Computabilidad y Complejidad (CMC)

Laura Ruiz Muñoz

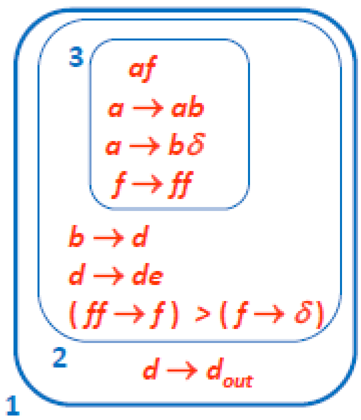
Ángel Carrascosa Beltrán

Práctica 3 – Computación con membranas. Sistemas P.

Actividades propuestas

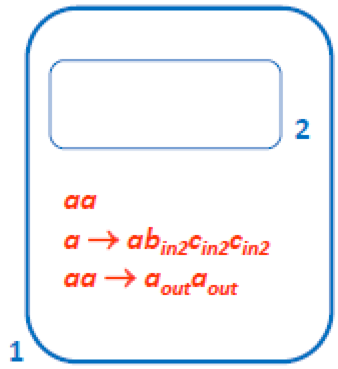
1. **Especifique el conjunto de naturales calculado por cada uno de los siguientes sistemas P**

(a)



$i_0 = 1$

(b)



$i_0 = 2$

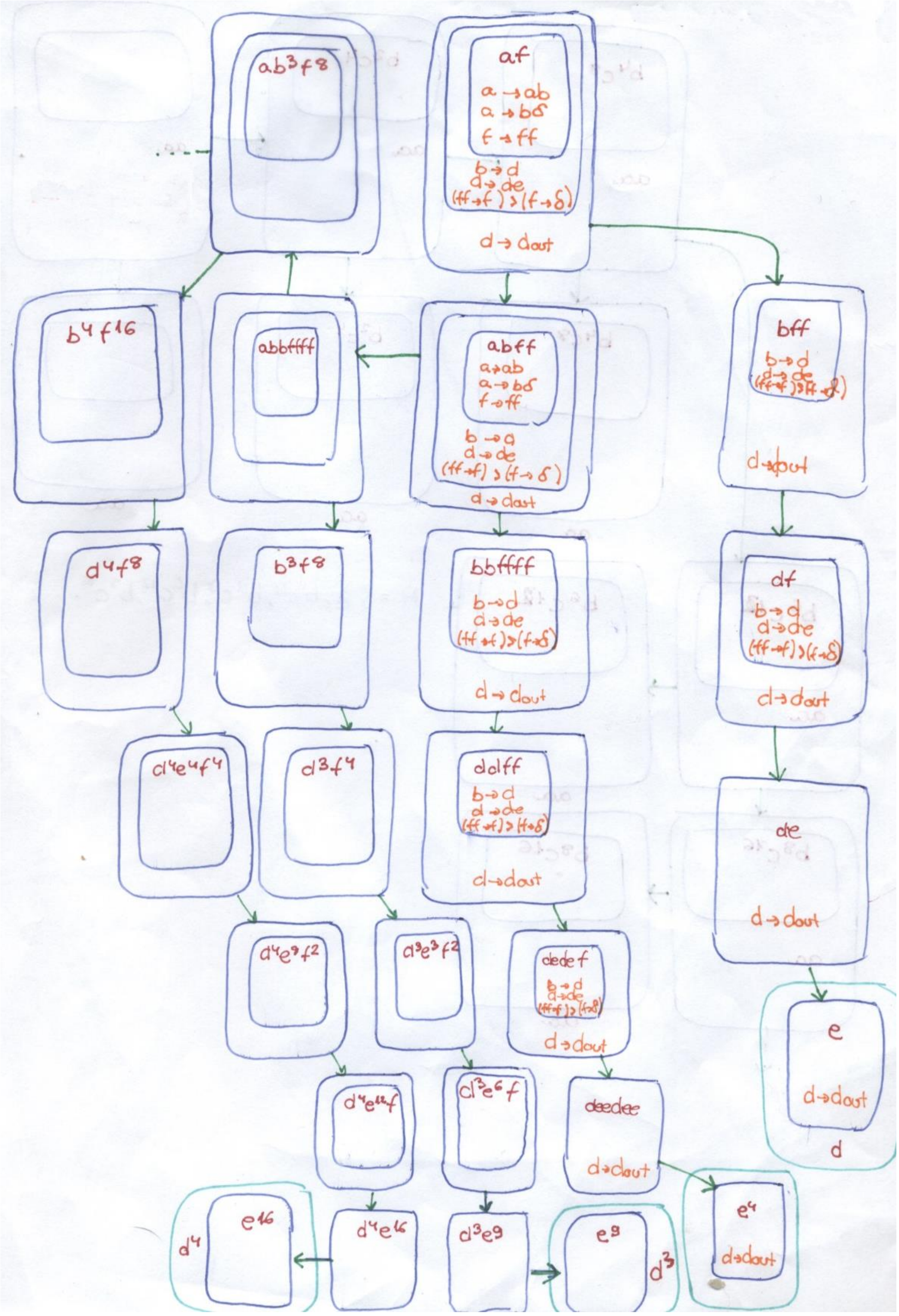
Justificación (a):

Inicialmente, podemos aplicar en la tercera membrana en paralelo dos distintas combinaciones de reglas, ya que ha dos posibles transiciones a partir de a, por lo que generaremos dos sistemas, uno de ellos con el doble de f que el anterior y una b más, y el segundo causará la ruptura de la tercera membrana, pasando la cadena a la membrana exterior (2) y aplicando a partir de ahora sus reglas correspondientes.

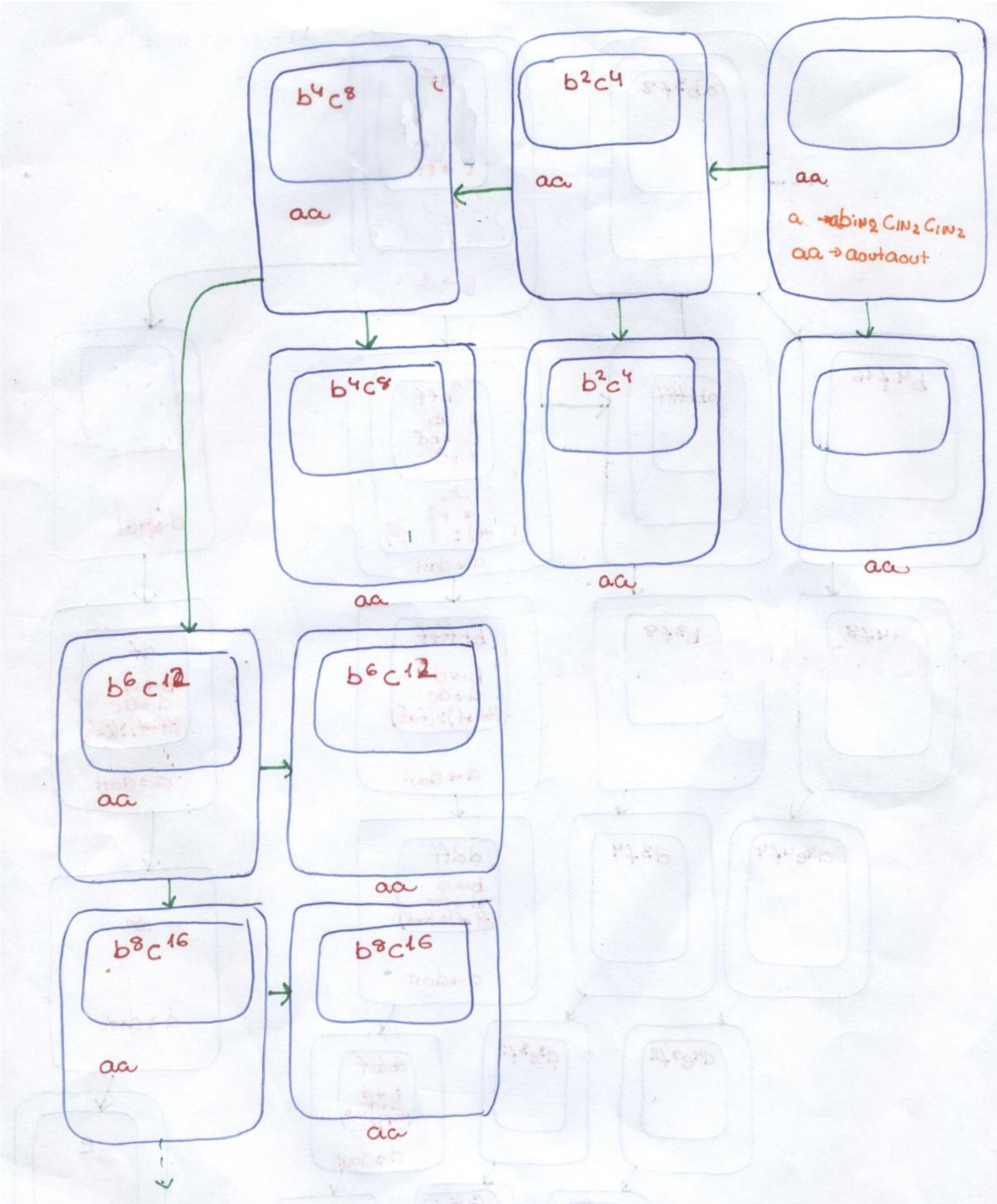
La primera configuración generada podrá generar a su vez otros dos sistemas, siguiendo las características previamente dichas que generará cada uno. La segunda configuración solo tendrá una posible ejecución de aquí al final. Primero sustituirá las b por d, posteriormente por cada d genere una e. Además, vamos reduciendo el número de f (haciendo que dos f se conviertan en una) hasta que quede una, en tal caso, se podrá aplicar la norma menos prioritaria y romper la membrana dos (δ), pasando a aplicar las normas de la membrana 1. La cual saca las d del sistema, dejando dentro solo las e. No hay más posibles configuraciones, y la membrana final indicada es la 1, así que hemos llegado al final de la ejecución de la segunda configuración.

Como se puede ver en el dibujo, al final de cada configuración se mostrará e^(n^2), siendo n el nº de la iteración donde se ha generado el camino único a esa configuración, por lo que el conjunto de naturales serán los cuadrados ∀n≥1,

M = {1, 4, 9, 16, 25…}



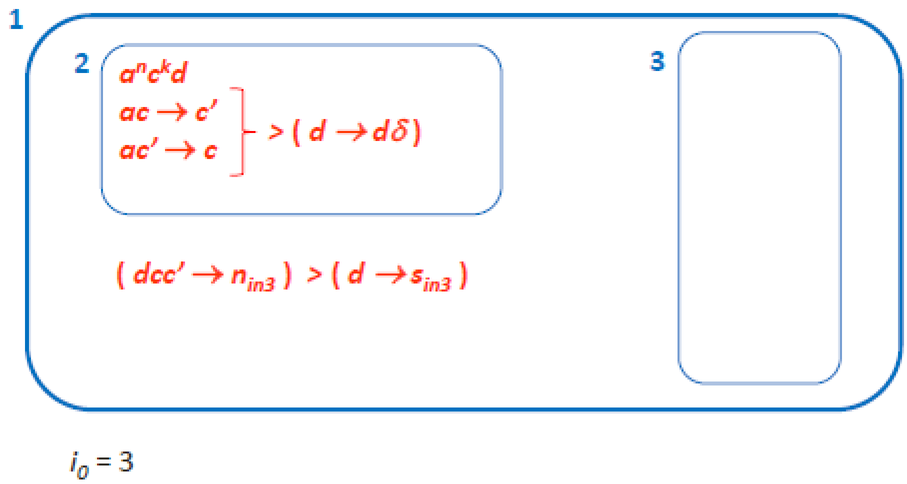
Justificación (b):



A partir de la configuración inicial se puede llegar con sus reglas a dos configuraciones diferentes. La primera configuración será final, la cual envía fuera del sistema las dos aa dentro de la primera membrana, la segunda configuración generará por cada a de la cadena una a que se quedará dentro de la primera membrana, y una b y dos c, ambas siendo enviadas a la membrana 2, la cual es la membrana que debe reflejar la cadena resultado. A partir de esta configuración pueden surgir dos configuraciones más realizando otra iteración, siguiendo las características anteriormente dichas. Por lo tanto, siendo n el nº de iteraciones, la cadena resultante en una configuración final será b^((n-1)*2)c^((n-1)*4). Sumando los naturales que se generan en cada iteración, cada iteración generará ((n-1)*2) + ((n-1)*4) es decir, los múltiplos de 6 más λ

M = { λ, 6, 12, 18, 24…} ∀n≥1

2. **Dado el siguiente sistema P, establezca cuándo el sistema calcula como salida "s" y cuándo calcula como salida "n" (considere la región número 3 como la de salida).**



Observamos que inicialmente tenemos tres posibilidades para la generación de una nueva configuración:

- k=n
- k>n
- k<n

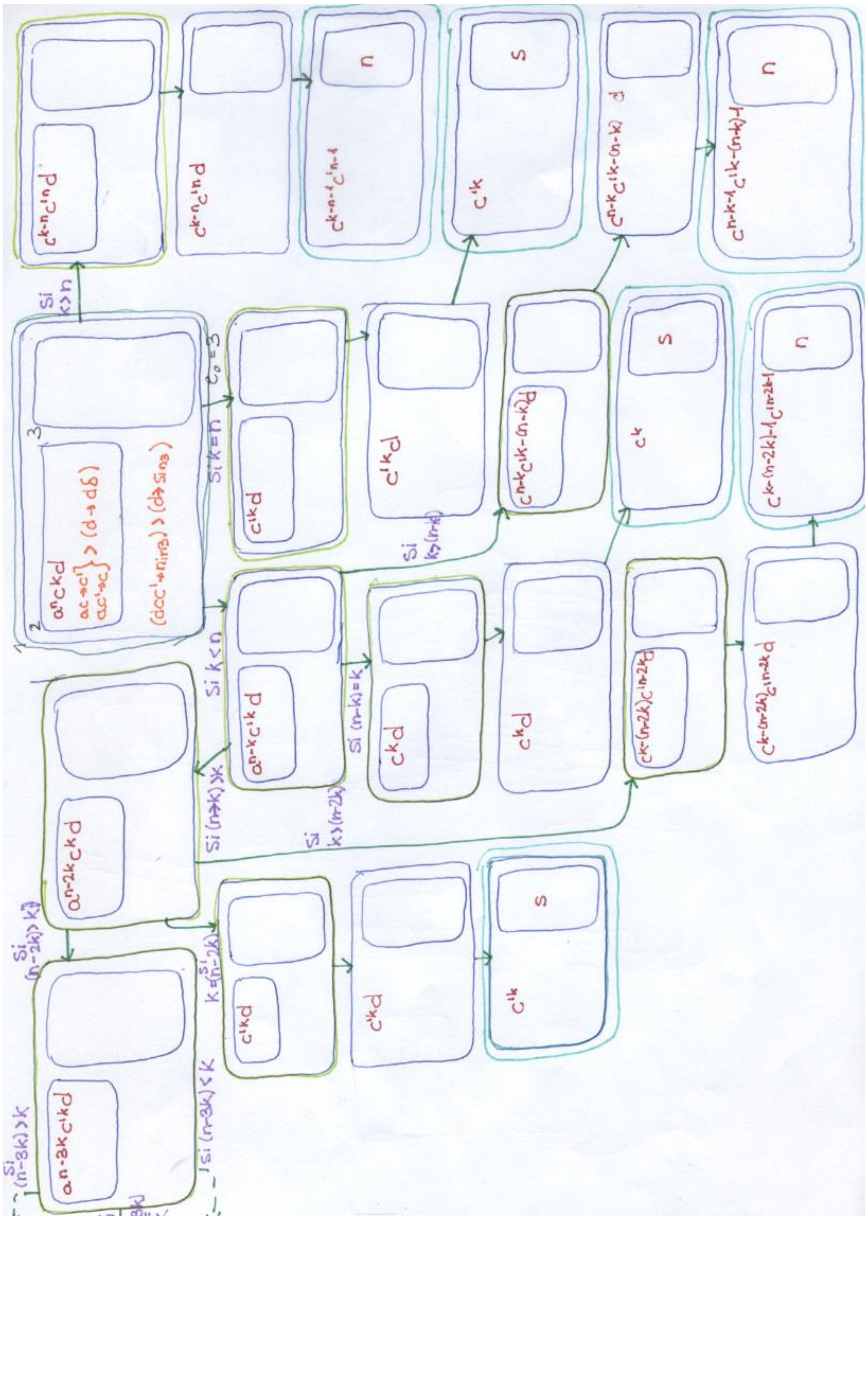
Siendo el primer caso (k=n), llegaremos a una configuración donde la cadena será c'^kd. A partir de esta última cadena, solo podremos aplicar la menos prioritaria (ya que no hay ninguna a en la cadena), que romperá la membrana, dejando la cadena en la membrana 1, donde solo podemos aplicar la regla menos prioritaria, la cual c'^k se quedará en la membrana 1 y d pasará a ser "**s**" en la membrana 3, siendo el resultado final.

En el segundo caso (k>n), llegamos a una configuración donde la cadena de la membrana 2 es $c^{k-n}c'^n$d. A partir de esta última cadena, solo podremos aplicar la menos prioritaria (ya que no hay ninguna a en la cadena), que romperá la membrana, dejando la cadena en la membrana 1, donde podremos aplicar la reglas más prioritaria, la cual dejará en la membrana 1 la cadena $c^{k-n-1}c'^{n-1}$ y en la membrana resultado pasará a estar una "**n**", siendo el final.

En el tercer caso (k<n), generará una configuración donde la cadena de la misma membrana sea $a^{n-k}c'^k$d. Aquí podrá volver a ejecutar las reglas más prioritarias, y así será hasta que desaparezcan las a de la cadena. Es decir, a partir de una configuración que hayamos llegado por k<n, podrán generarse las tres

posibilidades anteriormente nombradas, pasando a ser n=n-k y k manteniendo su valor.

NOTA:
Cabe destacar que, en el último caso, de seguir el camino k<n, se irán alternando c y c’ con tamaño k.
A partir de una configuración con c’, si la posibilidad fuera k=c, en la membrana 2 de la configuración final de esa rama acabaría con c, y viceversa.
Igualmente, a partir de una configuración con c’ y siendo la posibilidad k>n, en la membrana 2 de la configuración final sería $c'^{k-n-1}c^{n-1}$ y en el otro caso, sería $c^{k-n-1}c'^{n-1}$



3. **Diseñe un módulo Mathematica que, dado como entrada un valor entero n, proporcione como salida la configuración del sistema P del ejercicio 1(b) después de aplicar n transiciones.**

Hemos modificado la estructura original que se nos da en las diapositivas para representar el sistema P, resultando en la estructura alternativa que describiremos a continuación.

En la primera posición de la lista se muestra, i, la membrana donde se dará el resultado. En las siguientes posiciones tendremos tantas listas como membranas tenga el sistema. Dentro de cualquiera de estas listas, en la primera posición, tendremos la cadena que se encuentra dentro de la membrana, en la segunda posición habrá una lista con las transiciones de la membrana. Cada transición será una lista en la que la primera posición aparecerá el antecedente, en la segunda posición el consecuente, en la tercera posición, un booleano que indique si la transición acaba rompiendo la membrana en la que se encuentra (True) o no (False), y la cuarta y última posición será un número, el cual, estará a 0 si ninguna transición de esa membrana tiene prioridad, 1 si es prioridad sobre el resto de transiciones de la membrana, 2 si es segunda prioridad y así sucesivamente. En la última y tercera posición de la membrana, tendremos el número de membrana en la que estamos.

Introducimos una lista (insert = {b, b, c, c, c, c}), la cual será el consecuente de la única regla que modifica la cadena (no confundir con borrar) y guardamos en Reglas las reglas de la membrana con su cadena. Comprobamos que el número de iteraciones no sea 0, de ser así, devolveríamos el sistema sin ninguna modificación. Tras esta comprobación, pasaríamos a un bucle con fin en el número final de iteraciones pasadas como argumento. Guardamos en membranaIni y membranaRep las membranas en la que estamos operando y la membrana que modificaremos en el caso de la regla oportuna, respectivamente. Generamos un número aleatorio entre 1 y 2, señalando la regla que vamos a usar de las dos que tenemos dentro de la lista Reglas.
En el caso que el antecedente de la regla sea igual que la cadena, significa que estamos en la regla la cual envía la cadena fuera del sistema, por lo que borramos la cadena de la membrana y guardamos lo que tengamos en aux en la cadena de la membrana receptora, que en este caso es la 2. Fuera cual fuera la iteración, acabaríamos aquí la ejecución ya que nos es posible más configuraciones a partir de la actual.
En caso contrario, significa que aplicaremos la siguiente regla. Entonces guardaremos “insert” en “aux” y la guardaremos dentro de la cadena de membranaRep. En el caso de que hubiéramos llegado a la última iteración indicada devolvería el sistema indicando que no es nodo final y que tiene dos configuraciones más posibles a partir del mismo, calculables con más iteraciones. En caso contrario, volveríamos al comienzo del bucle.

```
SistemaP[niter_] :=
  Module[{Reglas, ReglaAUsar, NumeroReglaAUsar, iterFin, membranaRep, cadena, i, membranaIni, iter, sistemaFin,
  |módulo
    aux, insert, j, sistemaP},
   sistemaP = { 2, {{a, a}, { {{a}, {a, 1, b, 2, c, 2, c, 2}, False, 0}, {{a, a}, {a, 0, a, 0}, False, 0} }, 1},   {{}, {}, 2} };
                                                                 |falso                               |falso
   iterFin = niter;
   iter = 1;
   aux = {};
   insert = {b, b, c, c, c, c};
   Reglas = sistemaP[[2]][[2]];
   If[iterFin == 0,
   |si
    Print["Devolvemos el sistema inicial"];
    |escribe
    Return[sistemaP];
    |retorna
   ];
   While[iter <= iterFin,
   |mientras
    membranaIni = sistemaP[[2]];
    membranaRep = sistemaP[[3]];
    NumeroReglaAUsar = RandomInteger[{1, 2}];
                       |entero aleatorio
    (*Print[NumeroReglaAUsar];*)
    cadena = sistemaP[[2]][[1]];
    ReglaAUsar = Reglas[[NumeroReglaAUsar]];

    If[Length[ReglaAUsar[[1]]] == Length[cadena],
    |si |longitud                 |longitud
     membranaIni[[1]] = {};
     membranaRep[[1]] = aux;
     membranaRep[[1]] = Sort[membranaRep[[1]]];
                        |ordena
     sistemaFin = {2, membranaIni, membranaRep};
     Print["Estamos en la iteracion"];
     |escribe
     Print[iter];
     |escribe
     Print["Y es final"];
     |escribe
     Return[sistemaFin];
     |retorna
    ];
    If[Length[ReglaAUsar[[1]]] < Length [cadena],
    |si |longitud                 |longitud
     For[j = 1, j ≤ Length[insert], j++,
     |para cada   |longitud
       aux = Append[aux, insert[[j]]];
             |añade
      ];
    ];
    membranaRep[[1]] = aux;
    membranaRep[[1]] = Sort[membranaRep[[1]]];
    sistemaFin = {2, membranaIni, membranaRep};

    If[iter == iterFin,
    |si
     Print["No es nodo final"];
     |escribe
     Return[sistemaFin];
     |retorna
    ];
    ;
    iter++;
   ];
  ];
```