t_m^\sigma$ objetos r_0.*

Demostración. Por inducción sobre k.

- Caso base: $k = 0$.

  Basta observar que las únicas reglas aplicables a la configuración C^{n+1} son las de (6), en las membranas internas, y la de (7), en la membrana piel. Así, en la configuración C^{n+2} se verifica lo siguiente:

  - La membrana piel de μ^{n+2} tiene carga neutra y contiene un objeto # y 2^n objetos d_{n+2} y c_1.
  - Para cada valoración σ, la membrana interna $mb(\sigma)$ de μ^{n+2} tiene carga neutra y contiene t_i^σ objetos r_i, para cada $i = 1, \ldots, m$.

  Ahora, las únicas reglas aplicables a la configuración C^{n+2} son la (8) y la correspondiente de (14). Además, la regla (8) es de tipo (b), luego solamente se puede aplicar una por cada membrana interna. Así, en la configuración C^{n+3} se verifica lo siguiente:

  - La membrana piel de μ^{n+3} tiene carga neutra y contiene un objeto # y 2^n objetos d_{n+3}.
  - Para cada valoración σ, la membrana interna $mb(\sigma)$ de μ^{n+2} tiene carga negativa y contiene un objeto c_1 y t_i^σ objetos r_i, para cada $i = 1, \ldots, m$.

- Paso inductivo: $k \to k + 1$.

  Sea $k < m$ y supongamos por hipótesis de inducción que

  - La membrana piel de $\mu^{n+2\cdot k+3}$ tiene carga neutra y contiene un objeto # y 2^n objetos $d_{n+2\cdot k+3}$.
  - Para cada valoración, σ, la membrana interna, $mb(\sigma)$, de $\mu^{n+2\cdot k+3}$ tiene carga negativa. Además,
    1. Si existe $j \leq k$ tal que $t_j^\sigma = 0$ (es decir, tal que $\sigma(Cl_j) = 0$), entonces $mb(\sigma)$ contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma + \cdots + t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, con $j' = \min\{j : j \leq k \wedge t_j^\sigma = 0\}$.
    2. En caso contrario, $mb(\sigma)$ contiene el objeto c_{k+1} y el multiconjunto $r_0^{t_1^\sigma + \cdots + t_k^\sigma} + r_1^{t_{k+1}^\sigma} \ldots r_{m-k}^{t_m^\sigma}$.

Por tanto, la única regla aplicable en la configuración $C^{n+2\cdot k+3}$ en la membrana piel es la correspondiente regla de (14). Con respecto a las membranas internas, no existe ninguna regla aplicable tanto en las que se encuentran en el caso 1, como en las que no se encuentran en dicho caso, pero verifican que $t^{\sigma}_{k+1} = 0$. En el resto, puesto que contienen al menos un objeto r_1, es aplicable la regla (9). Sea t el número de membranas internas que se encuentran en esta última situación.

La configuración $C^{n+2\cdot k+4}$ verifica entonces lo siguiente:

- La membrana piel de $\mu^{n+2\cdot k+4}$ tiene carga neutra y contiene un objeto $\#$, 2^n objetos $d_{n+2\cdot k+4}$ y t objetos r_1.
- Para cada valoración, σ, la membrana interna, $mb(\sigma)$, de $\mu^{n+2\cdot k+4}$ cumple que:
  1. Si existe $j \leq k$ tal que $t^{\sigma}_j = 0$, entonces $mb(\sigma)$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t^{\sigma}_1+\cdots+t^{\sigma}_{j'-1}} + r_2^{t^{\sigma}_{j'+1}} \ldots r^{t^{\sigma}_m}_{m-j'+1}$, donde $j' = \min\{j : j \leq k \wedge t^{\sigma}_j = 0\}$.
  2. Si no existe $j \leq k$ tal que $t^{\sigma}_j = 0$, pero $t^{\sigma}_{k+1} = 0$, entonces $mb(\sigma)$ tiene carga negativa y contiene el objeto c_{k+1} y el multiconjunto $r_0^{t^{\sigma}_1+\cdots t^{\sigma}_k} + r_2^{t^{\sigma}_{k+2}} \ldots r^{t^{\sigma}_m}_{m-k}$.
  3. En caso contrario, tiene carga positiva y contiene el objeto c_{k+1} y el multiconjunto $r_0^{t^{\sigma}_1+\cdots+t^{\sigma}_k} + r_1^{t^{\sigma}_k} \ldots r^{t^{\sigma}_m}_{m-k}$.

Entonces, a dicha configuración solo es aplicable la regla correspondiente de (14) en la membrana piel, ninguna regla en las membranas internas que se encuentren en los casos 1 y 2 anteriores, y las reglas de (10),(11) y una sola de (12) en las membranas internas que se encuentren en el caso 3.

Por consiguiente, la configuración $C^{n+2\cdot(k+1)+3}$ verifica lo siguiente:

- La membrana piel de $\mu^{n+2\cdot(k+1)+3}$ tiene carga neutra y contiene un objeto $\#$ y 2^n objetos $d_{n+2\cdot(k+1)+3}$.
- Para cada valoración, σ, la membrana interna, $mb(\sigma)$, de la estructura $\mu^{n+2\cdot(k+1)+3}$ tiene carga negativa y, además,
  1. Si existe $j \leq k+1$ tal que $t^{\sigma}_j = 0$, entonces tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t^{\sigma}_1+\cdots+t^{\sigma}_{j'-1}} + r_2^{t^{\sigma}_{j'+1}} \ldots r^{t^{\sigma}_m}_{m-j'+1}$, donde $j' = \min\{j : j \leq k+1 \wedge t^{\sigma}_j = 0\}$.

2. En caso contrario, contiene el objeto c_{k+2} y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{k+1}^\sigma} + r_1^{t_{k+2}^\sigma} \ldots r_{m-(k+1)}^{t_m^\sigma}$.

Así pues, acabamos de probar que la configuración $C^{n+2\cdot m+3}$ verifica las siguientes propiedades:

- La membrana piel de $\mu^{n+2\cdot m+3}$ tiene carga neutra y contiene un objeto # y 2^n objetos $d_{n+2\cdot m+3}$.
- Dada una valoración, σ, relevante para ϕ, la membrana interna, $mb(\sigma)$, de $\mu^{n+2\cdot m+3}$ tiene carga negativa. Además,
  - Si existe $j \leq m$ tal que $t_j^\sigma = 0$, entonces la membrana $mb(\sigma)$ contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.
  - En caso contrario, la membrana $mb(\sigma)$ contiene el objeto c_{m+1} y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_m^\sigma}$.

Por tanto, las únicas reglas aplicables a esta configuración son la regla correspondiente de (14), en la membrana piel, y la regla (13), únicamente en las membranas internas $mb(\sigma)$ tales que $t_i^\sigma > 0$, para todo $i = 1, \ldots, m$. Así, la configuración $C^{n+2\cdot m+4}$ verifica lo que se pide. □

Para demostrar la corrección de la etapa de producción de la salida distinguimos dos casos, dependiendo de si existe o no una valoración que satisfaga la fórmula ϕ.

Proposición 11.9. *Supongamos que para toda valoración, σ, relevante para ϕ existe $j \leq m$ tal que $t_j^\sigma = 0$ (es decir, tal que $\sigma(Cl_j) = 0$). Entonces la configuración $C^{n+2\cdot m+7}$ verifica las siguientes propiedades:*

- *El entorno externo contiene un objeto No y un objeto #.*
- *La membrana piel tiene carga negativa y contiene $2^n - 1$ objetos $d_{n+2\cdot m+5}$.*
- *Cada una de las membranas internas, $mb(\sigma)$, de $\mu^{n+2\cdot m+7}$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.*

En consecuencia, ninguna regla es aplicable a esta configuración.

Demostración. De la hipótesis y de la proposición 11.8 se deduce que la configuración $C^{n+2\cdot m+4}$ verifica lo siguiente:

- La membrana piel de $\mu^{n+2\cdot m+4}$ tiene carga neutra y contiene un objeto $\#$ y 2^n objetos $d_{n+2\cdot m+4}$.
- Cada una de las membranas internas, $mb(\sigma)$, de $\mu^{n+2\cdot m+4}$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.

Por tanto, en la membrana piel la única regla aplicable a dicha configuración es la regla (16), y en las membranas internas no hay ninguna regla aplicable. Luego en la configuración $C^{n+2\cdot m+5}$ la membrana piel tiene carga neutra y contiene un objeto $\#$ y 2^n objetos $d_{n+2\cdot m+5}$ y las membranas internas no se han modificado.

Ahora a esta configuración solamente es aplicable la regla adecuada de (17) en la membrana piel, que expulsa al entorno un único objeto No y polariza negativamente la membrana. Así, en la configuración $C^{n+2\cdot m+6}$ la membrana piel tiene carga negativa y contiene un objeto $\#$ y $2^n - 1$ objetos $d_{n+2\cdot m+5}$, mientras que las membranas internas no se han modificado y el entorno externo contiene un objeto No.

Por último, a esta configuración solo es aplicable la regla (18). De donde se deduce que la configuración $C^{n+2\cdot m+7}$ cumple lo requerido. □

Proposición 11.10. *Supongamos que existen $t > 0$ valoraciones, σ, relevantes para ϕ tales que $t_i^\sigma > 0$, para todo $i = 1, \ldots, m$. Entonces la configuración $C^{n+2\cdot m+7}$ verifica las siguientes propiedades:*

- *El entorno externo contiene un objeto Yes y un objeto $\#$.*
- *La membrana piel tiene carga negativa y contiene $t - 1$ objetos c_{m+1} y $2^n - 1$ objetos $d_{n+2\cdot m+5}$.*
- *Dada una valoración, σ, relevante para ϕ,*
  - *Si existe $j \leq m$ tal que $t_j^\sigma = 0$, entonces $mb(\sigma)$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.*
  - *Si $t_j^\sigma > 0$, para todo $j = 1, \ldots, m$, entonces $mb(\sigma)$ tiene carga positiva y contiene $t_1^\sigma + \cdots + t_m^\sigma$ objetos r_0.*

En consecuencia, ninguna regla es aplicable a esta configuración.

Demostración. De la hipótesis y de la proposición 11.8 se deduce que la configuración $C^{n+2\cdot m+4}$ verifica lo siguiente:

- La membrana piel de $\mu^{n+2\cdot m+4}$ tiene carga neutra y contiene un objeto $\#$, 2^n objetos $d_{n+2\cdot m+4}$ y t objetos c_{m+1}.
- Dada una valoración, σ, relevante para ϕ, la membrana interna $mb(\sigma)$ de $\mu^{n+2\cdot m+4}$ cumple que:
  - Si existe $j \leq m$ tal que $t_j^\sigma = 0$, entonces $mb(\sigma)$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.
  - En caso contrario, la membrana $mb(\sigma)$ tiene carga positiva y contiene $t_1^\sigma + \cdots + t_m^\sigma$ objetos r_0.

Por tanto, en la membrana piel las reglas aplicables a dicha configuración son las reglas (15) y (16), y en las membranas internas no hay ninguna regla aplicable. Luego en la configuración $C^{n+2\cdot m+5}$ la membrana piel tiene carga positiva y contiene un objeto $\#$, 2^n objetos $d_{n+2\cdot m+5}$ y $t-1$ objetos c_{m+1}, mientras que las membranas internas no se han modificado.

Ahora a esta configuración solamente es aplicable la regla adecuada de (17) en la membrana piel, que expulsa al entorno un único objeto Yes y polariza negativamente la membrana. Así, en la configuración $C^{n+2\cdot m+6}$ la membrana piel tiene carga negativa y contiene un objeto $\#$, $2^n - 1$ objetos $d_{n+2\cdot m+5}$ y t objetos c_{m+1}, mientras que las membranas internas no se han modificado y el entorno externo contiene un objeto Yes.

Por último, a esta configuración solo es aplicable la regla (18), de donde se deduce que la configuración $C^{n+2\cdot m+7}$ cumple lo requerido. □

Finalmente, vamos a demostrar que la familia $\mathbf{\Pi_{SAT}}$ resuelve el problema **SAT** en tiempo lineal.

Teorema 11.11. *Consideremos la función par,* $\mathbf{p} : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$, *dada por* $\mathbf{p}(m,n) = \frac{(m+n+1)\cdot(m+n)}{2} + n$. *Entonces la familia* $\mathbf{\Pi} = \left(\Pi(\mathbf{p}(m,n))\right)_{m,n\in\mathbb{N}^+}$, *donde* $\Pi(\mathbf{p}(m,n)) = \Pi(m,n)$, *verifica las siguientes propiedades:*

1. *Para todo* $m, n \in \mathbb{N}^+$, *el sistema* $\Pi(\mathbf{p}(m,n)) \in \mathcal{AM}$.

2. *Para todo* $m, n \in \mathbb{N}^+$, *el sistema* $\Pi(\mathbf{p}(m,n))$ *tiene membrana de entrada.*

3. *La familia* $\mathbf{\Pi}$ *es polinomialmente uniforme, por máquinas de Turing.*

4. *Consideremos la función cod* : $I_{\mathbf{SAT}} \to \bigcup_{m,n\in\mathbb{N}^+} \Sigma(m,n)$ *definida, para cada fórmula* $\phi \in I_{\mathbf{SAT}}$, *como*

$$\begin{aligned} cod(\phi) =& \{x_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq n \wedge x_j \in Cl_i\} \cup \\ & \{\overline{x}_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq n \wedge \neg x_{i,j} \in Cl_i\} \end{aligned}$$

   y la función $s : I_{\mathbf{SAT}} \to \mathbb{N}^+$ *definida, para cada fórmula* $\phi \in I_{\mathbf{SAT}}$, *como* $s(\phi) = \mathbf{p}(m,n)$, *donde* m *es el número de cláusulas y* n *el número de variables de* ϕ. *Entonces,*

   - *Las funciones cod y s son computables en tiempo polinomial en el tamaño de la fórmula* ϕ.
   - *Para cada* $\phi \in I_{\mathbf{SAT}}$, *se tiene que* $cod(\phi) \in I_{\Pi(s(\phi))}$.
   - *Para cada fórmula* ϕ, *toda computación del sistema* $\Pi(s(\phi))$ *con entrada* $cod(\phi)$ *realiza un número de pasos* lineal *en el tamaño de la fórmula* ϕ.
   - *Si la fórmula* ϕ *es satisfacible, entonces toda computación del sistema* $\Pi(s(\phi))$ *con entrada* $cod(\phi)$ *es de aceptación.*
   - *Si existe una computación del sistema* $\Pi(s(\phi))$ *con entrada* $cod(\phi)$ *que es de aceptación, entonces la fórmula* ϕ *es satisfacible.*

Demostración.

1. Las proposiciones 11.9 y 11.10 nos aseguran que $\Pi(\mathbf{p}(m,n)) \in \mathcal{AM}$, para todo $m, n \in \mathbb{N}^+$; es decir, $\Pi(\mathbf{p}(m,n))$ es un sistema P con membranas activas que es válido como dispositivo de aceptación de lenguajes.

2. Por definición, el sistema $\Pi(\mathbf{p}(m,n))$ tiene membrana de entrada, para todo $m, n \in \mathbb{N}^+$.

3. Es claro que la familia $\mathbf{\Pi}$ es uniforme. Además, para cada $m, n \in \mathbb{N}^+$, el sistema $\Pi(\mathbf{p}(m,n))$ cumple que:

- El tamaño del alfabeto de entrada es $2 \cdot m \cdot n$.
- El tamaño del alfabeto de trabajo es $2 \cdot m \cdot n + 5 \cdot m + n + 11$.
- El tamaño del alfabeto de salida es 2.
- El tamaño del conjunto de etiquetas es 2.
- El grado de la estructura de membranas inicial es 2.
- El tamaño de los multiconjuntos de objetos asociados a las membranas iniciales es 1.
- El número total de reglas es $4 \cdot m \cdot n + 5 \cdot m + 2 \cdot n + 13$.
- La longitud máxima de las reglas es 7.

Por consiguiente, como la inversa de la función par es computable en tiempo polinomial, la familia Π se puede construir en tiempo polinomial.

4. Las funciones *cod* y s son trivialmente computables en tiempo polinomial en el tamaño de ϕ.

   - Por definición, $cod(\phi) \in \Sigma(m, n)$, para toda fórmula ϕ.
   - Sea ϕ una fórmula proposicional en forma normal conjuntiva. Las proposiciones 11.9 y 11.10 nos muestran que existe una única computación de $\Pi(s(\phi))$ con entrada $cod(\phi)$, y que realiza $n+2 \cdot m+7$ pasos, que es lineal en el tamaño de ϕ.
   - Sea ϕ una fórmula satisfacible. Entonces de la proposición 11.10 se deduce que la única computación de $\Pi(s(\phi))$ con entrada $cod(\phi)$ es de aceptación.
   - Sea ϕ una fórmula tal que la la única computación de $\Pi(s(\phi))$ con entrada $cod(\phi)$ es de aceptación. De la proposición 11.9 se deduce que ϕ es satisfacible.

□

De este resultado se deducen los siguientes corolarios.

Corolario 11.12. $\mathbf{SAT} \in \mathbf{PMC}_{\mathcal{AM}^{fi}}$.

Corolario 11.13. $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}^{fi}}$.

Demostración. Basta tener en cuenta que la clase $\mathbf{PMC}_{\mathcal{AM}^{fi}}$ es cerrada bajo reducibilidad en tiempo polinomial, el corolario 11.12 y que **SAT** es un problema **NP**–completo. □

11.3. Resolución del problema VALIDITY

En esta sección presentamos una familia de sistemas de la clase $\mathcal{AM}$, considerados como sistemas reconocedores, que resuelve el problema **VALIDITY** en tiempo *lineal*. Así, puesto que este problema es **coNP**–completo, se deducirá que la clase **coNP** está contenida en la clase $\mathbf{PMC}^{fi}_{\mathcal{AM}}$.

El problema de la validez (o tautología) de la lógica proposicional, **VALIDITY**, es el siguiente:

> *Dada una fórmula del cálculo proposicional, en forma normal conjuntiva, determinar si es o no una tautología; es decir, si toda valoración de verdad de sus variables hace cierta la fórmula.*

Consideramos ahora la familia de sistemas P con membranas activas de aceptación de lenguajes, $\mathbf{\Pi_{VAL}} = \big(\Pi(m,n)\big)_{m,n\in\mathbb{N}^+}$, de tal manera que el sistema $\Pi(m,n)$ se encarga de procesar todas aquellas fórmulas que poseen m cláusulas y n variables. Para cada $m,n \in \mathbb{N}^+$, el sistema $\Pi(m,n)$ se define como sigue:

$$\Pi(m,n) = \big(\Sigma(m,n), \Gamma(m,n), \Lambda, \#, LB, \mu_{\Pi(m,n)}, \mathcal{M}_1, \mathcal{M}_2, R(m,n), im, ext\big)$$

donde

$$\begin{aligned}
\Sigma(m,n) &= \{x_{i,j}, \overline{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\} \\
\Gamma(m,n) &= \Sigma(m,n) \cup \{e, e', Yes, No, \#\} \cup \{c_k : 1 \leq k \leq m+1\} \cup \\
&\quad \{d_k : 0 \leq k \leq n + 2 \cdot m + 7\} \cup \{r'_i : 1 \leq i \leq m\} \cup \\
&\quad \{r_i : 0 \leq i \leq m\} \\
\Lambda &= \{Yes, No\} \\
LB &= \{1, 2\} \\
\mu_{\Pi(m,n)} &= [_1\, [_2\,]_2^0\,]_1^0 \\
\mathcal{M}_1 &= \# \\
\mathcal{M}_2 &= d_0 \\
im &= 2 \\
R(m,n) &= R^1(m,n) \cup R^2(m,n) \cup R^3(m,n)
\end{aligned}$$

y, además,

$R^1(m,n)$ es el conjunto de las siguientes reglas:

(1) $[_2 d_0]_2^0 \to [_2 d_1]_2^+ [_2 d_1]_2^-$

(2) $[_2 d_k]_2^+ \to [_2 d_{k+1}]_2^+ [_2 d_{k+1}]_2^-$ $(1 \leq k < n)$

$[_2 d_k]_2^- \to [_2 d_{k+1}]_2^+ [_2 d_{k+1}]_2^-$ $(1 \leq k < n)$

(3) $[_2 x_{i,1} \to r'_i]_2^+, [_2 x_{i,1} \to \lambda]_2^-$ $(1 \leq i \leq m)$

$[_2 \overline{x}_{i,1} \to r'_i]_2^-, [_2 \overline{x}_{i,1} \to \lambda]_2^+$ $(1 \leq i \leq m)$

(4) $[_2 x_{i,j} \to x_{i,j-1}]_2^+, [_2 x_{i,j} \to x_{i,j-1}]_2^-$ $(1 \leq i \leq m, 2 \leq j \leq n)$

$[_2 \overline{x}_{i,j} \to \overline{x}_{i,j-1}]_2^+, [_2 \overline{x}_{i,j} \to \overline{x}_{i,j-1}]_2^-$ $(1 \leq i \leq m, 2 \leq j \leq n)$

(5) $[_2 d_n]_2^+ \to [_2\ ]_2^0 d_{n+1}, [_2 d_n]_2^- \to [_2\ ]_2^0 d_{n+1}$

$R^2(m,n)$ es el conjunto de las siguientes reglas:

(6) $[_2 r'_i \to r_i]_2^0$ $(1 \leq i \leq m)$

(7) $[_1 d_{n+1} \to d_{n+2} c_1]_1^0$

(8) $c_1 [_2\ ]_2^0 \to [_2 c_1]_2^-$

(9) $[_2 r_1]_2^- \to [_2\ ]_2^+ r_1$

(10) $[_2 r_i \to r_{i-1}]_2^+$ $(1 \leq i \leq m)$

(11) $[_2 c_k \to c_{k+1}]_2^+$ $(1 \leq k \leq m)$

(12) $r_1 [_2\ ]_2^+ \to [_2 r_0]_2^-$

(13) $[_2 c_{m+1}]_2^- \to [_2 c_{m+1}]_2^+$

(14) $[_1 d_k \to d_{k+1}]_1^0$ $(n+2 \leq k \leq n+2 \cdot m+2)$

(15) $[_1 d_{n+2 \cdot m+3} \to d_{n+2 \cdot m+4} e]_1^0$

$R^3(m,n)$ es el conjunto de las siguientes reglas:

(16) $e[_2\ ]_2^+ \to [_2 e]_2^+, e[_2\ ]_2^- \to [_2 e]_2^-, [_1 c_{m+1}]_1^0 \to [_1\ ]_1^+ c_{m+1}$

$[_1 d_{n+2 \cdot m+4} \to d_{n+2 \cdot m+5}]_1^0$

(17) $[_2 e]_2^- \to [_2\ ]_2^- e'$

$[_1 d_{n+2 \cdot m+5} \to d_{n+2 \cdot m+6}]_1^0, [_1 d_{n+2 \cdot m+5} \to d_{n+2 \cdot m+6}]_1^+$

(18) $[_1 e']_1^+ \to [_1\ ]_1^0 e'$

$[_1 d_{n+2 \cdot m+6} \to d_{n+2 \cdot m+7}]_1^0, [_1 d_{n+2 \cdot m+6} \to d_{n+2 \cdot m+7}]_1^+$

(19) $[_1 d_{n+2 \cdot m+7}]_1^+ \to [_1\ ]_1^- Yes, [_1 d_{n+2 \cdot m+7}]_1^0 \to [_1\ ]_1^- No$

(20) $[_1 \#]_1^- \to [_1\ ]_1^- \#$

Se trata de justificar que, dados $m, n \in \mathbb{N}^+$, el sistema $\Pi(m,n)$ decide la validez de todas las fórmulas con m cláusulas y n variables. Sea ϕ una tal fórmula, $\phi \equiv Cl_1 \wedge \cdots \wedge Cl_m$, en forma normal conjuntiva y cuyo conjunto de variables es $Var(\phi) = \{x_1, \ldots, x_n\}$.

Codificamos dicha fórmula por el multiconjunto sobre $\Sigma(m,n)$ siguiente,

$$\begin{aligned} cod(\phi) \equiv &\{x_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq n \wedge x_j \in Cl_i\} \cup \\ &\{\overline{x}_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq n \wedge \neg x_j \in Cl_i\} \end{aligned}$$

Es decir, este multiconjunto contiene un objeto $x_{i,j}$ por cada literal x_j, y un objeto $\overline{x}_{i,j}$ por cada literal $\neg x_j$, que aparezca en la cláusula Cl_i de la fórmula.

Veamos que existe una única computación de $\Pi(m,n)$ con entrada el multiconjunto $cod(\phi)$, y que se puede estructurar en tres etapas: una etapa de *generación y aplicación* de valoraciones, una etapa de *comprobación* de los valores obtenidos, y una etapa de *producción de la salida* del sistema.

Para ello, llamemos *membrana interna* de una configuración de $\Pi(m,n)$ a toda membrana de dicha configuración con etiqueta 2 y describamos con detalle cada una de las etapas anteriores.

Fase 1: En la etapa de generación y aplicación de valoraciones se generan las 2^n valoraciones relevantes para ϕ, a la vez que se determinan cuáles de las cláusulas de la fórmula van haciéndose verdaderas. Esta etapa se realiza de forma idéntica a la del sistema correspondiente para el problema **SAT**, descrito en la sección anterior.

Fase 2: En la etapa de comprobación de los valores obtenidos se determina si alguna de las membranas internas contiene objetos representando que todas las cláusulas de ϕ y, por tanto también dicha fórmula, son verdaderas. Esta etapa también es idéntica a la del sistema correspondiente para el problema **SAT**, salvo que en el último paso en lugar de aplicarse la regla de (14) se aplica la regla (15) que transforma los 2^n objetos $d_{n+2\cdot m+3}$ de la membrana piel en el mismo número de objetos $d_{n+2\cdot m+4}$ y de objetos e.

Fase 3: En la etapa de producción de la salida se expulsa al entorno externo un objeto Yes, si todas las membranas internas representan valoracio-

nes que hacen cierta la fórmula ϕ, o un objeto No, en caso contrario. Finalmente, se expulsa al entorno externo un objeto # en el último paso de la computación.

Pueden ocurrir tres casos: que ninguna valoración satisfaga la fórmula; que algunas de las valoraciones, pero no todas, satisfagan la fórmula; o bien que todas las valoraciones satisfagan la fórmula.

En el primer caso, las reglas de (16) introducen uno de los 2^n objetos e contenidos en la membrana piel en cada una de las 2^n membranas internas y convierten los objetos $d_{n+2\cdot m+4}$ en objetos $d_{n+2\cdot m+5}$. Sin embargo, como la membrana piel no contiene ningún objeto c_{m+1}, permanece con carga neutra. A continuación, las reglas de (17) hacen que los objetos e vuelvan a la piel como objetos e', ya que todas las membranas internas tienen carga negativa y, además, hacen que los objetos $d_{n+2\cdot m+5}$ se transformen en objetos $d_{n+2\cdot m+6}$, para que finalmente la única regla aplicable de (18) los transforme a su vez en objetos $d_{n+2\cdot m+7}$.

En el segundo caso, las reglas de (16) introducen uno de los 2^n objetos e contenidos en la membrana piel en cada una de las 2^n membranas internas, expulsan un objeto c_{m+1} al entorno externo, polarizando positivamente la membrana piel, y convierten los objetos $d_{n+2\cdot m+4}$ en objetos $d_{n+2\cdot m+5}$. A continuación, las reglas de (17) hacen que alguno de los objetos e vuelvan a la piel como objetos e' y que los objetos $d_{n+2\cdot m+5}$ se transformen en objetos $d_{n+2\cdot m+6}$. Finalmente, las reglas de (18) los transforman a su vez en objetos $d_{n+2\cdot m+7}$ y expulsan un objeto e' al entorno externo, para cambiar la carga de la membrana piel a neutra.

En el tercer caso, las reglas de (16) introducen uno de los 2^n objetos e contenidos en la membrana piel en cada una de las 2^n membranas internas, expulsan un objeto c_{m+1} al entorno externo, polarizando positivamente la membrana piel, y convierten los objetos $d_{n+2\cdot m+4}$ en objetos $d_{n+2\cdot m+5}$. A continuación, la única regla aplicable de (17) hace que los objetos $d_{n+2\cdot m+5}$ se transformen en objetos $d_{n+2\cdot m+6}$, para que finalmente la única regla aplicable de (18) los transforme a su vez en objetos $d_{n+2\cdot m+7}$.

En el penúltimo paso de la computación, por medio de las reglas de (19) un objeto $d_{n+2\cdot m+7}$ sale al entorno externo como un objeto Yes, si la piel está polarizada positivamente, o como un objeto No, si no está polarizada. Además, la carga de esta membrana cambia a negativa,

para que en el último paso de la computación la regla (20) expulse el objeto #.

Vamos a demostrar a continuación que la familia $\mathbf{\Pi_{VAL}}$ resuelve el problema **VALIDITY** en tiempo lineal. Para ello, sean ϕ una fórmula en forma normal conjuntiva, con m cláusulas y n variables, y $\mathcal{C} = \{C^i\}_{i<r}$ una computación de $\Pi(m,n)$ con entrada $cod(\phi)$.

Obsérvese que las proposiciones 11.6, 11.7 y 11.8 son válidas, con la única diferencia de que la membrana piel de la estructura $\mu^{n+2\cdot m+4}$ contiene además 2^n objetos e, como consecuencia de la aplicación de la regla (15) en lugar de la que correspondería de (14).

Para establecer la corrección de la etapa de producción de la salida, distinguimos los tres casos posibles, dependiendo del número de valoraciones que satisfagan la fórmula ϕ.

Proposición 11.14. *Supongamos que para toda valoración σ relevante para ϕ existe $j \leq m$ tal que $t_j^\sigma = 0$. Entonces la configuración $C^{n+2\cdot m+9}$ verifica las siguientes propiedades:*

- *El entorno externo contiene un objeto No y un objeto #.*
- *La membrana piel tiene carga negativa y contiene 2^n objetos e' y $2^n - 1$ objetos $d_{n+2\cdot m+7}$.*
- *Cada una de las membranas internas, $mb(\sigma)$, de $\mu^{n+2\cdot m+9}$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.*

En consecuencia, ninguna regla es aplicable a esta configuración.

Demostración. De la hipótesis y de la proposición 11.8 (con la salvedad mencionada anteriormente acerca de los objetos e), se deduce que la configuración $C^{n+2\cdot m+4}$ verifica lo siguiente:

- La membrana piel de $\mu^{n+2\cdot m+4}$ tiene carga neutra y contiene un objeto #, 2^n objetos $d_{n+2\cdot m+4}$ y 2^n objetos e.
- Cada una de las membranas internas, $mb(\sigma)$, de $\mu^{n+2\cdot m+4}$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.

Por tanto, las únicas reglas aplicables a dicha configuración son las de (16), salvo la regla $[_1 c_{m+1}]_1^0 \to [_1\ ]_1^+ c_{m+1}$. Luego en la configuración $C^{n+2\cdot m+5}$ la membrana piel tiene carga neutra y contiene un objeto # y 2^n objetos $d_{n+2\cdot m+5}$, mientras que a las membranas internas se les ha añadido un objeto e.

Ahora bien, a esta configuración solamente son aplicables las reglas de (17), por lo que en la configuración $C^{n+2\cdot m+6}$ la membrana piel tiene carga neutra y contiene un objeto #, 2^n objetos $d_{n+2\cdot m+6}$ y 2^n objetos e'. Las membranas internas, por su parte, siguen con carga negativa, pero pierden el objeto e añadido en el paso anterior.

Entonces resulta que a esta configuración únicamente se le puede aplicar la regla $[_1 d_{n+2\cdot m+6} \to d_{n+2\cdot m+7}]_1^0$ de (18). Así, en la configuración $C^{n+2\cdot m+7}$ la membrana piel contiene un objeto # y 2^n objetos $d_{n+2\cdot m+7}$ y e'. Las membranas internas, sin embargo, no se han modificado.

Finalmente, a esta configuración se le aplica la regla adecuada de (19), lo que induce que la membrana piel se polarice negativamente y expulse un objeto $d_{n+2\cdot m+7}$, transformado en un objeto No, al entorno externo. De esta manera, en la configuración $C^{n+2\cdot m+8}$ la piel tiene carga negativa y contiene un objeto #, 2^n objetos e' y $2^n - 1$ objetos $d_{n+2\cdot m+7}$. Entonces la computación termina aplicando la regla (20), que expulsa el objeto # de la membrana piel al entorno externo. De donde se deduce que la configuración $C^{n+2\cdot m+9}$ cumple lo requerido. □

Proposición 11.15. *Supongamos que existen $2^n > t > 0$ valoraciones, σ, relevante para ϕ tales que $t_j^\sigma > 0$, para todo $i = 1, \ldots, m$. Entonces la configuración $C^{n+2\cdot m+9}$ verifica las siguientes propiedades:*

- *El entorno externo contiene un objeto No y un objeto #.*
- *La membrana piel tiene carga negativa y contiene $t-1$ objetos c_{m+1}, $2^n - t - 1$ objetos e' y $2^n - 1$ objetos $d_{n+2\cdot m+7}$.*
- *Dada una valoración relevante, σ, para ϕ,*
  - *Si existe $j \leq m$ tal que $t_j^\sigma = 0$, entonces la membrana $mb(\sigma)$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma + \cdots + t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.*
  - *En caso contrario, $mb(\sigma)$ tiene carga positiva y contiene un objeto e y $t_1^\sigma + \cdots + t_m^\sigma$ objetos r_0.*

En consecuencia, ninguna regla es aplicable a esta configuración.

Demostración. De la hipótesis y de la proposición 11.8 (con la salvedad antes mencionada) se deduce que la configuración $C^{n+2\cdot m+4}$ verifica lo siguiente:

- La membrana piel de $\mu^{n+2\cdot m+4}$ tiene carga neutra y contiene un objeto $\#$, t objetos c_{m+1}, 2^n objetos $d_{n+2\cdot m+4}$ y 2^n objetos e.
- Dada una valoración, σ, relevante para ϕ,
  - Si existe $j \leq m$ tal que $t_j^\sigma = 0$, entonces la membrana $mb(\sigma)$ tiene carga negativa y contiene el objeto $c_{j'}$ y el multiconjunto $r_0^{t_1^\sigma+\cdots+t_{j'-1}^\sigma} + r_2^{t_{j'+1}^\sigma} \ldots r_{m-j'+1}^{t_m^\sigma}$, donde $j' = \min\{j : j \leq m \wedge t_j^\sigma = 0\}$.
  - En caso contrario, la membrana $mb(\sigma)$ tiene carga positiva y contiene $t_1^\sigma + \cdots + t_m^\sigma$ objetos r_0.

Por tanto, las únicas reglas aplicables a dicha configuración son las de (16). Luego en la configuración $C^{n+2\cdot m+5}$ la membrana piel tiene carga positiva y contiene un objeto $\#$, $t-1$ objetos c_{m+1} y 2^n objetos $d_{n+2\cdot m+5}$, mientras que a cada membrana interna se le ha añadido un objeto e.

Ahora bien, a esta configuración solamente son aplicables las reglas de (17), por lo que en la configuración $C^{n+2\cdot m+6}$ la membrana piel tiene carga positiva y contiene un objeto $\#$, $t-1$ objetos c_{m+1}, $2^n - t$ objetos e' y 2^n objetos $d_{n+2\cdot m+6}$. Las membranas internas con carga negativa, por su parte, pierden el objeto e añadido en el paso anterior.

A esta configuración únicamente se le puede aplicar las reglas de (18). Así, en la configuración $C^{n+2\cdot m+7}$ la membrana piel tiene carga neutra y contiene un objeto $\#$, $t-1$ objetos c_{m+1}, $2^n - t - 1$ objetos e' y 2^n objetos $d_{n+2\cdot m+7}$ y e'. Las membranas internas, sin embargo, no se han modificado.

Finalmente, a esta configuración se le aplica la regla adecuada de (19), lo que induce que la membrana piel se polarice negativamente y expulse un objeto $d_{n+2\cdot m+7}$, transformado en un objeto No, al entorno externo. De esta manera, en la configuración $C^{n+2\cdot m+8}$ la piel tiene carga negativa y contiene un objeto $\#$, $t-1$ objetos c_{m+1}, $2^n - t - 1$ objetos e' y $2^n - 1$ objetos $d_{n+2\cdot m+7}$. Entonces, la computación termina aplicando la regla (20), que expulsa el objeto $\#$ de la membrana piel al entorno externo. De donde se deduce que la configuración $C^{n+2\cdot m+9}$ cumple lo requerido. □

Proposición 11.16. *Supongamos que existen 2^n valoraciones, σ, relevante para ϕ tales que $t_i^\sigma > 0$, para todo $i = 1, \ldots, m$. Entonces la configuración $C^{n+2\cdot m+9}$ verifica las siguientes propiedades:*

- *El entorno externo contiene un objeto Yes y un objeto $\#$.*
- *La membrana piel tiene carga negativa y contiene $2^n - 1$ objetos c_{m+1} y objetos $d_{n+2\cdot m+7}$.*
- *Cada una de las membranas internas, $mb(\sigma)$, de $\mu^{n+2\cdot m+9}$ tiene carga positiva y contiene un objeto e y $t_1^\sigma + \cdots + t_m^\sigma$ objetos r_0.*

En consecuencia, ninguna regla es aplicable a esta configuración.

Demostración. De la hipótesis y de la proposición 11.8 (con la salvedad antes mencionada) se deduce que la configuración $C^{n+2\cdot m+4}$ verifica lo siguiente:

- La membrana piel de $\mu^{n+2\cdot m+4}$ tiene carga neutra y contiene un objeto $\#$ y 2^n objetos c_{m+1}, e y $d_{n+2\cdot m+4}$.
- Cada una de las membranas internas, $mb(\sigma)$, de $\mu^{n+2\cdot m+4}$ tiene carga positiva y contiene $t_1^\sigma + \cdots + t_m^\sigma$ objetos r_0.

Por tanto, las únicas reglas aplicables a dicha configuración son las de (16). Luego en la configuración $C^{n+2\cdot m+5}$ la membrana piel tiene carga positiva y contiene un objeto $\#$, $2^n - 1$ objetos c_{m+1} y 2^n objetos $d_{n+2\cdot m+5}$, mientras que a cada membrana interna se le ha añadido un objeto e.

Ahora bien, a esta configuración únicamente se puede aplicar la regla $[_1 d_{n+2\cdot m+5} \to d_{n+2\cdot m+5}]_1^+$ de (17), por lo que en la configuración $C^{n+2\cdot m+6}$ la membrana piel tiene carga positiva y contiene un objeto $\#$, $2^n - 1$ objetos c_{m+1} y 2^n objetos $d_{n+2\cdot m+6}$. Las membranas internas, por su parte, no se han modificado.

A esta configuración solo es aplicable la regla $[_1 d_{n+2\cdot m+6} \to d_{n+2\cdot m+7}]_1^+$ de (18). Así, en la configuración $C^{n+2\cdot m+7}$ la membrana piel contiene un objeto $\#$, $2^n - 1$ objetos c_{m+1} y 2^n objetos $d_{n+2\cdot m+7}$. Las membranas internas, sin embargo, no se han modificado.

Finalmente, a esta configuración se le aplica la regla adecuada de (19), lo que induce que la membrana piel se polarice negativamente y expulse un objeto $d_{n+2\cdot m+7}$, transformado en un objeto Yes, al entorno externo. De esta manera, en la configuración $C^{n+2\cdot m+8}$ la membrana piel tiene carga negativa

y contiene un objeto #, $2^n - 1$ objetos c_{m+1} y $d_{n+2\cdot m+7}$. Entonces la computación termina aplicando la regla (20), que expulsa el objeto # de la membrana piel al entorno externo. De donde se deduce que la configuración $C^{n+2\cdot m+9}$ cumple lo requerido. □

Finalmente, vamos a demostrar que la familia $\mathbf{\Pi}_{\mathbf{VAL}}$ resuelve el problema **VALIDITY** en tiempo lineal.

Teorema 11.17. *Consideremos la función par,* $\mathbf{p} : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$, *dada por* $\mathbf{p}(m,n) = \frac{(m+n+1)\cdot(m+n)}{2} + n$. *Entonces la familia* $\mathbf{\Pi} = \left(\Pi(\mathbf{p}(m,n))\right)_{m,n\in\mathbb{N}^+}$, *donde* $\Pi(\mathbf{p}(m,n)) = \Pi(m,n)$, *verifica las siguientes propiedades:*

1. *Para todo* $m,n \in \mathbb{N}^+$, *el sistema* $\Pi(\mathbf{p}(m,n)) \in \mathcal{AM}$.
2. *Para todo* $m,n \in \mathbb{N}^+$, *el sistema* $\Pi(\mathbf{p}(m,n))$ *tiene membrana de entrada.*
3. *La familia* $\mathbf{\Pi}$ *es polinomialmente uniforme, por máquinas de Turing.*
4. *Consideremos la función cod :* $I_{\mathbf{VAL}} \to \bigcup_{m,n\in\mathbb{N}^+} \Sigma(m,n)$ *definida, para cada fórmula* $\phi \in I_{\mathbf{VAL}}$, *como*

$$\begin{aligned} cod(\phi) = &\{x_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq m \wedge x_j \in Cl_i\} \cup \\ &\{\overline{x}_{i,j} : 1 \leq i \leq m \wedge 1 \leq j \leq m \wedge \neg x_{i,j} \in Cl_i\} \end{aligned}$$

y la función $s : I_{\mathbf{VAL}} \to \mathbb{N}^+$ *definida, para cada fórmula* $\phi \in I_{\mathbf{VAL}}$, *como* $s(\phi) = \mathbf{p}(m,n)$, *donde* m *es el número de cláusulas y* n *el número de variables de* ϕ. *Entonces,*

   - *Las funciones cod y s son computables en tiempo polinomial en el tamaño de* ϕ.
   - *Para cada* $\phi \in I_{\mathbf{VAL}}$, *se tiene que* $cod(\phi) \in I_{\Pi(s(\phi))}$.
   - *Para cada fórmula* ϕ, *toda computación del sistema* $\Pi(s(\phi))$ *con entrada* $cod(\phi)$ *realiza un número de pasos* lineal *en el tamaño de la fórmula* ϕ.
   - *Si la fórmula* ϕ *es válida, entonces toda computación del sistema* $\Pi(s(\phi))$ *con entrada* $cod(\phi)$ *es de aceptación.*
   - *Si existe una computación del sistema* $\Pi(s(\phi))$ *con entrada* $cod(\phi)$ *que es de aceptación, entonces la fórmula* ϕ *es válida.*

Demostración.

1. Las proposiciones 11.14, 11.15 y 11.16 nos aseguran que el sistema $\Pi(\mathbf{p}(m,n)) \in \mathcal{AM}$, para todo $m, n \in \mathbb{N}^+$.

2. Por definición, el sistema $\Pi(\mathbf{p}(m,n))$ tiene membrana de entrada, para todo $m, n \in \mathbb{N}^+$.

3. Es claro que la familia $\mathbf{\Pi}$ es uniforme. Además, para cada $m, n \in \mathbb{N}^+$, el sistema $\Pi(\mathbf{p}(m,n))$ cumple que:

   - El tamaño del alfabeto de entrada es $2 \cdot m \cdot n$.
   - El tamaño del alfabeto de trabajo es $2 \cdot m \cdot n + 5 \cdot m + n + 15$.
   - El tamaño del alfabeto de salida es 2.
   - El tamaño del conjunto de etiquetas es 2.
   - El grado de la estructura de membranas inicial es 2.
   - El tamaño de los multiconjuntos de objetos asociados a las membranas iniciales es 1.
   - El número total de reglas es $4 \cdot m \cdot n + 5 \cdot m + 2 \cdot n + 21$.
   - La longitud máxima de las reglas es 7.

   Por consiguiente, teniendo presente que la inversa de la función par es computable en tiempo polinomial, la familia Π se puede construir en tiempo polinomial.

4. Las funciones *cod* y s son trivialmente computables en tiempo polinomial en el tamaño de ϕ.

   - Por definición, $cod(\phi) \in \Sigma(m,n)$, para toda fórmula ϕ.
   - Sea ϕ una fórmula proposicional en forma normal conjuntiva. Las proposiciones 11.14, 11.15 y 11.16 nos muestran que existe una única computación de $\Pi(s(\phi))$ con entrada $cod(\phi)$, y que realiza $n + 2 \cdot m + 9$ pasos, que es lineal en el tamaño de ϕ.
   - Sea ϕ una fórmula válida. Entonces de la proposición 11.16 se deduce que la única computación de $\Pi(s(\phi))$ con entrada $cod(\phi)$ es de aceptación.

- Sea ϕ una fórmula tal que la la única computación de $\Pi(s(\phi))$ con entrada $cod(\phi)$ es de aceptación. De las proposiciones 11.14 y 11.15 se deduce que ϕ es válida.

□

De este resultado se deducen los siguientes corolarios.

Corolario 11.18. $\mathbf{VALIDITY} \in \mathbf{PMC}_{\mathcal{AM}^{fi}}$

Corolario 11.19. $\mathbf{coNP} \subseteq \mathbf{PMC}_{\mathcal{AM}^{fi}}$

Demostración. Basta tener en cuenta que la clase $\mathbf{PMC}_{\mathcal{AM}^{fi}}$ es cerrada bajo reducibilidad en tiempo polinomial, el corolario 11.18 y que **VALIDITY** es un problema **coNP**–completo. □

Conclusiones y trabajo futuro

Finalizamos esta memoria presentando ciertas conclusiones que se pueden extraer de la misma, y algunos caminos a seguir como trabajo futuro.

Conclusiones

La Computación Celular con Membranas es una disciplina joven aunque emergente dentro del campo de los modelos de computación no convencionales, en este caso biológicamente inspirados. Como tal, es natural que existan dentro de este campo un gran número de cuestiones a solucionar y tareas a realizar. En este trabajo hemos pretendido mostrar el camino para resolver y elaborar algunas de ellas.

En la literatura existente encontramos numerosas variantes de sistemas de computación celular, muchas de las cuales resultan ser computacionalmente completas sin necesidad de grandes artificios. Las pruebas de la correspondiente completitud computacional suelen deducirse de ciertos resultados provenientes de la *Teoría de Lenguajes Formales*. Además, estas pruebas son desarrolladas en un contexto informal, lejos de cualquier proceso de verificación formal.

En la primera parte de este trabajo hemos mostrado que para estos menesteres es posible considerar otros modelos clásicos de computación, como son las máquinas de Turing y las funciones recursivas. Estamos convencidos de que esta orientación nos podrá permitir aumentar nuestro conocimiento acerca de los sistemas P, trasladando algunos resultados obtenidos en esos modelos clásicos de forma que nos posibilite mejorar los ya conocidos en este nuevo modelo, o incluso demostrar propiedades inéditas de estos sistemas (especialmente en lo que respecta a problemas de indecidibilidad).

Por otro lado, la verificación de que los sistemas de computación con membranas construidos realizan correctamente la tarea que se les ha asigna-

do no es en absoluto trivial, y ello se debe fundamentalmente a dos razones: la primera es que estos sistemas no están descritos mediante la ejecución secuencial de una serie de operaciones básicas sobre un dato de entrada hasta obtener un resultado final, sino que se tratan de dispositivos computacionales que en cada paso se modifican a sí mismos hasta llegar a una configuración de parada. Así, no es posible utilizar ninguna de las técnicas de verificación desarrolladas para los algoritmos descritos mediante un lenguaje de programación imperativo. La segunda de las razones consideradas es el alto grado de paralelismo implementado por estos sistemas, lo que implica que el número de computaciones a estudiar puede ser exponencial respecto a los elementos de la configuración inicial.

No obstante, en esta memoria hemos conseguido verificar la corrección de la simulación de máquinas de Turing mediante sistemas P de transición con salida externa, y la corrección de la resolución de los problemas **SAT** y **VALIDITY** mediante sistemas P con membranas activas, basándonos en dos características fundamentales de los sistemas construidos:

- Toda computación se divide en varias fases debidamente estructuradas, cada una de las cuales lleva a cabo un cometido concreto.
- El comienzo y el final de estas fases se pueden identificar por la presencia de ciertos objetos en determinadas membranas.

De esta forma, para justificar la corrección del proceso efectuado por estos sistemas, el procedimiento a seguir hà consistido en estudiar por separado las fases atravesadas por las distintas computaciones, para finalmente «ensamblarlas» y concluir con la corrección global.

Es evidente que para que este procedimiento sea operativo es imprescindible la ausencia de ambigüedades en la descripción del funcionamiento de los sistemas. De ahí que sea completamente necesario formalizar con precisión cada uno de los modelos considerados.

Uno de los objetivos perseguidos con la introducción de este modelo de computación es investigar si puede resultar de utilidad para resolver instancias razonablemente grandes de problemas presuntamente intratables. Naturalmente el método consiste en efectuar un intercambio entre tiempo y espacio, aprovechando la densidad de información que podría almacenarse en una célula.

Una *Teoría de la Complejidad Computacional* para sistemas de computación celular debe comenzar con una definición rigurosa del concepto de

problema de decisión resoluble por un sistema P (o por una familia de sistemas P) utilizando una cantidad determinada de recursos computacionales. El siguiente paso a seguir consiste en definir clases de complejidad de problemas, a las que es natural exigirles que sean cerradas bajo una cierta operación de reducibilidad entre problemas.

Analizando en la literatura existente los resultados aparecidos que «resuelven» problemas **NP**–completos en tiempo polinomial (incluso lineal), observamos que en realidad en ellos se construye un sistema P para cada instancia del problema considerado, que determina en tiempo polinomial (o lineal, según el caso) si dicha instancia tiene o no solución. De esta forma, en esta memoria hemos fijado una definición de resolubilidad de problemas de decisión en tiempo polinomial, *por familias de sistemas P sin membrana de entrada.* Sin embargo, nada impide considerar *un sistema con membrana de entrada* que permita decidir todas las instancias de un problema, aunque en general es más sencillo construir *una familia de sistemas con membrana de entrada*, cada uno de los cuales decida todas las instancias de un mismo tamaño (en un cierto sentido).

Así, hemos introducido tres clases de complejidad, en función de la «maquinaria» utilizada para resolver los problemas. Además, hemos demostrado que estas clases de complejidad son cerradas bajo reducibilidad polinomial, por máquinas de Turing.

Como justificación de la adecuación de las definiciones realizadas, hemos estudiado las clases de complejidad determinadas por los sistemas P de transición con salida externa y por los sistemas P con membranas activas, mostrando diferencias cualitativas entre unas y otras. Además, hemos relacionado las clases de complejidad clásicas con estas nuevas clases, evidenciando así su posible utilidad a la hora de abordar problemas de la Teoría clásica de la Complejidad, en particular el problema $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

Trabajo futuro

Creemos que la investigación desarrollada en esta memoria debe tener una continuación en una serie de puntos que remarcamos como objetivos de posibles trabajos futuros, algunos de los cuales ya están siendo abordados por miembros del *Grupo de Investigación en Computación Natural* de la Universidad de Sevilla [27].

1. Estudio de cuestiones de decidibilidad relativas a sistemas P por medio de las simulaciones de modelos clásicos realizadas en esta memoria.

2. Simulación de otros modelos clásicos, como pueden ser las máquinas de registro o el λ–cálculo, por sistemas de computación celular con membranas.

3. Simulación de modelos de computación convencionales en variantes de sistemas P de menor potencia que las consideradas en esta memoria, como pueden ser los sistemas P con symport/antiport [30].

4. Perfeccionamiento de la metodología seguida para la verificación de sistemas de computación celular.

5. Estudio de las clases de complejidad determinadas por otras variantes de sistemas P.

6. Descripción de clases de complejidad clásicas (cerradas bajo complementario) (como $\mathbf{NP} \cup \mathbf{coNP}$ o **PSPACE**) a través de clases de complejidad celulares para ciertos tipos de sistemas P.

7. Definición del concepto de reducibilidad en tiempo polinomial mediante sistemas de computación celular con membranas.

8. Determinación de enumeraciones efectivas de sistemas de computación celular y diseño de sistemas P universales.

9. Desarrollo de clases de complejidad en las que para su definición solo se consideren sistemas de computación celular.

10. Resolución, en tiempo polinomial, de problemas **NP**–completos mediante un solo sistema P con membrana de entrada.

11. Diseño y verificación de sistemas de computación celular que resuelvan, en tiempo polinomial, problemas **NP**–completos de tipo *numérico*.

12. Definición adecuada de espacio empleado por un sistema P y definición de las clases de complejidad correspondientes.

13. Determinación de la relación existente entre el tiempo y el espacio empleado por un sistema P para resolver un problema (como las *alfombras de Sevilla* introducidas por G. Păun [7]).

14. Estudio de otras nociones de complejidad para sistemas P no deterministas (medidas de fiabilidad, entropía).

Bibliografía

[1] Adleman, L. Molecular Computation of Solutions to Combinatorial Problems. *Science*, 268, págs. 1021–1024 (1994).

[2] Arroyo, F.; Baranda, A.; Castellanos, J.; Păun, G. Membrane Computing: The Power of (Rule) Creation. *Journal of Universal Computer Science*, 8 (3), págs. 369–381 (2002).

[3] Balbontín Noval, D.; Pérez Jiménez, M. J.; Sancho Caparrini, F. A MzScheme implementation of Transition P systems. En *[40]*, págs. 58–73 (2003).

[4] Calude, C. S.; Dinneen, M. J.; Păun, G., eds. *Pre–proceedings of Workshop on Multiset Processing*, Informe Técnico 140. Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, New Zealand (2000).

[5] Calude, C. S.; Păun, G.; Rozenberg, G.; Salomaa, A., eds. *Multiset Processing (Mathematical, Computer Science, Molecular Computing Points of View)*, LNCS 2235. Springer–Verlag (2001).

[6] Cavaliere, M.; Martín-Vide, C.; Paun, G., eds. *Brainstorming Week on Membrane Computing*, Informe Técnico 26/03. Universitat Rovira i Virgili, Grup de Recerca en Lingüística Matemática, Tarragona, Spain (2003).

[7] Ciobanu, G.; Păun, G.; Stefănescu, G. Sevilla Carpets Associated with P Systems. En *[6]*, págs. 135–140 (2003).

[8] Cohen, D. E. *Computability and Logic*. Ellis Horwood (1987).

[9] Cordón Franco, A.; Gutiérrez Naranjo, M. A.; Pérez Jiménez, M. J.; Sancho Caparrini, F. A Prolog simulator for deterministic P systems with active membranes. En *[6]*, págs. 141–154 (2003).

[10] Cowan, J. D. Neural networks: the early days. En *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]* (D. S. Touretzky, ed.), págs. 828–842. Morgan Kaufmann (1990).

[11] DNA computing web pages. URL `http://www.liacs.nl/home/pier/webPagesDNA/index.html`.

[12] Feynman, R. P. There's Plenty of Room at the Bottom. *Journal of Microelectromechanical Systems*, 1 (1), págs. 60–66 (1992).

[13] Harel, D. *Computers Ltd. (What they really can't do)*. Oxford University Press (2000).

[14] Head, T. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49, págs. 737–759 (1987).

[15] Ionescu, M.; Martín Vide, C.; Păun, G. P Systems with Symport/Antiport Rules: The Traces of Objects. *Grammars*, 5 (2), págs. 65–79 (2002).

[16] Krishna, S. N. *Languages of P Systems: Computability and Complexity*. Tesis Doctoral, Indian Institute of Technology Madras (2001).

[17] Krishna, S. N.; Krithivasan, K.; Rama, R. P Systems with Picture Objects. *Acta Cybernetica*, 15 (1), págs. 53–74 (2001).

[18] Madhu, M.; Krithivasan, K. P Systems with Membrane Creation: Universality and Efficiency. En *[19]*, págs. 276–287 (2001).

[19] Margenstern, M.; Rogozhin, Y., eds. *Machines, Computations, and Universality (Third International Conference, MCU 2001 Chisinau, Moldava, May 23-27, 2001 Proceedings)*, LNCS 2055. Springer–Verlag (2001).

[20] Martín-Vide, C.; Păun, G. Computing with Membranes (P Systems): Universality Results. En *[19]*, págs. 82–101 (2001).

[21] Martín-Vide, C.; Păun, G. Elements of Formal Language Theory for Membrane Computing. Informe Técnico 21/01, Universidad Rovira i Virgili (2001).

[22] Martín-Vide, C.; Păun, G., eds. *Pre-proceedings of Workshop on Membrane Computing*, Informe Técnico 17/00. Grup de Recerca en Lingüística Matemática, Universitat Rovira i Virgili, Tarragona, Spain (2001).

[23] Martín-Vide, C.; Păun, G.; Pazos, J.; Rodríguez-Patón, A. Tissue P Systems. TUCS Technical Report 421, Turku Center for Computer Science (2001).

[24] Martín-Vide, C.; Păun, G.; Rozenberg, G. Membrane Systems with Carriers. *Theoretical Computer Science*, 270, págs. 779–796 (2002).

[25] Matiyasevich, Y. *Hilbert's Tenth Problem*. The MIT Press (1993).

[26] McHugh, J. A. *Algorithmic Graph Theory*. Prentice Hall (1990).

[27] Página Web del Grupo de Investigación en Computación Natural de la Universidad de Sevilla. URL `http://www.cs.us.es/gcn/`.

[28] Papadimitriou, C. H. *Computational Complexity*. Addison Wesley (1995).

[29] Păun, A. On P Systems with Global Rules. En *DNA Computing (7th International Workshop on DNA-Based Computers, DNA7, Tampa, FL, USA, June 10-13, 2001, Revised Papers)* (N. Jonoska; N. C. Seeman, eds.), LNCS 2340, págs. 329–339. Springer–Verlag (2001).

[30] Păun, A.; Păun, G. The Power of Communication: P Systems with Symport/Antiport. *New Generation Computing*, 20 (3), págs. 295–306 (2002).

[31] Păun, A.; Păun, G.; Rozenberg, G. Computing by Communication in Networks of Membranes. *International Journal of Foundations of Computer Science*, 13 (6), págs. 779–798 (2001).

[32] Păun, G. Computing with Membranes. *Journal of Computer and System Sciences*, 61 (1), págs. 108–143 (2000).

[33] Păun, G. Computing with Membranes – A Variant. *International Journal of Foundations of Computer Science*, 11 (1), págs. 167–182 (2000).

[34] Păun, G. P Systems with Active Membranes: Attacking NP–Complete Problems. *Journal of Automata, Languages and Combinatorics*, 6 (1), págs. 75–90 (2001).

[35] Păun, G. *Membrane Computing. An Introduction.* Springer–Verlag (2002).

[36] Păun, G.; Pérez Jiménez, M. J. Recent computing models inspired from Biology: DNA and membrane computing. *Theoria*, en prensa.

[37] Păun, G.; Pérez Jiménez, M. J.; Sancho Caparrini, F. On the Reachability Problem for P systems with Porters. En *Proceedings of the 10th International Conference on Automata and Formal Languages*, págs. 1–3 (2002).

[38] Păun, G.; Rozenberg, G.; Salomaa, A. *DNA Computing. New Computing Paradigms*. Springer (1998).

[39] Păun, G.; Rozenberg, G.; Salomaa, A. Membrane Computing with External Output. *Fundamenta Informaticae*, 41 (3), págs. 259–266 (2000).

[40] Păun, G.; Rozenberg, G.; Salomaa, A.; Zandron, C., eds. *Membrane Computing (International Workshop, WMC-CdeA 2002, Curtea de Arges, Romania, August 19-23, 2002, Revised Papers)*, LNCS 2597. Springer–Verlag (2003).

[41] Pérez Jiménez, A.; Pérez Jiménez, M. J.; Sancho Caparrini, F. Computing a partial mapping by a P system: Design and Verification. En *[6]*, págs. 247–260 (2003).

[42] Pérez Jiménez, A.; Pérez Jiménez, M. J.; Sancho Caparrini, F. Formal verification of a transition P system generating the set $\{2^n + n^2 + n : n \geq 1\}$. En *[6]*, págs. 261–269 (2003).

[43] Pérez Jiménez, M. J.; Romero Jiménez, A.; Sancho Caparrini, F. *Teoría de la complejidad en modelos de computación celular con membranas.* Editorial Kronos, Sevilla (2002).

[44] Pérez Jiménez, M. J.; Romero Jiménez, A.; Sancho Caparrini, F. Complexity classes in cellular computing with membranes. En *[6]*, págs. 270–278 (2003).

[45] Pérez Jiménez, M. J.; Romero Jiménez, Á.; Sancho Caparrini, F. Decision P systems and the $P \neq NP$ conjecture. En *[40]*, págs. 388–399 (2003).

[46] Pérez Jiménez, M. J.; Romero Jiménez, A.; Sancho Caparrini, F. Solving VALIDITY problem by active membranes with input. En *[6]*, págs. 279–290 (2003).

[47] Pérez Jiménez, M. J.; Sancho Caparrini, F. *Computación celular con membranas: Un modelo no convencional.* Editorial Kronos, Sevilla (2002).

[48] Pérez Jiménez, M. J.; Sancho Caparrini, F. A Formalization of Transition P Systems. *Fundamenta Informaticae*, 49 (1–3), págs. 261–272 (2002).

[49] Pérez Jiménez, M. J.; Sancho Caparrini, F. Verifying a P system generating squares. *Romanian Journal of Information Science and Technology*, 5 (1–2), págs. 181–191 (2002).

[50] Pérez Jiménez, M. J.; Sancho Caparrini, F. Verification of non deterministic transition P systems solving SAT problem. En *[6]*, págs. 291–304 (2003).

[51] Romero Jiménez, Á.; Pérez Jiménez, M. J. Generation of Diophantine Sets by Computing P Systems with External Output. En *Unconventional Models of Computation (Third International Conference, UMC 2002, Kobe, Japan, October 15-19, 2002, Proceedings)* (C. S. Calude; M. J. Dinneen; F. Peper, eds.), LNCS 2509, págs. 176–190. Springer–Verlag (2002).

[52] Romero Jiménez, A.; Pérez Jiménez, M. J. Simulating Turing Machines by P Systems with External Output. *Fundamenta Informaticae*, 49 (1–3), págs. 273–287 (2002).

[53] Rosen, K. H.; Michaels, J. G.; Gross, J. L.; Grossman, J. W.; Shier, D. R., eds. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press (2000).

[54] Sancho Caparrini, F. *Verificación de programas en modelos de computación no convencionales*. Tesis Doctoral, Universidad de Sevilla (2002).

[55] Sipser, M. The History and Status of the P versus NP Question. En *Proceedings of the 24th ACM Symposium on Theory of Computing*, págs. 603–618 (1992).

[56] Soare, R. I. Computability and Recursion. *The Bulletin of Symbolic Logic*, 2 (3), págs. 284–321 (1996).

[57] Spears, W. M.; De Jong, K. A.; Bäck, T.; Fogel, D. B.; de Garis, H. An Overview of Evolutionary Computation. En *Machine Learning: ECML-93 (European Conference on Machine Learning, Vienna, Austria, April 5-7, 1993. Proceedings)* (P. B. Brazdil, ed.), LNCS 667, págs. 442–459. Springer–Verlag (1993).

[58] The P Systems Web Page. URL `http://psystems.disco.unimib.it/`.

[59] Zandron, C. *A Model for Molecular Computing: Membrane Systems*. Tesis Doctoral, Università degli Studi di Milano (2001).

[60] Zandron, C.; Ferretti, C.; Mauri, G. Solving NP–Complete Problems Using P Systems with Active Membranes. En *Unconventional Models of Computation, UMC'2K (Proceedings of the Second International Conference, Brussel, Belgium, 13-16 December 2000)* (I. Antoniou; C. Calude; M. J. Dinneen, eds.), págs. 289–301. Springer–Verlag (2000).

UNIVERSIDAD DE SEVILLA

Reunido el Tribunal [illegible] los abajo firmantes en el día de la fecha para [illegible] la Tesis Doctoral de

D. Álvaro Romero Jiménez

titulada Complejidad y Universalidad en Modelos de Computación Celular

acordó otorgarle la calificación de Sobresaliente cum laude (por unanimidad)

Sevilla, 18 Junio 2003

El Vocal,

Rudolf Freund

El Vocal,

El Doctorado,

El Presidente

El Secretario,

Álvaro