Universidad de La Laguna

E. T. S. DE INGENIERÍA INFORMÁTICA
Departamento de Estadística, I.O. y Computación
Teoría de Autómatas y Lenguajes Formales

PRACTICA 8: Minimización del número de estados de un DFA

8.1. Introducción

Consideremos el DFA que se presenta en la Figura 8.1. En ese autómata, ciertos

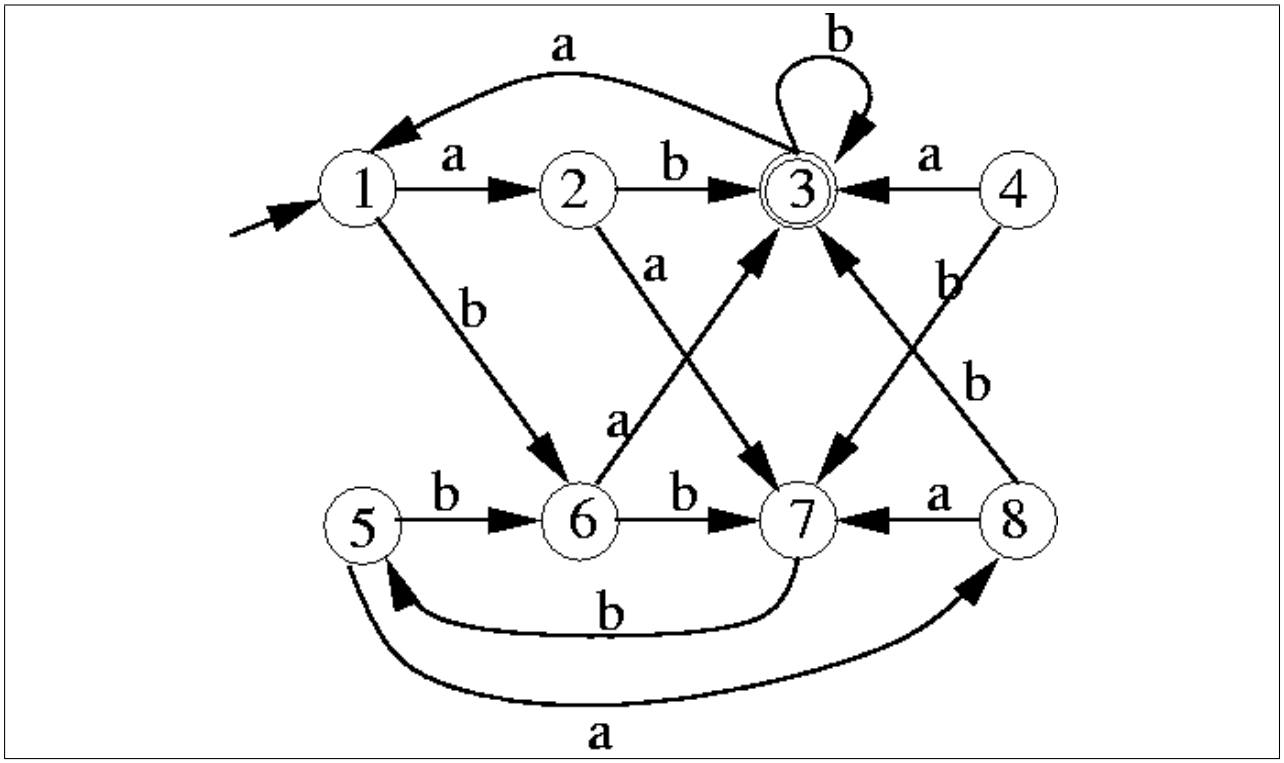

Figura 8.1: Un DFA con estados redundantes

estados se comportan del mismo modo para toda cadena de entrada. Por ejemplo, si el autómata está en el estado 2 u 8 y se lee cualquier cadena de entrada no vacía (el alfabeto es $\Sigma = \{a, b\}$) se llega siempre al mismo estado final. De algún modo, la presencia de los estados 2 y 8 es redundante. En ocasiones conviene obtener un DFA para un lenguaje que sea el DFA mínimo para ese lenguaje, en el sentido de que tenga un número mínimo de estados. El algoritmo que plantearemos lo que hará será eliminar todos los estados redundantes como los estados 2 y 8 de este ejemplo.

Sea un $DFA = (Q, \Sigma, q_0, F, \delta)$. Dos estados $p, q \in Q$ se dicen distinguibles si para alguna cadena $w \in \Sigma*$ ocurre que $\delta(p, w) \in F$ y $\delta(p, x) \notin F$, o viceversa.

Si todos los pares de estados del DFA son distinguibles, el autómata no tiene estados redundantes y por lo tanto será un DFA mínimo. Si el autómata contiene uno o más conjuntos de estados no distinguibles, se puede eliminar la redundancia reemplazando cada conjunto de estados por un único estado.

Los pares de estados no distinguibles se pueden hallar mediante una tabla en la que cada fila y columna corresponden a un estado. Se comienza por marcar como distinguibles las posiciones de la tabla correspondientes a un estado final y a otro no final. A partir de esa situación, para cada par de estados p y q que no se sabe si son distinguibles se consideran los estados $p_a = \delta(p, a)$ y $q_a = \delta(q, a)$ para todos los símbolos $a \in \Sigma$ del alfabeto. Si p_a y q_a son distinguibles por medio de la cadena $x \in \Sigma*$, entonces p y q son distinguibles por medio de la cadena ax.

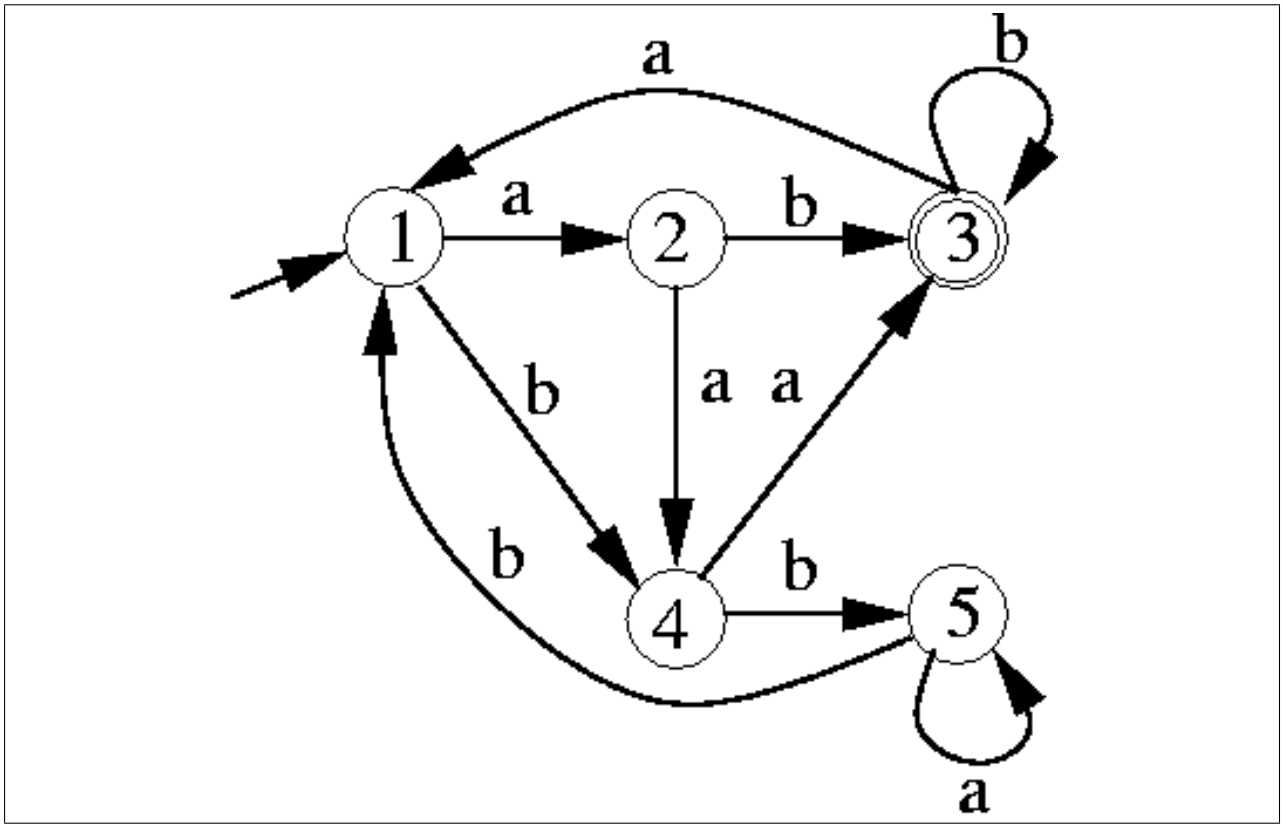

Figura 8.2: Un DFA mínimo equivalente al de la figura 8.1

De este modo, si la celda correspondiente a p_a y q_a está marcada para algún símbolo a, se marcará también la celda para p y q. Si $\forall a \in \Sigma$ la celda correspondiente a p_a y q_a no está marcada, introduciremos (p, q) en una lista asociada con (p_a, q_a) para todo símbolo a. Si luego se obtiene que p_a y q_a son distinguibles, se marcarán también p y q. Dado que las celdas simétricas con respecto a la diagonal de la tabla corresponden a los mismos pares de estados, podemos utilizar sólo la parte inferior de la tabla. De

hecho, las posiciones correspondientes a la diagonal de la tabla corresponde a pares de estados no distinguibles.

2	X						
3	X	X					
4	X	X	X				
5		X	X	X			
6	X	X	X		X		
7	X	X	X	X	X	X	
8	X		X	X	X	X	X
	1	2	3	4	5	6	7

Cuadro 8.1: La tabla correspondiente al DFA de la figura 8.1

La tabla 8.1 presenta los estados no distinguibles para el autómata de la figura 8.1. Los conjuntos de estados no distinguibles son para este ejemplo $\{1, 5\}$, $\{2, 8\}$, $\{4, 6\}$, $\{3\}$ y $\{7\}$. El DFA mínimo se obtiene sustituyendo cada conjunto de estados por un único estado. Así pues, el DFA mínimo para este ejemplo es el de la figura 8.2.

Veamos a continuación un algoritmo de minimización del número de estados de un DFA más apto para su codificación en un programa.

El algoritmo funciona hallando todos los conjuntos de estados que pueden ser diferenciados por una cadena de entrada. Cada grupo de estados no distinguibles se fusiona entonces en un único estado. El algoritmo trabajará manteniendo y refinando una partición del conjunto de estados. Cada grupo de estados dentro de la partición estará formado por estados que aún no han sido distinguidos, y todos los pares de estados elegidos de entre grupos diferentes han sido considerados distinguibles por alguna entrada.

Al comienzo del algoritmo la partición tiene dos conjuntos: los estados de aceptación y los de no aceptación. El paso básico del algoritmo consiste en tomar un conjunto de estados $C = \{e_1, e_2, \ldots, e_k\}$ y un símbolo del alfabeto, a y comprobar qué transiciones producen los estados del conjunto C con esa entrada. Si estas transiciones son hacia estados de dos o más conjuntos distintos de la partición considerada, entonces el conjunto C ha de dividirse para que las transiciones desde los subconjuntos de C queden todas restringidas en un único conjunto de la partición que se está considerando. Supongamos por ejemplo que los estados e_1 y e_2 transitan a los estados t_1 y t_2 con un determinado símbolo de entrada, a y que t_1 y t_2 están en diferentes conjuntos de la partición. En este caso ha de dividirse el conjunto C al menos en dos subconjuntos para conseguir que el estado e_1 esté en un subconjunto y e_2 en otro.

Este proceso de división de la partición considerada ha de repetirse hasta que no sea necesario dividir ningún conjunto. Un pseudocódigo de este algoritmo, estructurado en varios procedimientos se presenta en la figura 8.6.

Consideremos un ejemplo práctico de aplicación del algoritmo, para el DFA de la figura 8.3. La primera partición Π contiene dos conjuntos de estados: $\Pi = \{\{4\}, \{0, 1, 2, 3\}\}$ el primer conjunto de la partición contiene al estado de aceptación (4) y el otro al resto de estados. En el bucle de la línea 12 del algoritmo, primero se considera el conjunto $\{4\}$

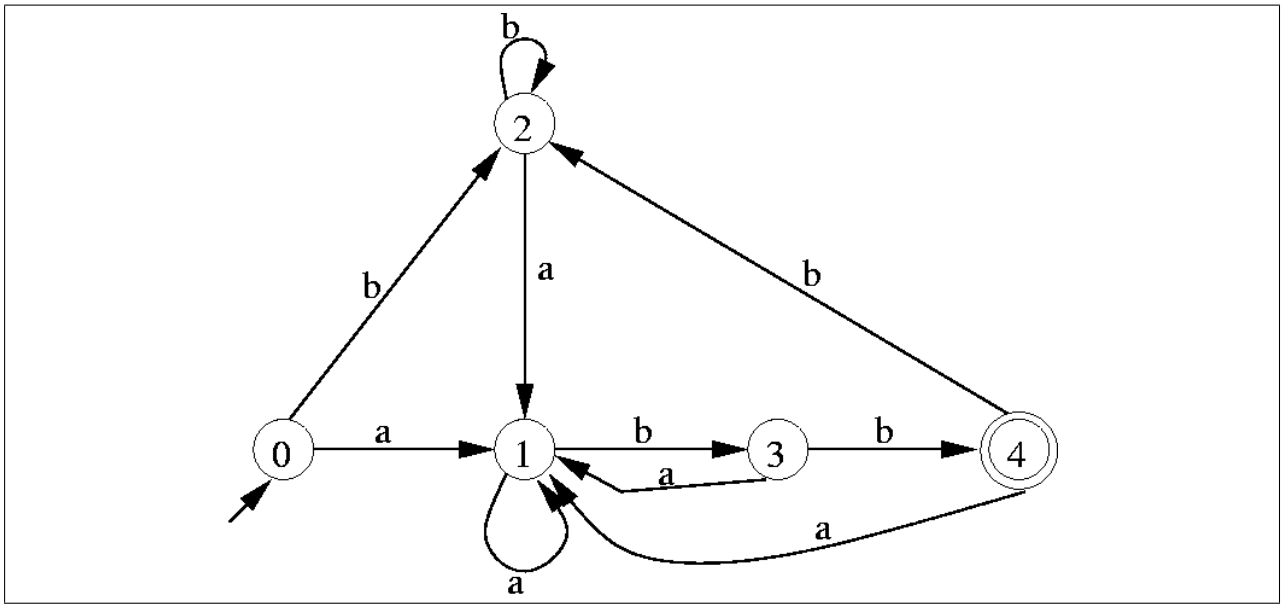

Figura 8.3: Otro ejemplo de DFA con estados redundantes

de la partición, pero este conjunto no puede ser particionado (usando el procedimiento *d*escomp()) porque sólo contiene un estado. Así pues se pasa al siguiente conjunto de la partición: $\{0, 1, 2, 3\}$. En el bucle de la línea 19 que recorre los símbolos del alfabeto, primero se considera el símbolo a. Con esta entrada, todos los estados del conjunto tienen una transición al estado 1, por lo cual podrían permanecer todos en el mismo conjunto. Sin embargo, para la entrada b los estados 0, 1 y 2 transitan a estados del conjunto $\{0, 1, 2, 3\}$ mientras que el estado 3 transita a 4 con esa entrada. Así pues habrá que dividir el conjunto y la partición pasará a ser ahora $\Pi = \{\{4\}, \{3\}, \{0, 1, 2\}\}$.

Estado	Símbolo de entrada	
	a	**b**
0	1	0
1	1	3
3	1	4
4	1	0

Cuadro 8.2: Tabla de transiciones del DFA mínimo

En la siguiente pasada del algoritmo se trata de refinar el conjunto $\{0, 1, 2\}$ (el resto de conjuntos tiene un único estado y no se va a dividir más). Con el símbolo a no se producen divisiones, pero para la entrada b los estados 0 y 2 transitan a 2 mientras que 1 transita a 3. Así pues hay que dividir el conjunto $\{0, 1, 2\}$ dando lugar a una nueva partición $\Pi = \{\{4\}, \{3\}, \{1\}, \{0, 2\}\}$. Al intentar dividir el conjunto $\{0, 2\}$ se observa que no es necesario puesto que con entrada a, 0 y 2 transitan a 1 y con entrada b ambos transitan al estado 2. El algoritmo finaliza teniendo como partición final la anterior: $\Pi =$

$\{\{4\},\{3\},\{1\},\{0,2\}\}$. Esto significa que el DFA mínimo tendrá 4 estados. La tabla de transiciones del DFA mínimo se muestra en la tabla 8.2.

Para comprender el funcionamiento del procedimiento descomp(G, P), considérese la figura 8.4. En ella se muestra cómo una partición inicial $\Pi_0 = \{G\}$ se va particionando ante los diferentes símbolos del alfabeto (en este caso son a y b) para dar lugar a sucesivas particiones: $\Pi_1 = \{G1, G2, G3\}$; $\Pi_2 = \{\{G11, G12\}\{G21\}, \{G31, G32, G33\}\} \ldots$

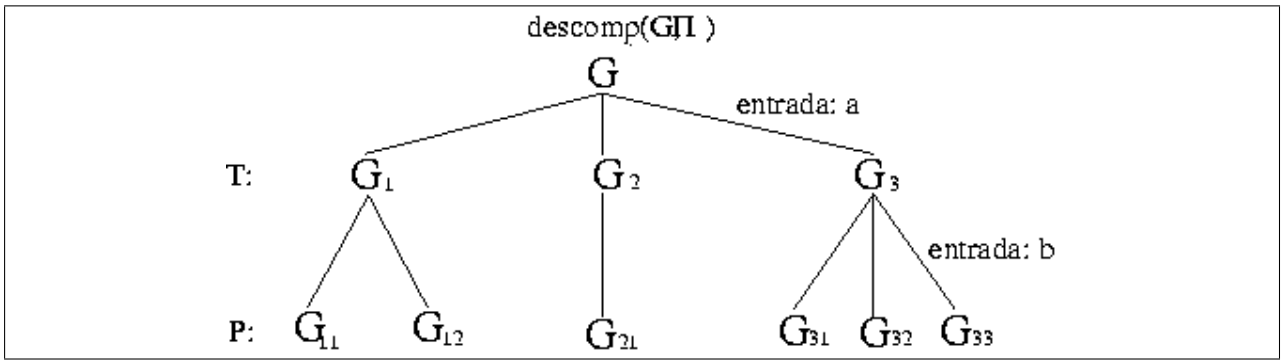

Figura 8.4: La función descomp()

La construcción de este árbol de particiones la lleva a cabo el algoritmo anterior. En este árbol, la última capa construida está representada en el código de la figura 8.6 por la partición T; mientras que la nueva capa que se construye está representada en el código por la variable P. Obsérvese que dentro del bucle que recorre los símbolos del alfabeto se comienza siempre con $P := \emptyset$ (como corresponde al hecho de que la capa está inicialmente vacía). Es el procedimiento $Part(G, a, \Pi)$ quien se encarga de generar los subárboles. (En nuestro ejemplo genera G_{11} y G_{12} a partir de G_1.

En la Figura 8.5 se muestra el proceso de creación de particiones para el DFA de la figura 8.3 que se ha utilizado como ejemplo. Téngase en cuenta que una partición sólo se podrá refinar si tiene más de un estado; por ello, en la figura 8.5 sólo se dividen aquellos conjuntos de estados que contienen más de un estado.

8.2. Práctica 8: Minimización del número de estados de un DFA

La práctica consistirá en la realización de un programa que lea de un fichero la definición de un DFA, le aplique el algoritmo de búsqueda de estados distinguidos para minimizar su número de estados y escriba el nuevo DFA en otro fichero.

La estructura de los ficheros de definición de DFA"s de entrada y salida será igual que la que se ha utilizado en prácticas anteriores.

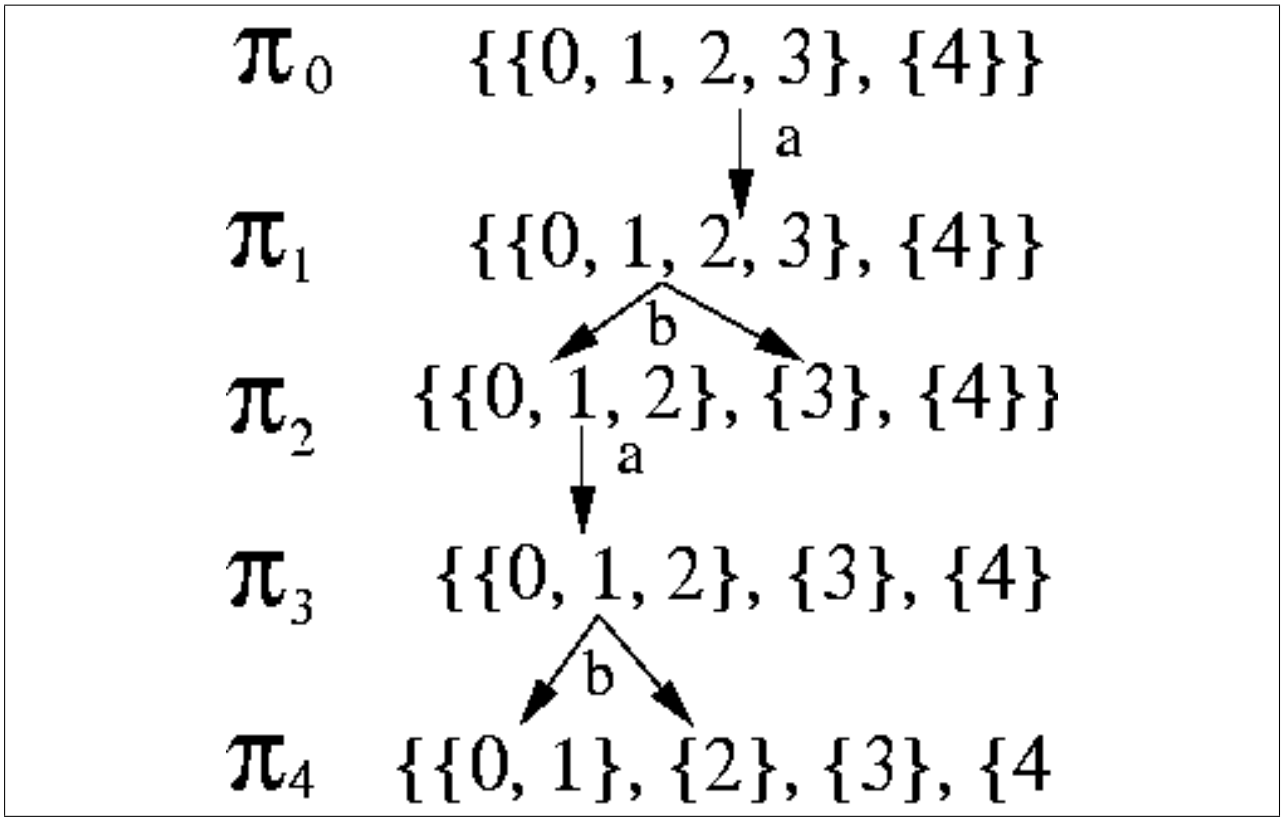

Figura 8.5: Particiones que se generan al minimizar el DFA de la figura 8.3

8.3. Notas

Al igual que en prácticas anteriores se supondrá que el alfabeto Σ estará constituido exclusivamente por todos aquellos caracteres con los que el DFA produzca alguna transición de estados.

En el servidor ftp del Centro se encuentran los programas

`ftp://ftp.csi.ull.es/pub/asignas/AUTOMALF/p03_minDFA/minimo.exe`
`ftp://ftp.csi.ull.es/pub/asignas/AUTOMALF/p03_minDFA/minimo`

(versiones DOS y unix respectivamente) que tienen el comportamiento que se pide. Se encuentran también en ese directorio una serie de ficheros

`ftp://ftp.csi.ull.es/pub/asignas/AUTOMALF/p03_minDFA/test?.dfa`

de definición de autómatas que se pueden utilizar como ejemplos de entrada para la aplicación que se desarrollará. El fichero

`ftp://ftp.csi.ull.es/pub/asignas/AUTOMALF/p03_minDFA/indice.txt`

contiene una descripción de los lenguajes que reconocen cada uno de los autómatas de ejemplo.

Con los ficheros de definición de autómatas se debe verificar la corrección del resultado de la práctica, minimizando el DFA y comprobando que el autómata minimizado reconoce el mismo lenguaje que el inicial. Para esta comprobación se utilizará el simulador de DFA que se ha desarrollado en una práctica anterior.

Para realizar la práctica se puede utilizar cualquiera de las librerías de funciones que se han desarrollado en prácticas anteriores. Una buena guía para la preparación de esta práctica es la referencia [Aho90].

```
1  begin
2  Π := {Q − F, F}                /* partición inicial */
3  repeat
4     Π_vieja := Π
5     Π := crear_nueva_particion(Π_vieja)
6  until Π = Π_vieja
7  construir_DFA(Π)               /* Construye el autómata */
8  end.

9  crear_nueva_particion(Π_vieja)
10 begin
11 Π := ∅
12 for all G ∈ Π_vieja do         /* Para todos los conjuntos de Π_vieja */
13    Π := Π∪ descomp(G, Π_vieja)
14 return(Π)
15 end

16 descomp(G, Π)                  /* Descomponer el cjto. de estados G */
17 begin
18 T := {G}                       /* T es la ”capa” actual del árbol */
19 for all a ∈ Σ do
20    begin
21    P := ∅                      /* P Es la nueva capa que se crea */
22    for all G ∈ T do
23       begin
24       T' := part(G, a, Π)
25       P := P ∪ T'
26       end
27    T := P
28    end
29 return(T)
30 end

31 part(G, a, Π)                  /* Partir el cjto. de estados G para la */
32 begin                          /* entrada a tomando Π como partición inicial */
33 T := ∅
34 for all H ∈ Π do
35    if ∃q ∈ G/δ(q, a) ∈ H then
36       T := T ∪{{q ∈ G/δ(q, a) ∈ H}}
37 return(T)
38 end
```

Figura 8.6: El algoritmo de minimización del número de estados de un DFA mediante búsqueda de estados distinguidos