5 PBE. MODELOS EN LÓGICA

Esta unidad tiene como objetivo continuar con la resolución de problemas a través de la modelización y simulación siguiendo el patrón PBE (P systems By Example) empleado en la unidad anterior, ilustrando en esta ocasión la aplicación a problemas relevantes en Lógica (en este caso un importante problema, SAT, consistente en la determinación de la satisfactibilidad de una fórmula lógica), haciendo uso de sistemas P con membranas activas.

1 Ejemplo 3. Satisfactibilidad de una fórmula lógica

Este ejemplo trata de presentar un problema muy sencillo a resolver mediante sistemas P, con el objetivo de ir familiarizándonos con el proceso y las herramientas a emplear.

1. 1 Escenario

El problema que se trata de resolver en este ejemplo es determinar si una determinada fórmula proposicional es satisfactible (SAT) o no. Se han dado varias soluciones eficientes a este problema en el ámbito de la computación celular con membranas, tanto trabajando a modo de células como a modo de tejidos y a modo de neuronas. Nos centraremos en este apartado en la resolución del problema mediante un simulador de sistemas P trabajando a modo de células, en este caso empleando la variante de membranas activas. En el siguiente apartado vamos a analizar el sistema P que resuelve el problema, pero antes vamos a introducir la descripción de los sistemas P con membranas activas a través de algunas definiciones básicas y descripciones detalladas extraídas de la tesis [4].

Los sistemas P de transición que se han descrito y empleado en unidades anteriores presentan un paralelismo a dos niveles: por un lado, las reglas asociadas a una membrana pueden aplicarse simultáneamente y por otro, en todas las membranas del sistema se realizan operaciones al mismo tiempo.

En este capítulo se introduce una nueva variante de sistemas celulares de

computación, los sistemas P con membranas activas [2], que presentan un paralelismo mejorado, en cierto sentido, y que permiten, además, la construcción de un espacio de tamaño exponencial en tiempo polinomial. De esta manera surge la posibilidad de atacar la resolubilidad eficiente de problemas **NP**–completos en el marco de estos nuevos dispositivos computacionales, intercambiando la complejidad en tiempo por complejidad en espacio.

A nivel conceptual, la modificación fundamental que introduce esta variante de sistemas P consiste en permitir que el número de membranas de la célula pueda aumentar (lo que está justificado desde el punto de vista biológico, por ejemplo, mediante la creación o división de membranas). Esto tiene dos consecuencias inmediatas en el diseño de sistemas P usando esta variante:

- Se considera la posibilidad de crear nuevas membranas por la acción de ciertos objetos. En este sentido, se pueden introducir reglas en las que un objeto produce, en el interior de la membrana a la que pertenece, una nueva membrana con un cierto contenido. Esto se traduce en el siguiente cambio con respecto a los sistemas P de transición que hemos visto: admitir entre sus reglas algunas del tipo $a \to [v]_i$, donde a es un objeto del alfabeto de trabajo, v es un multiconjunto de objetos, e i es la etiqueta de una membrana. Si en cierta membrana j existe un objeto a y la regla citada está asociada a dicha membrana, entonces la aplicación de la regla produce el siguiente efecto: se elimina el objeto a de la membrana j y, a la vez, se crea una nueva membrana (hija de la membrana j) con etiqueta i, y cuyo contenido es el multiconjunto de objetos v. De esta manera, estos sistemas celulares pueden fabricar un número exponencial de membranas en tiempo polinomial y, en consecuencia, son dispositivos computacionales apropiados para resolver problemas **NP**-completos en tiempo polinomial. Para una descripción más detallada de esta variante, junto con algunos ejemplos de aplicación, ver [3].

- Se toma inspiración también en el proceso de división celular (mitosis), que crea dos *copias* de una célula a partir de otra original. Este proceso permite obtener una cantidad exponencial de células idénticas en tiempo lineal; es decir, en n pasos se pueden generar 2^n copias de una célula.

A continuación plasmamos en las siguientes subsecciones la definición formal de este tipo de sistemas P con membranas activas con carácter informativo y de referencia, si bien puede obviarlas y pasar a la próxima sección si lo estima oportuno, consultando esta información cuando lo necesite.

1.0.1. Sintaxis de los sistemas P con membranas activas

Definición 1.1 *Un* sistema P con membranas activas *es una tupla*

$$\Pi = (\Gamma, H, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, R)$$

donde:

- Γ *es un alfabeto finito (el alfabeto de trabajo).*

- *H es un conjunto finito de etiquetas.*
- *μ_Π es una estructura de membranas de grado p, con etiquetas del conjunto H. Las membranas poseen carga eléctrica positiva $(+)$, negativa $(-)$ o neutra (0).*
- *$\mathcal{M}_1, \ldots, \mathcal{M}_p$ son multiconjuntos sobre Γ, que expresan los contenidos iniciales de las membranas de μ.*
- *R es un conjunto finito de* reglas de evolución, *que pueden ser de cada uno de los tipos siguientes:*
  - (a) $[a \to v]_l^\alpha$, *donde* $a \in \Gamma$, $v \in \Gamma^*$, $\alpha \in \{0, +, -\}$ (*reglas de evolución de objetos*). *Es una regla interna que no modifica nada fuera de la membrana l, ni tampoco la carga eléctrica de ésta. Su ejecución produce la sustitución de un objeto a por un multiconjunto v, dentro de una membrana etiquetada por l y con carga α.*
  - (b) $[a]_l^\alpha \to b[\ ]_l^\beta$, *donde* $a, b \in \Gamma$, $\alpha, \beta \in \{0, +, -\}$ (*reglas de comunicación*). *Un objeto a puede abandonar una membrana etiquetada por l y con carga α pasando a la membrana padre, pudiendo transformarse en un nuevo objeto b en el proceso. Al mismo tiempo, la ejecución de la regla puede producir un cambio en la carga de la membrana (de α a β), pero conservándose la etiqueta.*
  - (c) $a[\ ]_l^\alpha \to [b]_l^\beta$, *donde* $a, b \in \Gamma$, $\alpha, \beta \in \{0, +, -\}$ (*reglas de comunicación*). *Un objeto a puede penetrar en una membrana etiquetada por l y con carga α desde la región inmediatamente superior (es decir, desde la membrana padre), pudiendo transformarse en uno nuevo b en el proceso y, al mismo tiempo, cambiar la carga de la membrana atravesada (de α a β), pero conservándose la etiqueta.*
  - (d) $[a]_l^\alpha \to b$, *donde* $a, b \in \Gamma$, $\alpha \in \{0, +, -\}$, $l \neq skin$ (*reglas de disolución*). *Su ejecución produce la transformación de un objeto a dentro de una membrana etiquetada por l y con carga α en otro nuevo b. Además, simultáneamente, la membrana es disuelta (la piel no se puede disolver).*
  - (e) $[a]_l^\alpha \to [b]_l^\beta [c]_l^\gamma$, *donde* $a, b, c \in \Gamma$, $\alpha, \beta, \gamma \in \{0, +, -\}$, $l \neq skin$ (*reglas de 2-división para membranas elementales*). *Un objeto a situado en una membrana etiquetada por l y con carga α puede provocar la división de dicha membrana en otras dos, con la misma etiqueta pero, eventualmente, con distintas cargas eléctricas, de tal manera que el objeto a se transforma de manera independiente en cada una de ellas en nuevos objetos b y c (aparte de esos objetos, los contenidos de las membranas resultantes son idénticos). Este tipo de reglas no pueden ser ejecutadas en la piel.*

Obsérvese que las reglas del sistema están asociadas a etiquetas (es decir, la regla $[a \to v]_l^\alpha$ está asociada a la etiqueta $l \in H$) y no, propiamente, a las regiones del sistema. Nótese también que las reglas del tipo (e) permiten la existencia de varias membranas en el sistema con la misma etiqueta. Para cada etiqueta $l \in H$,

denotaremos por R_l el conjunto de reglas asociadas a la etiqueta l. Por tanto, $R = \bigcup\{R_l : l \in H\}$.

La siguiente subsección indica la semántica que sigue este sistema.

1.1. Semántica de los sistemas P con membranas activas

Las reglas se aplicarán de acuerdo con los siguientes principios (semántica informal de los sistemas P con membranas activas):

- Las reglas se usan como es habitual en el marco de la computación con membranas; esto es, de manera paralela y maximal. En cada paso, un objeto en una membrana *sólo* puede participar en la aplicación de *una* regla (elegida de manera no determinista en caso de que hubiera varias posibilidades), pero todo objeto que pueda evolucionar mediante alguna regla, debe hacerlo.
- Si una membrana es disuelta, su contenido (multiconjunto y membranas interiores) pasan a formar parte de la membrana (no disuelta) inmediatamente exterior (esto es, del primer antecesor no disuelto).
- Todos los objetos que no aparecen especificados en ninguna de las reglas que se aplican permanecerán inalterados.
- Una regla de división puede ejecutarse sobre una membrana y, simultáneamente y en el mismo paso, reglas de evolución de objetos se pueden aplicar sobre el contenido de la membrana. En este caso, podemos imaginar que "en primer lugar" evolucionan internamente los objetos y "posteriormente" se produce el proceso de división de la membrana, introduciendo copias de los resultados de dichas evoluciones en las dos membranas creadas (sobreentendiendo claramente que todo lo anterior se produce en un único paso de computación).
- Las reglas asociadas a una etiqueta i se pueden usar para todas las membranas que tengan dicha etiqueta. En el mismo paso se pueden aplicar reglas distintas a membranas distintas que tengan la misma etiqueta, pero una membrana puede ser objeto de, *a lo sumo*, una regla de entre las que no son de tipo (a).
- La membrana piel no se puede dividir nunca, aunque, como cualquier otra membrana, puede estar polarizada; es decir, cargada eléctricamente.

Obsérvese que en la descripción de esta nueva variante no se admite la cooperación entre objetos para disparar una regla, ni prioridad entre reglas. Además, la carga eléctrica de las membranas puede modificarse por la acción de ciertas reglas, pero no la etiqueta.

1. 2 Modelo

En esta sección se muestran una solución eficiente al problema **SAT** junto con sus respectiva codificación en P-Lingua. El sistema P que empleamos se ha tomado de [6], si bien aportamos a continuación la información extraída del artículo.

Se considera una fórmula proposicional genérica en forma normal conjuntiva $\varphi = C_1 \wedge \cdots \wedge C_m$ sobre n variables $x_1 \dots x_n$ compuesta por m cláusulas $C_j = y_{j,1} \vee \cdots \vee y_{j,k_j}$, $1 \leq j \leq m$, donde $y_{j,i} \in \{x_l, \neg x_l \mid 1 \leq l \leq n\}$, $1 \leq i \leq k_j$.

A continuación se detalla una solución al problema **SAT** usando una familia de sistemas P reconocedores con membranas activas y reglas de división. Esta solución fue presentada originalmente por M.J. Pérez–Jiménez y otros en [6].

La instancia φ será procesada por el sistema P $\Pi(s(\varphi))$ con entrada $cod(\varphi)$, siendo $s(\varphi) = \frac{(n+m)\cdot(n+m+1)}{2} + n$ denotado por $\langle n, m\rangle$.

La aplicación f de $\mathbb{N} \times \mathbb{N}$ en $\mathbb{N}$ definida por $f(m,n) = \frac{(m+n)\cdot(m+n+1)}{2} + m$ es una función computable y biyectiva entre los conjuntos citados. Usualmente, notaremos $f(m,n) = \langle m, n\rangle$. Para cada par de números naturales $m, n \in \mathbf{N}$ consideramos el sistema P reconocedor con membranas activas $\Pi(\langle m, n\rangle) = (\Gamma, \Sigma, \mu, \mathcal{M}_1, \mathcal{M}_2, R, 2)$ de grado 2, definido como sigue:

- El alfabeto de entrada es $\Sigma = \{x_{i,j}, \overline{x}_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\}$.
- El alfabeto de trabajo es

$$\begin{aligned}\Gamma \;=\; & \Sigma \cup \{c_k : 1 \leq k \leq m+2\} \cup \{d_k : 1 \leq k \leq 3n+2m+3\} \cup \\ & \cup \{r_{i,k} : 0 \leq i \leq m, 1 \leq k \leq 2n\} \cup \{e, t\} \cup n\{Yes, No\}\end{aligned}$$

- El conjunto de etiquetas es $\{1, 2\}$.
- La estructura inicial de membranas es $\mu = [\,[\,]_2\,]_1$.
- Los multiconjuntos iniciales son $\mathcal{M}_1 = \emptyset$ y $\mathcal{M}_2 = \{d_1\}$.
- La membrana de entrada es la etiquetada por 2.
- El conjunto R consiste de las siguientes reglas:
  - (a) $\{[d_k]_2^0 \to [d_k]_2^+ [d_k]_2^- : 1 \leq k \leq n\}$.
  - (b) $\{[x_{i,1} \to r_{i,1}]_2^+, [\,\overline{x}_{i,1} \to r_{i,1}]_2^- : 1 \leq i \leq m\}$. $\{[x_{i,1} \to \lambda]_2^-, [\overline{x}_{i,1} \to \lambda]_2^+ : 1 \leq i \leq m\}$.
  - (c) $\{[x_{i,j} \to x_{i,j-1}]_2^+, [x_{i,j} \to x_{i,j-1}]_2^- : 1 \leq i \leq m,\ 2 \leq j \leq n\}$. $\{[\,\overline{x}_{i,j} \to \overline{x}_{i,j-1}]_2^+, [\,\overline{x}_{i,j} \to \overline{x}_{i,j-1}]_2^- : 1 \leq i \leq m,\ 2 \leq j \leq n\}$.
  - (d) $\{[d_k]_2^+ \to [\,]_2^0 d_k, \ , [d_k]_2^- \to [\,]_2^0 d_k : 1 \leq k \leq n\}$. $\{d_k[\,]_2^0 \to [d_{k+1}]_2^0 : 1 \leq k \leq n-1\}$.
  - (e) $\{[r_{i,k} \to r_{i,k+1}]_2^0 : 1 \leq i \leq m,\ 1 \leq k \leq 2n-1\}$.
  - (f) $\{[d_k \to d_{k+1}]_1^0 : n \leq k \leq 3n-3\}$; $[d_{3n-2} \to d_{3n-1}e]_1^0$.
  - (g) $e[\,]_2^0 \to [c_1]_2^+$; $[d_{3n-1} \to d_{3n}]_1^0$.
  - (h) $\{[d_k \to d_{k+1}]_1^0 : 3n \leq k \leq 3n+2m+2\}$.

(i) $[r_{1,2n}]_2^+ \to [\,]_2^- r_{1,2n}$.
(j) $\{[r_{i,2n} \to r_{i-1,2n}]_2^- : 1 \leq i \leq m\}$.
(k) $r_{1,2n}[\,]_2^- \to [r_{0,2n}]_2^+$.
(l) $\{[c_k \to c_{k+1}]_2^- : 1 \leq k \leq m\}$.
(m) $[c_{m+1}]_2^+ \to [\,]_2^+ c_{m+1}$.
(n) $[c_{m+1} \to c_{m+2}t]_1^0$.
(o) $[t\,]_1^0 \to [\,]_1^+ t$.
(p) $[c_{m+2}]_1^+ \to [\,]_1^- Yes$.
(q) $[d_{3n+2m+3}]_1^0 \to [\,]_1^+ No$.

Sea $\varphi = C_1 \wedge \cdots \wedge C_m$ una fórmula proposicional en FNC tal que $Var(\varphi) = \{x_1, \ldots, x_n\}$ y $C_i = y_{i,1} \vee \cdots \vee y_{i,k_i}$, $1 \leq i \leq m$, donde $y_{i,i'} \in \{x_j, \neg x_j : 1 \leq j \leq n\}$ son los literales de φ. Consideramos las funciones $cod(\varphi) = \bigcup_{i=1}^m \{x_{i,j} : x_j \in C_i\} \cup \{\overline{x}_{i,j} : \neg x_j \in C_i\}$ y $s(\varphi) = \langle m, n\rangle = \frac{(m+n)\cdot(m+n+1)}{2} + m$. La ejecución del sistema $\Pi(s(\varphi))$ con entrada $cod(\varphi)$ se estructura en cuatro fases:

- *Fase de generación de valoraciones*: se generan todas las posibles valoraciones de verdad del conjunto $\{x_1, \ldots, x_n\}$ y se ejecuta en $3n - 1$.
- *Fase de sincronización*: se prepara al sistema para la fase de chequeo y consume $2n$ pasos.
- *Fase de chequeo*: se determina cuántas cláusulas son verdaderas por cada valoración y se ejecuta en $2m$ pasos.
- *Fase de salida*: el sistema proporciona la salida correspondiente en función del chequeo anterior.

1.1.1. Definición en P-Lingua

A continuación se va a mostrar el código del programa escrito en P-Lingua que especifica la familia de sistemas P antes descrita. En concreto, se va a considerar como ejemplo la siguiente fórmula para instanciar el problema `SAT` a resolver:

$$\varphi \equiv (x_1 + \neg x_2)(\neg x_2 + x_3 + \neg x_4)\, x_5\, \neg x_6$$

El módulo `main` puede ser modificado de manera sencilla para definir cualquier otro sistema P de la familia.

El código se estructura como sigue:

1. Módulo `main()`: define un sistema P reconocedor con membranas activas y reglas de división resolviendo el problema `SAT` para la fórmula descrita con 6 variables y 4 cláusulas. En primer lugar, este módulo llama al modulo

`Sat(n,m)` para $(n, m) \equiv (6, 4)$. A continuación, el módulo incluye el multiconjunto inicial de la membrana de entrada con $cod(\varphi)$ donde los objetos $x_{j,i}$ son escritos `x{j,i}`, representando que la variable x_i se encuentra en la cláusula C_j, y los objetos $nx_{j,i}$ se codifican por `nx{j,i}`, representando que la variable $\neg x_i$ está en la cláusula C_j.

2. Módulo `Sat(n,m)`: define una familia de sistemas P reconocedores con membranas activas y reglas de división resolviendo el problema `SAT` para cualquier instancia con n variables y m cláusulas.

El código es el siguiente:

```
@model<membrane_division>

def main()
{
        call Sat(6,4);
        call define_input();
}

def init_membrane_structure()
{
        @mu = [[]'2]'1;
}

def Sat(n,m)
{
        call init_membrane_structure();
        call init_multisets();
        call init_rules(m,n);
}

def init_multisets()
{
        @ms(2) = d{1};
}

def init_rules(m,n)
{
        [d{k}]'2 --> +[d{k}]-[d{k}] : 1 <= k <= n;

        {
        +[x{i,1} --> r{i,1}]'2;
        -[nx{i,1} --> r{i,1}]'2;
        -[x{i,1} --> #]'2;
        +[nx{i,1} --> #]'2;
        } : 1 <= i <= m;

        {
        +[x{i,j} --> x{i,j-1}]'2;
        -[x{i,j} --> x{i,j-1}]'2;
        +[nx{i,j} --> nx{i,j-1}]'2;
        -[nx{i,j} --> nx{i,j-1}]'2;
        } : 1<=i<=m, 2<=j<=n;

        {
        +[d{k}]'2 --> []d{k};
        -[d{k}]'2 --> []d{k};
        } : 1<=k<=n;

        d{k}[]'2 --> [d{k+1}] : 1<=k<=n-1;
        [r{i,k} --> r{i,k+1}]'2 : 1<=i<=m, 1<=k<=2*n-1;
        [d{k} --> d{k+1}]'1 : n <= k<= 3*n-3;
```

```
        [d{3*n-2} --> d{3*n-1},e]'1;
        e[]'2 --> +[c{1}];
        [d{3*n-1} --> d{3*n}]'1;
        [d{k} --> d{k+1}]'1 : 3*n <= k <= 3*n+2*m+2;
        +[r{1,2*n}]'2 --> -[]r{1,2*n};
        -[r{i,2*n} --> r{i-1,2*n}]'2 : 1<= i <= m;
        r{1,2*n}-[]'2 --> +[r{0,2*n}];
        -[c{k} --> c{k+1}]'2 : 1<=k<=m;
        +[c{m+1}]'2 --> +[]c{m+1};
        [c{m+1} --> c{m+2},t]'1;
        [t]'1 --> +[]t;
        +[c{m+2}]'1 --> -[]Yes;
        [d{3*n+2*m+3}]'1 --> +[]No;
}

def define_input()
{
    @ms(2) += x{1,1}, nx{1,2}, nx{2,2}, x{2,3}, nx{2,4}, x{3,5}, nx{4,6};
}
```

1. 3 Simulación

El sistema P descrito en el apartado anterior resuelve el problema **SAT** para la fórmula indicada. P-Lingua acepta el fichero anterior y dispone de simuladores de membranas activas preparados para su simulación. Puede emplear el mismo directamente con la misma aplicación general empleada en la unidad anterior.

Para llevar a cabo la simulación planteada en este ejemplo debemos hacer lo siguiente:

1. Arrancar MeCoSim, abrir la aplicación por defecto ("General") y cargar el modelo "Ejemplo3_SAT.pli". Este fichero indica en su primera línea el tipo de modelo "membrane_division", y al cargarlo indicará que el simulador seleccionado es "active_membranes", que es que queremos. Basta con simular con *Simulation* $>$ *Simulate!* y mostrará en la pestaña "Output" la salida. Si vamos hacia el final de la tabla, podremos encontrar en el entorno "Environment" el objeto "Yes", ya que se trata de una fórmula satisfactible.

2. Simular el mismo modelo de SAT pero para otra fórmula que sepamos insatisfactible. Por ejemplo, probemos para la fórmula:

$$\varphi \equiv (x_1 + \neg x_2)(\neg x_2 + x_3 + \neg x_4)\, x_5\, \neg x_5$$

, que podemos estar seguro de que no se satisface para ninguna valoración de verdad ya que x_5 no puede ser cierta y falsa a la vez, como se requiere en las dos últimas cláusulas. Modifique el fichero mediante el plugin "Edit P-Lingua file" o mediante su editor de texto favorito, guárdelo y simule mediante *Simulation* $>$ *Simulate!*. Ahora debe devolvernos el resultado "No". A continuación puede salir de la aplicación "General" y volver al listado de aplicaciones de MeCoSim.

3. Más interesante que resolver el problema para una instancia concreta, como estamos haciendo, es fijar el modelo empleado por parte del diseñador de sistemas P y dejar la aplicación en manos de la persona (usuario final) interesada en conocer la satisfactibilidad e ir introduciendo las fórmulas cuya satisfactibilidad quiera conocer. Para ello es necesario hacer algunos trabajos adicionales:

   - Adaptar el archivo de P-Lingua para que en lugar de fijar el sistema P con $n = 6$ y $m = 4$ deje ese valor pendiente de la fórmula concreta introducida, y sustituir el multiconjunto inicial de la membrana 2 por la fórmula que tengamos en cada caso. Para ello, se dejan n y m sin valor, y se modifica la función "define_input" conteniendo ahora:

     ```
     @ms(2) += nx{clause{i},variable{i}}*valn{i},x{clause{i},variable{i}}*val{i} : 1<=i<=nvals;
     ```

     Esto asignará para cada par $(cláusula, variable)$ de nuestra fórmula un objeto $x_{j,i}$ si la variable i aparece afirmada en la cláusula j y $nx_{j,i}$ si aparece negada. Este ejemplo se encuentra como el ejemplo anterior en el repositorio de modelos, en este caso con el nombre "Ejemplo3_SAT_parametrizado.pli", listo para su descarga. Añadámoslo a la espera de continuar los pasos siguientes. No se preocupe por los detalles que no comprenda, ya que se irá aclarando con las explicaciones siguientes. De momento sepa que tanto n y m como $nvals$ y los valores indexados $clausei$, $variablei$, $vali$ y $valni$ serán variables dependientes de la instancia concreta del problema (es decir, de la fórmula concreta), por lo que será necesario definir una aplicación basada en MeCoSim, adaptada al problema particular, y en la que definiremos qué entradas necesitaremos y cómo interpretar la información de la fórmula de entrada para que pase a complementar el modelo aportado, dando valores específicos a los parámetros anteriores.

   - Como se comentaba en el paso anterior, es necesario definir una aplicación basada en MeCoSim adaptada a este problema. No se preocupe, no es necesario realizar ningún desarrollo de software, sino que basta con configurar un archivo Excel que defina las entradas y salidas (tabuladas y gráficas) deseadas, junto con la generación de los parámetros de P-Lingua a partir de las entradas de MeCoSim. El archivo Excel correspondiente está disponible en el repositorio de apps "http://www.p-lingua.org/mecosim/matvida/apps/", de modo que añada este repositorio y seleccione la aplicación "Ejemplo 3 - SAT". Esto descargará la configuración de la aplicación al lugar adecuado, lista para que la añadamos si lo deseamos a la lista de nuestras aplicaciones (donde hasta el momento tenemos únicamente la aplicación "General". Para añadir la aplicación a nuestra lista desde la ventana principal de MeCoSim basta con pulsar en el botón "New", y desde ahí seleccionar el archivo "Ejemplo3_SAT.xls", lo que hará que en nuestro listado aparezca en primer lugar la aplicación "SAT", lista para ser lanzada.

- Previamente instamos a descargar la aplicación SAT y el modelo parametrizado, luego estamos listos para abrir la aplicación mediante el botón "Run". Verá que la aplicación ya no muestra la información de "General", sino una aplicación con distintas entradas y salidas, como se ve en la figura 1.

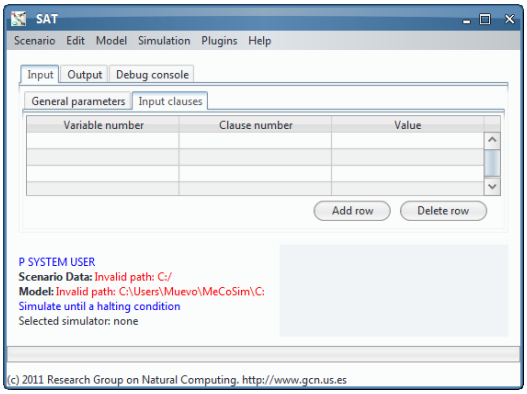

FIGURA 1: SAT - Ventana de la aplicación basada en MeCoSim.

Ahora podemos ver la aplicación como usuarios que deseamos saber si nuestras fórmulas son satisfactibles, pasando a indicar en las pestañas de entrada los parámetros generales (número de variables, n, y número de cláusulas, m) y las cláusulas de entrada (como vemos, la tabla de entrada espera que indiquemos para cada cláusula, para cada variable que aparezca, una fila de la tabla indicando en value si se trata de una variable afirmada, 1, o negada, -1). Así, para el ejemplo anterior, tendríamos $n = 6$, $m = 4$, y en la pestaña "Input clauses" una fila por cada variable que aparezca en cada cláusula (por ejemplo, la terna $(1, 1, 1)$ indicando que la variable $x1$ aparece en la cláusula 1 de manera $afirmativa$). Podemos introducir la información correspondiente de entrada y guardar el modelo con la opción de menú *Scenario* $>$ *Save as...* y asignar un nombre al fichero de datos del escenario, representando en este caso una fórmula concreta. O bien podemos salir de la aplicación SAT, y añadir el archivo "Ejemplo3_SAT.ec2" desde el repositorio de escenarios. Posteriormente podremos abrir de nuevo la aplicación SAT, y abrir este archivo mediante *Scenario* $>$ *Open*. En este momento podremos proceder a la simulación mediante *Simulation* $>$ *Simulate!*. ¿Qué cambios observa respecto a la salida ofrecida en la aplicación general?

- La aplicación anterior emplea el mecanismo de personalización de MeCoSim, basado en la configuración vía archivo Excel. No obstante, para este problema concreto parece que el usuario final podría encontrarse incómodo con la pestaña de entrada en la que introducir variables afir-

madas o negadas de cada cláusula, y preferirá introducir su fórmula de forma más intuitiva. Con ese fin, se ha desarrollado un plugin denominado "SAT plugin", que podemos instalar desde el *gestor de repositorios de MeCoSim* > *Plugins* (recordemos que tras instalarlo debemos salir del programa, y configurar el plugin para que aparezca en nuestro listado de plugins introduciendo las siguientes líneas en la ruta *MeCoSim* > *prop* > *plugins-properties*:

```
plugin-sat = mecosim.plugins.SATPlugin.SATFrame
pluginname-sat = SAT plugin
pluginmethod-sat = main
pluginorder-sat = 5
pluginjar-sat-1 = SATPlugin.jar
```

Este plugin permite introducir fórmulas y generar tablas de datos a partir de ellas, listas para hacer `Copy&Paste` mediante $Ctrl + c$ en la tabla generada por la ventana del plugin y $Ctrl + v$ en la pestaña de "Input clauses", añadiendo de antemano las filas vacías necesarias para que el número de filas copiado quepa en la tabla donde las deseamos copiar. Ya estamos listos para arrancar de nuevo MeCoSim, y entrar en la aplicación SAT. Una vez en ella, podemos probar el ejemplo anterior, como se muestra en la figura 2.

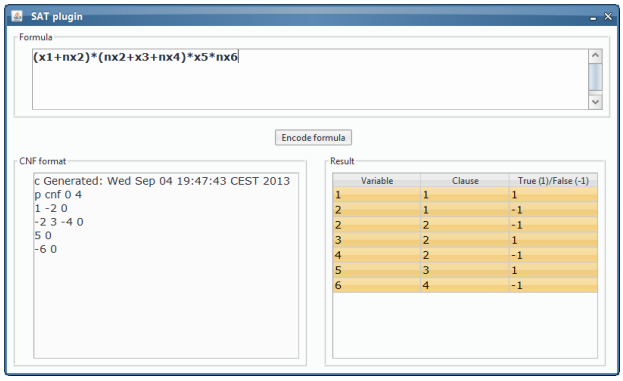

FIGURA 2: SAT - Ventana de SAT plugin para MeCoSim.

Hasta aquí el detalle de la simulación. En el próximo apartado se detallan los resultados esperados y el análisis de los mismos.

1. 4 Resultados

La idea de esta unidad es trabajar con un problema NP-completo resuelto con un sistema P con membranas activas, en este caso el problema SAT, además de ilustrar el mecanismo de personalización de aplicaciones de MeCoSim para acercar las aplicaciones al usuario final.

Se propone al alumno la realización de los apartados de la sección anterior, recopilando el resultado de los ejemplos concretos a partir del modelo original, y contrastando que el resultado de la simulación es el mismo empleando la aplicación a medida haciendo uso de la interfaz personalizada. Podrá aportar capturas de pantalla con detalles de las tablas y gráficos generados.

Además, se propone realizar experimentos virtuales con fórmulas de distintos tamaños dentro del listado de la tabla 1 (al menos 5 fórmulas de distintos tamaños), incluyendo los pasos:

1. Introducir la fórmula mediante el plugin de SAT.
2. Copiar la tabla generada por el plugin de SAT, pegarla a la tabla de entrada "Input clauses", indicar los parámetros generales n y m y guardar el fichero con *Scenario* $>$ *Save As....*
3. Simular mediante *Simulation* $>$ *Simulate!*.
4. Comprobar si el resultado (Yes o No) coincide con el esperado que aparezca en la tabla (T o F).
5. Recopile mediante capturas de pantalla o texto algunas fórmulas aportadas y las salidas tabuladas o gráficas obtenidas.

Esta tabla incluye también tiempos aproximados para la simulación para las distintas instancias del problema.

TABLA 1: *Fórmulas - Satisfactibilidad*

n	m	SAT	Time (s)	Fórmula
2	3	F	0,23	$(\bar{x}_1 + \bar{x}_2) \cdot x_1 \cdot x_2$
2	3	F	0,08	$(\bar{x}_1 + x_2) \cdot x_1 \cdot \bar{x}_2$
2	3	F	0,07	$(x_1 + x_2) \cdot \bar{x}_1 \cdot \bar{x}_2$
2	3	T	0,05	$(\bar{x}_1 + \bar{x}_2) \cdot x_2 \cdot (\bar{x}_1 + x_2)$
2	3	T	0,03	$\bar{x}_2 \cdot (\bar{x}_1 + x_2) \cdot (x_1 + \bar{x}_2)$
2	3	T	0,04	$(x_1 + \bar{x}_2) \cdot x_1 \cdot x_2$
3	4	F	0,14	$(x_1 + x_2) \cdot (x_1 + x_2 + \bar{x}_3) \cdot \bar{x}_1 \cdot \bar{x}_2$
3	4	F	0,13	$(\bar{x}_1 + x_2 + x_3) \cdot x_1 \cdot (\bar{x}_1 + \bar{x}_3) \cdot (\bar{x}_2 + x_3)$
3	4	F	0,12	$(x_1 + x_2) \cdot \bar{x}_2 \cdot (x_2 + x_3) \cdot \bar{x}_1$
3	4	T	0,14	$(\bar{x}_2 + x_3) \cdot x_2 \cdot (\bar{x}_1 + \bar{x}_2 + x_3) \cdot (x_1 + x_3)$
3	4	T	0,15	$(\bar{x}_1 + x_2) \cdot \bar{x}_1 \cdot x_3 \cdot (\bar{x}_1 + x_3)$
3	4	T	0,13	$\bar{x}_2 \cdot (\bar{x}_1 + \bar{x}_2) \cdot x_3 \cdot (\bar{x}_1 + x_2)$
4	5	F	0,51	$(\bar{x}_1 + \bar{x}_2) \cdot x_3 \cdot (\bar{x}_1 + \bar{x}_3) \cdot (\bar{x}_3 + x_4) \cdot (x_1 + \bar{x}_4)$
4	5	F	0,51	$(\bar{x}_1 + x_2 + \bar{x}_3 + x_4) \cdot (\bar{x}_2 + \bar{x}_3 + \bar{x}_4) \cdot (\bar{x}_1 + x_3) \cdot x_1 \cdot (\bar{x}_1 + \bar{x}_3)$
4	5	F	0,53	$(x_1 + x_4) \cdot (x_1 + \bar{x}_4) \cdot x_3 \cdot (x_2 + \bar{x}_3 + x_4) \cdot \bar{x}_1$
4	5	T	0,54	$(x_2 + x_3) \cdot (x_1 + x_3 + \bar{x}_4) \cdot (x_1 + x_2 + x_3 + \bar{x}_4) \cdot (x_1 + x_2 + x_3) \cdot x_4$
4	5	T	0,56	$(x_3 + \bar{x}_4) \cdot (\bar{x}_1 + x_2 + \bar{x}_3 + x_4) \cdot (x_1 + x_2) \cdot (\bar{x}_1 + x_2 + x_3 + x_4) \cdot (\bar{x}_1 + x_3)$

n	m	SAT	Time (s)	Fórmula
4	5	T	0,49	$(\bar{x}_1 + x_4) \cdot (\bar{x}_1 + x_2 + \bar{x}_3) \cdot x_2 \cdot (\bar{x}_1 + x_2 + \bar{x}_4) \cdot (\bar{x}_1 + \bar{x}_3 + x_4)$
5	6	F	2,05	$\bar{x}_5 \cdot (x_2 + \bar{x}_4 + x_5) \cdot (\bar{x}_2 + x_3) \cdot (x_1 + \bar{x}_5) \cdot (\bar{x}_1 + \bar{x}_3 + \bar{x}_5) \cdot x_1$
5	6	F	2,16	$(x_1 + \bar{x}_2 + x_3 + x_5) \cdot (\bar{x}_1 + x_4) \cdot (\bar{x}_2 + \bar{x}_4) \cdot x_4 \cdot x_2 \cdot (\bar{x}_1 + x_2 + \bar{x}_3 + x_4)$
5	6	F	2,02	$(\bar{x}_2 + x_5) \cdot (\bar{x}_4 + x_5) \cdot (x_4 + x_5) \cdot (x_2 + \bar{x}_3 + x_5) \cdot \bar{x}_5 \cdot (x_1 + x_2 + \bar{x}_4 + \bar{x}_5)$
5	6	T	2,18	$(\bar{x}_1 + \bar{x}_3 + x_4 + \bar{x}_5) \cdot (\bar{x}_1 + x_2 + \bar{x}_3 + x_5) \cdot (\bar{x}_2 + x_3) \cdot (\bar{x}_2 + x_3 + \bar{x}_5) \cdot (x_2 + x_3 + x_4 + \bar{x}_5) \cdot (\bar{x}_3 + x_5)$
5	6	T	1,98	$x_1 \cdot (x_2 + x_4) \cdot (x_2 + \bar{x}_3 + \bar{x}_5) \cdot (x_4 + \bar{x}_5) \cdot (x_2 + \bar{x}_5)$
5	6	T	1,99	$(x_3 + x_4) \cdot (x_4 + \bar{x}_5) \cdot (\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4) \cdot (x_1 + \bar{x}_2 + x_4) \cdot (x_1 + \bar{x}_3 + x_4) \cdot (x_3 + x_5)$
6	7	F	8,88	$(x_1 + \bar{x}_2) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_5) \cdot (x_1 + x_2 + x_3 + \bar{x}_4 + x_5) \cdot (\bar{x}_2 + x_5) \cdot x_2 \cdot (x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_5) \cdot (\bar{x}_1 + \bar{x}_6)$
6	7	F	8,3	$(x_2 + \bar{x}_3 + \bar{x}_5) \cdot (x_1 + x_3 + x_5) \cdot x_1 \cdot (\bar{x}_1 + \bar{x}_4 + x_5) \cdot (\bar{x}_1 + \bar{x}_4 + \bar{x}_5) \cdot (\bar{x}_1 + x_4) \cdot (x_2 + x_3 + \bar{x}_5 + x_6)$
6	7	F	8,58	$(x_3 + x_5 + x_6) \cdot (x_3 + \bar{x}_4 + x_5 + \bar{x}_6) \cdot \bar{x}_3 \cdot \bar{x}_6 \cdot (x_1 + \bar{x}_2 + \bar{x}_3 + x_5 + x_6) \cdot (x_1 + x_4 + x_5) \cdot (\bar{x}_5 + x_6)$
6	7	T	8,44	$(x_1 + x_3 + \bar{x}_4) \cdot (x_2 + x_3 + \bar{x}_4 + x_6) \cdot (x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 + x_5) \cdot (x_1 + \bar{x}_3 + x_5) \cdot (\bar{x}_3 + x_5 + x_6) \cdot (x_2 + x_3 + \bar{x}_4) \cdot (x_1 + x_6)$
6	7	T	9,15	$(x_1 + x_2 + \bar{x}_5 + \bar{x}_6) \cdot (x_1 + x_2 + \bar{x}_4 + \bar{x}_5 + x_6) \cdot (\bar{x}_2 + x_4 + \bar{x}_5 + \bar{x}_6) \cdot (x_1 + \bar{x}_3 + \bar{x}_4 + x_5) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_5) \cdot (\bar{x}_3 + x_5 + x_6) \cdot (\bar{x}_1 + \bar{x}_2 + x_3 + x_6)$
6	7	T	8,07	$(\bar{x}_1 + \bar{x}_2 + x_5) \cdot (x_2 + x_3) \cdot (x_3 + \bar{x}_5 + \bar{x}_6) \cdot (\bar{x}_1 + x_2 + \bar{x}_3 + x_4 + x_5 + x_6) \cdot (\bar{x}_2 + \bar{x}_3) \cdot (x_2 + x_3 + x_6) \cdot (x_1 + \bar{x}_2 + x_3 + x_4 + x_5 + x_6)$
7	8	F	30,36	$(x_2 + \bar{x}_7) \cdot (x_1 + x_4) \cdot (x_1 + \bar{x}_3 + x_4 + x_5 + \bar{x}_6) \cdot (\bar{x}_2 + \bar{x}_7) \cdot x_7 \cdot (x_2 + x_5 + x_6 + x_7) \cdot (x_1 + x_3 + \bar{x}_4 + x_5 + \bar{x}_6) \cdot (x_3 + x_5 + x_7)$
7	8	F	29,69	$(x_5 + x_7) \cdot x_2 \cdot (x_1 + x_2 + x_3 + \bar{x}_5 + \bar{x}_7) \cdot x_7 \cdot (\bar{x}_1 + \bar{x}_5 + x_6 + x_7) \cdot (\bar{x}_1 + \bar{x}_2 + x_6 + x_7) \cdot (\bar{x}_2 + \bar{x}_7) \cdot (x_1 + x_2 + \bar{x}_4 + x_5 + x_6)$
7	8	F	28,92	$(\bar{x}_5 + \bar{x}_6 + \bar{x}_7) \cdot (x_3 + \bar{x}_4 + x_7) \cdot (\bar{x}_1 + x_3 + x_5 + x_6 + \bar{x}_7) \cdot (x_1 + x_3 + \bar{x}_5 + x_6 + x_7) \cdot (x_2 + x_6) \cdot (x_2 + \bar{x}_6) \cdot \bar{x}_2 \cdot (x_2 + x_3 + x_4 + \bar{x}_5 + x_7)$
7	8	T	29,23	$(x_3 + \bar{x}_4 + x_5) \cdot (\bar{x}_2 + x_4) \cdot (x_1 + x_2 + \bar{x}_4 + x_6) \cdot (x_6 + x_7) \cdot (\bar{x}_3 + \bar{x}_5 + \bar{x}_7) \cdot (\bar{x}_4 + \bar{x}_5) \cdot (\bar{x}_1 + x_4 + x_5 + \bar{x}_6 + x_7) \cdot (x_4 + x_5 + \bar{x}_6)$
7	8	T	29,3	$(\bar{x}_2 + \bar{x}_3 + x_4 + x_5 + \bar{x}_6 + x_7) \cdot (x_1 + x_3 + \bar{x}_4 + x_6 + \bar{x}_7) \cdot (\bar{x}_3 + \bar{x}_4 + x_5) \cdot (\bar{x}_3 + \bar{x}_4 + x_6 + \bar{x}_7) \cdot (\bar{x}_1 + x_2 + \bar{x}_4 + x_5) \cdot (x_4 + x_5 + \bar{x}_6 + x_7) \cdot (x_2 + \bar{x}_3 + \bar{x}_4 + x_6 + \bar{x}_7) \cdot (\bar{x}_2 + x_3 + \bar{x}_4)$
7	8	T	29	$(\bar{x}_2 + x_5 + x_6 + x_7) \cdot (x_2 + \bar{x}_4 + \bar{x}_5 + \bar{x}_7) \cdot (x_1 + x_2 + \bar{x}_3 + \bar{x}_6 + x_7) \cdot (x_1 + x_2 + x_3 + \bar{x}_5 + x_6 + \bar{x}_7) \cdot (\bar{x}_3 + \bar{x}_5 + x_6 + \bar{x}_7) \cdot (x_1 + x_2 + \bar{x}_3 + \bar{x}_7) \cdot (\bar{x}_1 + x_2 + \bar{x}_4 + \bar{x}_6) \cdot (x_3 + x_5 + x_6 + \bar{x}_7)$
7	8	T	115,22	$(x_1 + x_3 + x_4 + x_5 + x_6 + \bar{x}_8) \cdot \bar{x}_5 \cdot (\bar{x}_2 + x_4 + \bar{x}_5 + x_6 + \bar{x}_8) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 + x_5 + \bar{x}_6) \cdot (\bar{x}_1 + x_2 + x_3 + x_6) \cdot (\bar{x}_2 + x_3 + \bar{x}_4 + x_6 + x_7 + x_8) \cdot (\bar{x}_3 + x_5) \cdot (\bar{x}_1 + x_2 + \bar{x}_4 + \bar{x}_6) \cdot (x_3 + x_5)$
8	9	F	112,65	$(x_1 + x_2 + x_3 + \bar{x}_7 + x_8) \cdot (x_3 + x_5 + x_6 + x_7 + x_8) \cdot (\bar{x}_5 + x_8) \cdot (x_1 + \bar{x}_6) \cdot (x_2 + x_4 + \bar{x}_6 + x_8) \cdot (\bar{x}_1 + \bar{x}_2 + x_4 + x_5 + x_6 + \bar{x}_7) \cdot x_5 \cdot (x_1 + x_2 + \bar{x}_3 + \bar{x}_6 + \bar{x}_8) \cdot (\bar{x}_5 + \bar{x}_8)$

n	m	SAT	Time (s)	Fórmula
8	9	F	119,3	$(x_3 + x_4 + \bar{x}_6 + \bar{x}_8) \cdot (x_6 + \bar{x}_7) \cdot (\bar{x}_2 + x_3 + \bar{x}_4 + x_5 + x_8) \cdot x_7 \cdot (x_1 + \bar{x}_2 + x_5 + \bar{x}_7 + \bar{x}_8) \cdot (x_2 + x_7 + x_8) \cdot (\bar{x}_6 + \bar{x}_7) \cdot (x_1 + x_5 + \bar{x}_8) \cdot (x_1 + \bar{x}_4 + x_5 + \bar{x}_6 + x_7)$
8	9	T	114,26	$(\bar{x}_2 + \bar{x}_3 + \bar{x}_4 + \bar{x}_6) \cdot (x_2 + \bar{x}_4 + x_5 + x_6 + x_7 + \bar{x}_8) \cdot (\bar{x}_1 + \bar{x}_4 + \bar{x}_6 + \bar{x}_8) \cdot (\bar{x}_4 + x_5 + \bar{x}_6 + x_8) \cdot (x_2 + \bar{x}_4 + \bar{x}_6) \cdot (x_1 + \bar{x}_4 + x_7 + \bar{x}_8) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_5 + x_6 + x_8) \cdot (\bar{x}_1 + x_2 + x_3 + x_6 + x_8) \cdot (x_2 + \bar{x}_3 + \bar{x}_4 + x_6 + x_7)$
8	9	T	121,2	$(x_1 + \bar{x}_5 + \bar{x}_6 + \bar{x}_7 + \bar{x}_8) \cdot (x_2 + x_3 + x_4 + \bar{x}_6 + \bar{x}_7 + x_8) \cdot (x_3 + x_4 + \bar{x}_5 + \bar{x}_6 + \bar{x}_7 + \bar{x}_8) \cdot (\bar{x}_1 + \bar{x}_3 + \bar{x}_4 + \bar{x}_5 + x_6 + \bar{x}_7 + x_8) \cdot (\bar{x}_3 + \bar{x}_7) \cdot (x_4 + x_5 + \bar{x}_7) \cdot (x_1 + x_3 + \bar{x}_4) \cdot (x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 + \bar{x}_5 + \bar{x}_6 + \bar{x}_7) \cdot (x_4 + \bar{x}_5 + \bar{x}_6 + x_7 + \bar{x}_8)$
8	9	T	120,8	$(x_5 + \bar{x}_7 + \bar{x}_8) \cdot (\bar{x}_1 + x_3 + \bar{x}_4 + \bar{x}_5) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_4 + \bar{x}_6 + x_8) \cdot (\bar{x}_1 + x_2 + x_6 + \bar{x}_7 + x_8) \cdot (x_2 + \bar{x}_3 + x_4 + x_6 + \bar{x}_8) \cdot (x_2 + \bar{x}_3 + x_4 + x_5) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_6) \cdot (x_2 + \bar{x}_4 + \bar{x}_6 + x_8) \cdot (x_2 + x_3 + x_4 + \bar{x}_5 + \bar{x}_6 + \bar{x}_8)$
9	10	F	406,95	$(\bar{x}_2 + \bar{x}_3 + x_5 + x_7) \cdot (x_2 + x_5 + x_6 + x_7 + x_9) \cdot (\bar{x}_3 + x_5 + x_7 + x_8) \cdot (x_1 + \bar{x}_4 + \bar{x}_5 + x_6 + x_8) \cdot (\bar{x}_2 + x_3 + x_5 + x_7 + x_8 + \bar{x}_9) \cdot (\bar{x}_2 + \bar{x}_4 + x_7 + x_9) \cdot (\bar{x}_2 + x_4 + x_6 + x_9) \cdot x_1 \cdot x_5 \cdot (\bar{x}_1 + \bar{x}_5)$
9	10	F	415,25	$\bar{x}_8 \cdot (x_2 + x_5 + x_6 + \bar{x}_7 + x_8) \cdot (\bar{x}_1 + \bar{x}_3 + \bar{x}_5 + \bar{x}_9) \cdot \bar{x}_4 \cdot (\bar{x}_1 + x_3 + \bar{x}_4 + x_5 + \bar{x}_8) \cdot (x_3 + \bar{x}_5 + \bar{x}_6) \cdot (x_4 + x_8) \cdot (x_1 + \bar{x}_4 + x_6 + x_7 + \bar{x}_9) \cdot (\bar{x}_1 + \bar{x}_2 + x_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + x_5 + \bar{x}_6 + x_7 + \bar{x}_9)$
9	10	F	415,36	$(x_6 + \bar{x}_7 + x_8 + \bar{x}_9) \cdot (\bar{x}_1 + x_7 + x_9) \cdot (x_1 + x_2 + \bar{x}_3 + x_4 + x_5 + \bar{x}_6 + x_8) \cdot (\bar{x}_5 + \bar{x}_8) \cdot (\bar{x}_1 + \bar{x}_2 + x_3 + x_6 + x_7 + \bar{x}_8 + \bar{x}_9) \cdot (\bar{x}_1 + x_2 + \bar{x}_7 + \bar{x}_8) \cdot (\bar{x}_1 + \bar{x}_2 + x_3 + \bar{x}_4 + \bar{x}_5 + \bar{x}_8) \cdot x_5 \cdot (x_1 + \bar{x}_4 + \bar{x}_8 + x_9) \cdot x_8$
9	10	T	417,38	$(x_3 + x_8) \cdot (x_1 + \bar{x}_2 + x_5 + \bar{x}_6 + x_9) \cdot (x_3 + x_6 + x_9) \cdot (x_3 + x_5 + \bar{x}_6 + \bar{x}_8) \cdot (x_1 + x_2 + \bar{x}_5 + x_7 + \bar{x}_8 + \bar{x}_9) \cdot (\bar{x}_1 + x_2 + \bar{x}_4 + x_5 + \bar{x}_6 + \bar{x}_7 + x_9) \cdot (x_1 + x_2 + x_4 + \bar{x}_6 + x_8 + \bar{x}_9) \cdot (\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4 + x_7 + \bar{x}_8) \cdot (\bar{x}_1 + x_2 + x_3 + x_5 + \bar{x}_6 + x_8 + \bar{x}_9) \cdot (x_2 + \bar{x}_3 + x_4 + \bar{x}_6 + \bar{x}_7 + \bar{x}_9)$
9	10	T	425,14	$(x_3 + x_6 + x_7 + \bar{x}_9) \cdot (x_6 + \bar{x}_8) \cdot (x_1 + x_3 + x_4 + \bar{x}_5 + \bar{x}_7 + \bar{x}_8 + \bar{x}_9) \cdot (\bar{x}_1 + x_2 + x_4 + x_5 + x_6 + x_8) \cdot (x_1 + \bar{x}_2 + x_5 + \bar{x}_7 + \bar{x}_8) \cdot (\bar{x}_2 + \bar{x}_3 + x_5 + \bar{x}_6 + x_9) \cdot (\bar{x}_2 + \bar{x}_4 + x_5 + x_6 + x_8 + x_9) \cdot (x_1 + x_3) \cdot (\bar{x}_1 + x_3 + \bar{x}_5 + \bar{x}_7 + x_8 + \bar{x}_9) \cdot (\bar{x}_1 + \bar{x}_4 + x_5 + x_7 + \bar{x}_8 + x_9)$
9	10	T	423,4	$(x_1 + \bar{x}_2 + x_3 + x_4 + \bar{x}_5 + x_6 + \bar{x}_8 + x_9) \cdot (\bar{x}_1 + x_2 + \bar{x}_4 + x_7 + x_9) \cdot (x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_6) \cdot (\bar{x}_1 + \bar{x}_3 + x_4 + x_7 + x_9) \cdot (x_1 + \bar{x}_3 + \bar{x}_4 + \bar{x}_7 + x_8) \cdot (\bar{x}_3 + \bar{x}_4 + \bar{x}_6 + \bar{x}_7 + \bar{x}_8 + \bar{x}_9) \cdot (\bar{x}_2 + x_4 + x_5 + \bar{x}_6 + x_8 + \bar{x}_9) \cdot (\bar{x}_4 + \bar{x}_7 + x_8) \cdot (\bar{x}_2 + \bar{x}_4 + x_5 + \bar{x}_6 + x_7 + \bar{x}_8) \cdot (x_3 + x_4 + x_7 + x_8)$

BIBLIOGRAFÍA

[1] M.J. Pérez–Jiménez, A. Romero–Jiménez, F. Sancho–Caparrini. A polynomial complexity class in P systems using membrane division. *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems, DCFS 2003*, 2003, 284–294.

[2] Gh. Păun. P Systems with active membranes: attacking NP complete problems, *Journal of Automata, Languages and Combinatorics* **6** (1), 2001, 75–90.

[3] Gh. Păun. *Membrane Computing. An introduction*. Springer-Verlag, Berlin, 2002.

[4] A. Riscos-Núñez. *Programación celular: resolución eficiente de problemas numéricos NP-completos*. Tesis doctoral, Universidad de Sevilla, 2004.

[5] I. Pérez-Hurtado. *Desarrollo y aplicaciones de un entorno de programación para Computación Celular: P-Lingua*. Ph.D. Thesis. Universidad de Sevilla, 2010.

[6] M.J. Pérez-Jiménez, A. Romero, F. Sancho. A polynomial complexity class in P systems using membrane division. *Journal of Automata, Languages and Combinatorics*, 11, 4 (2006), 423-434.