6 PBE. MODELOS EN MATEMÁTICAS

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 Matemáticas (posibilidad de colorear un grafo con 3 colores), haciendo uso sistemas P que trabajan a modo de tejidos.

1 Ejemplo 4. Coloración de grafos

Este ejemplo trata de presentar un problema clásico NP-completo conocido como 3-COL, es decir, la coloración de un grafo a través de 3 colores.

1. 1 Escenario

El problema `3-COL` se formula como sigue: "*dado un grafo no dirigido, determinar si existe una coloración válida del grafo con tres colores*". Recordemos que una coloración de un grafo no dirigido $\mathcal{G} = (V, E)$ con tres colores es una aplicación $f : V \to \{1, 2, 3\}$ y se dirá que una tal coloración es válida si satisface la siguiente condición adicional: para cada arista $\{u, v\} \in E$ del grafo se tiene que $f(u) \neq f(v)$; es decir, una coloración válida asigna colores distintos a nodos del grafo que sean adyacentes.

Por cuestiones de simplicidad de notación en la solución que se va a presentar, usaremos los elementos del conjunto $\{R, G, B\}$ para designar los tres colores $\{1, 2, 3\}$. Emplearemos justamente los colores rojo (**R**ed), verde (**G**reen) y azul (**B**lue), si bien podrían tomarse otros cualesquiera. El problema original es de hecho un problema de decisión, que debe respondernos simplemente si el grafo dado admite o no la 3-coloración.

Veamos por ejemplo el grafo de la Figura 3.

Si nos fijamos en el nodo 1, éste está conectado con 2 y 3, que a su vez están conectados. Por tanto, los colores de los tres nodos deberán ser diferentes. Ahora

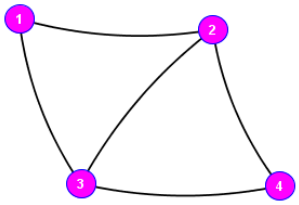

FIGURA 1: 3-COL. Instancia a resolver.

bien, el nodo 4 no está conectado con el nodo 1, luego estos dos nodos sí podrán compartir color.

A continuación vamos a presentar una solución eficiente del problema `3-COL` a través de una familia de sistemas P de tejidos reconocedores con división celular.

1. 2 Modelo

Para cada $m, n \in \mathbb{N}$, consideramos el sistema P de tejidos reconocedor con división celular $\Pi(\langle n, m\rangle) = (\Gamma, \mathcal{E}, \Sigma, \mathcal{M}_1, \mathcal{M}_2, \mathcal{R}, i_{in})$ definido como sigue:

- El alfabeto de trabajo Γ es el conjunto

  $\{A_i, R_i, G_i, B_i, T_i, \overline{R}_i, \overline{G}_i\, \overline{B}_i, : \ 1 \leq i \leq n\} \cup$
  $\{a_i \ : \ 1 \leq i \leq 2n + m + \lceil \log m \rceil + 11\} \ \cup \ \{c_i \ : \ 1 \leq i \leq 2n + 1\} \cup$
  $\{d_i \ : \ 1 \leq i \leq \lceil \log m \rceil + 1\} \ \cup \{z_i \ : \ 2 \leq i \leq m + \lceil \log m \rceil + 6\} \cup$
  $\{A_{ij}, P_{ij}, \overline{P}_{ij}, R_{ij}, G_{ij}, B_{ij} \ : \ 1 \leq i < j \leq n\} \cup \{b, D, \overline{D}, e, T, S, N, \natural, \texttt{yes}, \texttt{no}\}$

- El alfabeto del entorno es $\mathcal{E} = \Gamma - \{\texttt{yes}, \texttt{no}\}$

- El alfabeto de entrada Σ es el conjunto $\{A_{ij} \ : \ 1 \leq i < j \leq n\}$

- Los multiconjuntos iniciales contenidos en las células son: $\mathcal{M}_1 = \{a_1, b, c_1, \texttt{yes}, \texttt{no}\}$ y $\mathcal{M}_2(n) = \{D, A_1, \ldots, A_n\}$

- El conjunto $\mathcal{R}$ consta de las siguientes reglas:

  (a) $\{[A_i]_2 \to [R_i]_2[T_i]_2, \ [T_i]_2 \ \to \ [G_i]_2[B_i]_2 : \ 1 \leq i \leq n\}$
  (b) $\{(1, a_i/a_{i+1}, 0) : \ 1 \leq i \leq 2n + m + \lceil \log m \rceil + 10\}$
  (c) $\{(1, c_i/c_{i+1}^2, 0) : \ 1 \leq i \leq 2n\}$
  (d) $(1, c_{2n+1}/D, 2)$
  (e) $(2, c_{2n+1}/d_1\overline{D}, 0)$
  (f) $\{(2, d_i/d_{i+1}^2, 0) : \ 1 \leq i \leq \lceil \log m \rceil\}$
  (g) $(2, \overline{D}/e\, z_2, 0)$

(h) $\{(2, z_i/z_{i+1}, 0) : 1 \leq i \leq m + \lceil \log m \rceil + 5\}$

(i) $\{(2, d_{\lceil \log m \rceil + 1} A_{ij}/P_{ij}, 0) : 1 \leq i < j \leq n\}$

(j) $\{(2, P_{ij}/R_{ij}\overline{P}_{ij}, 0) : 1 \leq i < j \leq n\}$

(k) $\{(2, \overline{P}_{ij}/B_{ij}G_{ij}, 0) : 1 \leq i < j \leq n\}$

(l) $\{(2, R_iR_{ij}/R_i\overline{R}_j, 0) : 1 \leq i < j \leq n\}$

(m) $\{(2, B_iB_{ij}/B_i\overline{B}_j, 0) : 1 \leq i < j \leq n\}$

(n) $\{(2, G_iG_{ij}/G_i\overline{G}_j, 0) : 1 \leq i < j \leq n\}$

(o) $\{(2, \overline{R}_jR_j/\flat, 0) : 1 \leq j \leq n\}$

(p) $\{(2, \overline{B}_jB_j/\flat, 0) : 1 \leq j \leq n\}$

(q) $\{(2, \overline{G}_jG_j/\flat, 0) : 1 \leq j \leq n\}$

(r) $(2, e\,\flat/\lambda, 0)$

(s) $(2, e\ z_{m+\lceil \log m \rceil + 6}/T, 0)$

(t) $r_{21} \equiv (2, T/\lambda, 1)$

(u) $(1, b\,T/S, 0)$

(v) $(1, S\ \texttt{yes}/\lambda, 0)$

(x) $(1, b\ a_{2n+m+\lceil \log m \rceil + 11}/N, 0)$

(y) $(1, N\ \texttt{no}/\lambda, 0)$

- La célula de entrada es $i_{in} = 2$.

Sea $u = (V, E)$ un grafo no dirigido con n vértices y m aristas. Consideramos las funciones $cod(u) = \{\{A_{ij} : \{A_i, A_j\} \in E \wedge 1 \leq i < j \leq n\}\}$ y $s(u) = \langle m, n \rangle$. La ejecución del sistema $\Pi(s(u))$ con entrada $cod(u)$ se estructura en cuatro fases:

- *Fase de generación de coloraciones*: se generan todas las posibles coloraciones del grafo mediante la aplicación sucesiva de reglas de división en la célula 2. En esta fase se pude observar con bastante claridad el no determinismo de los sistemas P y, a la vez, la confluencia de los mismos.

- *Fase de pre-chequeo*: esta fase permite introducir objetos R_{ij}, G_{ij}, B_{ij}, para cada arista $\{i, j\}$ del grafo, en todas las células etiquetadas por 2, a fin de analizar la validez de la coloración que cada una de ellas codifica.

- *Fase de chequeo*: esta fase chequea en cada célula 2 si la coloración que codifica es válida, para ello hará uso de los objetos R_{ij}, G_{ij}, B_{ij}.

- *Fase de salida*: el sistema proporciona la salida correspondiente en función de lo analizado anteriormente.

1.0.1. 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. La entrada, el grafo concreto, dependerá de los datos introducidos por el usuario, lo cuál veremos en el siguiente apartado.
El código es el siguiente:

```
@model<tissue_psystems>

def main()
{
 /* tissue P system skeleton */
 call tricolor_tissue(n,m);
}

def tricolor_tissue(n,m)
{
  call init_cells();
  call init_multisets(n);
  call init_environment(n,m);
  call init_rules(n,m);
}

def init_cells()
{
  @mu = [[]'1 []'2]'0;
}

def init_rules(n,m)
{
  /* r1,i */ [A{i}]'2 --> [R{i}]'2 [T{i}]'2 : 1<=i<=n;
  /* r2,i */ [T{i}]'2 --> [G{i}]'2 [B{i}]'2 : 1<=i<=n;

  /* r3,i */ [a{i}]'1 <--> [a{i+1}]'0 : 1<=i<=2*n+m+@ceil(@log(m))+10;
  /* r4,i */ [c{i}]'1 <--> [c{i+1}*2]'0 : 1<=i<=2*n;
  /* r5 */ [c{2*n+1}]'1 <--> [D]'2;
  /* r6 */ [c{2*n+1}]'2 <--> [d{1},noD]'0;
  /* r7,i */ [d{i}]'2 <--> [d{i+1}*2]'0 : 1<=i<=@ceil(@log(m));
  /* r8 */ [noD]'2 <--> [e,z{2}]'0;
  /* r9,i */ [z{i}]'2 <--> [z{i+1}]'0 : 2<=i<=m+@ceil(@log(m))+5;
  /* r10,i,j */ [d{@ceil(@log(m))+1},A{i,j}]'2 <--> [P{i,j}]'0 : i<j<=n, 1<=i<n;
  /* r11,i,j */ [P{i,j}]'2 <--> [R{i,j},noP{i,j}]'0 : i<j<=n, 1<=i<n;
  /* r12,i,j */ [noP{i,j}]'2 <--> [B{i,j},G{i,j}]'0 : i<j<=n, 1<=i<n;
  /* r13,i,j */ [R{i},RR{i,j}]'2 <--> [R{i},noR{j}]'0 : i<j<=n, 1<=i<n;
  /* r14,i,j */ [B{i},BB{i,j}]'2 <--> [B{i},noB{j}]'0 : i<j<=n, 1<=i<n;
  /* r15,i,j */ [G{i},GG{i,j}]'2 <--> [G{i},noG{j}]'0 : i<j<=n, 1<=i<n;
  /* r16,j */ [noR{j},R{j}]'2 <--> [bb]'0 : 1<=j<=n;
  /* r17,j */ [noB{j},B{j}]'2 <--> [bb]'0 : 1<=j<=n;
  /* r18,j */ [noG{j},G{j}]'2 <--> [bb]'0 : 1<=j<=n;
  /* r19 */ [e,bb]'2 <--> []'0;
  /* r20 */ [e,z{m+@ceil(@log(m))+6}]'2 <--> [T]'0;
  /* r21 */ [T]'2 <--> []'1;
  /* r22 */ [b,T]'1 <--> [S]'0;
  /* r23 */ [S,yes]'1 <--> []'0;
  /* r24 */ [b,a{2*n+m+@ceil(@log(m))+11}]'1 <--> [N]'0;
  /* r25 */ [N,no]'1 <--> []'0;

}

def init_multisets(n)
{
        @ms(1) = a{1},b,c{1},yes,no;
        @ms(2) = D;
        @ms(2) += A{i} : 1<=i<=n;
```

```
        @ms(2) += A{e{i,1},e{i,2}} : 1<=i<=ne;
}

def init_environment(n,m)
{
  @ms(0) = A{i},R{i},G{i},B{i},T{i},noR{i},noG{i},noB{i} : 1<=i<=n;
  @ms(0) += a{i} : 1<=i<=2*n+m+@ceil(@log(m))+11;
  @ms(0) += c{i} : 1<=i<=2*n+1;
  @ms(0) += d{i} : 1<=i<=@ceil(@log(m))+1;
  @ms(0) += z{i} : 2<=i<=m+@ceil(@log(m))+6;
  @ms(0) += A{i,j},P{i,j},noP{i,j},RR{i,j},GG{i,j},BB{i,j} : i<j<=n,1<=i<n;
  @ms(0) += b,D,noD,e,T,S,N,bb;
}
```

1. 3 Simulación

El sistema P descrito en el apartado anterior resuelve el problema **3-COL** para la fórmula indicada. P-Lingua acepta el fichero anterior y dispone de simuladores preparados para su simulación. Ahora bien, como en el caso de la unidad anterior, el archivo contiene una serie de parámetros cuyos valores dependerán de la instancia que se quiera resolver.

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

1. Como en el ejemplo de SAT de la unidad anterior, es necesario definir una aplicación basada en MeCoSim adaptada a este problema (en este caso 3-COL). Recordemos que la personalización se lleva a cabo empleando un fichero de configuración Excel con las entradas y salidas (tabuladas y gráficas) deseadas y los parámetros para P-Lingua. El archivo Excel correspondiente está disponible como en la unidad anterior en el repositorio de apps "http://www.p-lingua.org/mecosim/matvida/apps/", de modo que seleccione este repositorio y seleccione la aplicación "Ejemplo 4 - 3-COL" para instalarse con doble click y salimos del repositorio de apps. Añadimos la aplicación con el botón "New", y desde ahí seleccionar el archivo "Ejemplo4_3col.xls", lo que hará que en nuestro listado aparezca en primer lugar la aplicación "3-COL", lista para ser lanzada.

2. Acudir al gestor de repositorios para importar el archivo de modelo ("Ejemplo4_3col.pli") y el de escenario ("Ejemplo4_3col.ec2"), para poder cargar el sistema P correspondiente expresado en lenguaje P-Lingua y aplicarlo sobre el grafo indicado.

3. Lance la aplicación 3-COL mediante "Run". Observe la apariencia de la aplicación adaptada para 3-COL, que contiene:

   - Entrada para parámetros generales: el número de nodos, n, y el número de aristas, m.
   - Entrada para las aristas, con dos columnas indicando los dos nodos que conecta cada arista.

- Salida detallada con los objetos presentes en cada membrana en cada paso.
- Salida "Result", que deberá indicar si el grafo admite una 3-coloración ("yes") o no ("No").

4. Cargue el modelo mediante *Model* > *Set Model*, abra el escenario descargado mediante *Scenario* > *Open*, y proceda a simular mediante *Simulation* > *Simulate!*. ¿Cuál es el resultado? ¿se admite la 3-coloración? ¿cuál es?. A continuación puede salir de la aplicación y volver al listado de aplicaciones de MeCoSim.

5. ¿Le ha costado responder la primera pregunta anterior? ¿Y a la segunda? Lo más probable es que haya encontrado fácil respuesta a la primera pregunta a través de alguna de las salidas proporcionadas, pero que le haya resultado más difícil responder a la segunda cuestión. Esto se debe a que el problema a resolver no pretendía hallar la coloración sino resolver el problema de decisión de determinar si el grafo puede ser coloreado o no empleando 3 colores. No obstante, podemos estar interesado en este u otros muchos problemas de grafos. Para mostrar información vsual sobre árboles y grafos se ha desarrollado un plugin denominado "Graphs plugin", que podemos instalar desde el *gestor de repositorios de MeCoSim* > *Plugins* (recordemos que tras instalarlo debemos salir del programa completo, y configurar el plugin para que aparezca en nuestro listado de plugins introduciendo las siguientes líneas en la ruta *MeCoSim* > *prop* > *plugins-properties*:

```
plugin-graphs = mecosim.plugins.graphsPlugin.GraphViewer
pluginname-graphs = Graph Viewer
pluginmethod-graphs = entryPoint
pluginorder-graphs = 6
pluginjar-graphs-1 = GraphsPlugin.jar
pluginparam-graphs-1 = numver
pluginparam-graphs-2 = graph
pluginparam-graphs-3 = twocolor
```

Este plugin permite visualizar información sobre grafos a partir de información numérica o textual generada por el mecanismo de generación de parámetros de MeCoSim. En este curso se empleará para visualizar tanto el grafo inicial sin colorear como los grafos codificados al final de la computación en cada membrana etiquetada con 2, conteniendo objetos de tipo R, G o B. Ya estamos listos para arrancar de nuevo MeCoSim, y entrar en la aplicación 3-COL. Una vez en ella, podemos probar el ejemplo anterior, como se muestra en la Figura 2.

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

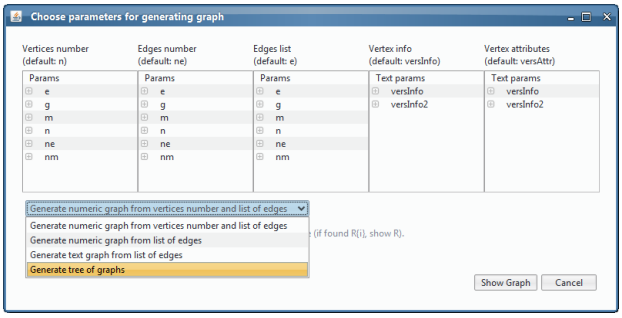

FIGURA 2: 3-COL - Selección de parámetros en Graphs plugin para MeCoSim.

La idea de esta unidad es trabajar con un problema NP-completo resuelto con un sistema P de tejidos con reglas de división celular, en este caso el problema `3-COL`, además de ilustrar el mecanismo de personalización de aplicaciones de MeCoSim para acercar las aplicaciones al usuario final.

En concreto, se va a considerar como ejemplo para instanciar el problema `3-COL` a resolver el grafo de la Figura 3.

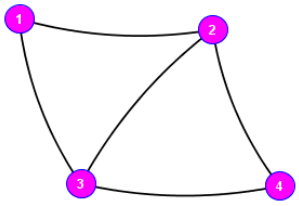

FIGURA 3: 3-COL. Instancia a resolver.

Deberá seguir los pasos descritos en el apartado de simulación hasta simular el sistema P introducido para el grafo del ejemplo, analizando la salida del sistema y empleando el plugin de visualización de grafos para mostrar tanto el grafo inicial (dejando las opciones por defecto en la ventana de selección de parámetros de la Figura 2, como los grafos finales obtenidos en cada membrana que codifica una combinación de colores para los nodos del grafo (modificando en la ventana de selección el tipo de grafo, cambiándolo por la opción del desplegable denominada "Generate tree of graphs"). Esta última opción muestra la información de forma jerárquica, de forma que seleccionemos una membrana del sistema, y un paso (en este caso tenemos establecida en la configuración de esta aplicación que solamente muestre el último paso de la computación) y nos muestre el grafo codificado en la membrana correspondiente. Esto es posible gracias al efecto combinado del mecanismo de generación de parámetros de MeCoSim y del plugin de visualización de grafos.

La Figura 4 muestra el grafo coloreado contenido en una de las membranas.

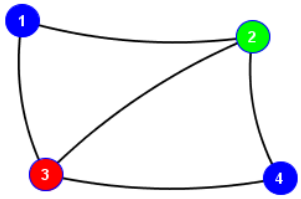

FIGURA 4: 3-COL. Instancia resuelta.

Se propone al alumno buscar un mínimo de 5 ejemplos de grafos que contengan entre 4 y 8 nodos (para simplificar tanto su comprensión de la evolución del sistema como su computación) y repita el proceso, analizando también mediante el mecanismo de depuración cómo se van disparando las reglas correspondientes a cada una de las fases descritas en el apartado de “Modelo”.