Universidad de La Laguna

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

PRACTICA 6: Autómatas Finitos No Deterministas

6.1. Requisito de codificación

Utilización de los espacios:
Una correcta utilización de los espacios en el código fuente contribuye claramente a mejorar la legibilidad del mismo. Recomendamos utilizar los espacios del modo más similar posible a como éstos se utilizan en la escritura convencional. Las reglas que debemos seguir son:

- Colocar siempre un espacio a cada lado de un operador binario.
- No poner espacios después de paréntesis abierto ni antes de paréntesis cerrado.
- Poner siempre un espacio después de las comas.

6.2. Introducción

Podemos introducir los autómatas finitos no deterministas (AFN) o en inglés Nondeterministic Finite Automaton, NFA como una modificación de los autómatas finitos deterministas en los que se permitirá cero, una o varias transiciones desde un estado con un símbolo del alfabeto de entrada. La Figura 6.1 presenta un primer ejemplo de NFA. Observamos que el estado 0, ante el símbolo a puede transitar sobre sí mismo o bien al estado 1. De forma análoga el estado 1 puede transitar con entrada a al estado 2 o bien al estado 3.

Se define formalmente un Autómata Finito No Determinista como una quíntupla $(\Sigma, Q, q_0, F, \delta)$ donde todas las componentes tienen el mismo significado que en un DFA, salvo la función de transición, δ que en este caso es una relación

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$

$(q, \sigma) \rightarrow \{q_1, q_2, \ldots, q_N\}$

es decir, asigna a un par (q, σ) un elemento de $\wp(Q)$ (partes de Q, el conjunto de todos los subconjuntos de Q). $q \in Q, \sigma \in (\Sigma \cup \epsilon)$.

$\delta(q, \sigma)$ es el conjunto de estados p tales que hay una transición etiquetada σ que va desde q hasta p y donde σ es ϵ o bien un símbolo del alfabeto Σ.

En el NFA de la Figura 6.1:

$\delta(2, a) = \emptyset$

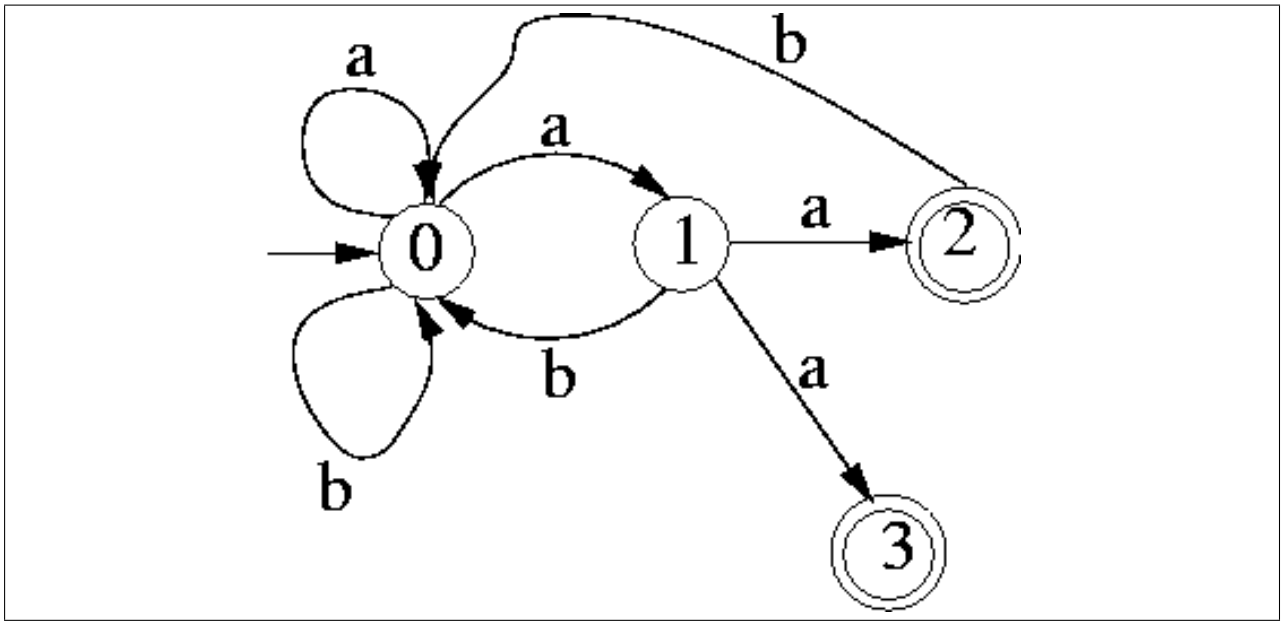

Figura 6.1: Ejemplo de NFA

$\delta(3, a) = \delta(3, b) = \emptyset$
$\delta(0, a) = \{0, 1\}$
La Tabla 6.1 muestra la función de transición del NFA de la Figura 6.1.

δ	a	b
0	$\{0, 1\}$	$\{0\}$
1	$\{2, 3\}$	$\{0\}$
2	$\emptyset$	$\{0\}$
3	$\emptyset$	$\emptyset$

Cuadro 6.1: Función de transición δ para el NFA de la Figura 6.1

El tipo de autómata que estamos definiendo se conoce como NFA con transiciones vacías (ϵ-transiciones) porque el autómata puede transitar sin necesidad de entrada (con una entrada vacía, ϵ). El no determinismo viene dado porque el autómata puede transitar con una misma entrada hacia diferentes estados. En particular, puede transitar al conjunto vacío (lo cual significaría que el autómata es incapaz de transitar).

De forma análoga a como se hizo para el caso de un DFA, se puede extender la función de transición δ de un NFA para que acepte como entrada un estado y una cadena de símbolos del alfabeto:
$\hat{\delta} : Q \times \Sigma^* \rightarrow \wp(Q)$
$(q, \sigma) \rightarrow \{q1, q2, ..., qN\}$
$\hat{\delta}(q, w)$ serán todos los estados p tales que se puede ir desde el estado q hasta p recorriendo un camino etiquetado con los símbolos de $w \in \Sigma^*$, incluyendo potencialmente arcos etiquetados ϵ.

Para construir $\hat{\delta}$ será importante calcular el conjunto de estados alcanzables a partir de un determinado estado q utilizando solamente ϵ-transiciones. Este cálculo es equivalente al de calcular qué vértices son alcanzables partiendo de uno dado en un grafo dirigido. El vértice origen será el vértice correspondiente al estado q en el diagrama de transición de estados y el grafo dirigido será el formado por los arcos etiquetados ϵ. Utilizaremos la notación $\epsilon - clausura(q)$ para denotar al conjunto de todos los vértices p tales que hay un camino desde q hasta p etiquetado ϵ.

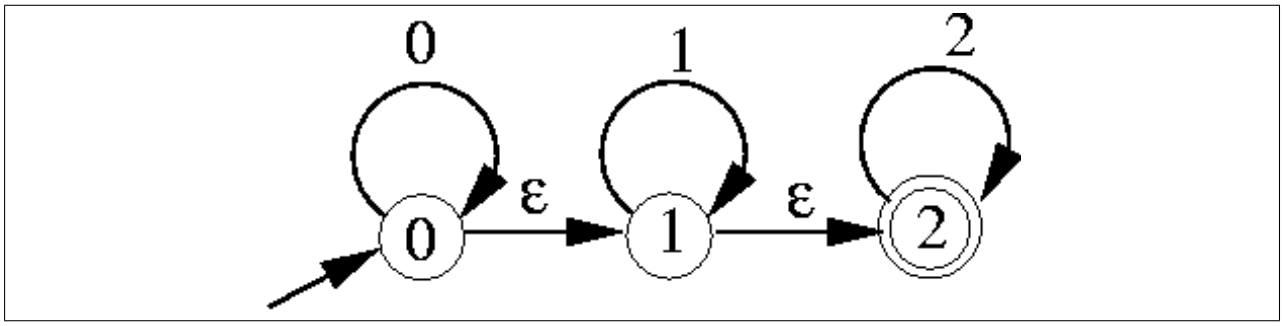

Figura 6.2: Un NFA que reconoce el lenguaje denotado por `0*1*2*`

En la Figura 6.2, $\epsilon - clausura(0) = \{0, 1, 2\}$ puesto que hay un camino desde el estado 0 hasta él mismo etiquetado con ϵ (no hay arcos en este camino), el camino $0 - 1$ etiquetado ϵ indica que el estado 1 está en la $\epsilon - clausura(0)$ y el camino $0 - 1 - 2$ etiquetado ϵ indica que $2 \in \epsilon - clausura(0)$.

De forma natural, se define $\epsilon - clausura(R)$ siendo R un conjunto de estados como $\bigcup_{q \in R} \epsilon - clausura(q)$. Con estos elementos podemos definir la extensión de la función de transición:

- $\hat{\delta}(q, \epsilon) = \epsilon - clausura(q)$
- Para $w \in \Sigma^*$ y $a \in \Sigma$, $\hat{\delta}(q, wa) = \epsilon - clausura(P)$ siendo $P = \{p|$ para algún r en $\hat{\delta}(q, w)$, p está en $\delta(r, a)\}$

También es conveniente extender δ y $\hat{\delta}$ a conjuntos de estados R:
$\delta(R, a) = \bigcup_{q \in R} \delta(q, a)$
$\hat{\delta}(R, w) = \bigcup_{q \in R} \hat{\delta}(q, w)$

Nótese que en el caso de un NFA $\hat{\delta}(q, a)$ no coincide necesariamente con $\delta(q, a)$ puesto que $\hat{\delta}(q, a)$ incluye todos los estados alcanzables a partir de q utilizando caminos etiquetados con a (incluyendo caminos con arcos etiquetados con ϵ) mientras que $\delta(q, a)$ incluye sólo aquellos estados alcanzables desde q a través de arcos etiquetados a.

Se define $L(M)$, el lenguaje aceptado por el NFA $M \equiv (\Sigma, Q, q_0, F, \delta)$ como el conjunto de cadenas $\{w|\hat{\delta}(q_0, w)$ contiene algún estado de F$\}$.

6.3. Simulación de un Autómata Finito No Determinista con ϵ-transiciones

La práctica consistirá en la realización de un programa en C++ que lea desde un fichero las especificaciones de un autómata finito no determinista, a continuación comience a imprimir una traza de los estados por los que transita mediante las entradas en forma de caracteres que se leerán desde otro fichero, y al final indique si llegó a un estado de aceptación o no. Tanto el fichero de definición del NFA como el de entradas para el mismo se leerán en la línea de comandos del programa.

El Listado 6.1 presenta el pseudocódigo de un algoritmo para calcular la $\epsilon - clausura$ de un conjunto de estados T. El cálculo de la $\epsilon - clausura$ se puede interpretar como el cálculo del conjunto de nodos alcanzables partiendo de un nodo dado en un grafo dirigido. En el caso del cálculo que nos ocupa, el conjunto de nodos del grafo son los estados del conjunto T y el grafo serían las aristas etiquetadas con ϵ en el diagrama de transición de estados del NFA. El algoritmo utiliza una pila y un conjunto de estados. El conjunto de estados se inicializa con todos los estados $q \in T$ y al final de la ejecución almacenará todos los estados de la $\epsilon - clausura(T)$.

```
For (cada estado q de T) do
  push(q)
epsilon-clausura(T) := T;
while (not pila vacía) do
  begin
  p := pop();
  for (cada estado u con una arista de p a u etiquetada epsilon) do
    if not (u in epsilon-clausura(T)) then
      begin
      epsilon-clausura(T) := epsilon-clausura(T) + {u}
      push(u);
      end;
  end;
```

Listado 6.1: Pseudocódigo del algoritmo para el cálculo de la $\epsilon - clausura$ de un conjunto de estados, T

```
S := epsilon-clausura({q0});
a := nextchar();
while (a <> eof) do
  begin
  S := epsilon-clausura(move(S, a));
  a := nextchar();
  end;
if (S intersección F <> vacio) then
  return("Cadena aceptada")
else
```

```
return("Cadena rechazada")
```

Listado 6.2: Pseudocódigo para el algoritmo de simulación de un NFA

El listado 6.2 presenta el pseudocódigo para el algoritmo de simulación que se implementará. El algoritmo opera sobre una cadena x de símbolos del alfabeto y realiza la construcción de subconjuntos en tiempo de ejecución. Calcula la transición de un conjunto actual de estados, S a un nuevo conjunto de estados en dos etapas: primero determina $move(S, a)$, todos los estados que pueden alcanzarse desde un estado de S mediante una transición con el símbolo de entrada, a. A continuación calcula la $\epsilon - clausura$ de $move(S, a)$, es decir, todos los estados que pueden alcanzarse desde $move(S, a)$ con cero o más ϵ-transiciones. En el pseudocódigo de la Figura 6.2 la función `nextchar()` se utiliza para leer el siguiente símbolo de la cadena x de entrada. El código que implementa la función $move()$ es muy similar al del cómputo de la $\epsilon - clausura$.

Todos los ficheros que se manejan en la práctica serán ficheros de texto. Los estados del autómata se representarán mediante números enteros sin signo (`unsigned`), y la numeración de los estados corresponderá a los primeros números naturales comenzando con 0.

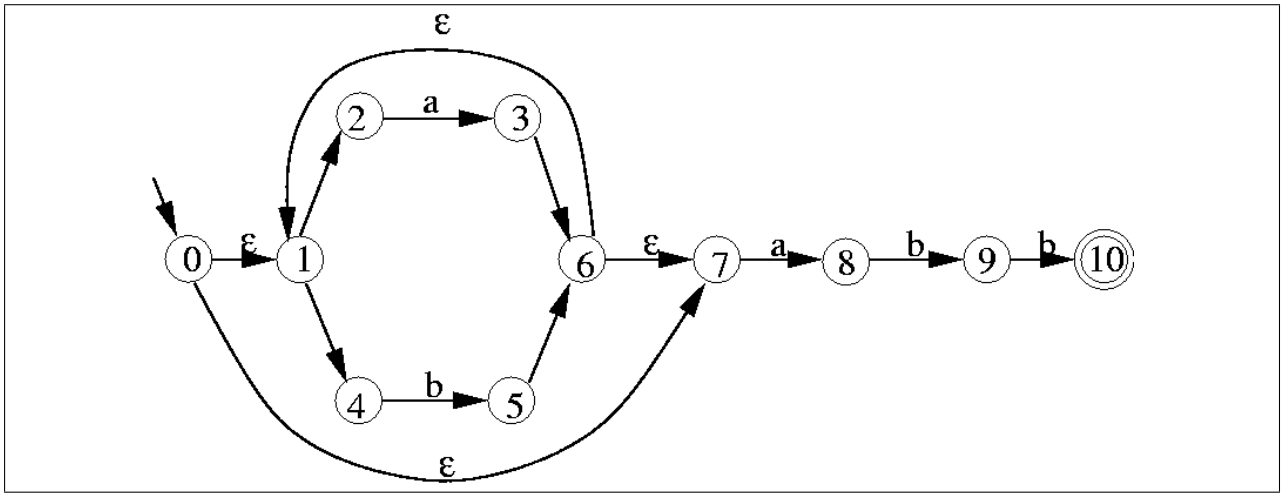

Figura 6.3: Ejemplo de NFA

6.3.1. Estructura de los ficheros de definición de NFA's

```
// Ejemplo de fichero de definición de NFAs
// Número de estados
11
// Estado inicial
0
// Definición de los estados
0 0 ~ 1 ~ 7
1 0 ~ 2 ~ 4
```

```
2 0 a 3
4 0 b 5
5 0 ~ 6
6 0 ~ 7 ~ 1
7 0 a 8
3 0 ~ 6
8 0 b 9
9 0 b 10
10 1
```

Listado 6.3: Fichero que representa al NFA de la Figura 6.3

Estos ficheros tendrán la extensión `.nfa`, tienen la misma estructura que los ficheros de defnición de DFAs y contendrán la siguiente información:

- 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.

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.

El fichero que se presenta en la Figura 6.3 es uno que podría representar al autómata de la Figura 6.3. Nótese que el orden en que se listan las transiciones del autómata es irrelevante y por tanto podría haber varios ficheros distintos que representen el mismo autómata.

6.3.2. Ficheros de entradas

Estos ficheros tendrán la extensión `.in` Se supone que el alfabeto de trabajo está constituido por todos los caracteres alfanuméricos (letras o dígitos). Las frases de entrada al autómata figuran en los ficheros de entrada sin separación entre los caracteres de la misma. Así para el autómata anterior podríamos tener la entrada: `ababbbbabb`

6.3.3. Trazas

```
Entrada: ababbbbabb

{\epsilon. Actuales} -->  Entrada -->  {E. Destino}
{ 0  1  2  4  7 } -->  a -->  { 3  8 }
{ 1  2  3  4  6  7  8 } -->  b -->  { 5  9 }
{ 1  2  4  5  6  7  9 } -->  a -->  { 3  8 }
{ 1  2  3  4  6  7  8 } -->  b -->  { 5  9 }
{ 1  2  4  5  6  7  9 } -->  b -->  { 5  10 }
{ 1  2  4  5  6  7  10 } -->  b -->  { 5 }
{ 1  2  4  5  6  7 } -->  b -->  { 5 }
{ 1  2  4  5  6  7 } -->  a -->  { 3  8 }
{ 1  2  3  4  6  7  8 } -->  b -->  { 5  9 }
{ 1  2  4  5  6  7  9 } -->  b -->  { 5  10 }
Estados finales: { 1  2  4  5  6  7  10 }
Estado de aceptación: 10
Cadena ACEPTADA.
```

Listado 6.4: Evolución del NFA de la Figura 6.3 para la entrada `ababbbbabb`

A medida que el autómata lee entradas deberá imprimirse una lista que contenga los números identificadores de los estados activos actualmente, la entrada leída y la lista de los números identificadores de los nuevos estados a los que se transita, previamente al cálculo de la $\epsilon - clausura$, finalizando todo ello con un mensaje que indique si se ha aceptado o rechazado la cadena de entrada. La Figura 6.4 ilustra la salida del programa para el autómata y entrada anteriores.

6.3.4. Notas

En la sección de descargas del portal de la asignatura tienen un fichero conteniendo varios ficheros de descripción de NFAs con los que se puede verificar la implementación que se realice. Es muy importante que l@s alumn@s prueben el correcto comportamiento del simulador que diseñen usando diferentes autómatas desarrollados por sí mism@s.