Universidad de La Laguna

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

PRACTICA 7: La construcción de subconjuntos: conversión de un NFA en un DFA

7.1. Introducción

El algoritmo de construcción de subconjuntos ya fue empleado en una práctica anterior para simular un NFA. Ahora vamos a utilizarlo para convertir un NFA en un DFA equivalente (que reconozca el mismo lenguaje).

Mientras que en la tabla de transiciones de un NFA cada entrada es un conjunto de estados, en la de un DFA cada casilla contiene un único estado. La idea que se utiliza para convertir un NFA en un DFA consiste en asociar cada estado del DFA con un conjunto de estados del NFA. El DFA utilizará un estado para representar todos los posibles estados en los que se puede encontrar el NFA después de leer cada símbolo de la entrada. Dicho de otro modo, después de leer la entrada $x_1, x_2, \ldots, x_m$ el DFA se encontrará en un estado que representa al subconjunto M de los estados del NFA alcanzables desde el estado de arranque del NFA siguiendo algún camino etiquetado con los símbolos de la cadena de entrada. El número de estados del DFA puede crecer de forma exponencial con respecto al número de estados del NFA equivalente, pero en la práctica este caso peor no es frecuente.

El algoritmo que utilizaremos se basa en construir la tabla de transiciones $tranDFA$ del DFA. Cada estado del DFA será un conjunto de estados del NFA y se construye $tranDFA$ de modo que el DFA simulará .en paralelo"todos los posibles movimientos que el NFA pueda realizar con una determinada frase de entrada. El algoritmo utiliza las operaciones $\epsilon - clausura()$ y $move()$ que se introdujeron en la práctica anterior.

Antes de leer el primer símbolo de la cadena de entrada, el NFA puede estar en cualquiera de los estados de $\epsilon - clausura(s_0)$ siendo s_0 el estado de arranque del NFA. Si suponemos que los estados alcanzables partiendo de s_0 con una determinada secuencia de símbolos de entrada son los del conjunto T y suponemos también que a es el siguiente símbolo de la entrada, al leer a el NFA puede cambiar su estado a cualquiera de los estados del conjunto $move(T, a)$. Si además tenemos en cuenta que puede haber ϵ-transiciones, el NFA puede cambiar a cualquiera de los estados de $\epsilon - clausura(move(T, a))$.

Construiremos $estadosD$, el conjunto de estados del DFA y $tranDFA$, su tabla de transiciones del siguiente modo: Cada estado del DFA corresponde a un conjunto de estados del NFA en los que éste podría estar después de leer alguna secuencia de símbolos de entrada, incluidas todas las posibles e-transiciones anteriores o posteriores a la lectura de símbolos. El estado inicial del DFA será $\epsilon - clausura(s_0)$. Añadiremos

estados y transiciones al DFA utilizando el algoritmo (construcción de subconjuntos) que se presenta en la 7.1.

```
1  q_0 := ε − clausura(s_0);
2  desmarcar(q_0);
3  estadosDFA := {q_0};
4  while (∃T ∈ estadosDFA && not marcado(T)) do
5     begin
6     marcar(T);
7     for (cada símbolo de entrada, a) do
8        begin
9        H := ε − clausura(move(T, a));
10       if H ∉ estadosDFA then
11          begin
12          desmarcar(H);
13          estadosDFA := estadosDFA ∪ H
14          end
15       tranDFA[T, a] := H;
16       end
17    end
```

Figura 7.1: El algoritmo de construcción de subconjuntos

Un estado del DFA será de aceptación si se corresponde con un conjunto de estados del NFA que contenga al menos un estado de aceptación del NFA. Veamos un ejemplo del funcionamiento del algoritmo de la construcción de subconjuntos. Consideremos el NFA de la 7.2 que acepta el lenguaje representado por la expresión regular $(a|b) * abb$.

El estado inicial del DFA equivalente es $\epsilon - clausura(0) = A = \{0, 1, 2, 4, 7\}$ puesto que los estados de A son los alcanzables con ϵ-transiciones partiendo del estado 0. Siguiendo el algoritmo de la construcción de subconjuntos, se marca el conjunto A y se calcula
$\epsilon - clausura(move(A, a)) = \epsilon - clausura(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B$
y la correspondiente entrada en la tabla de transiciones del DFA será $tranDFA[A, a] = B$. Siguiendo con el bucle de la línea 7 del algoritmo, utilizamos ahora el siguiente símbolo del alfabeto, b para calcular
$\epsilon - clausura(move(A, b)) = \epsilon - clausura(\{5\}) = \{1, 2, 4, 5, 6, 7\} = C$
y tendremos también que $tranDFA[A, b] = C$

Se continúa el proceso con el conjunto B que está sin marcar, y que se marca en este momento:
$\epsilon - clausura(move(B, a)) = \epsilon - clausura(\{3, 8\}) = B$
$tranDFA[B, a] = B$

Utilizamos el siguiente símbolo del alfabeto con el conjunto B:
$\epsilon - clausura(move(B, b)) = \epsilon - clausura(\{5, 9\}) = \{1, 2, 4, 5, 6, 7, 9\} = D\ tranDFA[B, b] = D$
y se pasa al siguiente conjunto sin marcar, C, marcándolo:

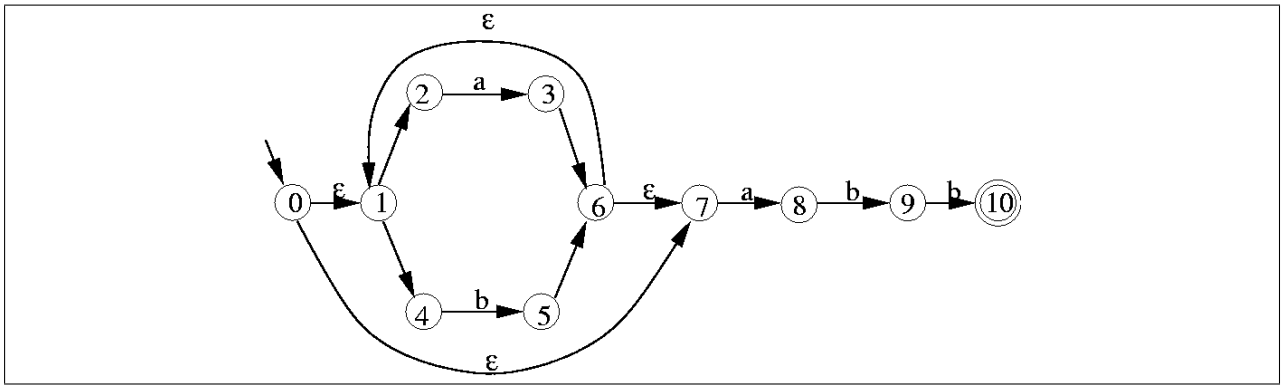

Figura 7.2: Ejemplo de NFA

$\epsilon - clausura(move(C, a)) = \epsilon - clausura(\{3, 8\}) = B$
$tranDFA[C, a] = B$
$\epsilon - clausura(move(C, b)) = \epsilon - clausura(\{5\}) = C$
$tranDFA[C, b] = C$

Ahora usamos el estado D que no está marcado:
$\epsilon - clausura(move(D, a)) = \epsilon - clausura(\{3, 8\}) = B$
$tranDFA[D, a] = B$
$\epsilon - clausura(move(D, b)) = \epsilon - clausura(\{5, 10\}) = \{1, 2, 4, 5, 6, 7, 10\} = E$
$tranDFA[D, b] = E$

Como el estado E no está marcado, se calcula
$\epsilon - clausura(move(E, a)) = \epsilon - clausura(\{3, 8\}) = B$
$tranDFA[E, a] = B$
$\epsilon - clausura(move(E, b)) = \epsilon - clausura(\{5\}) = C$
$tranDFA[E, b] = C$

y aquí se detiene el algoritmo porque no se cumple la condición del bucle de la línea 4, ya que todos los estados del DFA están ahora marcados.

Los conjuntos A, B, C, D, y E son los estados del DFA que aparece en la Figura 7.5, aunque en esa figura aparecen etiquetados respectivamente como 0, 1, 2, 3 y 4.

7.2. Práctica: Conversión de un NFA en un DFA

La práctica consistirá en la realización de un programa que lea desde un fichero las especificaciones de un NFA y utilizando el algoritmo de la construcción de subconjuntos transforme dicho autómata a en un DFA equivalente, y por último escriba las especificaciones de éste en un fichero.

7.2.1. Estructura de los ficheros de definición de NFAs

Estos ficheros tendrán la extensión `.nfa`, y contendrán la misma información que los ficheros que se utilizaron en una práctica anterior para representar NFAs, es decir:

- Línea 1: Número total de estados del NFA.
- Línea 2: Estado de arranque del NFA.

A continuación viene una línea para cada estado, conteniendo los siguientes números, separados entre sí por espacios en blanco.

- Número identificador del estado.
- Un valor 1 ó 0 que indica si el estado es de aceptación (1) o no (0).

A continuación para cada una de las transiciones y separado por espacios:

- Símbolo del alfabeto necesario para que se produzca la transición.
- Estado de destino de la transición.

También, al igual que en prácticas anteriores, una ϵ-transición se representará mediante el carácter ~ (código ASCII 126).

En los ficheros que representan NFAs, cualquier línea que comience con los caracteres // será ignorada, puesto que representará un comentario. Los comentarios se utilizarán para anotar en los ficheros características significativas de los autómatas representados.

```
// Fichero que representa a un NFA
11
 0
 0 0 ~ 1 ~ 7
 6 0 ~ 7 ~ 1
 7 0 a 8
 1 0 ~ 4
 2 0 a 3
 4 0 b 5
 3 0 ~ 6
 5 0 ~ 6
10 1
 8 0 b 9
 9 0 b 10
```

Figura 7.3: Fichero con una posible representación del NFA de la Figura 7.2

7.2.2. Fichero de salida

El fichero de salida tendrá extensión .dfa y una estructura igual a la que se utilizó en una práctica anterior para definir un DFA. Al tratarse de un DFA ya no ha de haber ϵ-transiciones. Recordemos que la estructura del fichero de salida será por tanto:

- Línea 1: El número total de estados del DFA.
- Línea 2: Estado de arranque del DFA.

A continuación viene una línea para cada estado, conteniendo los siguientes números, separados entre sí por espacios en blanco.

- Número identificador del estado.
- Un 1 si se trata de un estado de aceptación o un 0 si es un estado de rechazo.

A continuación (en la misma línea) para cada una de las transiciones y separados por espacios:

- Símbolo de entrada necesario para que se produzca la transición.
- Estado destino de la transición.

Se utilizarán los simuladores de DFA y NFA, que se han diseñado en las prácticas anteriores para la comprobación de resultados. Para el NFA de la figura 7.2, el fichero de salida generado por el programa será el que se muestra en la Figura 7.4, que define el DFA de la Figura 7.5.

```
// Los ficheros de representación de DFAs también admiten comentarios
5
0
0 0 2 a 1 b 2
1 0 2 a 1 b 3
2 0 2 a 1 b 2
3 0 2 a 1 b 4
4 1 2 a 1 b 2
```

Figura 7.4: DFA de salida

7.3. Notas

Para la implementación del algoritmo de Construcción de Subconjuntos se precisa conocer el alfabeto de entrada del NFA. Supondremos que este alfabeto estará constituido exclusivamente por todos aquellos símbolos con los que el NFA produzca alguna transición de estados (excluido ϵ).

Para representar el alfabeto en su forma definitiva se recomienda utilizar un vector de caracteres. Sin embargo, conforme se lee el fichero de definición del NFA, quizás convenga representar el alfabeto mediante un conjunto.

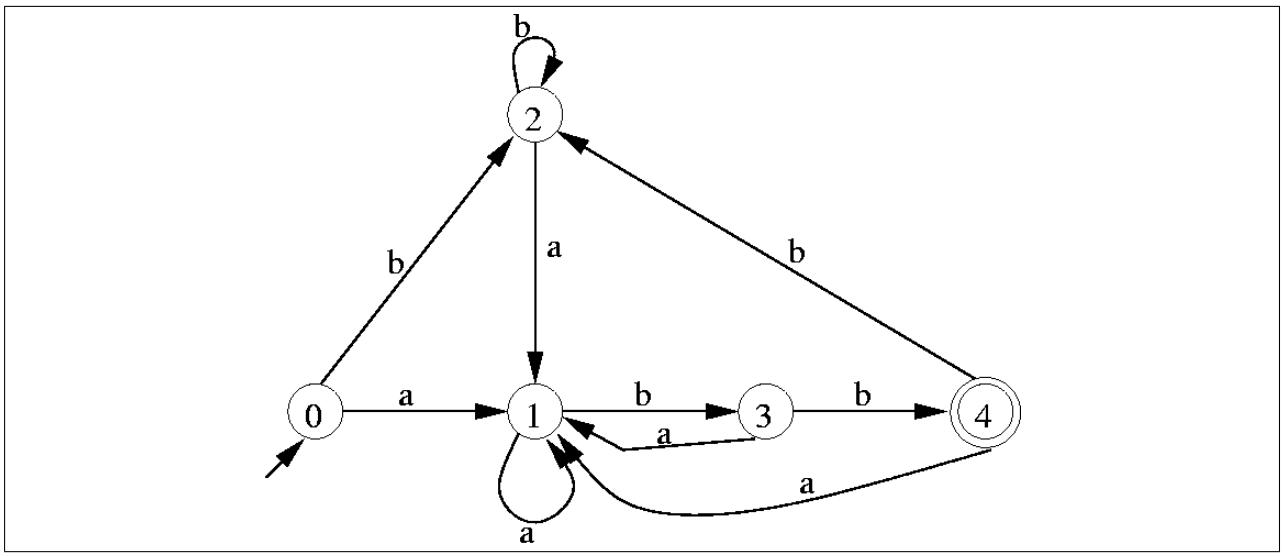

Figura 7.5: El DFA de la Figura 7.4

```
ftp://ftp.csi.ull.es/pub/asignas/AUTOMALF/p03_NFA2DFA/unix/thompson
ftp://ftp.csi.ull.es/pub/asignas/AUTOMALF/p03_NFA2DFA/dos/thompson.exe

ftp://ftp.csi.ull.es/pub/asignas/AUTOMALF/p03_NFA2DFA/unix/nfa2dfa
ftp://ftp.csi.ull.es/pub/asignas/AUTOMALF/p03_NFA2DFA/dos/nfa2dfa.exe
```

Recomendamos las obras [Aho90] y [Kel95] como material de trabajo para la preparación de esta práctica.

7.4. Bibliografía

[Aho90] Aho, A., Sethi, R., y Ullman, J. Compiladores. Principios, Técnicas y Herramientas. Addison-Wesley Iberoamericana, 1990. ISBN: 0-201-62903-8 [Kel95] Kelley, D. Teoría de Autómatas y Lenguajes Formales. Prentice-Hall, 1995. ISBN: 0-13-518705-2