Máquinas de estado finito y autómatas

Prof. Enrique Vílchez Quesada

Universidad Nacional de Costa Rica

Introducción

Las máquinas de estado finito constituyen un modelo abstracto para explicar el funcionamiento de una computadora o una máquina con una memoria simple o primitiva. A diferencia de otras formas de representación teóricas, las máquinas de estado finito incluyen el factor de la memoria como una condicionante para establecer las acciones subsiguientes que efectuaría un ordenador. Otros modelos como los circuitos combinatorios, que no serán desarrollados en este texto, establecen relaciones lógicas entre los datos de entrada para producir un conjunto de datos de salida, sin tomar en consideración, los cambios de estado en el sistema. De allí, que se aprecie a las máquinas de estado finito como un modelo teórico más completo.

Máquinas de estado finito

Prof. Enrique Vílchez Quesada

Universidad Nacional de Costa Rica

Máquinas de estado finito

En general una máquina de estado finito es un $6-$tupla formada por una lista de estados como las condiciones que podría tomar la máquina en el tiempo, un conjunto de símbolos de entrada correspondientes a los valores del alfabeto que podrían ser ingresados a la máquina, un conjunto de símbolos de salida representando los datos procesados en el sistema, un estado inicial del cual parte la máquina y dos funciones: una que determina el comportamiento de transición de los estados y la otra, el procesamiento de los símbolos de entrada traducido en una secuencia de símbolos de salida.

Definición 7.1. Máquina de estado finito

Definition (7.1)

Una máquina de estado finito denotada *MEF* es una 6−tupla M, $M = (\sigma, \tau, \delta, \sigma^*, \Delta, \Omega)$ donde:

1. σ se llama conjunto de estados de la *MEF*.
2. τ es el conjunto de símbolos de entrada.
3. δ es el conjunto de símbolos de salida de la máquina de estado finito.
4. σ^* es un estado del conjunto σ llamado estado inicial.

Definición 7.1. Continuación

5. Δ es una función de transición de estados, tal que: $\Delta : \sigma \times \tau \to \sigma$, es decir, la función toma como un elemento del dominio, un par ordenado constituido por un estado y un dato de entrada, retornando como imagen, otro estado. Esta imagen representa el estado del sistema después del procesamiento indicado por el par ordenado $(estado, entrada)$.
6. Ω es una función de salida, $\Omega : \sigma \times \tau \to \delta$, es decir, al igual que la función Δ, un elemento del dominio es un par ordenado $(estado, entrada)$ con la diferencia de devolver como imagen, un símbolo de salida. Este símbolo es el "output" del sistema en ese instante del tiempo.

Para comprender mejor la definición 1 abordaremos algunos ejemplos.

Example (7.1)

Determine si el siguiente arreglo $M = (\sigma, \tau, \delta, \sigma^*, \Delta, \Omega)$ es una *MEF*, con $\sigma = \{\sigma_0, \sigma_1, \sigma_2\}$, $\tau = \{a, b\}$, $\delta = \{0, 1, 2, 3\}$, $\sigma^* = \sigma_0$ y:

	Δ		Ω	
	a	b	a	b
σ_0	σ_1	σ_0	0	2
σ_1	σ_1	σ_2	1	0
σ_2	σ_0	σ_1	3	2

Solución del ejemplo 7.1

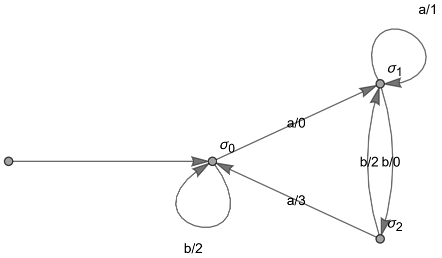
Por la definición 1 M es una máquina de estado finito pues está constituida por las seis componentes requeridas, donde Δ es una función tal que: $\Delta : \sigma \times \tau \to \sigma$ y Ω es otra función con: $\Omega : \sigma \times \tau \to \delta$.

Nota

Las máquinas de estado finito tienen una forma de representación a través de un digrafo, donde los nodos son los estados del conjunto σ. Un lado dirigido que une un estado μ con otro μ^*, se agrega al grafo si existe un par ordenado $(\mu, \mathit{entrada})$ cuya imagen en la función Δ da como resultado μ^*. Las aristas tienen etiquetas asociadas de estructura *entrada / salida* donde la imagen del par $(\mu, \mathit{entrada})$ en Ω devuelve el símbolo *salida* de δ. Además, se añade una fecha que indica cuál es el estado inicial σ^* de la máquina M.

Solución del ejemplo 7.1

A este grafo dirigido se le llama: "diagrama de transición" de M. El lector puede observar que en un sentido estricto, el diagrama de transición no es un digrafo por la flecha que apunta al estado σ^*, pese a ello, en adelante se abusará un poco del concepto desarrollado anteriormente, al asumir que todo diagrama de transición sí cumple con la definición de grafo dirigido. En este ejemplo, el diagrama de transición de la máquina M corresponde a:



(1)

Solución del ejemplo 7.1

El paquete **VilCretas** provee el comando `MaquinaToDiagrama` encargado de generar el diagrama de transición respectivo de una *MEF* de interés, al recibir cada una de sus seis componentes. El siguiente código genera el digrafo mostrado en 1:

In[ ] :=

```
MaquinaToDiagrama[{σ0, σ1, σ2}, {a, b}, {0, 1, 2, 3}, σ0,
{{σ0, a, σ1, 0}, {σ0, b, σ0, 2}, {σ1, a, σ1, 1},
{σ1, b, σ2, 0}, {σ2, a, σ0, 3}, {σ2, b, σ1, 2}}]
```

Solución del ejemplo 7.1

Se aprecia que las funciones Δ y Ω en la instrucción `MaquinaToDiagrama` se pasan juntas mediante una matriz. Cada fila de esta matriz contiene en sus primeras dos entradas un par ordenado $(estado, entrada)$ de $\sigma \times \tau$, la tercera entrada simboliza la imagen de Δ y la cuarta entrada representa la imagen de Ω, en el par ordenado $(estado, entrada)$ correspondiente.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-160.zip

Uso de la sentencia `MaquinaToDiagrama`

Explicación en video

https://youtu.be/L3S0mklctzE

Example (7.2)

Construya el diagrama de transición de la máquina de estado finito $M = (\sigma, \tau, \delta, \sigma^*, \Delta, \Omega)$, con $\sigma = \{\sigma_0, \sigma_1, \sigma_2\}$, $\tau = \{a, b, c\}$, $\delta = \{0, 1\}$, $\sigma^* = \sigma_2$ y:

	Δ			Ω		
	a	b	c	a	b	c
σ_0	σ_0	σ_1	σ_2	0	1	0
σ_1	σ_1	σ_0	σ_0	1	1	1
σ_2	σ_2	σ_1	σ_0	1	0	0

Solución del ejemplo 7.2

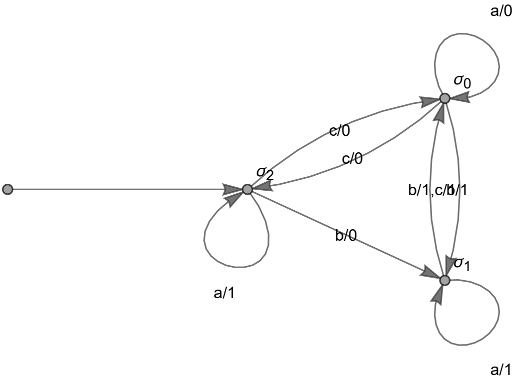
Se resolverá el ejemplo mediante el uso de software. En *Wolfram Mathematica*:

In[ ] :=

MaquinaToDiagrama[{σ_0, σ_1, σ_2}, {a, b, c}, {0, 1}, σ_2, {{σ_0, a, σ_0, 0}, {σ_0, b, σ_1, 1}, {σ_0, c, σ_2, 0}, {σ_1, a, σ_1, 1}, {σ_1, b, σ_0, 1}, {σ_1, c, σ_0, 1}, {σ_2, a, σ_2, 1}, {σ_2, b, σ_1, 0}, {σ_2, c, σ_0, 0}}]

Solución del ejemplo 7.2

Out[ ] =



Nota

Como se visualiza el comando `MaquinaToDiagrama` tiene la característica de crear una única arista por cada conjunto de lados múltiples y etiquetar con todos los símbolos *entrada / salida* contenidos, separados por una coma.

Solución del ejemplo 7.2

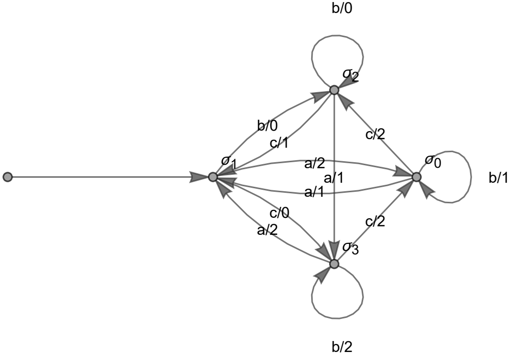
En este ejemplo, esto ocurre en el estado σ_1 con los símbolos de entrada b y c. Allí, no se dibujaron dos aristas de σ_1 a σ_0, se generó un solo lado dirigido con las etiquetas $b/1$ y $c/1$ separadas por una coma. La ventaja de ello, consiste en evitar una sobrecarga visual de aristas en el diagrama y la desventaja, reside en la sobreposición de textos producida algunas veces (como en este caso), aspecto no concialiable al emplear la sentencia `MaquinaToDiagrama`.

Descargue un archivo

`https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-161.zip`

Example (7.3)

Dado el diagrama de transición mostrado a continuación, de una máquina de estado finito M, $M = (\sigma, \tau, \delta, \sigma^*, \Delta, \Omega)$, encuentre explícitamente cada una de sus seis componentes.



(2)

Solución del ejemplo 7.3

Del análisis de esta representación se deduce que:

- $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$.
- $\tau = \{a, b, c\}$.
- $\delta = \{0, 1, 2\}$.
- $\sigma^* = \sigma_1$.
-

	Δ			Ω		
	a	b	c	a	b	c
σ_0	σ_1	σ_0	σ_2	1	1	2
σ_1	σ_0	σ_2	σ_3	2	0	0
σ_2	σ_3	σ_2	σ_1	1	0	1
σ_3	σ_1	σ_3	σ_0	2	2	2

.

- Como ya se ha señalado, la idea principal de una máquina de estado finito es representar el funcionamiento de una computadora con una memoria primitiva, es decir, el comportamiento de un sistema simple para procesar un conjunto de datos de entrada. Este concepto se relaciona con el procesamiento de hileras o "strings" del alfabeto τ en la máquina. La siguiente definición lo introduce.

Definición 7.2. Hilera de salida

Definition (7.2)

Sea $M = (\sigma, \tau, \delta, \sigma^*, \Delta, \Omega)$ una máquina de estado finito y $\alpha = x_1 x_2 \dots x_n$ una hilera de símbolos de entrada. El "string" de salida de α es una hilera β de caracteres del conjunto δ, $\beta = y_1 y_2 \dots y_n$ para los cuales existe un conjunto de estados σ_0, σ_1, ..., σ_n de σ, tales que:

$$
\begin{array}{l}
\sigma^* = \sigma_0 \\
\Delta(\sigma_0, x_1) = \sigma_1 \text{ y } \Omega(\sigma_0, x_1) = y_1 \\
\Delta(\sigma_1, x_2) = \sigma_2 \text{ y } \Omega(\sigma_1, x_2) = y_2 \\
\vdots \\
\Delta(\sigma_{n-1}, x_n) = \sigma_n \text{ y } \Omega(\sigma_{n-1}, x_n) = y_n
\end{array}
$$

Comentario sobre la definición 5

La definición 5 propone que una máquina de estado finito trata un conjunto de datos α del conjunto τ, usando la función de transición de estados para conocer el siguiente estado de la memoria del sistema y la función de salida encargada de proporcionar el "output" que formará parte de la hilera β de símbolos del conjunto δ. Como se verá, esta tarea es más fácil de ejecutar utilizando el diagrama de transición de la máquina. Cualquier máquina de estado finito tiene como objetivo procesar hileras de símbolos de entrada, generando como respuesta de ello una hilera de símbolos de salida. Las máquinas compartidas en los ejercicios mostrados con anterioridad son *MEF* genéricas pues la salida de un procesamiento de símbolos de entrada no tiene ningún significado en particular. En algunas ocasiones, este procesamiento puede tener un objetivo específico, como por ejemplo, realizar una operación aritmética, o bien, el copiado de un conjunto de caracteres. A este respecto, más adelante, se mostrará al lector una máquina de estado finito sumadora de números binarios.

Comentario sobre la definición 5

Otro aspecto importante de aclarar reside en el hecho de que en una *MEF* siempre será posible procesar de forma única cualquier hilera de símbolos de entrada. En los diagramas de transición mostrados en los ejemplos 2, 3 y 4 se infiere una propiedad muy importante: en cada uno de los estados en el digrafo que representa a la *MEF*, siempre existe una arista saliente para cada uno de los símbolos de entrada. Esto significa que si $\tau = \{a, b, c\}$ de cada estado en el diagrama de transición tendrán que salir tres aristas, una para el símbolo a, otra para b y otra para c (se invita al alumno a corroborar esta característica en los diagramas de los ejemplos 2, 3 y 4). La condición provoca que toda hilera de símbolos de τ recibida por la *MEF* genere una hilera β de símbolos de δ y como de cada estado solo sale una flecha por cada símbolo de entrada, se deriva una hilera β única.

Consideremos algunos ejemplos.

Example (7.4)

Sea la máquina de estado finito del ejemplo 4, procese el "string" de símbolos de entrada $\alpha = aabbacccaabbcba$.

Solución del ejemplo 7.4

Si se desea procesar la hilera $\alpha = aabbacccaabbcba$, el diagrama de transición de la máquina constituye una herramienta esencial para producir la hilera de salida β de interés. Al tomar el primer símbolo de entrada de α, el caracter a, el diagrama compartido en la página 18, señala que iniciando en el estado σ_1, el símbolo a produce 2 como salida y un estado siguiente σ_0. Esto se puede representar simbólicamente así:

$$\sigma_1 \xrightarrow{a/2} \sigma_0$$

Solución del ejemplo 7.4

Ahora, visualizando nuevamente el digrafo, en σ_0 si entra otra a (el segundo caracter de α) la salida es 1 con un estado siguiente σ_1 :

$$\sigma_1 \overset{a/2}{\to} \sigma_0 \overset{a/1}{\to} \sigma_1$$

Luego, el diagrama indica que si en σ_1 se tiene el caracter b (tercer elemento de α) se genera el estado σ_2 y el símbolo de salida 0 :

$$\sigma_1 \overset{a/2}{\to} \sigma_0 \overset{a/1}{\to} \sigma_1 \overset{b/0}{\to} \sigma_2$$

Solución del ejemplo 7.4

Posteriormente, si en σ_2 se recibe b (el cuarto caracter de α) se obtiene la salida 0 y el mismo estado σ_2 :

$$\sigma_1 \overset{a/2}{\longrightarrow} \sigma_0 \overset{a/1}{\longrightarrow} \sigma_1 \overset{b/0}{\longrightarrow} \sigma_2 \overset{b/0}{\longrightarrow} \sigma_2$$

El procedimiento continuaría así, realizando la lectura de los caracteres subsiguientes de α en el diagrama de transición de la máquina, lo cual produce el recorrido:

$$\begin{array}{l} \sigma_1 \overset{a/2}{\longrightarrow} \sigma_0 \overset{a/1}{\longrightarrow} \sigma_1 \overset{b/0}{\longrightarrow} \sigma_2 \overset{b/0}{\longrightarrow} \sigma_2 \overset{a/1}{\longrightarrow} \sigma_3 \overset{c/2}{\longrightarrow} \sigma_0 \overset{c/2}{\longrightarrow} \sigma_2 \overset{c/1}{\longrightarrow} \\ \sigma_1 \overset{a/2}{\longrightarrow} \sigma_0 \overset{a/1}{\longrightarrow} \sigma_1 \overset{b/0}{\longrightarrow} \sigma_2 \overset{b/0}{\longrightarrow} \sigma_2 \overset{c/1}{\longrightarrow} \sigma_1 \overset{b/0}{\longrightarrow} \sigma_2 \overset{a/1}{\longrightarrow} \sigma_3 \end{array}$$

Solución del ejemplo 7.4

Se concluye entonces que:

$$\beta = 210012212100101$$

Al ser la máquina de estado finito genérica, la hilera β no tiene ningún significado en particular.
La librería **VilCretas** contiene la instrucción `StringSalida` que automatiza la construcción de una hilera β de salida, al tener un "string" α de símbolos de τ. El comando presenta la opción `trace -> True` en caso de querer conocer el recorrido sobre cada uno de los estados en la *MEF*, al crear la hilera β.

Solución del ejemplo 7.4

En este ejercicio, la solución en *Wolfram* se detalla a continuación:

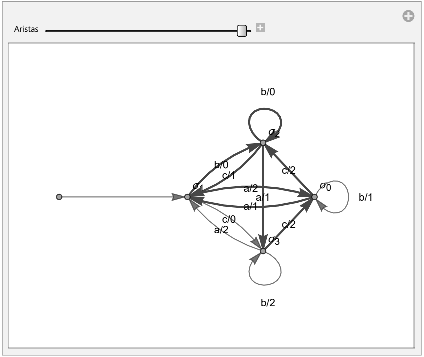
In[ ] :=

```
α = {a, a, b, b, a, c, c, c, a, a, b, b, c, b, a};
StringSalida[{σ0, σ1, σ2, σ3}, {a, b, c}, {0, 1, 2}, σ1,
{{σ0, a, σ1, 1}, {σ0, b, σ0, 1}, {σ0, c, σ2, 2},
{σ1, a, σ0, 2}, {σ1, b, σ2, 0}, {σ1, c, σ3, 0},
{σ2, a, σ3, 1}, {σ2, b, σ2, 0}, {σ2, c, σ1, 1},
{σ3, a, σ1, 2}, {σ3, b, σ3, 2}, {σ3, c, σ0, 2}}, α]
StringSalida[{σ0, σ1, σ2, σ3}, {a, b, c}, {0, 1, 2}, σ1,
{{σ0, a, σ1, 1}, {σ0, b, σ0, 1}, {σ0, c, σ2, 2},
{σ1, a, σ0, 2}, {σ1, b, σ2, 0}, {σ1, c, σ3, 0},
{σ2, a, σ3, 1}, {σ2, b, σ2, 0}, {σ2, c, σ1, 1},
{σ3, a, σ1, 2}, {σ3, b, σ3, 2}, {σ3, c, σ0, 2}}, α,
trace -> True]
```

Solución del ejemplo 7.4

Out[ ] =

$\{2, 1, 0, 0, 1, 2, 2, 1, 2, 1, 0, 0, 1, 0, 1\}$
$\{\{2, 1, 0, 0, 1, 2, 2, 1, 2, 1, 0, 0, 1, 0, 1\}, \{\sigma_1, \sigma_0, \sigma_1, \sigma_2, \sigma_2, \sigma_3, \sigma_0, \sigma_2, \sigma_1, \sigma_0, \sigma_1, \sigma_2, \sigma_2, \sigma_1, \sigma_2, \sigma_3\}\}$



Solución del ejemplo 7.4

Como se aprecia, la opción `trace -> True` construye una animación de recorrido paso a paso sobre cada uno de los estados al devolver la hilera β.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-162.zip

Empleo de la sentencia `StringSalida`.

Explicación en video

https://youtu.be/9ue-iXfE4Bc

Example (7.5)

Con ayuda de software, determine los "strings" de salida β obtenidos al emplear la *MEF* y la hilera α del ejemplo 6, asumiendo que $\sigma^* = \sigma_0$, $\sigma^* = \sigma_2$ y $\sigma^* = \sigma_3$.

Solución del ejemplo 7.5

Los comandos `Table` y `StringSalida` brindan la posibilidad de resolver el ejercicio:

In[ ] :=

α = {a, a, b, b, a, c, c, c, a, a, b, b, c, b, a};
Table[StringSalida[{σ_0, σ_1, σ_2, σ_3}, {a, b, c}, {0, 1, 2},
σ_j, {{σ_0, a, σ_1, 1}, {σ_0, b, σ_0, 1}, {σ_0, c, σ_2, 2},
{σ_1, a, σ_0, 2}, {σ_1, b, σ_2, 0}, {σ_1, c, σ_3, 0},
{σ_2, a, σ_3, 1}, {σ_2, b, σ_2, 0}, {σ_2, c, σ_1, 1},
{σ_3, a, σ_1, 2}, {σ_3, b, σ_3, 2}, {σ_3, c, σ_0, 2}}, α],
{j, {0, 2, 3}}]

Solución del ejemplo 7.5

Out[ ] $=$

$\{\{1, 2, 1, 1, 1, 0, 2, 2, 1, 2, 0, 0, 1, 0, 1\},$
$\{1, 2, 0, 0, 1, 2, 2, 1, 2, 1, 0, 0, 1, 0, 1\},$
$\{2, 2, 1, 1, 1, 0, 2, 2, 1, 2, 0, 0, 1, 0, 1\}\}$

En resumen:

σ^*	β
σ_0	121110221200101
σ_2	120012212100101
σ_3	221110221200101

Descargue un archivo

`https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-163.zip`

Comentario sobre los ejemplos 6 y 7

Las hileras β encontradas en los ejemplos 6 y 7, muestran que a partir de estados iniciales distintos teniendo la misma máquina y la misma hilera de símbolos de entrada, la salida o el comportamiento del sistema puede cambiar. Lo anterior significa, que las hileras β no serán necesariamente iguales, de hecho, en la *MEF* de los ejercicios citados, estos "strings" de salida son todos distintos. Al asumir estados iniciales diferentes, se presupone que la condición de la memoria en la máquina es desigual, por lo que es natural que esto pueda provocar un comportamiento de salida dispar al procesar información de entrada. Por esta razón, al inicio de este capítulo se señaló que las máquinas de estado finito conforman un modelo que toma en cuenta el factor de la memoria para representar el funcionamiento de un ordenador.

Example (7.6)

Sea la máquina de estado finito del ejemplo 3 halle la hilera de salida β para $\alpha = ccbbaabaccbbaacbcabc$.

Solución del ejemplo 7.6

Al utilizar el diagrama de transición mostrado en la página 15 y el "string" α del enunciado, se obtiene:

$$\sigma_2 \overset{c/0}{\to} \sigma_0 \overset{c/0}{\to} \sigma_2 \overset{b/0}{\to} \sigma_1 \overset{b/1}{\to} \sigma_0 \overset{a/0}{\to} \sigma_0 \overset{a/0}{\to} \sigma_0 \overset{b/1}{\to} \sigma_1 \overset{a/1}{\to} \sigma_1 \overset{c/1}{\to} \sigma_0 \overset{c/0}{\to}$$
$$\sigma_2 \overset{b/0}{\to} \sigma_1 \overset{b/1}{\to} \sigma_0 \overset{a/0}{\to} \sigma_0 \overset{a/0}{\to} \sigma_0 \overset{c/0}{\to} \sigma_2 \overset{b/0}{\to} \sigma_1 \overset{c/1}{\to} \sigma_0 \overset{a/0}{\to} \sigma_0 \overset{b/1}{\to} \sigma_1 \overset{c/1}{\to} \sigma_0$$

En consecuencia, la hilera β de salida corresponde a:

$$\beta = 000100111001000001011$$

Solución del ejemplo 7.6

De manera complementaria, el estudiante podría corroborar la respuesta anterior usando el software *Mathematica*:

In[ ] :=

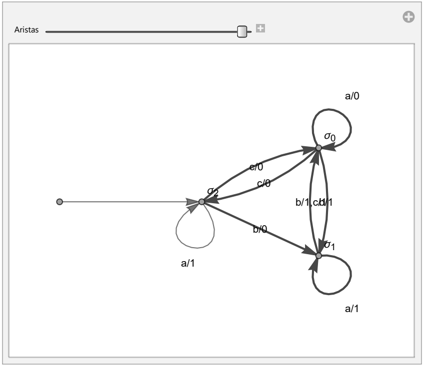
```
α = {c, c, b, b, a, a, b, a, c, c, b, b, a, a, c, b, c, a, b, c};
StringSalida[{σ0, σ1, σ2}, {a, b, c}, {0, 1}, σ2,
{{σ0, a, σ0, 0}, {σ0, b, σ1, 1}, {σ0, c, σ2, 0},
{σ1, a, σ1, 1}, {σ1, b, σ0, 1}, {σ1, c, σ0, 1},
{σ2, a, σ2, 1}, {σ2, b, σ1, 0}, {σ2, c, σ0, 0}}, α]
StringSalida[{σ0, σ1, σ2}, {a, b, c}, {0, 1}, σ2,
{{σ0, a, σ0, 0}, {σ0, b, σ1, 1}, {σ0, c, σ2, 0},
{σ1, a, σ1, 1}, {σ1, b, σ0, 1}, {σ1, c, σ0, 1},
{σ2, a, σ2, 1}, {σ2, b, σ1, 0}, {σ2, c, σ0, 0}}, α,
trace -> True]
```

Solución del ejemplo 7.6

Out[ ] =

$\{0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1\}$

$\{\{0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1\}, \{\sigma_2, \sigma_0, \sigma_2, \sigma_1, \sigma_0, \sigma_0, \sigma_0, \sigma_1, \sigma_1, \sigma_0, \sigma_2, \sigma_1, \sigma_0, \sigma_0, \sigma_0, \sigma_2, \sigma_1, \sigma_0, \sigma_0, \sigma_1, \sigma_0\}\}$



Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-164.zip

El ejercicio que prosigue es interesante dado que presenta una máquina de estado finito no genérica. En este caso, la función de la *MEF* compartida consiste en realizar la suma de dos números binarios.

Example (7.7)

Construya una máquina de estado finito que resuelva al procesar una hilera de símbolos de entrada, la suma entre dos números binarios. Para ello, se debe tomar en cuenta la tabla dada a continuación, donde se presenta las sumas básicas entre dos *bits*:

$$\begin{array}{|l|} \hline 0+0=0 \\ \hline 1+0=1 \\ \hline 0+1=1 \\ \hline 1+1=10 \\ \hline \end{array} \tag{3}$$

Solución del ejemplo 7.7

El algoritmo de la suma binaria, se fundamenta en realizar la operación dígito por dígito, de derecha a izquierda y llevar cantidades si es necesario. Por ejemplo:

$$\begin{array}{r} 0110 \\ +\ 0111 \\ \hline \end{array}$$

El primer paso, reside en sumar los dos primeros *bits* de ambos números binarios, $0+1=1$ de acuerdo con la tabla 3. No se lleva ningún dígito y se escribe el resultado:

$$\begin{array}{r} 0110 \\ +\ 0111 \\ \hline 1 \end{array}$$

Solución del ejemplo 7.7

Luego, se adicionan los segundos dígitos, $1+1=10$, se deja el 0 como el nuevo *bit* de la suma y se lleva el uno a la tercera columna, es decir:

$$\begin{array}{rr} & 01^{1}10 \\ + & 0111 \\ \hline & 01 \end{array}$$

Al llevar el 1, en 1^{1}, se suma $1+1=10$ por la tabla 3 y se resuelve aparte $10+1$, donde:

$$\begin{array}{rr} & 10 \\ + & 1 \\ \hline & 11 \end{array}$$

Solución del ejemplo 7.7

Volviendo a la tercera columna de la suma original, se deja como tercer *bit* de la adición a 1 y se lleva 1 :

$$\begin{array}{r} 0^{1}110 \\ + \quad 0111 \\ \hline 101 \end{array}$$

Finalmente, operando $0+1=1$ y $1+0=1$, se obtiene:

$$\begin{array}{r} 0110 \\ +\ 0111 \\ \hline 1101 \end{array}$$

Solución del ejemplo 7.7

En este ejemplo, se desea crear una máquina de estado finito que realice este tipo de operación, al procesar una hilera de símbolos de entrada. Construiremos el diagrama de transición de la máquina de estado finito requerida. La *MEF* debe ser capaz de recibir dos *bits* y efectuar su suma, por esta razón, el conjunto de símbolos de entrada se toma como todas las posibles parejas ordenadas de *bits*, es decir, $\tau = \{00, 10, 01, 11\}$. Además, los símbolos de salida vienen dados por el conjunto $\delta = \{0, 1\}$ que contiene el primer dígito de los resultados de las sumas básicas de cada elemento de τ, de acuerdo con la tabla 3. Los estados de la máquina adquieren dos posibilidades: llevar un 1 que será representado como "SLL" (se lleva) o no llevar nada, denotado como "NSLL" (no se lleva). Las aristas del diagrama simbolizarán si se lleva o no un 1 al adicionar cada elemento de τ.

Solución del ejemplo 7.7

Al estar ubicados en el estado "NSLL" y considerar el conjunto τ, se tiene:

- Con 00 hay un lazo, pues $0+0=0$ y por consiguiente no se lleva ningún *bit*. Además, por el resultado de la suma de estos dos dígitos binarios el símbolo de salida será igual a 0.
- En 10 también hay un lazo en "NSLL" pues por la tabla 3, $1+0=1$ y esto implica que no se lleva nada. Lo mismo ocurre con 01, $0+1=1$ y esto indica que hay un lazo en "NSLL". En ambos casos, el símbolo de salida vinculado con la etiqueta de estas aristas corresponde a 1, al ser el resultado de la suma de este par de dígitos binarios.

Solución del ejemplo 7.7

- Al tomar 11, la tabla 3 muestra que $1 + 1 = 10$, es decir, se conserva 0 y se lleva un 1, por lo que en el diagrama de transición de esta máquina al encontrarnos en el estado "NSLL" con el símbolo de entrada 11, se debe dibujar una arista de "NSLL" a "SLL" representando con ello, que se lleva ese 1. El símbolo de salida relacionado con la etiqueta de esta arista es 0, al ser el primer dígito del resultado de la suma $1 + 1 = 10$.

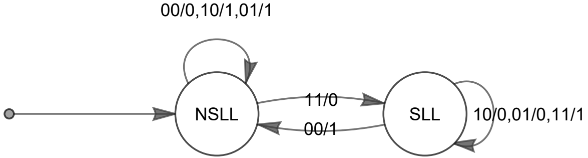
Solución del ejemplo 7.7

Luego, en el estado "SLL" al analizar la dirección que ocupan cada una de las aristas asociadas a los elementos de τ, se infiere:

- En 00 estando en el estado "SLL", significa que se debe realizar la suma $0+0=0$ y a este resultado adicionar el 1 que se lleva. Este 1 se considera pues la ubicación en "SLL" (se lleva) lo implica. Como $0+1=1$ (no se lleva nada), se dibuja una arista del estado "SLL" al estado "NSLL" con símbolo de salida 1.
- Con 10, $1+0=1$ y este 1 se suma al 1 que se lleva en el estado "SLL", donde $1+1=10$, lo que produce la presencia de un lazo con símbolo de salida 0. Lo mismo ocurre en 01.
- Ahora, en 11, $1+1=10$ y se debe sumar a 10 el 1 que se lleva. Se observa que $10+1=11$, razón por la cual, hay un ciclo en "SLL" con 11 y símbolo de salida igual a 1 (el primer dígito de la suma $10+1=11$).

Solución del ejemplo 7.7

Como consecuencia de este razonamiento, el diagrama de transición de la máquina de estado finito sumadora de binarios corresponde a:



El estado inicial se ha direccionado a "NSLL" pues naturalmente al comenzar una suma binaria no se lleva nada.

Solución del ejemplo 7.7

El comando `MaquinaSumadoraBinarios` del paquete **VilCretas** facilita el despliegue del diagrama anterior, al ejecutar:

In[ ] :=

```
MaquinaSumadoraBinarios[]
```

Nota

La *MEF* creada permite sumar dos números binarios a y b a través del procesamiento de una hilera de símbolos de entrada. Al construir el "string" de símbolos de entrada α, tomando el primer dígito de a y b, luego el segundo de a y b, y así sucesivamente hasta terminar con todos los dígitos, la hilera α resultante, procesada en la máquina de estado finito, forma un "string" β de símbolos de salida, donde se añade un 1 al final de la hilera, si el último estado es "SLL", o se deja sin modificar, si se finaliza en "NSLL". Todo esto deriva en un conjunto de símbolos de $\delta = \{0, 1\}$, que escrito al revés, corresponde a la suma binaria buscada.

Solución del ejemplo 7.7

Por ejemplo, si se desea sumar $(0110)_2 + (0111)_2$ el alumno puede comprobar cómo se forma $\alpha = \boxed{01}\boxed{11}\boxed{11}\boxed{00}$. Las casillas se han incluido en α para facilitar la lectura del "string" de datos de entrada. El procesamiento de α en la *MEF* es:

$$\text{NSLL} \overset{01/1}{\rightarrow} \text{NSLL} \overset{11/0}{\rightarrow} \text{NSLL} \overset{11/1}{\rightarrow} \text{NSLL} \overset{00/1}{\rightarrow} \text{NSLL} \tag{4}$$

Obteniéndose $\beta = 1011$, donde no se agrega un 1 al final pues se está terminando en el estado "NSLL". El "string" β escrito de derecha a izquierda es el resultado de la suma binaria buscada, por lo que, $(0110)_2 + (0111)_2 = (1101)_2$.

Solución del ejemplo 7.7

En *Wolfram Mathematica* empleando la instrucción `StringSalida` es factible corroborar lo expuesto en 4:

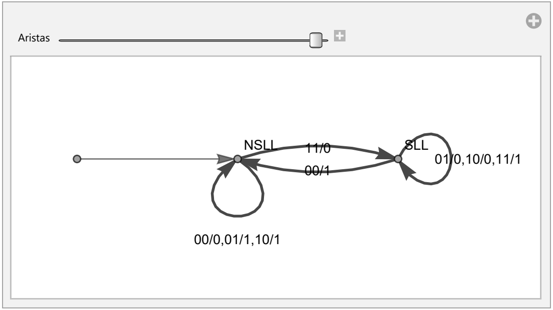
In[ ] :=

```
α = {"01", "11", "11", "00"};
StringSalida[{NSLL, SLL}, {"00", "01", "10", "11"}, {0, 1},
NSLL, {{NSLL, "00", NSLL, 0}, {NSLL, "01", NSLL, 1},
{NSLL, "10", NSLL, 1}, {NSLL, "11", SLL, 0},
{SLL, "00", NSLL, 1}, {SLL, "01", SLL, 0},
{SLL, "10", SLL, 0}, {SLL, "11", SLL, 1}}, α,
trace -> True]
```

Solución del ejemplo 7.7

Out[ ] =

{{1, 0, 1, 1}, {NSLL, NSLL, SLL, SLL, NSLL}}



Solución del ejemplo 7.7

En el código se hace necesario incluir los elementos de τ como "strings" (uso de comillas) pues de lo contrario *Mathematica* transforma por defecto 00 en 0, lo cual no es consistente con el diseño de la máquina.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-165.zip

Utilización del comando `MaquinaSumadoraBinarios`.

Explicación en video

https://youtu.be/BzxQbYfj_jw

Example (7.8)

Recurriendo a la *MEF* del ejemplo 9 realice la suma $(110101111101011)_2 + (1101101101111)_2$.

Solución del ejemplo 7.8

Los números binarios a sumar no tienen la misma longitud, el primero contiene 15 dígitos y el segundo 13. Para poder construir la hilera α a procesar en la máquina sumadora de binarios del ejemplo 9, se requiere una extensión idéntica para ambos números. Esto es fácil de ajustar, añadiendo al segundo número binario, dos bits iguales a 0 como sus últimos dígitos. En consecuencia, la suma de interés se puede expresar así:

$$(110101111101011)_2 + (001101101101111)_2$$

Luego:

$$\alpha = \boxed{11}\boxed{11}\boxed{01}\boxed{11}\boxed{00}\boxed{11}\boxed{11}\boxed{10}\boxed{11}\boxed{11}\boxed{00}\boxed{11}\boxed{01}\boxed{10}\boxed{10}$$

Solución del ejemplo 7.8

Al procesar α en la máquina sumadora de binarios, se obtiene:

$$\text{NSLL} \overset{11/0}{\to} \text{SLL} \overset{11/1}{\to} \text{SLL} \overset{01/0}{\to} \text{SLL} \overset{11/1}{\to} \text{SLL} \overset{00/1}{\to} \text{NSLL} \overset{11/0}{\to} \text{SLL} \overset{11/1}{\to} \text{SLL} \overset{10/0}{\to}$$
$$\text{SLL} \overset{11/1}{\to} \text{SLL} \overset{11/1}{\to} \text{SLL} \overset{00/1}{\to} \text{NSLL} \overset{11/0}{\to} \text{SLL} \overset{01/0}{\to} \text{SLL} \overset{10/0}{\to} \text{SLL} \overset{10/0}{\to} \text{SLL}$$

En este caso, se está finalizando en el estado "SLL" (se lleva) por lo que se debe incluir un 1 al final de la hilera de salida $\beta = 010110101110000$ y posteriormente darle vuelta a la secuencia, con el objetivo de encontrar el resultado de la suma binaria. Por lo tanto:

$$(11010111101011)_2 + (1101101101111)_2 = (1000011101011010)_2$$

Solución del ejemplo 7.8

Una verificación en *Wolfram* se efectuaría así:

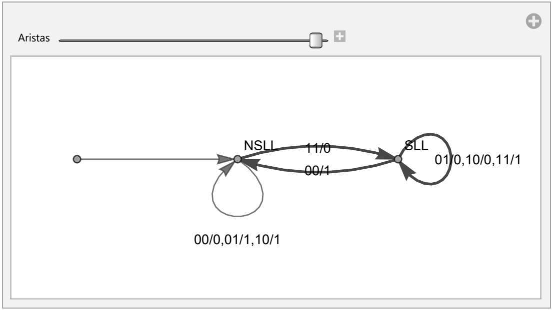
In[ ] :=

```
α = StringJoin /@ Reverse[Thread[{ToString /@ {1, 1, 0, 1,
0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1}, ToString /@ {0, 0, 1, 1,
0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1}}]]
StringSalida[{NSLL, SLL}, {"00", "01", "10", "11"}, {0, 1},
NSLL, {{NSLL, "00", NSLL, 0}, {NSLL, "01", NSLL, 1},
{NSLL, "10", NSLL, 1}, {NSLL, "11", SLL, 0},
{SLL, "00", NSLL, 1}, {SLL, "01", SLL, 0},
{SLL, "10", SLL, 0}, {SLL, "11", SLL, 1}}, α,
trace -> True]
```

Solución del ejemplo 7.8

Out[ ] =

{11, 11, 01, 11, 00, 11, 11, 10, 11, 11, 00, 11, 01, 10, 10}
{{0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0}, {NSLL, SLL, SLL, SLL, SLL, NSLL, SLL, SLL, SLL, SLL, SLL, NSLL, SLL, SLL, SLL, SLL}}



Solución del ejemplo 7.8

La sentencia `Thread` en el código es propia del software *Mathematica* y cumple la tarea de asociar las parejas de dígitos binarios correspondientes para la creación del "string" α de símbolos de τ.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-166.zip

CDF: máquinas de estado finito.

Un interesante documento con un formato computable se comparte en la siguiente descarga. El *CDF* en cuestión, recibe las seis componentes de una máquina de estado finito y una hilera de símbolos de entrada, retornando su diagrama de transición, el "string" β de salida y el recorrido sobre cada uno de los estados que permitieron construir a β. La figura 1 muestra la funcionalidad del *CDF* sobre una *MEF* en particular. La librería **VilCretas** integra esta misma herramienta mediante el comando `CDFMaquinaEF`.

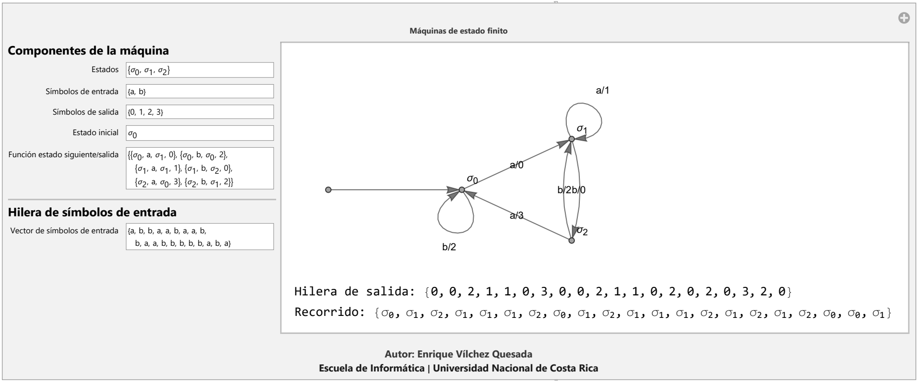


Figura: Funcionamiento del *CDF* “máquinas de estado finito”

CDF: máquinas de estado finito

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/CDFs/MEF.cdf.zip

- En la sección que continua se estudiará cierto tipo de máquinas de estado finito llamadas autómatas de estado finito determinísticos. Los autómatas tienen importantes aplicaciones en computación y una de ellas será abarcada en el capítulo de lenguajes y gramáticas.

Autómatas de estado finito determinísticos

Prof. Enrique Vílchez Quesada

Universidad Nacional de Costa Rica

Definición 7.3. Autómata de estado finito determinístico

Un autómata de estado finito determinístico representado con el acrónimo *ADF* es un tipo especial de máquina de estado finito. La definición próxima formaliza este concepto.

Definition (7.3)

Un autómata de estado finito determinístico denotado *ADF* es una máquina de estado finito $A = (\sigma, \tau, \delta, \sigma^*, \Delta, \Omega)$ con $\delta = \{0, 1\}$ y donde, en el diagrama de transición de A, todas las aristas que entran a un estado μ, poseen el mismo símbolo de salida. Un estado μ donde todos los lados entrantes tienen símbolo de salida 1, se llama "estado aceptado" y en caso contrario, símbolo de salida 0, se denomina "estado no aceptado"

Comentario sobre la definición 11

En el contexto de la definición 11 un autómata de estado finito determinístico es una *MEF* donde los únicos símbolos de salida posibles son el 0 y el 1 y además, si $\Delta\left(\mu_i, a\right) = \mu$ y $\Delta\left(\mu_j, b\right) = \mu$ entonces $\Omega\left(\mu_i, a\right) = \Omega\left(\mu_j, b\right)$, $\forall \mu_i,\ \mu_j \in \sigma$, es decir, en el diagrama de transición del autómata las aristas entrantes a un estado μ siempre tendrán el mismo símbolo de salida.

Definición 7.4. Componentes de un autómata de estado finito determinístico

Estas dos características hacen que el concepto de autómata de estado finito determinístico dependa no de seis componentes sino más bien de cinco (una 5−tu-pla). Como $\delta = \{0, 1\}$ se puede omitir este conjunto de las seis componentes del ADF y además, en lugar de incluir una función de salida (Ω) se sustituye por un conjunto de estados aceptados que se denotará como $\widehat{A}$. Veamos la definición 12.

Definition (7.4)

Un autómata de estado finito determinístico es una 5−tupla $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$ con $\widehat{A}$ el conjunto de estados aceptados.

Comentario sobre la definición 12

En la definición 12 se presupone que los símbolos de salida de la *MEF* son 0 o 1, por lo tanto, se excluye de A la especificación del conjunto δ. Además, la función de salida Ω, como ya se señaló, se reemplaza por el conjunto $\widehat{A}$ formado por todos los estados cuyos lados entrantes en el diagrama de transición de A, tienen como símbolo de salida un 1. Como es natural aquellos estados que no pertenecen a $\widehat{A}$ son no aceptados y por consiguiente, sus aristas entrantes en el diagrama de transición tienen como símbolo de salida un 0. Lo anterior significa que el conjunto $\widehat{A}$ es capaz de relevar a la función Ω pues con solo conocer cuáles son los estados aceptados de A (y por lo tanto, también los no aceptados) es suficiente para describir el comportamiento de los símbolos de salida al tener un estado y un símbolo de entrada.

- Con todo ello, el concepto de autómata de estado finito determinístico ha sufrido algunas modificaciones en comparación con la definición de máquina de estado finito. Bajo esta perspectiva, el diagrama de transición de un autómata, como es de esperarse, también presenta ciertas diferencias.

En un autómata de estado finito determinístico su diagrama de transición se realiza colocando una doble circunferencia sobre los estados aceptados y omitiendo en las etiquetas de las aristas los símbolos de salida.

Example (7.9)

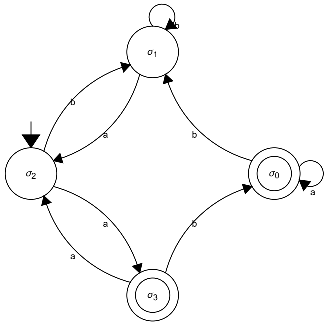
Sea el autómata de estado finito determinístico $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$, con $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$, $\tau = \{a, b\}$, $\sigma^* = \sigma_2$, $\widehat{A} = \{\sigma_0, \sigma_3\}$ y:

Δ	a	b
σ_0	σ_0	σ_1
σ_1	σ_2	σ_1
σ_2	σ_3	σ_1
σ_3	σ_2	σ_0

Construya su diagrama de transición.

Solución del ejemplo 7.9

El diagrama de transición se elabora tal y como se explicó en el caso de una máquina de estado finito con el detalle de usar una doble circunferencia en los estados σ_0 y σ_3, al ser aceptados y omitiendo los símbolos de salida en las etiquetas de los lados del digrafo. En este ejemplo, el diagrama corresponde a:



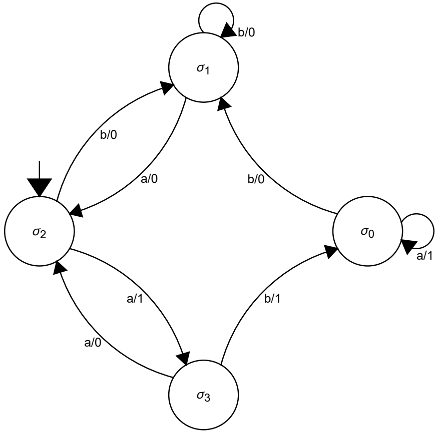
(5)

Nota

El estudiante debe apreciar que todo autómata de estado finito determinístico es una *MEF* según la definición 11, a razón de ello, todo autómata se podrá representar también, mediante un diagrama de transición correspondiente a una máquina de estado finito.

Solución del ejemplo 7.9

En este sentido, el siguiente digrafo representa el *ADF* de este ejemplo, si se pensara en el diagrama de transición de una *MEF*:



(6)

Solución del ejemplo 7.9

El digrafo 6 se ha obtenido del diagrama de transición 5, al incluir un 1 en las etiquetas de los lados entrantes a los estados aceptados σ_0 y σ_3 y, al incorporar un símbolo de salida 0 en las etiquetas restantes, correspondientes a las aristas que llegan a los estados no aceptados σ_1 y σ_2. Además, se ha eliminado la doble circunferencia en los estados σ_0 y σ_3. No es usual elaborar el diagrama de transición de un *ADF* como una *MEF*, sin embargo, aquí es importante aclarar la posibilidad de hacerlo, de acuerdo con el concepto de autómata.

Solución del ejemplo 7.9

En *Wolfram Mathematica* el diagrama de transición de un autómata de estado finito se puede elaborar recurriendo a dos instrucciones del paquete **VilCretas**: `Automata` y `AutomataToDiagrama`. La primera permite crear un autómata en el software y la segunda desplegar su diagrama de transición. `Automata` recibe la lista de estados σ, el conjunto de símbolos de entrada τ, el estado inicial σ^*, la función de transición Δ (como un vector de tripletes del estado origen, un elemento de τ y el estado de destino) y el conjunto de estados aceptados $\widehat{A}$. Además, `AutomataToDiagrama` tiene la opción `forma -> "rectangular"` que especifica la colocación de los estados en un formato rectangular (de oficio, esto se realiza de manera circular).

Solución del ejemplo 7.9

El código expuesto a continuación retorna como salida el digrafo 5:

In[ ] :=

A = Automata[{σ_0, σ_1, σ_2, σ_3}, {a, b}, σ_2, {{σ_0, a, σ_0}, {σ_0, b, σ_1}, {σ_1, a, σ_2}, {σ_1, b, σ_1}, {σ_2, a, σ_3}, {σ_2, b, σ_1}, {σ_3, a, σ_2}, {σ_3, b, σ_0}}, {σ_0, σ_3}]
AutomataToDiagrama[A]

Solución del ejemplo 7.9

La sentencia `Automata` por defecto siempre devuelve el mensaje - Automaton - para indicar al usuario que el autómata se creó exitosamente. Si el mensaje no se muestra en el **Out[ ]** es porque hay algún error en los argumentos recibidos por `Automata`. Además, `AutomataToDiagrama` cuenta con la opción `colores -> True` que añade un color en particular a cada una de las aristas salientes de un mismo estado, esto con la finalidad de dar una mejor lectura al procesamiento de una hilera de símbolos de τ.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-167.zip

Uso del comando `Automata`.

Explicación en video

https://youtu.be/R9Lyq-vEmBE

Empleo de la instrucción `AutomataToDiagrama`.

Explicación en video

https://youtu.be/G64e1pyavd4

El diagrama de transición de un autómata de estado finito es menos cargado al compararlo con el diagrama de una máquina de estado finito, dado que se omiten los símbolos de salida en las etiquetas de las aristas sustituyéndolos por una doble circunferencia en los estados aceptados.

Example (7.10)

Elabore mediante el uso de software el diagrama de transición del autómata de estado finito $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$, con $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$, $\tau = \{a, b, c\}$, $\sigma^* = \sigma_0$, $\widehat{A} = \{\sigma_1, \sigma_2\}$ y:

Δ	a	b	c
σ_0	σ_3	σ_0	σ_1
σ_1	σ_0	σ_3	σ_2
σ_2	σ_1	σ_1	σ_3
σ_3	σ_0	σ_0	σ_3

Solución del ejemplo 7.10

En *Mathematica*:

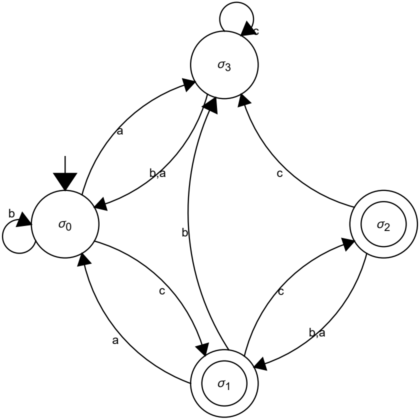
In[ ] :=

```
A = Automata[{σ0, σ1, σ2, σ3}, {a, b, c}, σ0, {{σ0, a, σ3},
{σ0, b, σ0}, {σ0, c, σ1}, {σ1, a, σ0}, {σ1, b, σ3},
{σ1, c, σ2}, {σ2, a, σ1}, {σ2, b, σ1}, {σ2, c, σ3},
{σ3, a, σ0}, {σ3, b, σ0}, {σ3, c, σ3}}, {σ1, σ2}]
AutomataToDiagrama[A]
```

Solución del ejemplo 7.10

Out[ ] $=$

- Automaton -



Solución del ejemplo 7.10

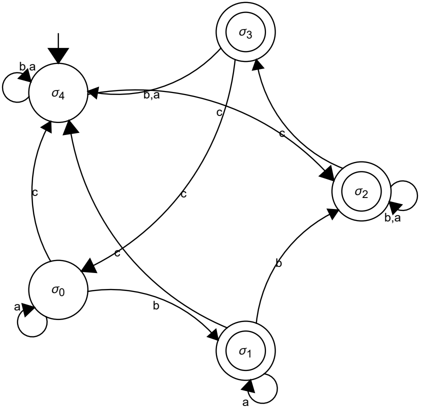
Las etiquetas "b, a" en el digrafo, señalan la existencia de aristas múltiples entre los estados respectivos.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-168.zip

Example (7.11)

Determine las componentes como una 5−tupla, del autómata de estado finito dado por el diagrama de transición adjunto.



Solución del ejemplo 7.11

Observando el digrafo del enunciado se concluye:

- $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$.
- $\tau = \{a, b, c\}$.
- $\sigma^* = \sigma_4$.
-

Δ	a	b	c
σ_0	σ_0	σ_1	σ_4
σ_1	σ_1	σ_2	σ_4
σ_2	σ_2	σ_2	σ_3
σ_3	σ_4	σ_4	σ_0
σ_4	σ_4	σ_4	σ_2

Solución del ejemplo 7.11

La librería **ViCretas** facilita el comando `ComponentesAutomata`, que recibe un autómata y genera las cinco componentes que lo caracterizan. Se invita al alumno a experimentar su utilización.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-169.zip

Uso de la instrucción `ComponentesAutomata`.

Explicación en video

https://youtu.be/7MhN_IwJrOM

- Los autómatas de estado finito al igual que las *MEF* procesan hileras de símbolos de entrada. En un autómata de estado finito, interesa si la hilera recibida es "aceptada" o "no aceptada". Esto depende del último estado del recorrido de la hilera en el diagrama de transición del autómata. Consideremos la definición que prosigue.

Definición 7.5. Hilera aceptada en ADF

Definition (7.5)

Sea $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$ un autómata de estado finito determinístico y $\alpha = x_1 x_2 \ldots x_n$ una hilera de símbolos de entrada, para la cual existe un conjunto de estados σ_0, σ_1, ..., σ_n de σ, tales que:

$$
\begin{aligned}
&\sigma^* = \sigma_0 \\
&\Delta\left(\sigma_0, x_1\right) = \sigma_1 \\
&\Delta\left(\sigma_1, x_2\right) = \sigma_2 \\
&\vdots \\
&\Delta\left(\sigma_{n-1}, x_n\right) = \sigma_n
\end{aligned}
$$

Se dice que α es aceptada si el último estado σ_n es aceptado, es decir, $\sigma_n \in \widehat{A}$. En caso contrario, α es una hilera no aceptada

Comentario sobre la definición 16

La definición 16 propone que en un autómata de estado finito determinístico se procesa una hilera α de símbolos de τ realizando el recorrido implicado por α en el diagrama de transición de la máquina y observando si se finaliza o no en un estado aceptado. Si se termina en un estado aceptado se dice que α es una hilera de símbolos de entrada aceptada y en caso contrario, α sería no aceptada. A diferencia de lo planteado en la definición 5, en un ADF no interesa retornar una hilera β de símbolos de salida, solo se requiere analizar si α es aceptada o no.

Example (7.12)

Establezca si la hilera de símbolos de entrada $\alpha = abcbbaccabccabb$ es aceptada en los autómatas de los ejemplos 14 y 15.

Solución del ejemplo 7.12

En el *ADF* del ejemplo 14, se obtiene al recorrer $\alpha = abcbbaccabccabb$ en el diagrama de transición, lo siguiente:

$$\begin{array}{l}\sigma_0 \xrightarrow{a} \sigma_3 \xrightarrow{b} \sigma_0 \xrightarrow{c} \sigma_1 \xrightarrow{b} \sigma_3 \xrightarrow{b} \sigma_0 \xrightarrow{a} \sigma_3 \xrightarrow{c} \sigma_3 \xrightarrow{c} \\ \sigma_3 \xrightarrow{a} \sigma_0 \xrightarrow{b} \sigma_0 \xrightarrow{c} \sigma_1 \xrightarrow{c} \sigma_2 \xrightarrow{a} \sigma_1 \xrightarrow{b} \sigma_3 \xrightarrow{b} \boxed{\sigma_0}\end{array}$$

Al ser el estado σ_0 no aceptado se concluye que la hilera α es no aceptada. Por otra parte, en el autómata del ejemplo 15, al analizar en el digrafo que lo representa, el camino provisto por la hilera α, se infiere:

$$\begin{array}{l}\sigma_4 \xrightarrow{a} \sigma_4 \xrightarrow{b} \sigma_4 \xrightarrow{c} \sigma_2 \xrightarrow{b} \sigma_2 \xrightarrow{b} \sigma_2 \xrightarrow{a} \sigma_2 \xrightarrow{c} \sigma_3 \xrightarrow{c} \\ \sigma_0 \xrightarrow{a} \sigma_0 \xrightarrow{b} \sigma_1 \xrightarrow{c} \sigma_4 \xrightarrow{c} \sigma_2 \xrightarrow{a} \sigma_2 \xrightarrow{b} \sigma_2 \xrightarrow{b} \boxed{\sigma_2}\end{array}$$

El estado σ_2 es aceptado por lo que la hilera α es aceptada.

Solución del ejemplo 7.12

La instrucción `StringAceptadaQ` del paquete **VilCretas** conforma una útil herramienta para determinar en un autómata de estado finito si una hilera α de símbolos de τ es aceptada o no. El comando brinda la opción `trace -> True` si se desea visualizar la trayectoria de estados definida por α. En el contexto de este ejercicio se procedería en *Wolfram*, así:

In[ ] :=

```
A1 = Automata[{σ0, σ1, σ2, σ3}, {a, b, c}, σ0, {{σ0, a, σ3},
{σ0, b, σ0}, {σ0, c, σ1}, {σ1, a, σ0}, {σ1, b, σ3},
{σ1, c, σ2}, {σ2, a, σ1}, {σ2, b, σ1}, {σ2, c, σ3},
{σ3, a, σ0}, {σ3, b, σ0}, {σ3, c, σ3}}, {σ1, σ2}];
α = {a, b, c, b, b, a, c, c, a, b, c, c, a, b, b};
StringAceptadaQ[A1, α]
StringAceptadaQ[A1, α, trace -> True]
```

Solución del ejemplo 7.12

```
A2 = Automata[{σ0, σ1, σ2, σ3, σ4}, {a, b, c}, σ4,
{{σ0, a, σ0}, {σ0, b, σ1}, {σ0, c, σ4}, {σ1, a, σ1},
{σ1, b, σ2}, {σ1, c, σ4}, {σ2, a, σ2}, {σ2, b, σ2},
{σ2, c, σ3}, {σ3, a, σ4}, {σ3, b, σ4}, {σ3, c, σ0},
{σ4, a, σ4}, {σ4, b, σ4}, {σ4, c, σ2}}, {σ1, σ2, σ3}];
StringAceptadaQ[A2, α]
StringAceptadaQ[A2, α, trace -> True]
```

Solución del ejemplo 7.12

Out[ ] $=$

False

$\{$False, $\{\sigma_0, \sigma_3, \sigma_0, \sigma_1, \sigma_3, \sigma_0, \sigma_3, \sigma_3, \sigma_3, \sigma_0, \sigma_0, \sigma_1, \sigma_2, \sigma_1, \sigma_3, \sigma_0\}\}$

True

$\{$True, $\{\sigma_4, \sigma_4, \sigma_4, \sigma_2, \sigma_2, \sigma_2, \sigma_2, \sigma_3, \sigma_0, \sigma_0, \sigma_1, \sigma_4, \sigma_2, \sigma_2, \sigma_2, \sigma_2\}\}$

El False indica que la hilera α es no aceptada por el autómata `A1` y el valor lógico True muestra que α sí es aceptada por el autómata `A2`.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-170.zip

Empleo de la sentencia `StringAceptadaQ`.

Explicación en video

https://youtu.be/knGX4lSUKZY

Example (7.13)

Determine si las hileras $\alpha_1 = aabbbbabaababaaaaaaa$ y $\alpha_2 = aaaaabbaaabbaabbbbaa$ son aceptadas en el autómata de estado finito determinístico del ejemplo 13.

Solución del ejemplo 7.13

En $\alpha_1 = aabbbbabaababaaaaaa$ se produce el recorrido:

$$\sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{b} \sigma_1 \xrightarrow{b} \sigma_1 \xrightarrow{b} \sigma_1 \xrightarrow{b} \sigma_1 \xrightarrow{a} \sigma_2 \xrightarrow{b} \sigma_1 \xrightarrow{a} \sigma_2 \xrightarrow{a}$$
$$\sigma_3 \xrightarrow{b} \sigma_0 \xrightarrow{a} \sigma_0 \xrightarrow{b} \sigma_1 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{a} \boxed{\sigma_2}$$

El estado σ_2 es no aceptado por lo que la hilera α_1 se considera no aceptada.
En $\alpha_2 = aaaaabbaaabbaabbbbaa$ se genera la trayectoria:

$$\sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{b} \sigma_0 \xrightarrow{b} \sigma_1 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{a}$$
$$\sigma_2 \xrightarrow{b} \sigma_1 \xrightarrow{b} \sigma_1 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{b} \sigma_0 \xrightarrow{b} \sigma_1 \xrightarrow{b} \sigma_1 \xrightarrow{b} \sigma_1 \xrightarrow{a} \sigma_2 \xrightarrow{a} \boxed{\sigma_3}$$

Como σ_3 es aceptado la hilera α_2 es aceptada por el autómata de estado finito.

Solución del ejemplo 7.13

Una verificación en *Mathematica* de estos resultados, se resolvería como sigue:

In[ ] :=

```
α1 = {a, a, b, b, b, b, a, b, a, a, b, a, b, a, a, a, a, a,
a, a};
StringAceptadaQ[A, α1]
StringAceptadaQ[A, α1, trace -> True]
α2 = {a, a, a, a, a, b, b, a, a, a, b, b, a, a, b, b, b, b,
a, a};
StringAceptadaQ[A, α2]
StringAceptadaQ[A, α2, trace -> True]
```

Solución del ejemplo 7.13

Out[ ] $=$

False

$\{$False, $\{\sigma_2, \sigma_3, \sigma_2, \sigma_1, \sigma_1, \sigma_1, \sigma_1, \sigma_2, \sigma_1, \sigma_2, \sigma_3, \sigma_0, \sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_2, \sigma_3, \sigma_2, \sigma_3, \sigma_2\}\}$

True

$\{$True, $\{\sigma_2, \sigma_3, \sigma_2, \sigma_3, \sigma_2, \sigma_3, \sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_2, \sigma_1, \sigma_1, \sigma_2, \sigma_3, \sigma_0, \sigma_1, \sigma_1, \sigma_1, \sigma_2, \sigma_3\}\}$

Solución del ejemplo 7.13

En el código anterior, `A` es una variable que contiene el autómata concebido con la instrucción `Automata`.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-171.zip

Definición 7.6. Autómatas equivalentes

Otro concepto fundamental es el de autómatas equivalentes.

Definition (7.6)

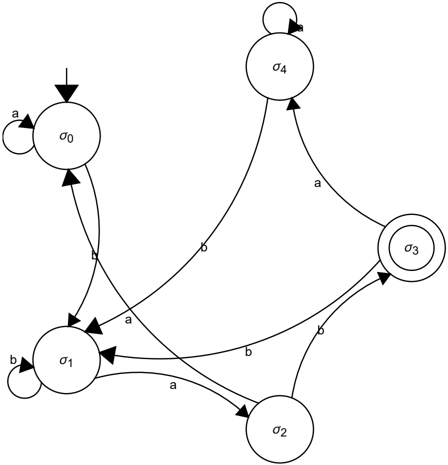
Dos autómatas A_1 y A_2 se dice que son equivalentes si aceptan las mismas hileras de símbolos de entrada. Además, al conjunto de hileras aceptadas por un autómata A, denotado A^o, se le llama "lenguaje del autómata"

Comentario sobre definición 19

El estudiante debe entender a cabalidad el concepto de lenguaje de un autómata A. De acuerdo con la definición 19, éste corresponde al conjunto A^o constituido por todas las hileras de símbolos de entrada que el ADF puede aceptar. En algunas ocasiones A^o será un conjunto infinito y en otras, un conjunto finito.

Example (7.14)

Conjeture cuál es el lenguaje del autómata dado a continuación.



Solución del ejemplo 7.14

En este ejemplo, se requiere analizar la forma que toman todas las hileras aceptadas por el *ADF*, procurando encontrar una o varias estructuras generales. A este respecto, la sentencia `LenguajeStrings` del paquete **VilCretas**, ofrece un interesante recurso para conocer diferentes hileras aceptadas por un autómata de estado finito con una longitud igual o menor o igual a n, $n \epsilon \mathbb{N}$. En este ejercicio `LenguajeStrings` constituye un mecanismo de resolución para la elaboración de la conjetura deseada.

Solución del ejemplo 7.14

Luego, en *Wolfram*:

In[ ] :=

```
A = Automata[{σ0, σ1, σ2, σ3, σ4}, {a, b}, σ0, {{σ0, a, σ0},
{σ0, b, σ1}, {σ1, a, σ2}, {σ1, b, σ1}, {σ2, a, σ0},
{σ2, b, σ3}, {σ3, a, σ4}, {σ3, b, σ1}, {σ4, a, σ4},
{σ4, b, σ1}}, {σ3}];
LenguajeStrings[A, 6, limite -> True]
```

Out[ ] =

{{b, a, b}, {a, b, a, b}, {b, b, a, b}, {a, a, b, a, b}, {a, b, b, a, b},
{b, b, b, a, b}, {a, a, a, b, a, b}, {a, a, b, b, a, b}, {a, b, b, b, a, b},
{b, a, a, b, a, b}, {b, a, b, b, a, b}, {b, b, b, b, a, b}}

Solución del ejemplo 7.14

La opción `limite -> True` solicita al software retornar todas las hileras aceptadas por `A`, de longitud menor o igual a 6. Si se prescinde de esta opción, el programa computa únicamente las hileras aceptadas de longitud igual a 6. Al observar la salida proveída, se conjetura un patrón en las hileras aceptadas: todas ellas terminan en "*bab*". Como consecuencia, se deduce:

$$A^o = \{\alpha \mid \alpha \text{ es una hilera de símbolos de } \tau \text{ que finaliza en } bab\} \tag{7}$$

Solución del ejemplo 7.14

Se insta al alumno a cambiar el parámetro 6 de la sentencia `LenguajeStrings` por números naturales mayores, con la intención de visualizar el mismo comportamiento descrito en 7, sobre otros grupos de hileras aceptadas por `A`.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-172.zip

Utilización del comando `LenguajeStrings`.

Explicación en video

https://youtu.be/ZiICN5lrODs

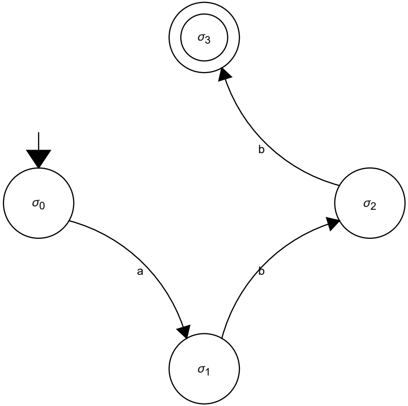
- Los ejemplos que se aborarán a continuación son de mucha relevancia. Describen algunas técnicas de trabajo para diseñar un autómata de estado finito determinístico conociendo de previo cuál es su lenguaje, es decir, el objetivo consiste en la construcción de un *ADF* que acepte cierto tipo de hileras de símbolos de entrada. Como se apreciará en los seis ejemplos compartidos, el proceso de diseño se fundamenta en observar cuál es la hilera más pequeña que el autómata debe de aceptar, recordando que por cada uno de los estados que lo caracterizan existe un lado saliente en correspondencia con cada uno de los símbolos del conjunto τ.

Example (7.15)

Diseñe un autómata de estado finito que acepte únicamente hileras que finalizan en *abb*.

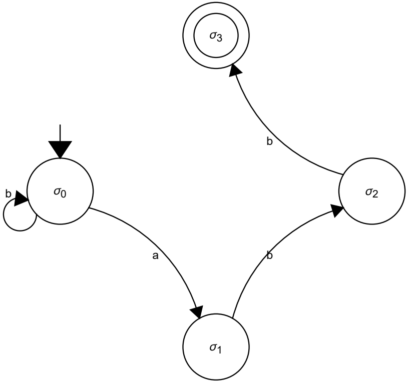
Solución del ejemplo 7.15

La hilera más pequeña que el autómata debe de aceptar es *abb*. Se inicia tomando cuatro estados cuya función es recorrer esa hilera, hasta llegar a un estado aceptado:



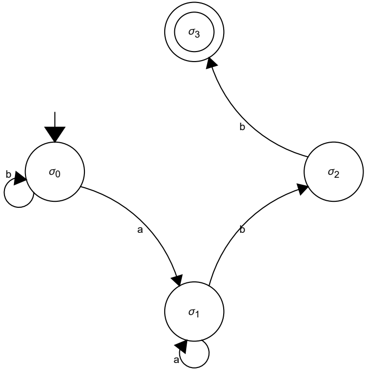
Solución del ejemplo 7.15

Ahora, se completa el autómata de estado finito con las aristas faltantes. En σ_0 se tiene la ausencia del lado saliente con símbolo de entrada b. Allí, se incluye un lazo con b, al estar comenzando el recorrido:



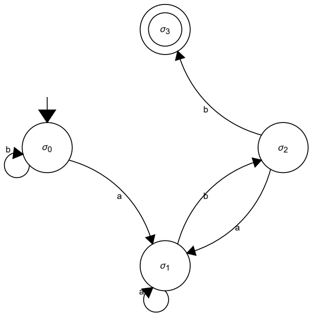
Solución del ejemplo 7.15

En σ_1 se debe agregar una arista de salida para el símbolo a. Resulta conveniente poner un lazo con a y no regresar al estado σ_0, pues si se retorna a σ_0, por ejemplo la hilera *abb*, finalizaría en ese estado y por lo tanto, no sería aceptada. Luego:



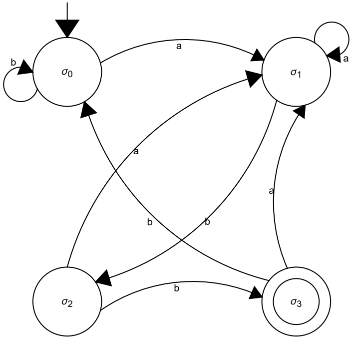
Solución del ejemplo 7.15

En σ_2 hay que añadir un lado saliente asociado al símbolo *a*. No es factible incluir un lazo con *a* pues se pierde la secuencia "*ab*" lograda en ese punto. Usualmente en este tipo de ejercicios, donde se pretende desarrollar un autómata que acepte hileras "finalizadas en", al no poder incorporar un lazo, se envía la arista a un estado anterior donde se encuentre un ciclo con el caracter respectivo. En este ejemplo con *a*, ese estado es σ_1 :



Solución del ejemplo 7.15

Finalmente en σ_3, faltan las dos aristas vinculadas con a y b. Con el símbolo a la dirección que toma el lado es hacia el estado σ_1 y con b, hacia el estado σ_0, donde ocurren los lazos con cada caracter a y b, respectivamente. Se concluye que el diagrama de transición del automáta diseñado es el siguiente:



Solución del ejemplo 7.15

En *Wolfram Mathematica* es posible ejecutar una prueba para establecer la correctitud del *ADF* creado, al analizar una serie de hileras de símbolos de entrada aceptadas, devueltas por la sentencia `LenguajeStrings`. El código que se muestra tiene ese propósito:

In[ ] :=

```
A = Automata[{σ0, σ1, σ2, σ3}, {a, b}, σ0, {{σ0, a, σ1},
{σ0, b, σ0}, {σ1, a, σ1}, {σ1, b, σ2}, {σ2, a, σ1},
{σ2, b, σ3}, {σ3, a, σ1}, {σ3, b, σ0}}, {σ3}];
```

Solución del ejemplo 7.15

```
Prueba[automata_, n_] := Module[{
L = LenguajeStrings[automata, n, limite -> True],
valorLogico = True, i, m}, m = Length[L];
For[i = 1, i <= m,
If[ToString[Take[L[[i]], -3]] != ToString[{a, b, b}],
valorLogico = False; Break[]]; i++];
Print[If[valorLogico == False, i, i - 1], "pruebas: ",
valorLogico]]
Prueba[A, 20]
```

Solución del ejemplo 7.15

Out[ ] =

262143 pruebas: True
El software analizó 262143 hileras de símbolos de entrada aceptadas por el autómata, de longitud menor o igual a 20 y todas ellas, cumplieron con la condición de finalizar en *abb*. En el código, el comando `Take` de *Wolfram*, se encarga de extraer en cada iteración los últimos tres caracteres de la hilera aceptada a estudiar. Si bien es cierto, la salida de *Mathematica* no es una demostración formal, el método de trabajo expuesto, al menos integra una verificación del diseño sobre algunos “strings”.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-173.zip

Example (7.16)

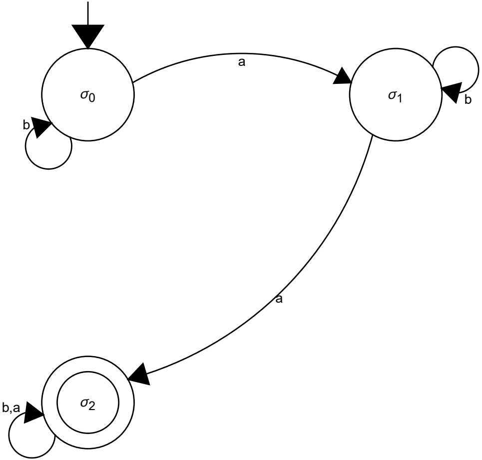
Elabore un *ADF* donde las hileras con al menos dos letras *a* sean aceptadas. Suponga $\tau = \{a, b\}$.

Solución del ejemplo 7.16

Al analizar el ejercicio, la hilera más corta de símbolos de τ que el autómata acepta es aa, de allí que se asume la existencia de tres estados σ_0, σ_1 y σ_2 para poderla recorrer, siendo σ_2 un estado aceptado. En el estado inicial σ_0 se coloca un lazo con símbolo de entrada b y otra arista al estado σ_1 con símbolo de entrada a. Posteriormente en σ_1, si entra otra a, la hilera debe ser aceptada (pues hay al menos dos a), por lo tanto, se direcciona la secuencia a σ_2. Si en σ_1 el dato de entrada es b, se toma un ciclo sobre este nodo pues el enunciado no señala que las letras a deben estar juntas. Finalmente en σ_2, cualquier símbolo de entrada debe quedarse en ese estado aceptado, dado que la hilera ya contendría como mínimo dos a.

Solución del ejemplo 7.16

El diagrama de transición del *ADF* descrito es:



Solución del ejemplo 7.16

Para probar el autómata creado en algunos casos particulares, se realizará un experimento que cuenta sobre el conjunto de hileras aceptadas de longitud menor o igual a 20 (podría ser otro tamaño), la cantidad de apariciones de *a*. Veamos:

In[ ] :=

A = Automata[$\{\sigma_0, \sigma_1, \sigma_2\}$, {a, b}, σ_0, $\{\{\sigma_0$, a, $\sigma_1\}$, $\{\sigma_0$, b, $\sigma_0\}$, $\{\sigma_1$, a, $\sigma_2\}$, $\{\sigma_1$, b, $\sigma_1\}$, $\{\sigma_2$, a, $\sigma_2\}$, $\{\sigma_2$, b, $\sigma_2\}\}$, $\{\sigma_2\}$];

Solución del ejemplo 7.16

```
Prueba[automata_, n_] := Module[{
L = LenguajeStrings[automata, n, limite -> True],
valorLogico = True, i, m}, m = Length[L];
For[i = 1, i <= m, If[Count[L[[i]], a] < 2,
valorLogico = False; Break[]]; i++];
Print[If[valorLogico == False, i, i - 1], "pruebas: ",
valorLogico]]
Prueba[A, 20]
```

Solución del ejemplo 7.16

Out[ ] $=$

2096920 pruebas: True

La verificación resultó ser exitosa, a razón de que el conteo de la cantidad de letras a en 2096920 "strings" aceptados, siempre fue mayor o igual a 2. El comando `Count` de *Mathematica*, en este sentido, cumple la función de contar el número de $a's$ en las cadenas analizadas.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-174.zip

Example (7.17)

Construya el diagrama de transición de un *ADF* que acepte hileras que contengan exactamente dos *a* y dos *b*.

Solución del ejemplo 7.17

El software *Wolfram Mathematica* posee la sentencia `Permutations` que retorna todos los posibles ordenamientos sobre los elementos de un conjunto. En el presente ejemplo, `Permutations[{a, a, b, b}]` devuelve el lenguaje del autómata de interés:

In[ ] :=

```
Permutations[{a, a, b, b}]
```

Out[ ] =

{{a, a, b, b}, {a, b, a, b}, {a, b, b, a}, {b, a, a, b}, {b, a, b, a}, {b, b, a, a}}

Solución del ejemplo 7.17

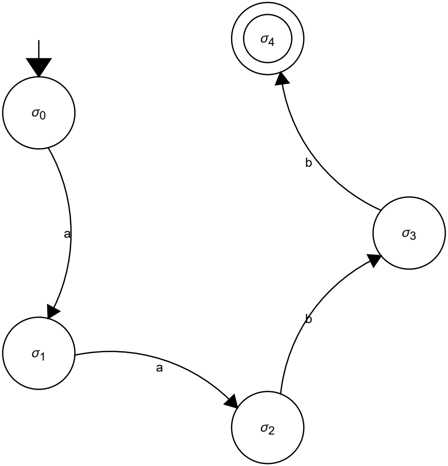
Luego, el conjunto de hileras que el autómata A a diseñar debe aceptar, corresponde a:

$$A^o = \{aabb, abab, abba, baab, baba, bbaa\} \tag{8}$$

El proceso de construcción de este autómata se circunscribe en crear una serie de subautómatas que acepten cada hilera del conjunto A^o y que todas ellas, finalicen en un único estado aceptado. Si la hilera procesada por el autómata no se encuentra en A^o, se enviará el recorrido a otro estado no aceptado.

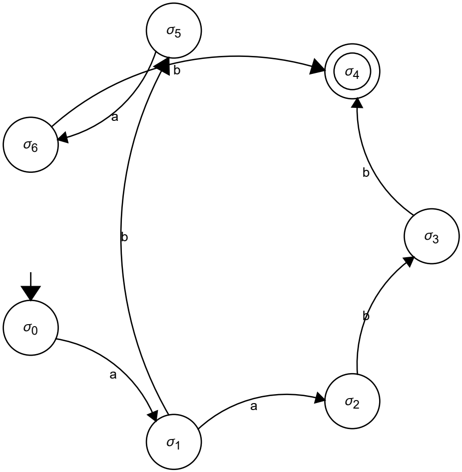
Solución del ejemplo 7.17

Al considerar el "string" *aabb* se tiene:



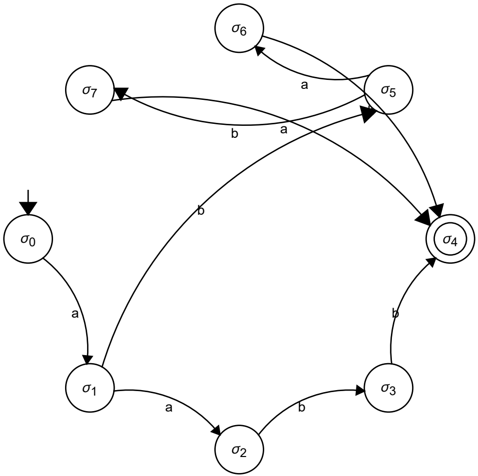
Solución del ejemplo 7.17

Luego, *abab* se incluye en el diagrama así:



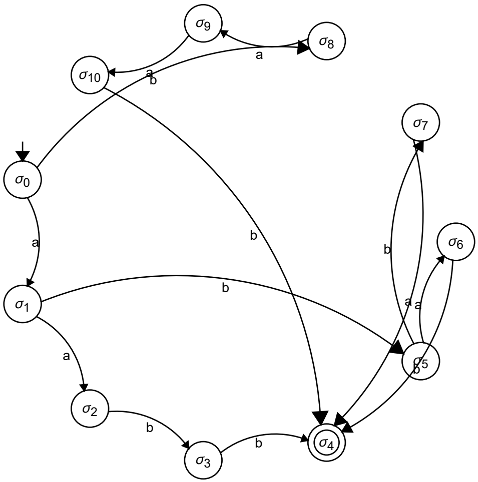
Solución del ejemplo 7.17

La hilera *abba* se integra en el digrafo como sigue:



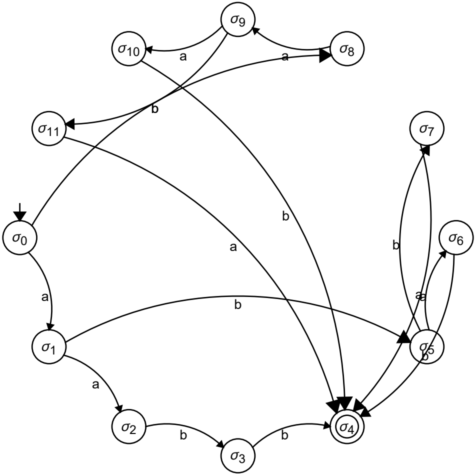
Solución del ejemplo 7.17

Con respecto a *baab*, se incorpora al diagrama de esta manera:



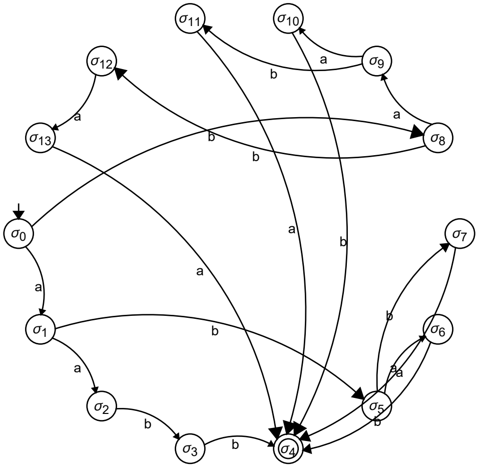
Solución del ejemplo 7.17

El "string" *baba* se añade así:



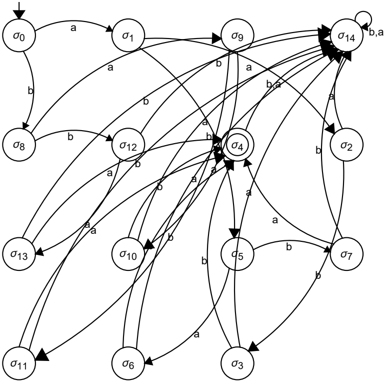
Solución del ejemplo 7.17

Además, al agregar la hilera *bbaa* se tiene:



Solución del ejemplo 7.17

Finalmente, se requiere un estado más σ_{14} en el diagrama de transición. Este estado es no aceptado y todas las aristas faltantes en el digrafo deben llegar a él, con el objetivo de aceptar únicamente las 6 hileras señaladas en 8. Al realizar lo anterior, se termina del diseño del *ADF*:



Solución del ejemplo 7.17

En el software una verificación de la correctitud de este autómata es:
In[ ] :=

A = Automata[{σ_0, σ_1, σ_2, σ_3, σ_4, σ_5, σ_6, σ_7, σ_8, σ_9, σ_{10}, σ_{11}, σ_{12}, σ_{13}, σ_{14}}, {a, b}, σ_0, {{σ_0, a, σ_1}, {σ_0, b, σ_8}, {σ_1, a, σ_2}, {σ_1, b, σ_5}, {σ_2, a, σ_{14}}, {σ_2, b, σ_3}, {σ_3, a, σ_{14}}, {σ_3, b, σ_4}, {σ_4, a, σ_{14}}, {σ_4, b, σ_{14}}, {σ_5, a, σ_6}, {σ_5, b, σ_7}, {σ_6, a, σ_{14}}, {σ_6, b, σ_4}, {σ_7, a, σ_4}, {σ_7, b, σ_{14}}, {σ_8, a, σ_9}, {σ_8, b, σ_{12}}, {σ_9, a, σ_{10}}, {σ_9, b, σ_{11}}, {σ_{10}, a, σ_{14}}, {σ_{10}, b, σ_4}, {σ_{11}, a, σ_4}, {σ_{11}, b, σ_{14}}, {σ_{12}, a, σ_{13}}, {σ_{12}, b, σ_{14}}, {σ_{13}, a, σ_4}, {σ_{13}, b, σ_{14}}, {σ_{14}, a, σ_{14}}, {σ_{14}, b, σ_{14}}}, {σ_4}];

Solución del ejemplo 7.17

```
Prueba[automata_, n_] := Module[{
L = LenguajeStrings[automata, n, limite -> True],
valorLogico = True}, If[n >= 4,
If[ToString[L] != ToString[Permutations[{a, a, b, b}]],
valorLogico = False]; valorLogico, Print["La longitud debe
ser mayor o igual a 4"]]]
Prueba[A, 20]
```

Solución del ejemplo 7.17

Out[ ] =

True

El True en el **Out[ ]** advierte que las hileras aceptadas por el autómata de longitud menor o igual a 20, están constituidas únicamente por los elementos de `Permutations[{a, a, b, b}]`. En esta prueba no se proporciona la cantidad de elementos del conjunto `LenguajeStrings[A, 20, limite -> True]` al realizarse una comparación directa. También es importante mencionar que el comando `LenguajeCantidad` del paquete **VilCretas**, permite corroborar en el autómata de este ejercicio, una cardinalidad igual a 6 en el conjunto `LenguajeStrings[A, 20, limite -> True]`, al ejecutar `LenguajeCantidad[A, 20, limite -> True]`.

Solución del ejemplo 7.17

Por otra parte, la librería **VilCretas** facilita además, la sentencia `AutomataExactamente` que automatiza el desarrollo de un *ADF* que acepta hileras con "exactamente" cierta cantidad de caracteres, mostrando su diagrama de transición. Por defecto la instrucción almacena el autómata en la variable `G`. El *ADF* diseñado en este ejemplo, se crearía con este comando así:

In[ ] :=

```
AutomataExactamente[{{a, 2}, {b, 2}}]
ComponentesAutomata[G]
```

`ComponentesAutomata[G]` devuelve la 5−tupla asociada con el autómata de estado finito generado por `AutomataExactamente`.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-175.zip

Empleo de la instrucción `LenguajeCantidad`.

Explicación en video

https://youtu.be/VIAfPcPExNE

Uso del comando `AutomataExactamente`.

Explicación en video

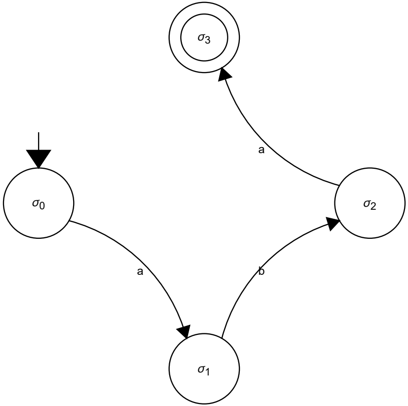
https://youtu.be/9LGayM0Y4ds

Example (7.18)

Desarrolle un autómata de estado finito que acepte cualquier hilera de símbolos de entrada que inicie con ab y termine con ba.

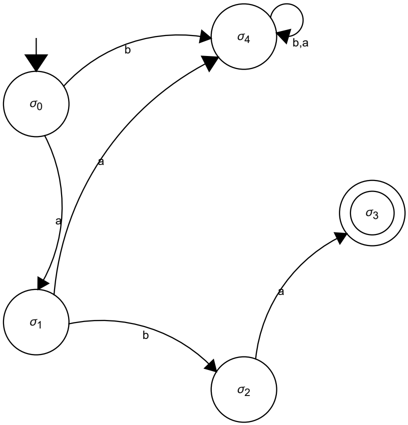
Solución del ejemplo 7.18

La solución propuesta se basa en tomar una sucesión de cuatro estados para recorrer la hilera *aba* que conforma el "string" más pequeño a aceptar por parte del autómata:



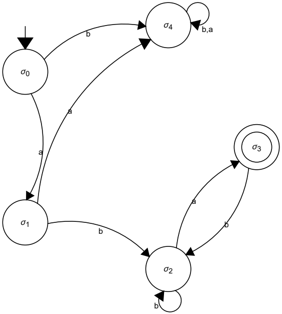
Solución del ejemplo 7.18

Se agregará ahora, un estado no aceptado σ_4 con el objetivo de enviar allí cualquier hilera de símbolos de entrada que no comience con ab :



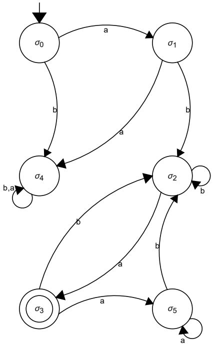
Solución del ejemplo 7.18

Como se desea aceptar "strings" que finalicen en *ba*, el razonamiento de trabajo que prosigue es equivalente al compartido en el ejemplo 21. En el estado σ_2 se incluye un lazo para el símbolo b pues con este ciclo, no se pierde el caracter b heredado del estado anterior. Además, en σ_3 se direcciona la arista con b, precisamente al estado σ_2 que contiene un lazo con ese símbolo:



Solución del ejemplo 7.18

Al no existir a partir del estado σ_2 un ciclo con a, a diferencia de lo que se hizo en el ejemplo 21, es necesario incluir un estado más σ_5, donde se añadirá ese lazo. En el estado σ_5 el lado con b toma la dirección hacia σ_2, garantizándose así, que si la hilera pasa por σ_5 y termina en ba, va a ser aceptada. Luego, el *ADF* finalmente queda de la siguiente manera:



Solución del ejemplo 7.18

Para verificar la correctitud del autómata diseñado, se utilizará una prueba con el propósito de conocer la lista de todas las hileras aceptadas de longitud menor o igual a 20 y por medio de un ciclo, ir comprobando si cada “string” de la lista, satisface las características de este problema: la hilera debe iniciar con *ab* y terminar con *ba*. En *Mathematica*:

In[ ] :=

```
A = Automata[{σ0, σ1, σ2, σ3, σ4, σ5}, {a, b}, σ0,
{{σ0, a, σ1}, {σ0, b, σ4}, {σ1, a, σ4}, {σ1, b, σ2},
{σ2, a, σ3}, {σ2, b, σ2}, {σ3, a, σ5}, {σ3, b, σ2},
{σ4, a, σ4}, {σ4, b, σ4}, {σ5, a, σ5}, {σ5, b, σ2}}, {σ3}];
```

Solución del ejemplo 7.18

```
Prueba[automata_, n_] := Module[{
L = LenguajeStrings[automata, n, limite -> True],
valorLogico = True, i, m}, m = Length[L];
For[i = 1, i <= m, If[ToString[Take[L[[i]], 2]] !=
ToString[{a, b}] || ToString[Take[L[[i]], -2]] !=
ToString[{b, a}], valorLogico = False; Break[]]; i++];
Print[If[valorLogico == False, i, i - 1], "pruebas: ",
valorLogico]]
Prueba[A, 20]
```

Solución del ejemplo 7.18

Out[ ] =

131072 pruebas: True

`Take` facilita la extracción de los dos caracteres iniciales y finales en las 131072 hileras analizadas. Todas ellas cumplieron con las condiciones de aceptación.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-176.zip

Example (7.19)

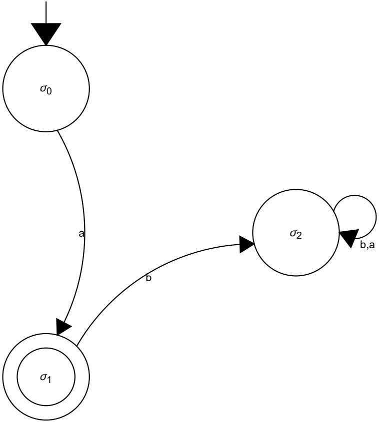
Elabore el diagrama de transición de un *ADF* que acepte cualquier hilera de símbolos de entrada que no inicie con *ab* y no finalice con *bb*.

Solución del ejemplo 7.19

Un mecanismo de resolución en el diseño del autómata solicitado, reside en negar las características descritas para una hilera aceptada. Por ley de *De Morgan*, la negación de "no iniciar con *ab* y no finalizar con *bb*" corresponde a "iniciar con *ab* o finalizar con *bb*". De acuerdo con ello, la lógica de construcción del *ADF* se sustenta en crear un autómata que no acepte hileras donde se inicie con *ab* o se finalice con *bb*.

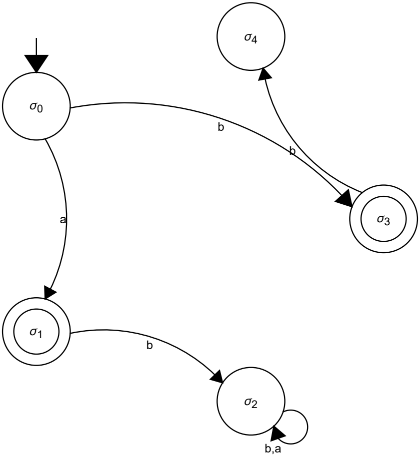
Solución del ejemplo 7.19

Para no aceptar hileras que comienzan con *ab* se requieren tres estados:



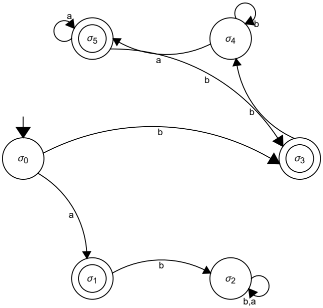
Solución del ejemplo 7.19

Por otra parte, si se desea no aceptar "strings" de entrada que finalicen con *bb* se usan dos estados más:



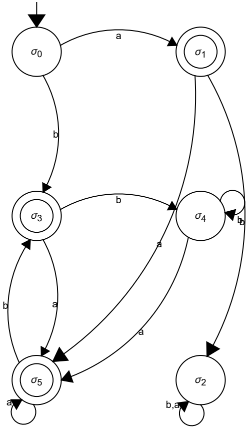
Solución del ejemplo 7.19

En el estado σ_4 se agrega un lazo con b pues si entran más $b's$ en esa posición, la hilera debe quedar no aceptada, al finalizar en bb. En σ_4 con a se necesita conectar a un nuevo estado aceptado σ_5, donde se colocará un lazo para el símbolo a (algo similar a lo realizado en el ejemplo 24). Además, con b se traza una arista de σ_5 a σ_3 con el objetivo de no aceptar cualquier hilera que pase por σ_5 y termine en bb :



Solución del ejemplo 7.19

El diseño se cierra añadiendo dos aristas salientes con el símbolo a en los estados σ_1 y σ_3. Estos lados se conectarán con el estado σ_5 dado que allí, existe un lazo con a :



Solución del ejemplo 7.19

Una prueba de verificación en *Wolfram* se comparte a continuación:

In[ ] :=

```
A = Automata[{σ0, σ1, σ2, σ3, σ4, σ5}, {a, b}, σ0,
{{σ0, a, σ1}, {σ0, b, σ3}, {σ1, a, σ5}, {σ1, b, σ2},
{σ2, a, σ2}, {σ2, b, σ2}, {σ3, a, σ5}, {σ3, b, σ4},
{σ4, a, σ5}, {σ4, b, σ4}, {σ5, a, σ5}, {σ5, b, σ3}},
{σ1, σ3, σ5}];
```

Solución del ejemplo 7.19

```
Prueba[automata_, n_] := Module[{
L = LenguajeStrings[automata, n, limite -> True],
valorLogico = True, i, m}, m = Length[L];
For[i = 1, i <= m, If[Length[L[[i]]] == 2,
If[ToString[L[[i]]] == ToString[{a, b}] ||
ToString[L[[i]]] == ToString[{b, b}], valorLogico = False;
Break[]], If[Length[L[[i]]] >= 3, If[ToString[Take[L[[i]],
2]] == ToString[{a, b}] ||
ToString[Take[L[[i]], -2]] == ToString[{b, b}],
valorLogico = False; Break[]]]]; i++];
Print[If[valorLogico == False, i, i - 1], "pruebas: ",
valorLogico]]
Prueba[A, 20]
```

Solución del ejemplo 7.19

Out[ ] $=$

1179648 pruebas: True

El experimento parte del conjunto de hileras aceptadas de longitud menor o igual a 20 y verifica sobre 1179648 "strings" que cada uno no comienza con *ab* y no termina con *bb*.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-177.zip

Example (7.20)

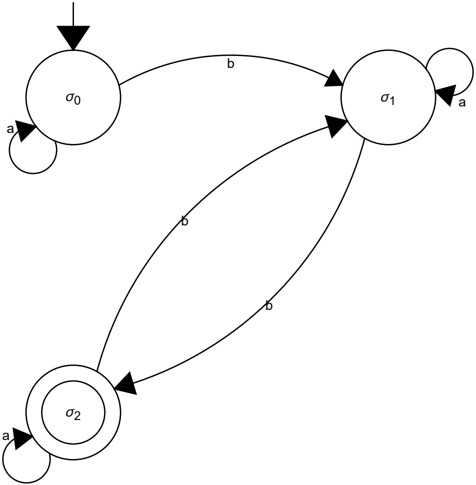
Diseñe el diagrama de transición de un autómata de estado finito que acepte hileras de τ, donde exista un número par de letras b. Se asume $\tau = \{a, b\}$.

Solución del ejemplo 7.20

En este ejercicio, se toman tres estados σ_0, σ_1 y σ_2 para recorrer bb, el "string" más pequeño a ser aceptado por la máquina. En σ_0 se forma un lazo con el símbolo de entrada a (no se avanza hasta que aparezca una b). Si el símbolo es b se continúa al estado σ_1. En σ_1, si la letra es una a se toma un ciclo (las $b's$ no necesariamente estarán juntas, el enunciado no lo demanda) y si es una b, se avanza al único estado aceptado del autómata, representado por σ_2. En σ_2, un símbolo a significaría que se preserva el número par de letras b, por lo tanto, se toma un lazo para aceptar la hilera. Finalmente, si el símbolo de entrada en σ_2 es una b, se tendría una cantidad impar de $b's$, lo cual obligaría a regresar a σ_1.

Solución del ejemplo 7.20

Luego:



Solución del ejemplo 7.20

Un experimento que permite comprobar si el autómata está bien desarrollado es el siguiente:

In[ ] :=

```
A = Automata[{σ0, σ1, σ2}, {a, b}, σ0, {{σ0, a, σ0},
{σ0, b, σ1}, {σ1, a, σ1}, {σ1, b, σ2}, {σ2, a, σ2},
{σ2, b, σ1}}, {σ2}]
```

Solución del ejemplo 7.20

```
Prueba[automata_, n_] := Module[{
L = LenguajeStrings[automata, n, limite -> True],
valorLogico = True, m}, m = Length[L];
If[ToString[Select[Table[Count[L[[i]], b], {i, m}], OddQ[#]
&]] != "{}", valorLogico = False]; Print[m, "pruebas: ",
valorLogico]]
Prueba[A, 20]
```

Solución del ejemplo 7.20

Out[ ] =

1048555 pruebas: True

En este código, el comando `OddQ` de *Mathematica* verifica si existe un entero impar en un vector cuyas componentes constituyen el número de $b's$ para 1048555 "strings" aceptados por el ADF. El vector generado fue igual a vacío, por lo que la prueba está indicando que toda hilera estudiada contuvo siempre una cantidad par de letras b.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-178.zip

- Ya hemos señalado cómo en el diagrama de transición de un autómata de estado finito determinístico (al igual que en una *MEF* general), cada estado tiene una única arista de salida que corresponde a cada uno de los símbolos de entrada de la máquina. No todos los autómatas satisfacen esta propiedad, en algunos casos, los estados no tienen ningún lado de salida asociado a un símbolo de entrada y en otros, hay múltiples aristas. A estos autómatas se les denomina “no determinísticos” y el tema será tratado en la siguiente sección.

Autómatas no determinísticos

Prof. Enrique Vílchez Quesada

Universidad Nacional de Costa Rica

Autómatas no determinísticos

Los autómatas no determinísticos denotados con las siglas *ADFN* son autómatas donde la función de transición de estados (Δ) toma como codominio el conjunto potencia de σ. Esto quiere decir, que una imagen de un par ordenado $(estado, entrada)$ en Δ es un subconjunto de estados de σ, incluyendo a ϕ. Un subconjunto imagen de σ de cardinalidad mayor que uno, indicaría la existencia de un símbolo de entrada para el cual, hay varios estados siguientes. En dicho caso, dada una trayectoria definida por una hilera α de τ, se seleccionaría cualquiera de estos estados para el recorrido posterior, es decir, la ruta en el diagrama de transición del autómata establecida por α deja de ser única. Una imagen de Δ igual a ϕ, significaría que para el símbolo de entrada correspondiente, no hay ninguna arista en el diagrama de transición del *ADFN*. Consideremos la definición de autómata no determinístico.

Definición 7.7. Autómata no determinístico

Definition (7.7)

Un autómata no deteminístico denotado *ADFN* es una 5−tupla $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$, donde: $\Delta : \sigma \times \tau \to P(\sigma)$, siendo $P(\sigma)$ el conjunto potencia de σ, es decir, $P(\sigma)$ contiene todos los subconjuntos del conjunto de estados σ.

Comentario sobre la definición 27

Un autómata no determinístico posee las mismas componentes de otro determinístico según la definición 27, con la diferencia de regresar como imagen en la función de transición Δ: uno, varios o ningún estado. Esta disimilitud propone cambios interesantes en el funcionamiento interno de la máquina, pues podrían existir en el *ADFN* hileras de símbolos de entrada no procesables y de la misma manera, hileras que tengan varias trayectorias en el diagrama de transición. A diferencia de lo enunciado en la definición 27, hay que recordar que en un *ADF* al tener una hilera de símbolos de τ, siempre existirá una trayectoria en el digrafo que lo representa y esta ruta es única. En los autómatas de estado finito no determinísticos, por lo tanto, el diagrama de transición no satisface la propiedad de contener en cada estado, una cantidad de lados de salida igual al número de símbolos de entrada. La cantidad de aristas de salida podría corresponder a la cardinalidad del conjunto τ, un valor mayor o menor, o bien, ser igual a cero.

ADFN en *Wolfram Mathematica*

En *Wolfram Mathematica* el diagrama de transición de un *ADFN* se traza utilizando las instrucciones: `Automata` y `AutomataToDiagrama`.

- El procedimiento es equivalente al explicado en la sección de autómatas de estado finito determinísticos, con la diferencia de que al ingresar la función de estado siguiente como un triplete de un estado, un símbolo de entrada y la imagen de este par ordenado, ésta siempre será un conjunto de estados, es decir, un elemento de $P(\sigma)$. Si el conjunto imagen es vacío se añade en *Mathematica* escribiendo "$\{\}$".

Example (7.21)

Elabore en *Wolfram* el diagrama de transición del autómata de estado finito no determinístico $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$, con $\sigma = \{\sigma_0, \sigma_1, \sigma_2\}$, $\tau = \{a, b, c\}$, $\sigma^* = \sigma_0$, $\widehat{A} = \{\sigma_2\}$ y:

Δ	a	b	c
σ_0	$\{\sigma_1\}$	$\{\sigma_0, \sigma_1\}$	σ
σ_1	$\{\sigma_0\}$	ϕ	$\{\sigma_1, \sigma_2\}$
σ_2	$\{\sigma_2\}$	ϕ	ϕ

Solución del ejemplo 7.21

En el software:

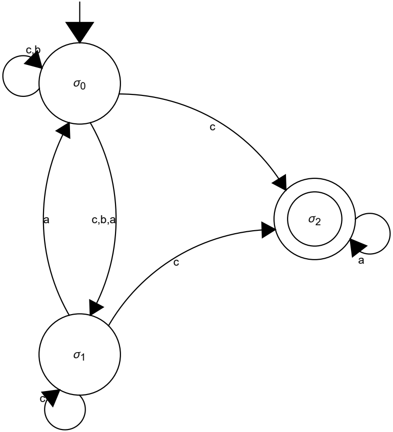
In[ ] :=

```
A = Automata[{σ0, σ1, σ2}, {a, b, c}, σ0, {{σ0, a, {σ1}},
{σ0, b, {σ0, σ1}}, {σ0, c, {σ0, σ1, σ2}}, {σ1, a, {σ0}},
{σ1, b, {}}, {σ1, c, {σ1, σ2}}, {σ2, a, {σ2}}, {σ2, b, {}},
{σ2, c, {}}}, {σ2}]
AutomataToDiagrama[A]
```

Solución del ejemplo 7.21

Out[ ] =

- Automaton -



Solución del ejemplo 7.21

La opción `colores -> True` de `AutomataToDiagrama` mencionada anteriormente, también es válida si los argumentos recibidos conforman un autómata de estado finito no determinístico.

Nota

En este ejercicio, se observa cómo algunas hileras de símbolos de τ, no pueden ser procesadas por la máquina. Esto ocurre por ejemplo en $\alpha_1 = aaab$, donde el recorrido en el diagrama de transición de A, se detiene en la letra b. Al llegar al estado σ_1 con el símbolo de entrada b, el autómata no tiene la capacidad de responder al no existir una arista saliente con el símbolo b.

También, hay "strings" de símbolos de entrada con varias posibilidades de procesamiento. Por ejemplo en $\alpha_2 = bacc$, hay varias rutas en el diagrama de transición, a saber:

$$\sigma_0 \xrightarrow{b} \sigma_0 \xrightarrow{a} \sigma_1 \xrightarrow{c} \sigma_1 \xrightarrow{c} \sigma_1$$
$$\sigma_0 \xrightarrow{b} \sigma_1 \xrightarrow{a} \sigma_0 \xrightarrow{c} \sigma_0 \xrightarrow{c} \sigma_0$$
$$\sigma_0 \xrightarrow{b} \sigma_0 \xrightarrow{a} \sigma_1 \xrightarrow{c} \sigma_1 \xrightarrow{c} \sigma_2$$

Nota

De hecho, existen más caminos para $\alpha_2 = bacc$, sin embargo, con las tres trayectorias ya expuestas se aprecia la multiplicidad de formas de recorrido que una misma hilera de símbolos de τ puede tener en un *ADFN*. Éstos son aspectos comunes en los autómatas de estado finito no determinísticos: la inexistencia de una trayectoria para un "string" de símbolos de entrada o eventualmente, la presencia de muchas rutas.

Solución del ejemplo 7.21

La definición que prosigue formaliza el concepto de hilera aceptada en un autómata de estado finito no determinístico. Al respecto, en un *ADFN* se considera que una hilera de símbolos de entrada es acepta, si existe al menos una trayectoria que finalice en un estado aceptado. La hilera $\alpha_2 = bacc$ se observa que algunas veces sí termina en el estado aceptado σ_2 mientras que el "string" $\alpha_1 = aaab$ nunca lo hace, por lo que se considera a $\alpha_1 = aaab$ no aceptada y a $\alpha_2 = bacc$ aceptada en el autómata no determinístico de este ejemplo.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-179.zip

Definición 7.8. Hilera aceptada en *ADFN*

Como ya señaló, en un *ADFN* el concepto de hilera aceptada debe ser modificado, pues para un "string" de símbolos de entrada pueden existir varias trayectorias o ninguna en su diagrama de transición.

Definition (7.8)

Sea $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$ un autómata de estado finito no determinístico y $\alpha = x_1 x_2 \ldots x_n$ una hilera de símbolos de entrada. Se dice que α es aceptada si existe al menos una ruta obtenida a través de la función Δ, donde α finalice en un estado aceptado.

Consideremos algunos ejemplos de uso de la definición 29.

Example (7.22)

Construya el diagrama de transición del autómata no determinístico $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$, con $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$, $\tau = \{a, b\}$, $\sigma^* = \sigma_0$, $\widehat{A} = \{\sigma_1, \sigma_3\}$ y:

Δ	a	b
σ_0	$\{\sigma_1, \sigma_2\}$	$\{\sigma_0, \sigma_3\}$
σ_1	$\{\sigma_1, \sigma_3\}$	ϕ
σ_2	$\{\sigma_0, \sigma_3\}$	$\{\sigma_1\}$
σ_3	$\{\sigma_2\}$	$\{\sigma_2, \sigma_3\}$

Determine además, si las siguientes hileras de símbolos de entrada $\alpha_1 = bbaabaabaaaabaa$ y $\alpha_2 = babbbbaaabbabbb$ son aceptadas.

Solución del ejemplo 7.22

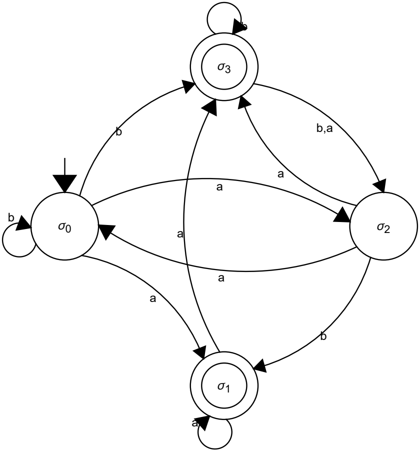
El diagrama de transición del autómata A se genera así:

In[ ] :=

```
A = Automata[{σ0, σ1, σ2, σ3}, {a, b}, σ0,
{{σ0, a, {σ1, σ2}}, {σ0, b, {σ0, σ3}}, {σ1, a, {σ1, σ3}},
{σ1, b, {}}, {σ2, a, {σ0, σ3}}, {σ2, b, {σ1}}, {σ3, a, {σ2}},
{σ3, b, {σ2, σ3}}}, {σ1, σ3}];
AutomataToDiagrama[A]
```

Solución del ejemplo 7.22

Out[ ] =



Solución del ejemplo 7.22

Ahora, al tomar el "string" $\alpha_1 = bbaabaabaaaabaa$ se observa que es aceptado por el autómata al existir al menos una ruta que finaliza en un estado aceptado:

$$\sigma_0 \xrightarrow{b} \sigma_3 \xrightarrow{b} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{b} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{b}$$
$$\sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_3 \xrightarrow{b} \sigma_3 \xrightarrow{a} \sigma_2 \xrightarrow{a} \boxed{\sigma_3}$$

Por otro lado, en $\alpha_2 = babbbbaaabbabbb$ se concluye que la hilera no es aceptada, al realizar distintas pruebas de recorrido en el diagrama de transición del autómata y notar que no es posible encontrar un camino que finalice en los estados aceptados σ_1, o bien, σ_3. Se insta al estudiante a comprobar distintas rutas.

Nota

Formalmente, para justificar de manera precisa la no aceptación de un "string" en un *ADFN* se deben hallar todas las posibles trayectorias de la hilera en el autómata y comprobar que ninguna termina en un estado aceptado. Pese a ello, esta tarea es normalmente muy exhaustiva, por lo que, se propondrá más adelante el teorema 32 que permite encontrar un autómata determinístico equivalente, por medio del cual el procesamiento de una hilera de τ se simplifica, al ser el recorrido único.

Solución del ejemplo 7.22

El comando `StringAceptadaQ` explicado en capítulos anteriores, funciona también, cuando el autómata recibido es no determinístico. Al emplearlo en el presente ejercicio:

In[ ] :=

α_1 = {b, b, a, a, b, a, a, b, a, a, a, a, b, a, a};
StringAceptadaQ[A, α_1]
α_2 = {b, a, b, b, b, b, a, a, a, b, b, a, b, b, b};
StringAceptadaQ[A, α_2]

Out[ ] =

True
False

Solución del ejemplo 7.22

El **Out[ ]** True indica que α_1 es un "string" aceptado y la salida False muestra la no aceptación de la hilera α_2.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-180.zip

Example (7.23)

Elabore el diagrama de transición del *ADFN* $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$, con $\sigma = \{\sigma_0, \sigma_1, \sigma_2\}$, $\tau = \{a, b, c, d\}$, $\sigma^* = \sigma_2$, $\widehat{A} = \{\sigma_0, \sigma_1\}$ y:

Δ	a	b	c	d
σ_0	σ	$\{\sigma_1, \sigma_2\}$	$\{\sigma_0\}$	σ
σ_1	ϕ	$\{\sigma_0, \sigma_2\}$	$\{\sigma_0, \sigma_1\}$	ϕ
σ_2	$\{\sigma_2\}$	$\{\sigma_0, \sigma_2\}$	$\{\sigma_0\}$	$\{\sigma_1\}$

Establezca si las hileras de símbolos de entrada $\alpha_1 = dcacbcadcbadbbaacd$ y $\alpha_2 = bbbaaadcdacccadbaa$ son aceptadas.

Solución del ejemplo 7.23

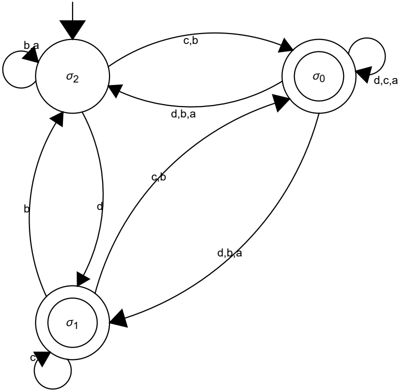
El digrafo que representa el autómata A se construye en *Wolfram* como sigue:

In[ ] :=

```
A = Automata[{σ0, σ1, σ2}, {a, b, c, d}, σ2,
{{σ0, a, {σ0, σ1, σ2}}, {σ0, b, {σ1, σ2}}, {σ0, c, {σ0}},
{σ0, d, {σ0, σ1, σ2}}, {σ1, a, {}}, {σ1, b, {σ0, σ2}},
{σ1, c, {σ0, σ1}}, {σ1, d, {}}, {σ2, a, {σ2}},
{σ2, b, {σ0, σ2}}, {σ2, c, {σ0}}, {σ2, d, {σ1}}}, {σ0, σ1}];
AutomataToDiagrama[A, forma -> "rectangular"]
```

Solución del ejemplo 7.23

Out[ ] =



En esta figura se ha otorgado al diagrama una forma rectangular.

Solución del ejemplo 7.23

Los "strings" de símbolos de entrada $\alpha_1 = dcacbcadcbadbbaacd$ y $\alpha_2 = bbbaaadcdacccadbaa$ son aceptados, al existir trayectorias que terminan en estados aceptados. En α_1 :

$$\sigma_2 \xrightarrow{d} \sigma_1 \xrightarrow{c} \sigma_0 \xrightarrow{a} \sigma_0 \xrightarrow{c} \sigma_0 \xrightarrow{b} \sigma_1 \xrightarrow{c} \sigma_0 \xrightarrow{a} \sigma_0 \xrightarrow{d} \sigma_0 \xrightarrow{c}$$
$$\sigma_0 \xrightarrow{b} \sigma_2 \xrightarrow{a} \sigma_2 \xrightarrow{d} \sigma_1 \xrightarrow{b} \sigma_0 \xrightarrow{b} \sigma_2 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_2 \xrightarrow{c} \sigma_0 \xrightarrow{d} \boxed{\sigma_0}$$

y en α_2 :

$$\sigma_2 \xrightarrow{b} \sigma_2 \xrightarrow{b} \sigma_2 \xrightarrow{b} \sigma_2 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_2 \xrightarrow{a} \sigma_2 \xrightarrow{d} \sigma_1 \xrightarrow{c} \sigma_0 \xrightarrow{d}$$
$$\sigma_0 \xrightarrow{a} \sigma_0 \xrightarrow{c} \sigma_0 \xrightarrow{c} \sigma_0 \xrightarrow{c} \sigma_0 \xrightarrow{a} \sigma_2 \xrightarrow{d} \sigma_1 \xrightarrow{b} \sigma_0 \xrightarrow{a} \sigma_0 \xrightarrow{a} \boxed{\sigma_1}$$

Solución del ejemplo 7.23

En *Mathematica*, la aceptación de los "strings" α_1 y α_2 se puede comprobar así:

In[ ] :=

```
α1 = {d, c, a, c, b, c, a, d, c, b, a, d, b, b, a, a, c, d};
StringAceptadaQ[A, α1]
α2 = {b, b, b, a, a, a, d, c, d, a, c, c, c, a, d, b, a, a};
StringAceptadaQ[A, α2]
```

Out[ ] =

True
True

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-181.zip

- Todo autómata de estado finito no determinístico es equivalente a otro de caracter determinístico. Esta equivalencia es de utilidad práctica por la dificultad asociada a la multiplicidad de rutas que podría presentar una hilera de símbolos de entrada, al querer determinar su aceptación en el *ADFN*. El teorema próximo da garantía de esta propiedad.

Teorema 7.1 Autómata determinístico equivalente

Theorem (7.1)

Sea $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$ *un ADFN entonces el autómata de estado finito determinístico* $A' = \left(P(\sigma), \tau, \{\sigma^*\}, \Delta', \widehat{A'}\right)$ *es equivalente a A, con:*

1. $P(\sigma)$ *el conjunto potencia de* σ.
2. $\Delta' : P(\sigma) \times \tau \to P(\sigma)$, *tal que:* $\Delta'(\phi, x) = \phi$ *y* $\Delta'(B, x) = \bigcup_{\mu \in B} \Delta(\mu, x), \forall x,\ x \in \tau,\ \forall B,\ B \in P(\sigma)$.
3. $\widehat{A'} = \left\{B \in P(\sigma) \mid \widehat{A} \cap B \neq \phi\right\}$.

Comentario sobre el teorema 32

El teorema 32 expone un método para encontrar un *ADF* A' equivalente a un autómata de estado finito no determinístico A dado. El algoritmo descrito inicia tomando el conjunto de estados del *ADF* como el conjunto de todos los subconjuntos de σ. Los símbolos de entrada son iguales en ambas máquinas. El estado inicial del nuevo autómata A' es el conjunto formado por el estado inicial del *ADFN* A. La función estado siguiente en A', toma un subconjunto B de σ y un símbolo de entrada x y, para cada elemento contenido en B, se calcula su imagen en la función Δ de A, la unión de estos conjuntos da como resultado la imagen de Δ' en el par ordenado (B, x). Por otra parte, si el subconjunto B es igual ϕ, por el teorema su imagen será igual al conjunto vacío. Finalmente, los estados aceptados en A' están constituidos por todos los subcojuntos de σ, que poseen al menos un estado aceptado de A.

El empleo del teorema 32 se aclarará mediante el desarrollo de algunos ejercicios.

Example (7.24)

Considere el autómata de estado finito no determinístico del ejemplo 28, encuentre un ADF A' equivalente.

Solución del ejemplo 7.24

De acuerdo con el teorema 32, se tiene:

- Los estados del nuevo autómata A' son:

$$\begin{array}{ll} \mu_0 = \{\} & \mu_4 = \{\sigma_0, \sigma_1\} \\ \mu_1 = \{\sigma_0\} & \mu_5 = \{\sigma_0, \sigma_2\} \\ \mu_2 = \{\sigma_1\} & \mu_6 = \{\sigma_1, \sigma_2\} \\ \mu_3 = \{\sigma_2\} & \mu_7 = \{\sigma_0, \sigma_1, \sigma_2\} \end{array}$$

Estos estados se han etiquetado con $\mu' s$ para facilitar la exposición de las componentes restantes del autómata determinístico buscado. El alumno debe notar que el conjunto potencia de σ, $P(\sigma)$, es tal que:

$$P(\sigma) = \{\mu_0, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6, \mu_7\}$$

$P(\sigma)$ posee una cantidad de elementos igual a 8 por la fórmula $2^3 = 8$, siendo 3 la cardinalidad de $\sigma = \{\sigma_0, \sigma_1, \sigma_2\}$.

Solución del ejemplo 7.24

La instrucción `Subsets` de *Mathematica* es de utilidad para retornar el conjunto potencia $P(\sigma)$, en este caso:

In[ ] :=

`Subsets[`$\{\sigma_0,\ \sigma_1,\ \sigma_2\}$`]`

Out[ ] =

$\{\{\}, \{\sigma_0\}, \{\sigma_1\}, \{\sigma_2\}, \{\sigma_0, \sigma_1\}, \{\sigma_0, \sigma_2\}, \{\sigma_1, \sigma_2\}, \{\sigma_0, \sigma_1, \sigma_2\}\}$

Solución del ejemplo 7.24

- Los símbolos de entrada de A' son: $\{a, b, c\}$.
- El estado inicial del autómata equivalente es: $\{\sigma_0\} = \mu_1$.
- Los estados aceptados de A' son: $\widehat{A'} = \{\mu_3, \mu_5, \mu_6, \mu_7\}$. Aquí es esencial indicar que μ_3, μ_5, μ_6 y μ_7 se consideran aceptados en el *ADF*, pues ellos contienen el estado aceptado σ_2 del *ADFN* del enunciado.

Solución del ejemplo 7.24

- Con respecto a la función de transición Δ' del autómata de estado finito determinístico A', se concluye:

$\Delta'(\mu_0, a) = \{\} = \mu_0$
$\Delta'(\mu_0, b) = \{\} = \mu_0$
$\Delta'(\mu_0, c) = \{\} = \mu_0$
$\Delta'(\mu_1, a) = \Delta(\sigma_0, a) = \{\sigma_1\} = \mu_2$
$\Delta'(\mu_1, b) = \Delta(\sigma_0, b) = \{\sigma_0, \sigma_1\} = \mu_4$
$\Delta'(\mu_1, c) = \Delta(\sigma_0, c) = \{\sigma_0, \sigma_1, \sigma_2\} = \mu_7$
$\Delta'(\mu_2, a) = \Delta(\sigma_1, a) = \{\sigma_0\} = \mu_1$
$\Delta'(\mu_2, b) = \Delta(\sigma_1, b) = \{\} = \mu_0$
$\Delta'(\mu_2, c) = \Delta(\sigma_1, c) = \{\sigma_1, \sigma_2\} = \mu_6$
$\Delta'(\mu_3, a) = \Delta(\sigma_2, a) = \{\sigma_2\} = \mu_3$
$\Delta'(\mu_3, b) = \Delta(\sigma_2, b) = \{\} = \mu_0$
$\Delta'(\mu_3, c) = \Delta(\sigma_2, c) = \{\} = \mu_0$
$\Delta'(\mu_4, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) = \{\sigma_1\} \cup \{\sigma_0\} = \{\sigma_0, \sigma_1\} = \mu_4$

Solución del ejemplo 7.24

$\Delta'(\mu_4, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) = \{\sigma_0, \sigma_1\} \cup \{\} = \{\sigma_0, \sigma_1\} = \mu_4$
$\Delta'(\mu_4, c) = \Delta(\sigma_0, c) \cup \Delta(\sigma_1, c) = \{\sigma_0, \sigma_1, \sigma_2\} \cup \{\sigma_1, \sigma_2\} =$
$\{\sigma_0, \sigma_1, \sigma_2\} = \mu_7$
$\Delta'(\mu_5, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_2, a) = \{\sigma_1\} \cup \{\sigma_2\} = \{\sigma_1, \sigma_2\} = \mu_6$
$\Delta'(\mu_5, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_2, b) = \{\sigma_0, \sigma_1\} \cup \{\} = \{\sigma_0, \sigma_1\} = \mu_4$
$\Delta'(\mu_5, c) = \Delta(\sigma_0, c) \cup \Delta(\sigma_2, c) = \{\sigma_0, \sigma_1, \sigma_2\} \cup \{\} = \{\sigma_0, \sigma_1, \sigma_2\} = \mu_7$
$\Delta'(\mu_6, a) = \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) = \{\sigma_0\} \cup \{\sigma_2\} = \{\sigma_0, \sigma_2\} = \mu_5$
$\Delta'(\mu_6, b) = \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) = \{\} \cup \{\} = \{\} = \mu_0$
$\Delta'(\mu_6, c) = \Delta(\sigma_1, c) \cup \Delta(\sigma_2, c) = \{\sigma_1, \sigma_2\} \cup \{\} = \{\sigma_1, \sigma_2\} = \mu_6$
$\Delta'(\mu_7, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) = \{\sigma_1\} \cup \{\sigma_0\} \cup \{\sigma_2\} =$
$\{\sigma_0, \sigma_1, \sigma_2\} = \mu_7$
$\Delta'(\mu_7, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) = \{\sigma_0, \sigma_1\} \cup \{\} \cup \{\} =$
$\{\sigma_0, \sigma_1\} = \mu_4$
$\Delta'(\mu_7, c) = \Delta(\sigma_0, c) \cup \Delta(\sigma_1, c) \cup \Delta(\sigma_2, c) =$
$\{\sigma_0, \sigma_1, \sigma_2\} \cup \{\sigma_1, \sigma_2\} \cup \{\} = \{\sigma_0, \sigma_1, \sigma_2\} = \mu_7$

Solución del ejemplo 7.24

En resumen:

Δ'	a	b	c
μ_0	μ_0	μ_0	μ_0
μ_1	μ_2	μ_4	μ_7
μ_2	μ_1	μ_0	μ_6
μ_3	μ_3	μ_0	μ_0
μ_4	μ_4	μ_4	μ_7
μ_5	μ_6	μ_4	μ_7
μ_6	μ_5	μ_0	μ_6
μ_7	μ_7	μ_4	μ_7

(9)

Solución del ejemplo 7.24

Para finalizar el proceso, se mostrará el diagrama de transición del autómata A'. En este tipo de ejercicios de uso del teorema 32 es usual que la cantidad de estados y aristas del diagrama sea considerablemente grande. Aquí, el digrafo respectivo contiene ocho estados y de cada uno salen tres lados, uno con el símbolo a, otro con el caracter b y un tercero con el símbolo c.

Nota

En algunas ocasiones es posible simplificar el diagrama de transición de A', incluyendo únicamente aquellos estados por los que se puede transitar partiendo del estado inicial. Si existen estados sobre los cuales es imposible su visitación, resulta natural excluirlos del digrafo. De hecho, esta lógica es aplicable sobre cualquier diagrama de transición, sin embargo, antes no había sido necesario simplificar o reducir esta forma de representación, pues en ejemplos previos el tamaño del digrafo fue relativamente pequeño.

Solución del ejemplo 7.24

El diagrama de A' se reduce o se simplifica comenzando en el estado inicial μ_1. En la tabla 9, μ_1 toma tres tipos de direcciones hacia μ_2, μ_4 y μ_7. En μ_2 según 9, se logran visitar los estados μ_0, μ_1 y μ_6. En μ_4 es posible transitar por μ_4 y μ_7. En μ_7 de acuerdo con 9, se pueden visitar los estados μ_4 y μ_7. En μ_0 hay un único estado siguiente que vuelve a ser μ_0. En μ_6 es factible transitar por μ_0, μ_5 y μ_6. En μ_5 la visitación ocurre en μ_4, μ_6 y μ_7. Finalmente, se comprueba mediante la tabla 9, que es inasequible pasar por el estado μ_3. De esta manera, una versión más simplificada o reducida del diagrama de transición, se obtiene al eliminar μ_3 y todas sus aristas de salida vinculadas.

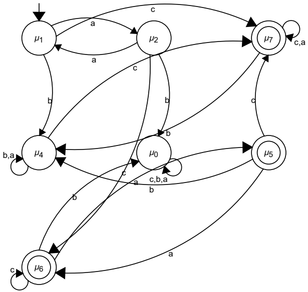
Solución del ejemplo 7.24

Como estrategia de análisis es viable en la tabla 9 marcar con una flecha los estados sobre los cuáles es factible pasar, señalando además, el estado inicial:

Δ'	a	b	c
$\rightarrow \mu_0$	μ_0	μ_0	μ_0
$\boxed{\mu_1}$	μ_2	μ_4	μ_7
$\rightarrow \mu_2$	μ_1	μ_0	μ_6
μ_3	μ_3	μ_0	μ_0
$\rightarrow \mu_4$	μ_4	μ_4	μ_7
$\rightarrow \mu_5$	μ_6	μ_4	μ_7
$\rightarrow \mu_6$	μ_5	μ_0	μ_6
$\rightarrow \mu_7$	μ_7	μ_4	μ_7

Solución del ejemplo 7.24

Como el alumno preverá, dependiendo del autómata A' su diagrama de transición podría no poderse reducir o simplificar. En este ejercicio, la reducción ha consistido en suprimir únicamente un estado (μ_3), sin embargo, en otros ejemplos cabe también la posibilidad de tener un ahorro mayor de estados y lados salientes. Luego, el diagrama de transición simplificado de A' corresponde a:



Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-182.zip

Example (7.25)

Tomando como base el autómata no determinístico $A = \left(\sigma, \tau, \sigma^*, \Delta, \widehat{A}\right)$ dado, determine un ADF A' equivalente y construya su diagrama de transición reducido, con $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$, $\tau = \{a, b\}$, $\sigma^* = \sigma_0$, $\widehat{A} = \{\sigma_1\}$ y:

Δ	a	b
σ_0	$\{\sigma_1\}$	$\{\sigma_3\}$
σ_1	$\{\sigma_1, \sigma_2\}$	$\{\sigma_3\}$
σ_2	ϕ	$\{\sigma_1, \sigma_2, \sigma_3\}$
σ_3	ϕ	ϕ

Solución del ejemplo 7.25

Usando el teorema 32:

- Los estados del nuevo autómata A' son:

$$
\begin{array}{ll}
\mu_0 = \{\} & \mu_8 = \{\sigma_1, \sigma_2\} \\
\mu_1 = \{\sigma_0\} & \mu_9 = \{\sigma_1, \sigma_3\} \\
\mu_2 = \{\sigma_1\} & \mu_{10} = \{\sigma_2, \sigma_3\} \\
\mu_3 = \{\sigma_2\} & \mu_{11} = \{\sigma_0, \sigma_1, \sigma_2\} \\
\mu_4 = \{\sigma_3\} & \mu_{12} = \{\sigma_0, \sigma_1, \sigma_3\} \\
\mu_5 = \{\sigma_0, \sigma_1\} & \mu_{13} = \{\sigma_0, \sigma_2, \sigma_3\} \\
\mu_6 = \{\sigma_0, \sigma_2\} & \mu_{14} = \{\sigma_1, \sigma_2, \sigma_3\} \\
\mu_7 = \{\sigma_0, \sigma_3\} & \mu_{15} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}
\end{array}
$$

Los $\mu's$ anteriores se logran verificar en *Wolfram*, ejecutando `Subsets[{`σ_0`, `σ_1`, `σ_2`, `σ_3`}]`.

Solución del ejemplo 7.25

- Los símbolos de entrada de A' corresponden a: $\{a, b\}$.
- El estado inicial del autómata equivalente es: $\{\sigma_0\} = \mu_1$.
- Los estados aceptados de A' son: $\widehat{A'} = \{\mu_2, \mu_5, \mu_8, \mu_9, \mu_{11}, \mu_{12}, \mu_{14}, \mu_{15}\}$. Los $\mu's$ señalados en el conjunto $\widehat{A'}$ tienen la característica de contener a σ_1, el único estado aceptado del autómata A.

Solución del ejemplo 7.25

- La función de transición Δ' de A' es tal que:

$\Delta'(\mu_0, a) = \{\} = \mu_0$
$\Delta'(\mu_0, b) = \{\} = \mu_0$
$\Delta'(\mu_1, a) = \Delta(\sigma_0, a) = \{\sigma_1\} = \mu_2$
$\Delta'(\mu_1, b) = \Delta(\sigma_0, b) = \{\sigma_3\} = \mu_4$
$\Delta'(\mu_2, a) = \Delta(\sigma_1, a) = \{\sigma_1, \sigma_2\} = \mu_8$
$\Delta'(\mu_2, b) = \Delta(\sigma_1, b) = \{\sigma_3\} = \mu_4$
$\Delta'(\mu_3, a) = \Delta(\sigma_2, a) = \{\} = \mu_0$
$\Delta'(\mu_3, b) = \Delta(\sigma_2, b) = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$
$\Delta'(\mu_4, a) = \Delta(\sigma_3, a) = \{\} = \mu_0$
$\Delta'(\mu_4, b) = \Delta(\sigma_3, b) = \{\} = \mu_0$
$\Delta'(\mu_5, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) = \{\sigma_1\} \cup \{\sigma_1, \sigma_2\} = \{\sigma_1, \sigma_2\} = \mu_8$
$\Delta'(\mu_5, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) = \{\sigma_3\} \cup \{\sigma_3\} = \{\sigma_3\} = \mu_4$
$\Delta'(\mu_6, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_2, a) = \{\sigma_1\} \cup \{\} = \{\sigma_1\} = \mu_2$

Solución del ejemplo 7.25

$\Delta'(\mu_6, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_2, b) = \{\sigma_3\} \cup \{\sigma_1, \sigma_2, \sigma_3\} = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$

$\Delta'(\mu_7, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_3, a) = \{\sigma_1\} \cup \{\} = \{\sigma_1\} = \mu_2$

$\Delta'(\mu_7, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_3, b) = \{\sigma_3\} \cup \{\} = \{\sigma_3\} = \mu_4$

$\Delta'(\mu_8, a) = \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) = \{\sigma_1, \sigma_2\} \cup \{\} = \{\sigma_1, \sigma_2\} = \mu_8$

$\Delta'(\mu_8, b) = \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) = \{\sigma_3\} \cup \{\sigma_1, \sigma_2, \sigma_3\} = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$

$\Delta'(\mu_9, a) = \Delta(\sigma_1, a) \cup \Delta(\sigma_3, a) = \{\sigma_1, \sigma_2\} \cup \{\} = \{\sigma_1, \sigma_2\} = \mu_8$

$\Delta'(\mu_9, b) = \Delta(\sigma_1, b) \cup \Delta(\sigma_3, b) = \{\sigma_3\} \cup \{\} = \{\sigma_3\} = \mu_4$

$\Delta'(\mu_{10}, a) = \Delta(\sigma_2, a) \cup \Delta(\sigma_3, a) = \{\} \cup \{\} = \{\} = \mu_0$

$\Delta'(\mu_{10}, b) = \Delta(\sigma_2, b) \cup \Delta(\sigma_3, b) = \{\sigma_1, \sigma_2, \sigma_3\} \cup \{\} = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$

$\Delta'(\mu_{11}, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) = \{\sigma_1\} \cup \{\sigma_1, \sigma_2\} \cup \{\} = \{\sigma_1, \sigma_2\} = \mu_8$

Solución del ejemplo 7.25

$\Delta'(\mu_{11}, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) =$
$\{\sigma_3\} \cup \{\sigma_3\} \cup \{\sigma_1, \sigma_2, \sigma_3\} = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$
$\Delta'(\mu_{12}, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) \cup \Delta(\sigma_3, a) = \{\sigma_1\} \cup \{\sigma_1, \sigma_2\} \cup \{\} =$
$\{\sigma_1, \sigma_2\} = \mu_8$
$\Delta'(\mu_{12}, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) \cup \Delta(\sigma_3, b) = \{\sigma_3\} \cup \{\sigma_3\} \cup \{\} =$
$\{\sigma_3\} = \mu_4$
$\Delta'(\mu_{13}, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_2, a) \cup \Delta(\sigma_3, a) = \{\sigma_1\} \cup \{\} \cup \{\} =$
$\{\sigma_1\} = \mu_2$
$\Delta'(\mu_{13}, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_2, b) \cup \Delta(\sigma_3, b) = \{\sigma_3\} \cup \{\sigma_1, \sigma_2, \sigma_3\} \cup \{\} =$
$\{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$
$\Delta'(\mu_{14}, a) = \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) \cup \Delta(\sigma_3, a) = \{\sigma_1, \sigma_2\} \cup \{\} \cup \{\} =$
$\{\sigma_1, \sigma_2\} = \mu_8$
$\Delta'(\mu_{14}, b) = \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) \cup \Delta(\sigma_3, b) = \{\sigma_3\} \cup \{\sigma_1, \sigma_2, \sigma_3\} \cup \{\} =$
$\{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$

Solución del ejemplo 7.25

$\Delta'(\mu_{15}, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) \cup \Delta(\sigma_3, a) =$
$\{\sigma_1\} \cup \{\sigma_1, \sigma_2\} \cup \{\} \cup \{\} = \{\sigma_1, \sigma_2\} = \mu_8$
$\Delta'(\mu_{15}, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) \cup \Delta(\sigma_3, b) =$
$\{\sigma_3\} \cup \{\sigma_3\} \cup \{\sigma_1, \sigma_2, \sigma_3\} \cup \{\} = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$
En resumen:

Δ'	a	b
μ_0	μ_0	μ_0
μ_1	μ_2	μ_4
μ_2	μ_8	μ_4
μ_3	μ_0	μ_{14}
μ_4	μ_0	μ_0
μ_5	μ_8	μ_4
μ_6	μ_2	μ_{14}
μ_7	μ_2	μ_4

Δ'	a	b
μ_8	μ_8	μ_{14}
μ_9	μ_8	μ_4
μ_{10}	μ_0	μ_{14}
μ_{11}	μ_8	μ_{14}
μ_{12}	μ_8	μ_4
μ_{13}	μ_2	μ_{14}
μ_{14}	μ_8	μ_{14}
μ_{15}	μ_8	μ_{14}

Solución del ejemplo 7.25

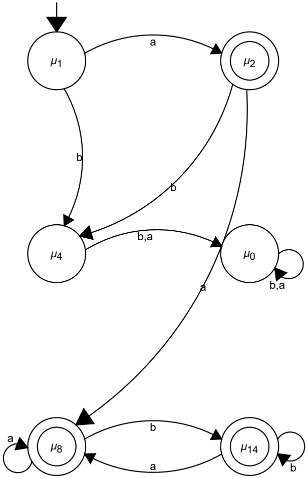
Por otra parte, al analizar comenzando en el estado inicial μ_1, los posibles estados de visitación con el objetivo de simplificar el diagrama de transición de A', se infiere:

Δ	a	b
$\to \mu_0$	μ_0	μ_0
$\boxed{\mu_1}$	μ_2	μ_4
$\to \mu_2$	μ_8	μ_4
μ_3	μ_0	μ_{14}
$\to \mu_4$	μ_0	μ_0
μ_5	μ_8	μ_4
μ_6	μ_2	μ_{14}
μ_7	μ_2	μ_4

Δ	a	b
$\to \mu_8$	μ_8	μ_{14}
μ_9	μ_8	μ_4
μ_{10}	μ_0	μ_{14}
μ_{11}	μ_8	μ_{14}
μ_{12}	μ_8	μ_4
μ_{13}	μ_2	μ_{14}
$\to \mu_{14}$	μ_8	μ_{14}
μ_{15}	μ_8	μ_{14}

Solución del ejemplo 7.25

Luego, el digrafo reducido del nuevo autómata es:



Solución del ejemplo 7.25

En este ejemplo, la simplificación ha permitido eliminar diez estados por los cuáles nunca se pasará iniciando en μ_1 y sus aristas de salida.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-183.zip

Example (7.26)

Sea el *ADFN* del ejemplo 30, halle un autómata determinístico equivalente A' y elabore su diagrama simplificado.

Solución del ejemplo 7.26

Recurriendo al teorema 32:

- Los estados de A' son:

$$
\begin{array}{ll}
\mu_0 = \{\} & \mu_8 = \{\sigma_1, \sigma_2\} \\
\mu_1 = \{\sigma_0\} & \mu_9 = \{\sigma_1, \sigma_3\} \\
\mu_2 = \{\sigma_1\} & \mu_{10} = \{\sigma_2, \sigma_3\} \\
\mu_3 = \{\sigma_2\} & \mu_{11} = \{\sigma_0, \sigma_1, \sigma_2\} \\
\mu_4 = \{\sigma_3\} & \mu_{12} = \{\sigma_0, \sigma_1, \sigma_3\} \\
\mu_5 = \{\sigma_0, \sigma_1\} & \mu_{13} = \{\sigma_0, \sigma_2, \sigma_3\} \\
\mu_6 = \{\sigma_0, \sigma_2\} & \mu_{14} = \{\sigma_1, \sigma_2, \sigma_3\} \\
\mu_7 = \{\sigma_0, \sigma_3\} & \mu_{15} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}
\end{array}
$$

Solución del ejemplo 7.26

- Los símbolos de entrada del nuevo autómata corresponden a: $\{a, b\}$.
- El estado inicial de A' es: $\{\sigma_0\} = \mu_1$.
- Los estados aceptados de A' son: $\widehat{A'} = \{\mu_2, \mu_4, \mu_5, \mu_7, \mu_8, \mu_9, \mu_{10}, \mu_{11}, \mu_{12}, \mu_{13}, \mu_{14}, \mu_{15}\}$. Los $\mu's$ del conjunto $\widehat{A'}$ se caracterizan por contener a σ_1 o σ_3 (uno de ellos o ambos), siendo σ_1 y σ_3 los estados aceptados de A.

Solución del ejemplo 7.26

- La función de transición Δ' del autómata equivalente, satisface:

$\Delta'(\mu_0, a) = \{\} = \mu_0$
$\Delta'(\mu_0, b) = \{\} = \mu_0$
$\Delta'(\mu_1, a) = \Delta(\sigma_0, a) = \{\sigma_1, \sigma_2\} = \mu_8$
$\Delta'(\mu_1, b) = \Delta(\sigma_0, b) = \{\sigma_0, \sigma_3\} = \mu_7$
$\Delta'(\mu_2, a) = \Delta(\sigma_1, a) = \{\sigma_1, \sigma_3\} = \mu_9$
$\Delta'(\mu_2, b) = \Delta(\sigma_1, b) = \{\} = \mu_0$
$\Delta'(\mu_3, a) = \Delta(\sigma_2, a) = \{\sigma_0, \sigma_3\} = \mu_7$
$\Delta'(\mu_3, b) = \Delta(\sigma_2, b) = \{\sigma_1\} = \mu_2$
$\Delta'(\mu_4, a) = \Delta(\sigma_3, a) = \{\sigma_2\} = \mu_3$
$\Delta'(\mu_4, b) = \Delta(\sigma_3, b) = \{\sigma_2, \sigma_3\} = \mu_{10}$
$\Delta'(\mu_5, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) = \{\sigma_1, \sigma_2\} \cup \{\sigma_1, \sigma_3\} =$
$\{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$
$\Delta'(\mu_5, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) = \{\sigma_0, \sigma_3\} \cup \{\} = \{\sigma_0, \sigma_3\} = \mu_7$

Solución del ejemplo 7.26

$\Delta'(\mu_6, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_2, a) = \{\sigma_1, \sigma_2\} \cup \{\sigma_0, \sigma_3\} =$
$\{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \mu_{15}$
$\Delta'(\mu_6, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_2, b) = \{\sigma_0, \sigma_3\} \cup \{\sigma_1\} = \{\sigma_0, \sigma_1, \sigma_3\} = \mu_{12}$
$\Delta'(\mu_7, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_3, a) = \{\sigma_1, \sigma_2\} \cup \{\sigma_2\} = \{\sigma_1, \sigma_2\} = \mu_8$
$\Delta'(\mu_7, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_3, b) = \{\sigma_0, \sigma_3\} \cup \{\sigma_2, \sigma_3\} =$
$\{\sigma_0, \sigma_2, \sigma_3\} = \mu_{13}$
$\Delta'(\mu_8, a) = \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) = \{\sigma_1, \sigma_3\} \cup \{\sigma_0, \sigma_3\} =$
$\{\sigma_0, \sigma_1, \sigma_3\} = \mu_{12}$
$\Delta'(\mu_8, b) = \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) = \{\} \cup \{\sigma_1\} = \{\sigma_1\} = \mu_2$
$\Delta'(\mu_9, a) = \Delta(\sigma_1, a) \cup \Delta(\sigma_3, a) = \{\sigma_1, \sigma_3\} \cup \{\sigma_2\} = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$
$\Delta'(\mu_9, b) = \Delta(\sigma_1, b) \cup \Delta(\sigma_3, b) = \{\} \cup \{\sigma_2, \sigma_3\} = \{\sigma_2, \sigma_3\} = \mu_{10}$
$\Delta'(\mu_{10}, a) = \Delta(\sigma_2, a) \cup \Delta(\sigma_3, a) = \{\sigma_0, \sigma_3\} \cup \{\sigma_2\} = \{\sigma_0, \sigma_2, \sigma_3\} = \mu_{13}$
$\Delta'(\mu_{10}, b) = \Delta(\sigma_2, b) \cup \Delta(\sigma_3, b) = \{\sigma_1\} \cup \{\sigma_2, \sigma_3\} = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$

Solución del ejemplo 7.26

$\Delta'(\mu_{11}, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) =$
$\{\sigma_1, \sigma_2\} \cup \{\sigma_1, \sigma_3\} \cup \{\sigma_0, \sigma_3\} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \mu_{15}$
$\Delta'(\mu_{11}, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) = \{\sigma_0, \sigma_3\} \cup \{\} \cup \{\sigma_1\} =$
$\{\sigma_0, \sigma_1, \sigma_3\} = \mu_{12}$
$\Delta'(\mu_{12}, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) \cup \Delta(\sigma_3, a) =$
$\{\sigma_1, \sigma_2\} \cup \{\sigma_1, \sigma_3\} \cup \{\sigma_2\} = \{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$
$\Delta'(\mu_{12}, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) \cup \Delta(\sigma_3, b) = \{\sigma_0, \sigma_3\} \cup \{\} \cup \{\sigma_2, \sigma_3\} =$
$\{\sigma_0, \sigma_2, \sigma_3\} = \mu_{13}$
$\Delta'(\mu_{13}, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_2, a) \cup \Delta(\sigma_3, a) =$
$\{\sigma_1, \sigma_2\} \cup \{\sigma_0, \sigma_3\} \cup \{\sigma_2\} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \mu_{15}$
$\Delta'(\mu_{13}, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_2, b) \cup \Delta(\sigma_3, b) =$
$\{\sigma_0, \sigma_3\} \cup \{\sigma_1\} \cup \{\sigma_2, \sigma_3\} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \mu_{15}$

Solución del ejemplo 7.26

$\Delta'(\mu_{14}, a) = \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) \cup \Delta(\sigma_3, a) =$
$\{\sigma_1, \sigma_3\} \cup \{\sigma_0, \sigma_3\} \cup \{\sigma_2\} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \mu_{15}$
$\Delta'(\mu_{14}, b) = \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) \cup \Delta(\sigma_3, b) = \{\} \cup \{\sigma_1\} \cup \{\sigma_2, \sigma_3\} =$
$\{\sigma_1, \sigma_2, \sigma_3\} = \mu_{14}$
$\Delta'(\mu_{15}, a) = \Delta(\sigma_0, a) \cup \Delta(\sigma_1, a) \cup \Delta(\sigma_2, a) \cup \Delta(\sigma_3, a) =$
$\{\sigma_1, \sigma_2\} \cup \{\sigma_1, \sigma_3\} \cup \{\sigma_0, \sigma_3\} \cup \{\sigma_2\} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \mu_{15}$
$\Delta'(\mu_{15}, b) = \Delta(\sigma_0, b) \cup \Delta(\sigma_1, b) \cup \Delta(\sigma_2, b) \cup \Delta(\sigma_3, b) =$
$\{\sigma_0, \sigma_3\} \cup \{\} \cup \{\sigma_1\} \cup \{\sigma_2, \sigma_3\} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \mu_{15}$

Solución del ejemplo 7.26

En resumen:

Δ'	a	b
μ_0	μ_0	μ_0
μ_1	μ_8	μ_7
μ_2	μ_9	μ_0
μ_3	μ_7	μ_2
μ_4	μ_3	μ_{10}
μ_5	μ_{14}	μ_7
μ_6	μ_{15}	μ_{12}
μ_7	μ_8	μ_{13}

Δ'	a	b
μ_8	μ_{12}	μ_2
μ_9	μ_{14}	μ_{10}
μ_{10}	μ_{13}	μ_{14}
μ_{11}	μ_{15}	μ_{12}
μ_{12}	μ_{14}	μ_{13}
μ_{13}	μ_{15}	μ_{15}
μ_{14}	μ_{15}	μ_{14}
μ_{15}	μ_{15}	μ_{15}

Solución del ejemplo 7.26

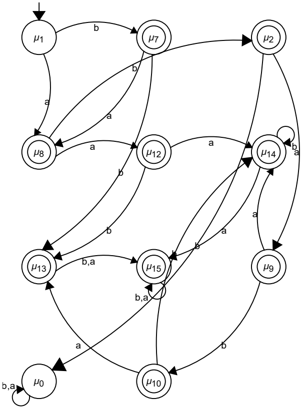
Ahora, con la finalidad de reducir el digrafo que representa a A', se obtiene la siguiente tabla de estados factibles a ser visitados, iniciando en μ_1 :

Δ	a	b
$\to \mu_0$	μ_0	μ_0
$\boxed{\mu_1}$	μ_8	μ_7
$\to \mu_2$	μ_9	μ_0
μ_3	μ_7	μ_2
μ_4	μ_3	μ_{10}
μ_5	μ_{14}	μ_7
μ_6	μ_{15}	μ_{12}
$\to \mu_7$	μ_8	μ_{13}

Δ	a	b
$\to \mu_8$	μ_{12}	μ_2
$\to \mu_9$	μ_{14}	μ_{10}
$\to \mu_{10}$	μ_{13}	μ_{14}
μ_{11}	μ_{15}	μ_{12}
$\to \mu_{12}$	μ_{14}	μ_{13}
$\to \mu_{13}$	μ_{15}	μ_{15}
$\to \mu_{14}$	μ_{15}	μ_{14}
$\to \mu_{15}$	μ_{15}	μ_{15}

Solución del ejemplo 7.26

Por consiguiente, se pueden eliminar cinco estados y sus lados de salida, al elaborar el diagrama de transición simplificado del autómata determinístico A' :



Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-184.zip

Example (7.27)

Considere el autómata de estado finito no determinístico del ejemplo 31, encuentre con ayuda de *Wolfram Mathematica* un *ADF* A' equivalente.

Solución del ejemplo 7.27

El algoritmo descrito en el teorema 32 se ha automatizado en la instrucción `AutomataDeterministicoEquivalente` del paquete **VilCretas**. El comando recibe un autómata no determinístico creado con la sentencia `Automata` y de forma opcional el atributo `colores -> True` ya explicado en la página 81, desplegando como salida el diagrama de transición del autómata no determinístico dado, las cinco componentes de A', su digrafo de representación completo y su diagrama reducido.

Solución del ejemplo 7.27

En el presente ejemplo, al utilizar `AutomataDeterministicoEquivalente` se obtiene:

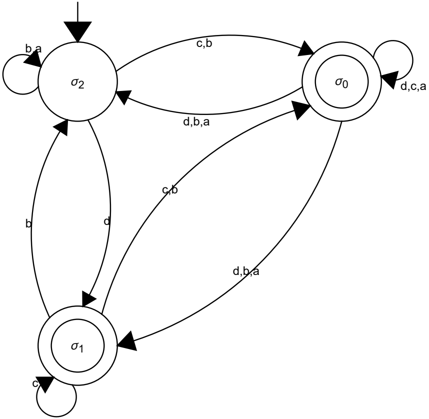
In[ ] :=

```
A = Automata[{σ0, σ1, σ2}, {a, b, c, d}, σ2,
{{σ0, a, {σ0, σ1, σ2}}, {σ0, b, {σ1, σ2}}, {σ0, c, {σ0}},
{σ0, d, {σ0, σ1, σ2}}, {σ1, a, {}}, {σ1, b, {σ0, σ2}},
{σ1, c, {σ0, σ1}}, {σ1, d, {}}, {σ2, a, {σ2}},
{σ2, b, {σ0, σ2}}, {σ2, c, {σ0}}, {σ2, d, {σ1}}}, {σ0, σ1}];
AutomataDeterministicoEquivalente[A]
```

Solución del ejemplo 7.27

Out[ ] =

Diagrama de transición del autómata no determinístico:



Solución del ejemplo 7.27

Estados del nuevo autómata:
$\mu_0 = \{\}$
$\mu_1 = \{\sigma_0\}$
$\mu_2 = \{\sigma_1\}$
$\mu_3 = \{\sigma_2\}$
$\mu_4 = \{\sigma_0, \sigma_1\}$
$\mu_5 = \{\sigma_0, \sigma_2\}$
$\mu_6 = \{\sigma_1, \sigma_2\}$
$\mu_7 = \{\sigma_0, \sigma_1, \sigma_2\}$
Estado inicial: μ_3

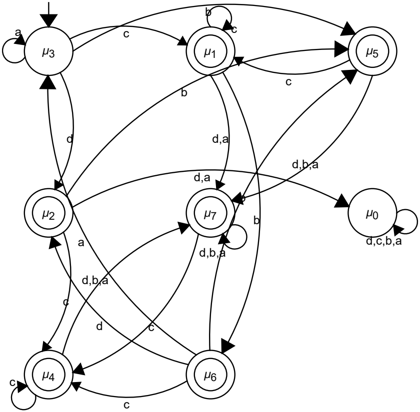
Solución del ejemplo 7.27

Estados aceptados: $\{\mu_1, \mu_2, \mu_4, \mu_5, \mu_6, \mu_7\}$

$$
\begin{array}{c|c}
 & \begin{array}{cccc} a & b & c & d \end{array} \\
\begin{array}{c} \mu_0 \\ \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6 \\ \mu_7 \end{array} &
\begin{pmatrix}
\mu_0 & \mu_0 & \mu_0 & \mu_0 \\
\mu_7 & \mu_6 & \mu_1 & \mu_7 \\
\mu_0 & \mu_5 & \mu_4 & \mu_0 \\
\mu_3 & \mu_5 & \mu_1 & \mu_2 \\
\mu_7 & \mu_7 & \mu_4 & \mu_7 \\
\mu_7 & \mu_7 & \mu_1 & \mu_7 \\
\mu_3 & \mu_5 & \mu_4 & \mu_2 \\
\mu_7 & \mu_7 & \mu_4 & \mu_7
\end{pmatrix}
\end{array}
$$

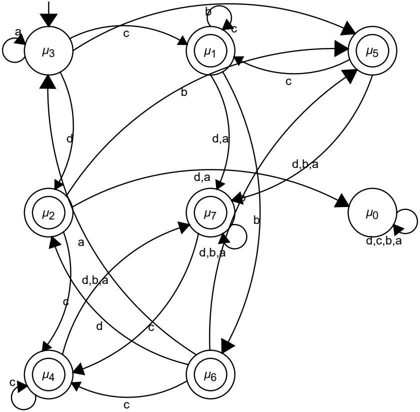
Solución del ejemplo 7.27

Diagrama de transición del nuevo autómata:



Solución del ejemplo 7.27

Diagrama de transición reducido del nuevo autómata:



Solución del ejemplo 7.27

En este ejercicio a diferencia de los ejemplos 33, 34 y 35 el estado inicial deja de ser μ_1 pues $\sigma^* = \sigma_2$ y por ende, el estado inicial de A' corresponde a $\{\sigma_2\} = \mu_3$. Otro aspecto importante a destacar, reside en el hecho de que el diagrama de transición de A' no es simplificable. Al comparar el digrafo completo y el diagrama reducido en el **Out[ ]** anterior, ambos son iguales, es decir, comenzando en el estado μ_3 es posible transitar por todos los estados restantes en el autómata A'.
Se sugiere al estudiante resolver este ejemplo también, sin ayuda del software *Mathematica*, con la intención de repasar una vez más, la aplicación del teorema 32. El alumno debe ser cauto en el empleo de la sentencia `AutomataDeterministicoEquivalente`, ésta cobra sentido (como otras instrucciones de **VilCretas**) siempre y cuando constituya única y exclusivamente un recurso de verificación de resultados.

Solución del ejemplo 7.27

Si el estudiante utiliza `AutomataDeterministicoEquivalente` de manera directa sin analizar por adelantado la solución, aportará muy poco en su proceso de aprendizaje.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/File-185.zip

Uso del comando `AutomataDeterministicoEquivalente`.

Explicación en video

`https://youtu.be/76X0AHeZc5g`

¡Recuerde siempre dar la milla extra!

Un tema interesante relacionado con las máquinas de estado finito y los autómatas que no será cubierto en este libro, consiste en las máquinas de *Turing*. El modelo de las máquinas de *Turing* fue concebido por el matemático inglés *Alan Turing* en el año de 1936.
En esta época aún no aparecía en el escenario científico, una computadora electromecánica u electrónica. *Turing* fue un visionario de su tiempo, al definir un modelo que establece si un problema dado es computable o indecidible (no resuelto por medio de un ordenador), en una fase de la historia de la humanidad, donde se carecía de la existencia física de estos dispositivos. En general, se presupone que cualquier proceso computable es equivalente a una máquina de *Turing*, es decir, existe una máquina de *Turing* que lo resuelve. Esta hipótesis se conoce como la tesis de *Church-Turing* y a pesar de no estar demostrada formalmente, suele ser aceptada de forma consensuada.

Cuaderno interactivo sobre los contenidos del capítulo.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Cuadernos/Maquinas.pdf.rar

Quiz interactivo de repaso sobre los contenidos del capítulo.

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/Quiz_maquinas.rar

Webmix sobre el empleo de los comandos del paquete **VilCretas** estudiados en el capítulo.

Abra un sitio web

https://www.symbaloo.com/mix/vilcretasmaquinas

¡Recuerde resolver los ejercicios asignados!

Descargue un archivo

https://www.escinf.una.ac.cr/discretas/Archivos/Maquinas/Exercises.zip

enrique.vilchez.quesada@una.cr
http://www.escinf.una.ac.cr/discretas