>>> IF013 - Fundamentos Teóricos de Informática
>>> Licenciatura de Sistemas - UNPSJB - Sede Trelew

Name: Celia Cintas[†], Pablo Navarro[‡], Samuel Almonacid[§]
Date: August 9, 2017

[†]cintas@cenpat-conicet.gob.ar, cintas.celia@gmail.com, @RTFMCelia
[‡]pnavarro@cenpat-conicet.gob.ar, pablo1n7@gmail.com
[§]almonacid@cenpat-conicet.gob.ar, almonacid.samuel.tw@gmail.com

>>> Unidad 1

1. Autómatas finitos. Reconocedores. Traductores. Diagrama de estados. Autómatas finitos no deterministas.
2. Equivalencia entre autómatas finitos deterministas y no deterministas. Morfismos sobre autómatas. Autómata Cociente.
3. Propiedades de lenguajes aceptados por Autómatas Finitos. Expresiones y lenguajes regulares.
4. Propiedades algebraicas de los lenguajes regulares. Equivalencia entre autómatas finitos y lenguajes regulares.
5. Teorema de Kleene. Gramáticas regulares. Relación entre gramáticas regulares y autómatas finitos.
6. Usos y aplicaciones de los autómatas finitos y lenguajes regulares.

>>> Autómatas Finitos No Determinísticos

Hay dos formas posibles de entender cómo funciona un AFND.

* Cuando hay varias alternativas, el AFND elige alguna de ellas.
* Imaginarse que el AFND está en varios estados a la vez. Si luego de leer la cadena puede estar en un estado final, acepta la cadena.

En cualquier caso, es bueno por un rato no pensar en cómo implementar un AFND.

Definición

Un AFND es la 5-upla $M = (K, \Sigma, \delta, S, F)$, donde K, Σ, δ y F tienen el mismo significado que en AFD, pero $\delta : K \times \Sigma \to P(K)$.

Definición

La función de transición se puede generalizar para que acepte cadenas en Σ, es decir $\hat{\delta} : K \times \Sigma^* \to P(K)$.

$$\hat{\delta}(q, \lambda) = \{q\}$$
$$\hat{\delta}(q, xa) = \{p : \exists r \in \hat{\delta}(q, x) | p \in \delta(r, a)\} \text{ con } x \in \Sigma^* \text{ y } a \in \Sigma$$

Definición

Se dice que una cadena x es aceptada por un AFND $M = (K, \Sigma, \delta, S, F)$ si y solo si $\delta(S, x) \cap F \neq \emptyset$

Definición

Dado un AFND $M = (K, \Sigma, \delta, S, F)$, el lenguaje aceptado por M, el cual se denotará $L(M)$, es el conjunto de cadenas aceptadas por M y se define como:

$$L(M) = \{x : \delta(S, x) \cap F \neq \emptyset\}$$

Podemos extender la función de transición aún más, haciendo que mapee conjuntos de estados y cadenas en conjuntos de estados, es decir:

Definición

Función de transición $\delta : P(K) \times \Sigma^* \to P(K)$, dada por:

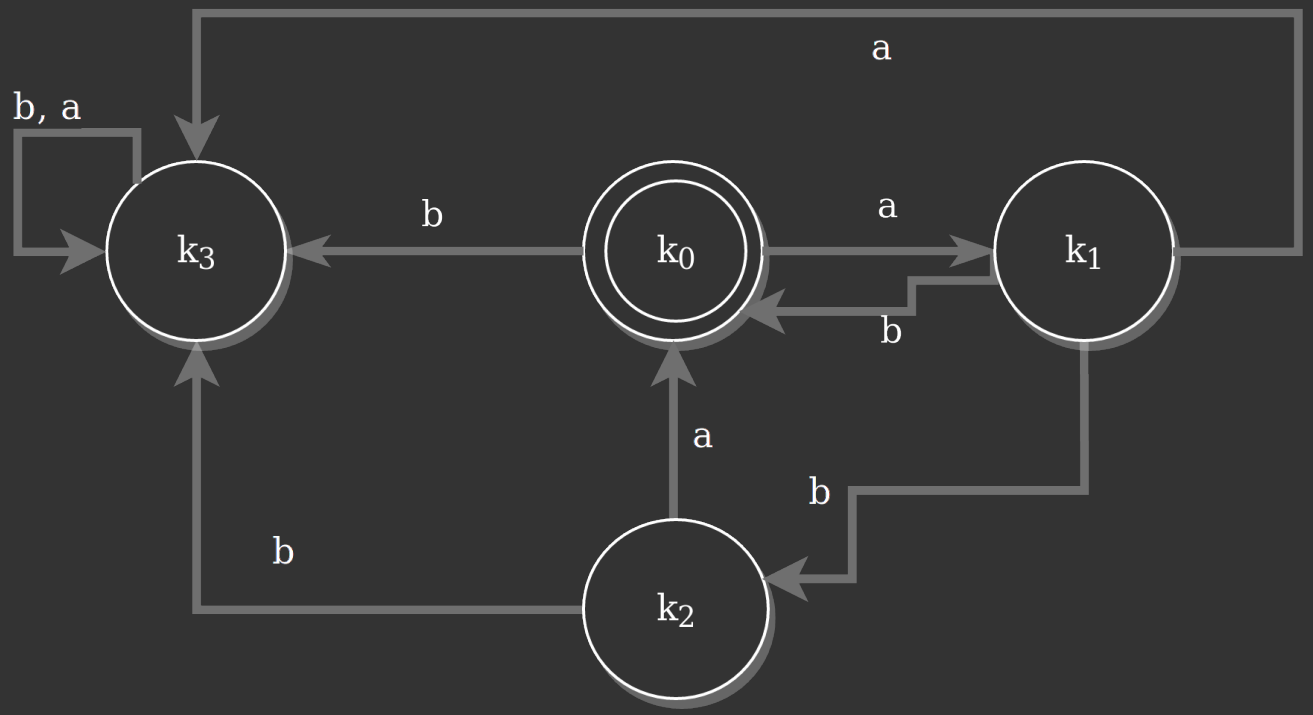
$$\delta(P, x) = \bigcup_{k \in P} \delta(k, x)$$

Para todo AFD existe un AFND y para cada AFND existe un AFD equivalente.

Ejemplo: AFND que acepte el lenguaje de las cadenas formadas por concatenación de 0 o más cadenas de ab o aba (sin importar el orden).

$$M = (K, \Sigma, \delta, S, F)$$
$$K = \{k_0, k_1, k_2, k_3\}$$
$$\Sigma = \{a, b\}$$
$$F = \{k_0\}$$
$$S = k_0$$

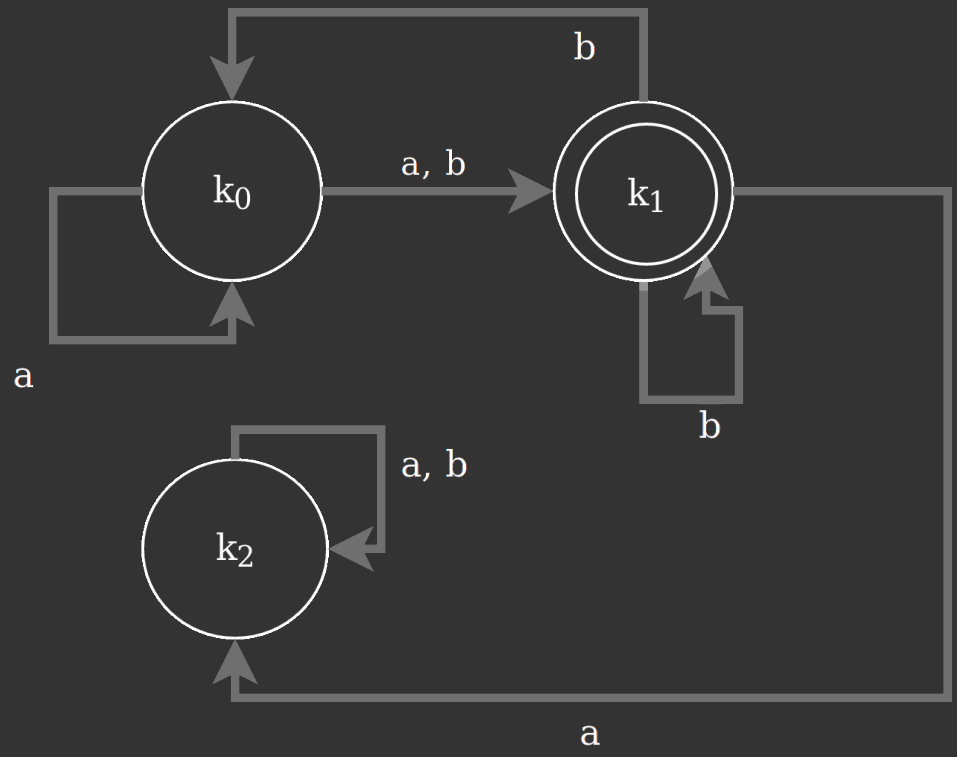
δ	a	b
k_0	$\{k_1\}$	$\{k_3\}$
k_1	$\{k_3\}$	$\{k_0, k_2\}$
k_2	$\{k_0\}$	$\{k_3\}$
k_3	$\{k_3\}$	$\{k_3\}$



Sea un AFND $M = (K, \Sigma, \delta, S, F)$ donde:

$$K = \{k_0, k_1, k_2\}$$
$$\Sigma = \{a, b\}$$
$$F = \{k_1\}$$
$$S = k_0$$

δ	a	b
k_0	$\{k_0, k_1\}$	$\{k_1\}$
k_1	$\{k_2\}$	$\{k_0, k_1\}$
k_2	$\{k_2\}$	$\{k_2\}$



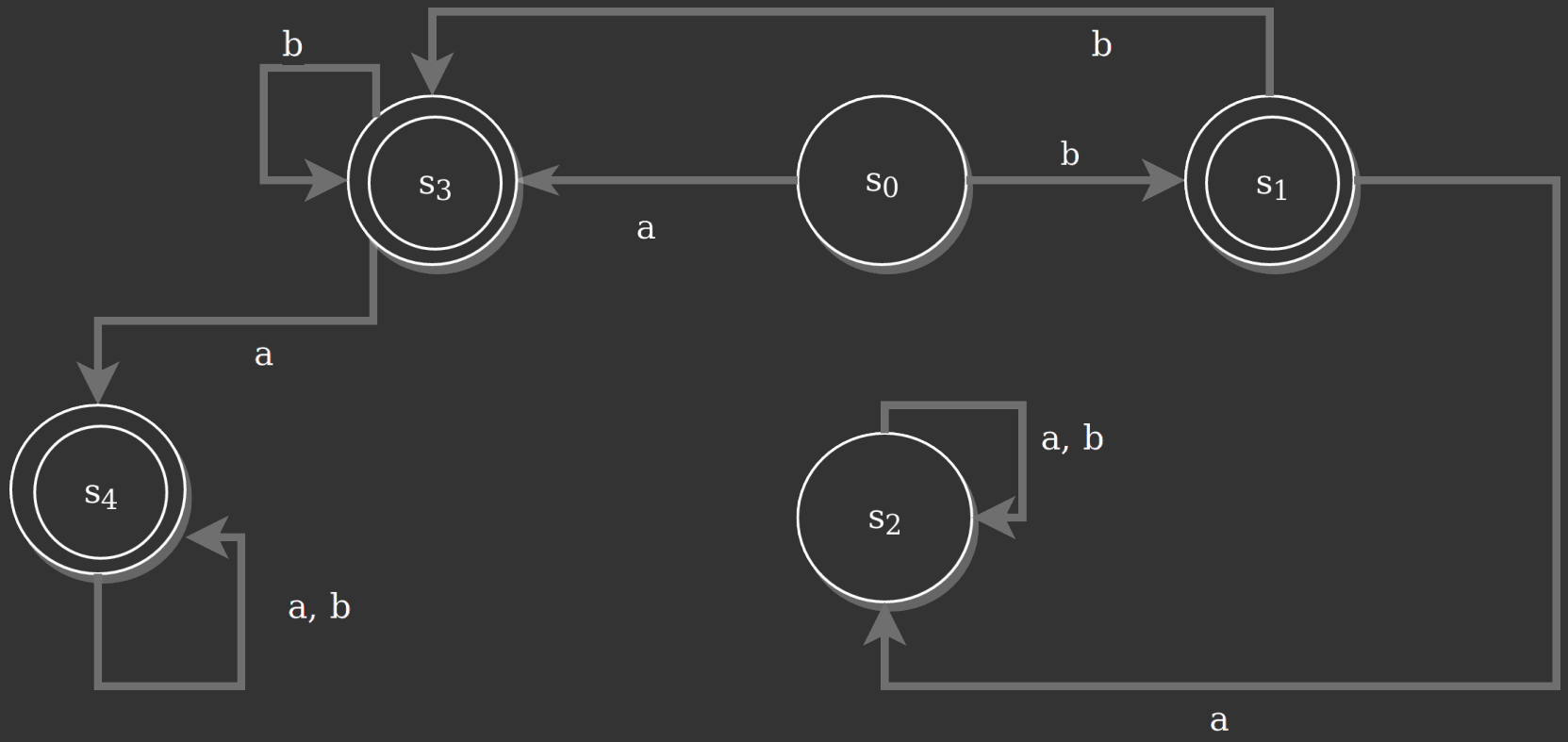
Obtengamos un AFD M' que reconozca el mismo lenguaje utilizando el teorema anterior.

Construyamos un AFD $M' = (K', \Sigma', \delta', S', F')$ donde:

$$K' = \{\{k_0\}, \{k_1\}, \{k_2\}, \{k_0, k_1\}, \{k_0, k_1, k_2\}\}$$
$$\Sigma' = \Sigma$$
$$F' = \{\{k_1\}, \{k_0, k_1\}, \{k_0, k_1, k_2\}\}$$
$$S' = \{k_0\}$$

δ	a	b	Aceptador
$s_0 = \{k_0\}$	$\{k_0, k_1\}$	$\{k_1\}$	0
$s_1 = \{k_1\}$	$\{k_2\}$	$\{k_0, k_1\}$	1
$s_2 = \{k_2\}$	$\{k_2\}$	$\{k_2\}$	0
$s_3 = \{k_0, k_1\}$	$\{k_0, k_1, k_2\}$	$\{k_0, k_1\}$	1
$s_4 = \{k_0, k_1, k_2\}$	$\{k_0, k_1, k_2\}$	$\{k_0, k_1, k_2\}$	1

>>> Conversión de AFND a AFD (cont.)



>>> Equivalencia entre AFD y AFND

Teorema

Sea L un lenguaje aceptado por un autómata finito no determinista, entonces existe un autómata finito determinista que también acepta L.

Demostración

Sea $M = (S, \Sigma, \delta, s_0, F)$ un autómata finito no determinista que acepta L. Para probar el teorema debemos probar:

* que podemos construir M' determinista, a partir de M.
* M' acepta el mismo lenguaje que M.

Demostración

Construimos un autómata finito determinista $M' = (S', \Sigma, \delta', s'_0, F')$ tal que :

* $S' \subseteq P(S)$. Los estados de M' serán identificados con subconjuntos del conjunto de estados de M. Notación: un elemento de S' se notará como $\langle s_1, s_2, \cdots s_i \rangle$ donde $\{s_1, s_2, \cdots, s_i\} \subseteq S$.
* $F' = \{s'_i \in S' | s'_i \supseteq \{s_j\}\ y\ s_j \in F\}$. F' es el conjunto de todos los estados de S' que contengan por lo menos un elemento de F (es decir, un estado aceptador de M).
* Σ es el alfabeto de entrada.
* $s'_0 = \langle s_0 \rangle = \{s_0\}$. s'_0 será el conjunto que contiene como único estado al inicial de M.

Demostración

la función δ' es la extensión de δ a conjuntos:
$\delta'(\langle s_1, s_2, \cdots, s_i \rangle, a) = \langle p_1, p_2, \cdots, p_j \rangle$ si y solo si
$\delta(\{s_1, s_2, \cdots, s_i\}, a) = \{p_1, p_2, \cdots, p_j\}$.

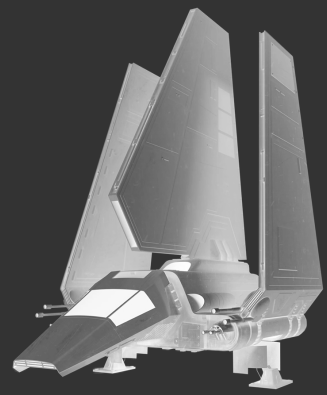
$$\delta'(\zeta, a) = \bigcup_{i=1}^{n} \delta(s_i, a), \text{ donde } \zeta = \langle s_1, s_2, \cdots, s_n \rangle$$

Con esto tenemos definido M' a partir de M.

Demostración

Ahora necesitamos probar que $\delta'(s'_0, x) = \langle q_1, q_2, \cdots, q_i \rangle$ si y solo si $\delta(s_0, x) = \{q_1, q_2, \cdots, q_i\}$ para toda cadena x.

* base inductiva: $long(x) = 0$, $x = \lambda$ y la demostración trivial es que $s'_0 = \langle s_0 \rangle$.
* paso inductivo: $long(x) = l$ es decir $\delta'(s'_0, x) = \langle p_1, p_2, \cdots, p_n \rangle$ si y solo si $\delta(s_0, x) = \{p_1, p_2, \cdots, p_n\}$, sea $a \in \Sigma$, debemos probar que vale para $long(xa) = l + 1$.
  Por definición de δ':
  $\delta'(s'_0, xa) = \delta'(\delta'(s'_0, x), a)$
  por hipótesis de inducción, tenemos:
  $\delta'(s'_0, x) = \langle p_1, p_2, \cdots, p_n \rangle$ si y solo si
  $\delta(s_0, x) = \{p_1, p_2, \cdots, p_n\}$, reemplazando la definición anterior tenemos que :
  $\delta'(\langle p_1, \cdots, p_n \rangle, a) = \langle q_1, \cdots, q_k \rangle$ si y solo si
  $\delta'(\{p_1, \cdots, p_n\}, a) = \{q_1, \cdots, q_k\}$



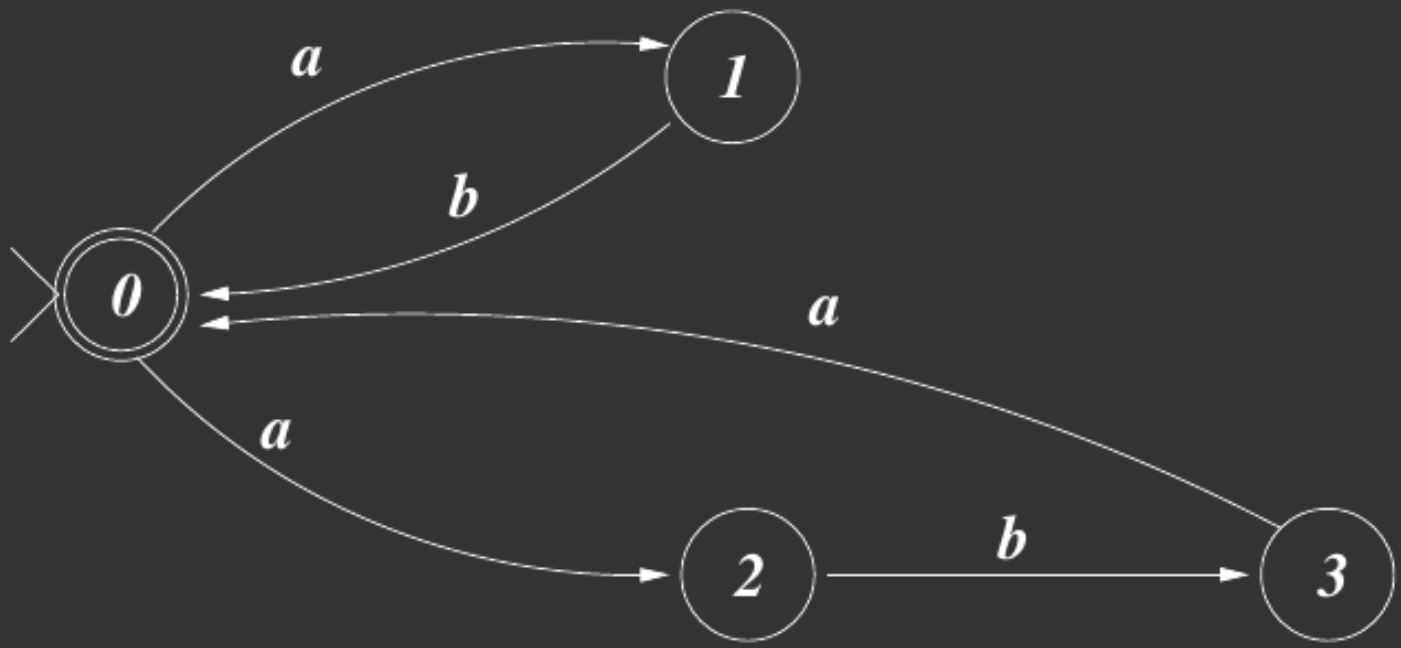
Definición

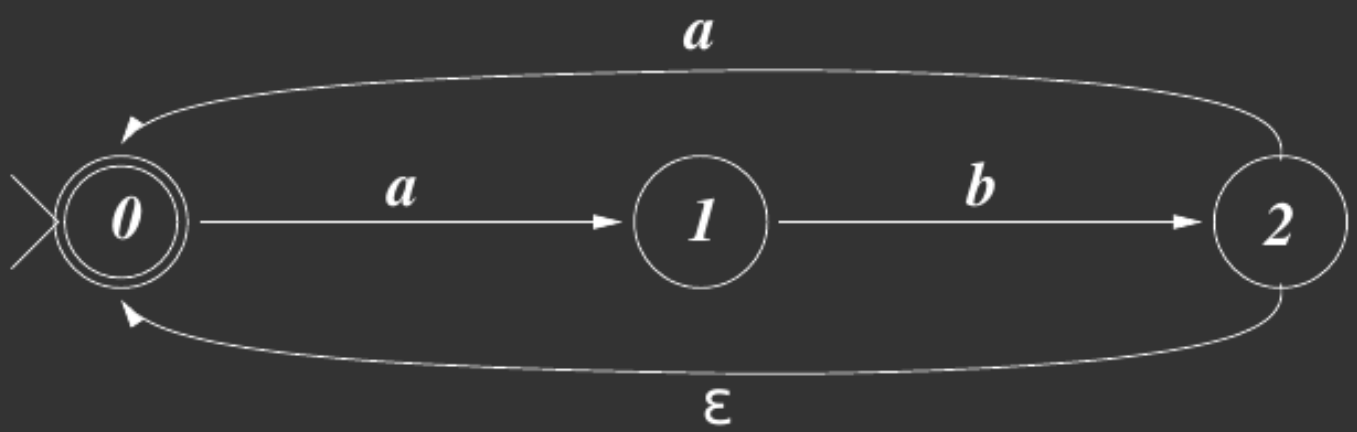
Un Autómata Finito No Deterministico con transiciones λ es una quintupla $(Q, \Sigma, \delta, q_0, F)$. Donde Q, Σ, q_0, F tienen el mismo significado que en AFND pero δ se define como:

$$\delta : Q \times (\Sigma \cup \lambda) \to P(Q)$$

Definición

La Clausura-λ de un estado q, se denota como $Cl_\lambda(q)$ es el conjunto de estados alcanzables desde q con transiciones λ. El estado q pertenece a su clausura.


a
1
b
0
a
a
2
b
3


a
0
a
1
b
2
ε

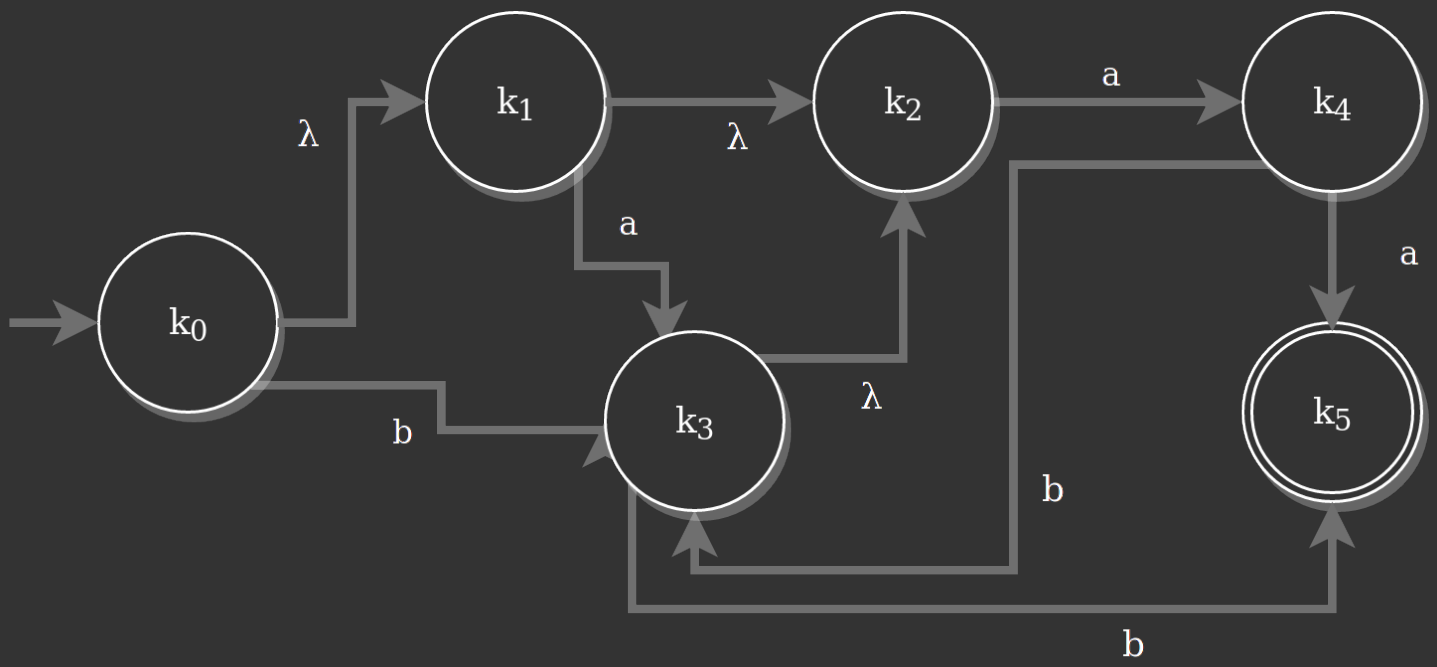
>>> Conversión Autómatas Finitos $AFND-\lambda$ a AFND

Dado un AFND-λ es posible construir un AFND equivalente sin transiciones vacías que reconoce el mismo lenguaje.

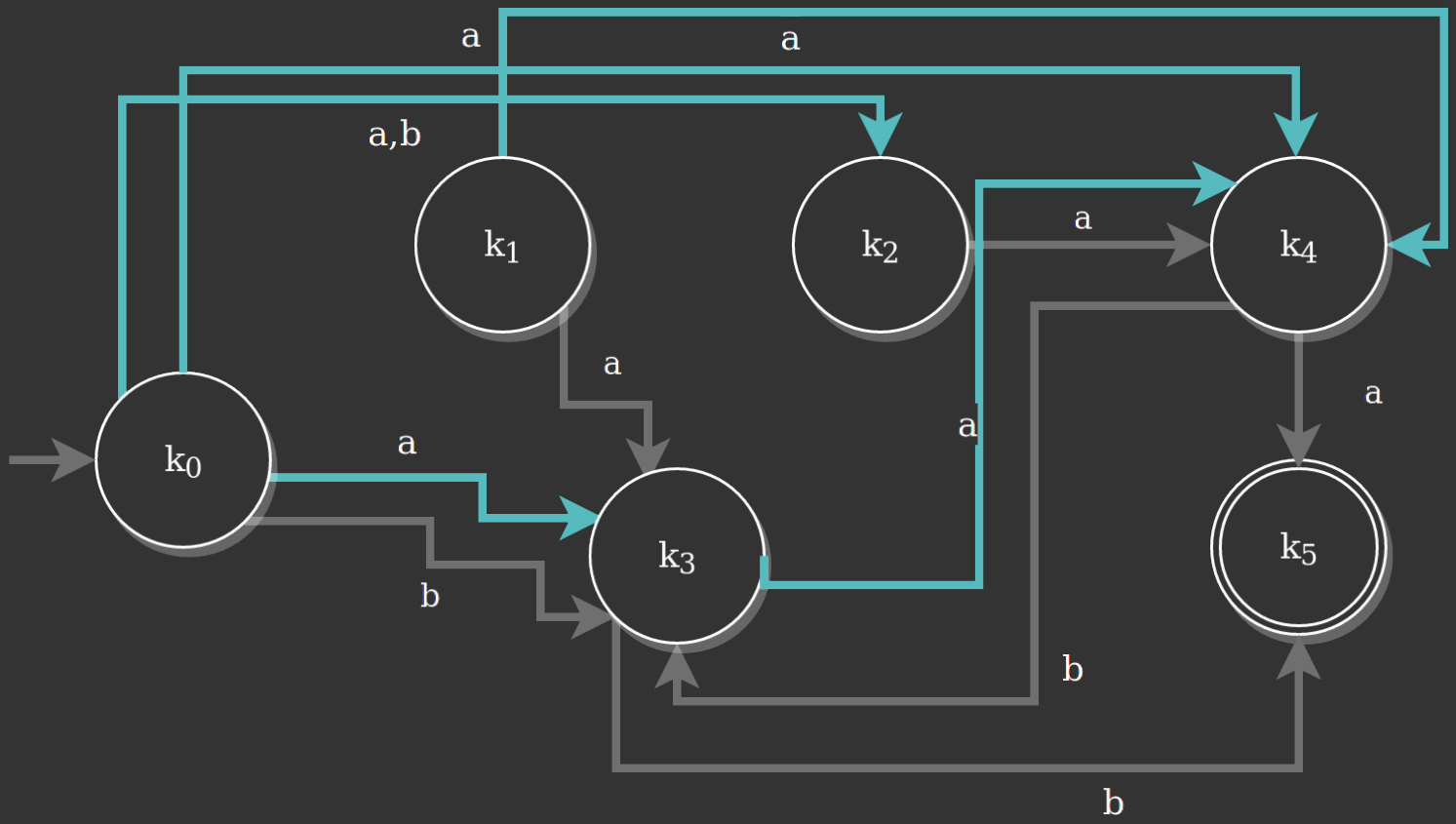
* Estados importantes del AFND$-\lambda$ y estado inicial del AFND$-\lambda$.
* Σ de AFND $=$ Σ de AFND$-\lambda$.
* Estado inicial de AFND = AFND$-\lambda$.
* $\delta(e_i, x) = e_k$ siendo e_i, e_k estados importantes.
* Estados finales del AFND : estados finales del AFND$-\lambda$ y todos los estados e_i del AFND$-\lambda$ para los cuales existe un camino con transiciones λ, en el AFND$-\lambda$, a algún estado final del AFND$-\lambda$.

>>> Conversión Autómatas Finitos $AFND - \lambda$ a AFND (Cont.)

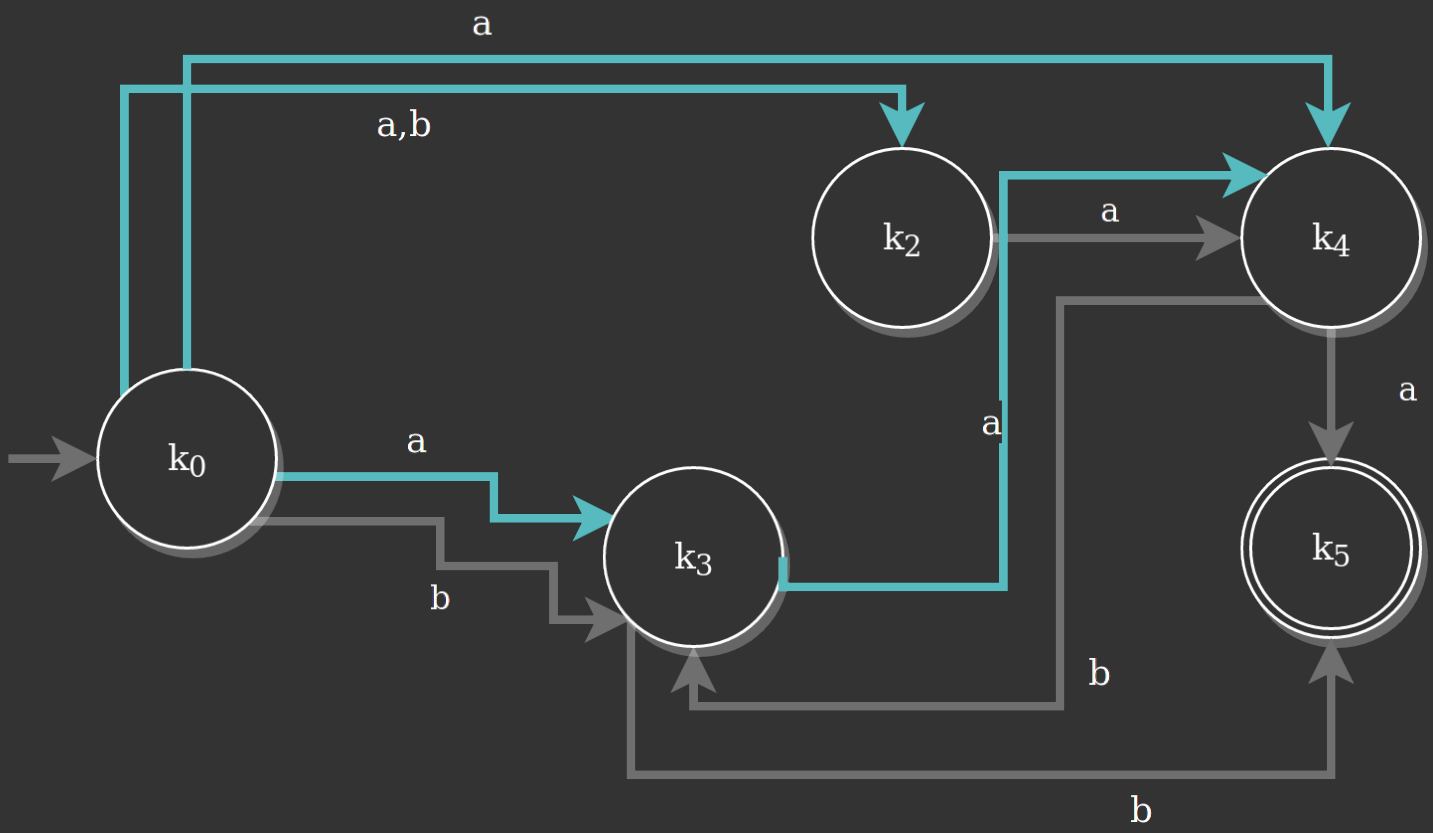
Cómo sería este Autómata sin transiciones λ ?



>>> Conversión Autómatas Finitos $AFND - \lambda$ a AFND (Cont.)



>>> Conversión Autómatas Finitos $AFND - \lambda$ a AFND (Cont.)

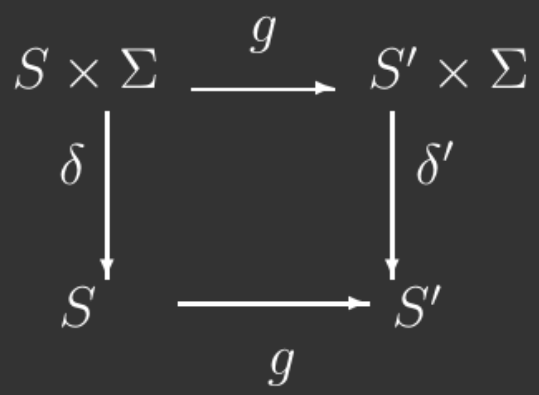


Definición

Sean $M = (S, \Sigma, s_0, \delta, f_0)$ y $M' = (S', \Sigma, s'_0, \delta', f'_0)$ dos AFD reconocedores.

Un homomorfismo g del autómata finito M al autómata finito M' es una función $g : S \to S'$ tal que para cualquier $i \in \Sigma$ y $s \in S$:

$$g(s_0) = s'_0$$
$$g(\delta(s, i)) = \delta'(g(s), i)$$
$$f_0(s) = f'_0(g(s))$$



Cómo esta definido f_0?

>>> Autómata Cociente

Definición

Sean $M = (S, \Sigma, s_0, \delta, f_0)$ y M' dos AFR y g un homomorfismo entre los autómatas M y M'. Denominaremos Autómata cociente $M/g = (S'', \Sigma, [s_0], \delta'', f_0'')$ al AFR tal que:

* S'' es el conjunto de estados de M/g, tal que cada uno es una clase de equivalencia $[s]$ de estados de M determinada por g. Todos los elementos de $[s]$ tienen asociado el mismo estado en M y la misma salida.
* Σ el alfabeto de entrada de M.
* El estado inicial $[s_0]$ es la clase del estado inicial de M.
* f_0'' es la función de salida.
* δ'' es la función de próximo estado definida mediante $\delta([s_j], i) = [\delta(s_j, i)]$.

Definición

Dado un autómata finito $M = (S, \Sigma, s_0, \delta, f_0)$ y un estado $s_i \in S$ tal que $s_i \neq s_0$, se dice que si es un estado inalcanzable de M si no existe una cadena $w \in \Sigma$ tal que $\delta(s_0, w) = s_i$.

$$M = (S, \Sigma, \delta, s_0, f_0)$$
$$S = \{s_0, s_1, s_2, s_3\}$$
$$\Sigma = \{0, 1\}$$

δ	0	1	f_0
s_0	s_1	s_3	0
s_1	s_3	s_0	0
s_2	s_1	s_3	1
s_0	s_0	s_1	1

Definición

Dos estados s_i, s_j de un AF M son equivalentes si para toda cadena de entrada, $w \in \Sigma^*$, $\delta(s_i, w) = s_k$ y $\delta(s_j, w) = s_l$, donde $f_0(s_k) = f_0(s_l)$.

Definición

Dos estados s_i, s_j de un AF M son k-equivalentes si para toda cadena de entrada, $w \in \Sigma^*$, con $long(w) \leq k$, $\delta(s_i, w) = s_k$ y $\delta(s_j, w) = s_l$, donde $f_0(s_k) = f_0(s_l)$.

>>> Minimización

El objetivo será aprender a establecer un homomorfismo entre dos AFD M y M', lo cual a su vez nos permitirá definir un método para simplificar autómatas. La idea es que dado un AF M, podamos estudiar si existe un AF M' que realize lo mismo que M pero tenga menos estados.

>>> Algoritmo de Minimización

Entrada: un AF $M = (S, \Sigma, \delta, s_0, F)$ determinista completo.
Salida: el AF M minimizado.

1. Eliminar del conjunto de estados S a los estados inalcanzables y llamar a este nuevo conjunto T .
2. Particionar T en dos clases formadas por los estados $0-$equivalentes. Por tratarse de un AF reconocedor tendremos una clase de estados aceptadores y otra de estados no aceptadore
3. Asumir $k = 0$.
4. Repetir:
   * Determinar las clases $(k+1)-$equivalentes como un refinamiento de las k equivalentes, es decir: s_i, s_j son $(k+1)-$equivalentes si y solo si s_i, s_j son $k-$equivalentes y $\delta(s_i, i)$, $\delta(s_j, i)$ son $k-$equivalentes, para todo $i \in \Sigma$.
   * Incrementar k en 1. hasta que las clases $(k+1)-$equivalentes sean iguales a las $k-$equivalentes.
5. Usar las clases $k-$equivalentes determinadas para definir el autómata cociente de M.

$$M = (S, \Sigma, \delta, s_0, f_0)$$
$$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$$
$$\Sigma = \{0, 1\}$$

δ	0	1	f_0
s_0	s_2	s_3	0
s_1	s_3	s_2	1
s_2	s_0	s_4	0
s_3	s_1	s_5	1
s_4	s_6	s_5	1
s_5	s_2	s_0	0
s_6	s_4	s_0	1

$T = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$

M tiene estados inalcanzables? ($T = S$)

$$K_0 = \{s_0, s_2, s_5\}\{s_1, s_3, s_4, s_6\}$$
$$K_1 = \{s_0, s_2\}\{s_5\}\{s_1, s_3, s_4, s_6\}$$
$$K_2 = \{s_0, s_2\}\{s_5\}\{s_1, s_6\}\{, s_3, s_4\}$$
$$K_3 = \{s_0, s_2\}\{s_5\}\{s_1, s_6\}\{, s_3, s_4\}$$

Definición

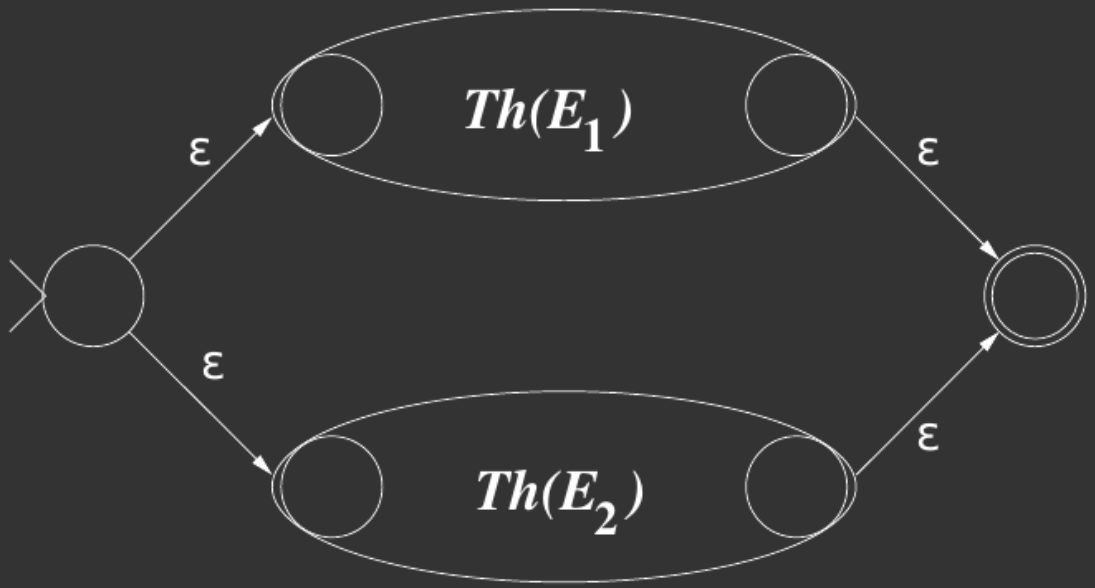
La función Th convierte ERs en AFNDs según las siguientes reglas:

* Para $c \in \Sigma$, $Th(c)$:

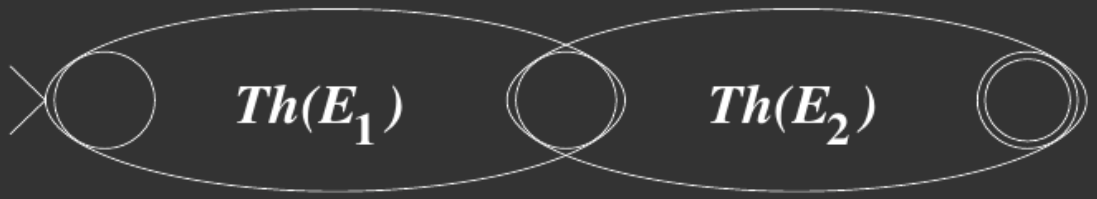
* Para $Th(\emptyset)$:

Definición

* Para $Th(E_1|E_2)$:

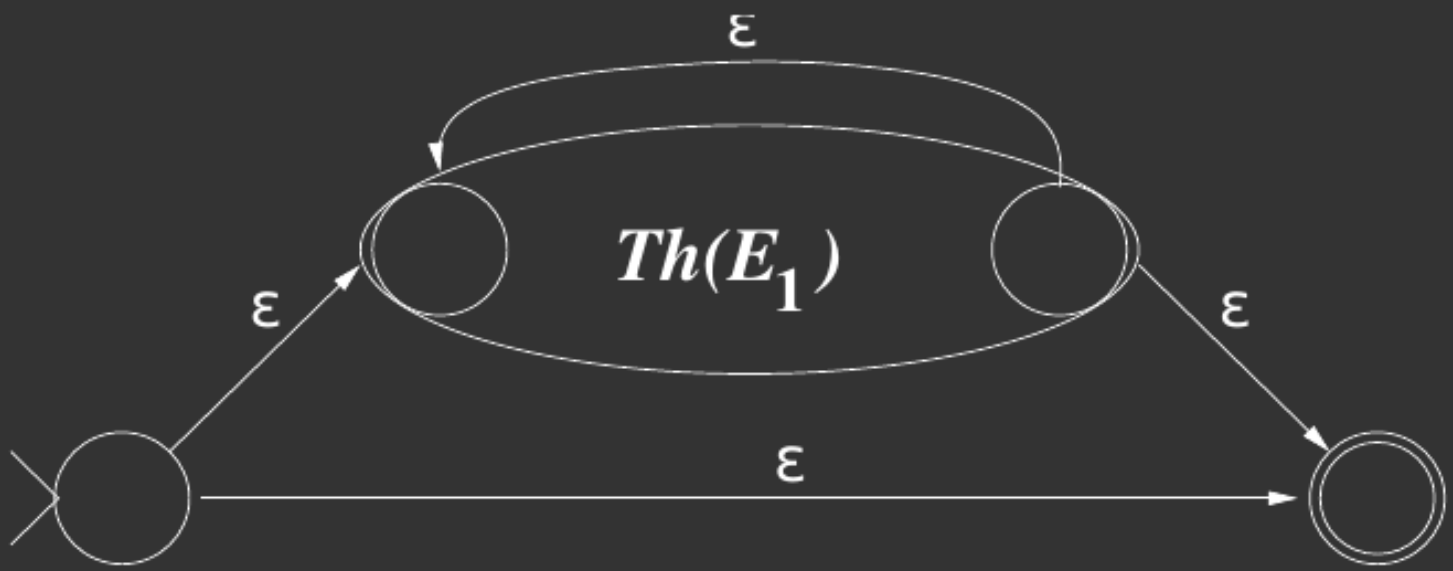


* Para $Th(E_1.E_2)$:



Definición

* $Th(E_1^*)$:

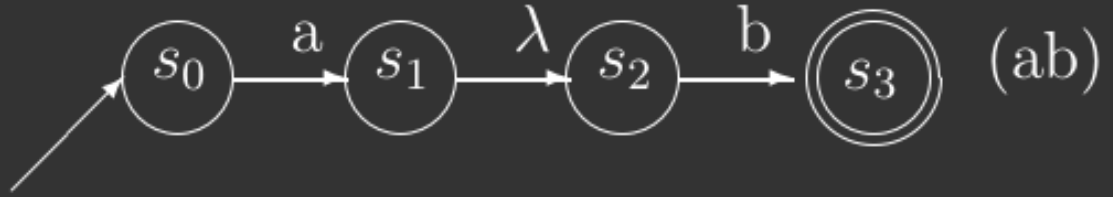


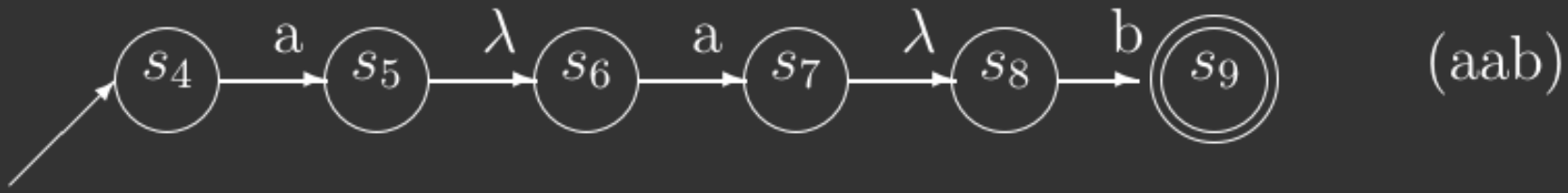
* $Th((E_1)) = Th(E_1)$

$$(ab|aab)^*$$

* a, b:

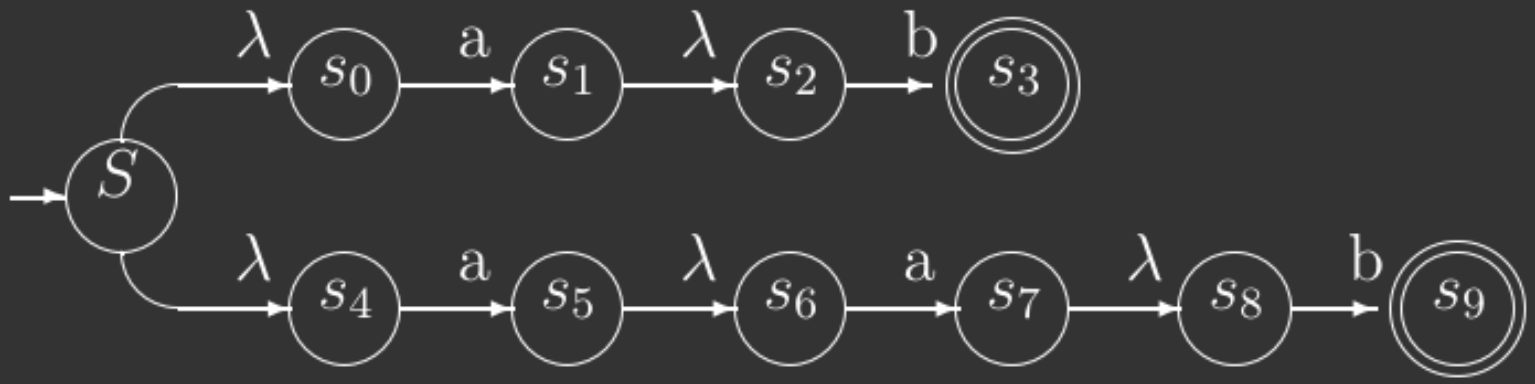
* concatenación para obtener ab y aab:



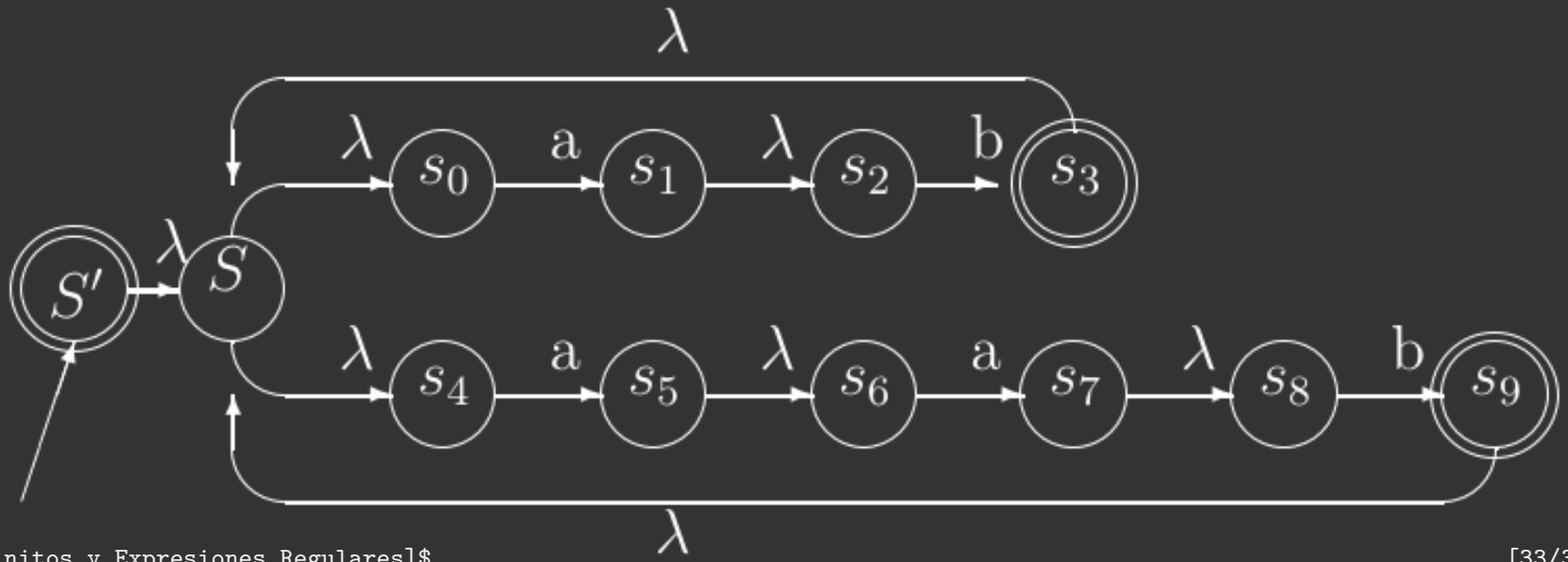


$$(ab|aab)^*$$

* unión para obtener $(ab + aab)$:



* estrella de Kleene para obtener $(ab + aab)^*$



Teorema

Dado un AFD $M = (\{q_0, \cdots, q_n\}, \Sigma, \delta, q_0, F)$ que acepta el lenguaje L, existe una ER que denota el mismo lenguaje.

Definición

Si ω, β y γ son expresiones regulares sobre un alfabeto Σ, $\gamma \neq \lambda$, entonces la ecuación $\omega = \beta + \omega\gamma$ tiene una única solución y ésta es $\omega = \beta\gamma^*$. (Algoritmo I)

Definición

Si ω, β y γ son expresiones regulares sobre un alfabeto Σ, $\gamma \neq \lambda$, entonces la ecuación $\omega = \beta + \omega\gamma$ tiene una única solución y ésta es $\omega = \gamma^*\beta$. (Algoritmo II)

>>> Algoritmo I para determinar ER a partir de AF

Entrada: un AF, $M = (S, \Sigma, s_0, , F)$
Salida: una ER que denota el Lenguaje Regular reconocido por M

* Se plantea una ecuación por cada estado, como unión de n términos. Cada término representa un tipo de cadena que llega al estado. Estos términos son la concatenación de la variable correspondiente al estado origen del arco con el símbolo que rotula al arco. Para $\delta(s_j, a) = s_i$, uno de los términos de la ecuación ω_{s_i} será: $\omega_{s_j} \cdot a$. En la ecuación planteada para el estado inicial se agrega el término λ.
* Se despejan las ecuaciones según el lema: Si $\omega = \beta + \omega\gamma$ entonces $\omega = \beta\gamma^*$
* La ER es la unión de las soluciones para todos los estados aceptadores del AF.

Ejercicio con Método I: Sea $\Sigma = \{a, b\}$ y $L = \{w \in \Sigma^* | w \; tiene \; 3k + 1 bs, \; k \geq 0\}$

$$M = (S, \Sigma, \delta, s_0, F)$$
$$S = \{s_0, s_1, s_2\}$$
$$\Sigma = \{a, b\}$$
$$F = \{s_1\}$$

δ	a	b
s_0	s_0	s_1
s_1	s_1	s_2
s_2	s_2	s_0

>>> Algoritmo II para determinar ER a partir de AF

Entrada: un AF, $M = (S, \Sigma, s_0, , F)$
Salida: una ER que denota el Lenguaje Regular reconocido por M

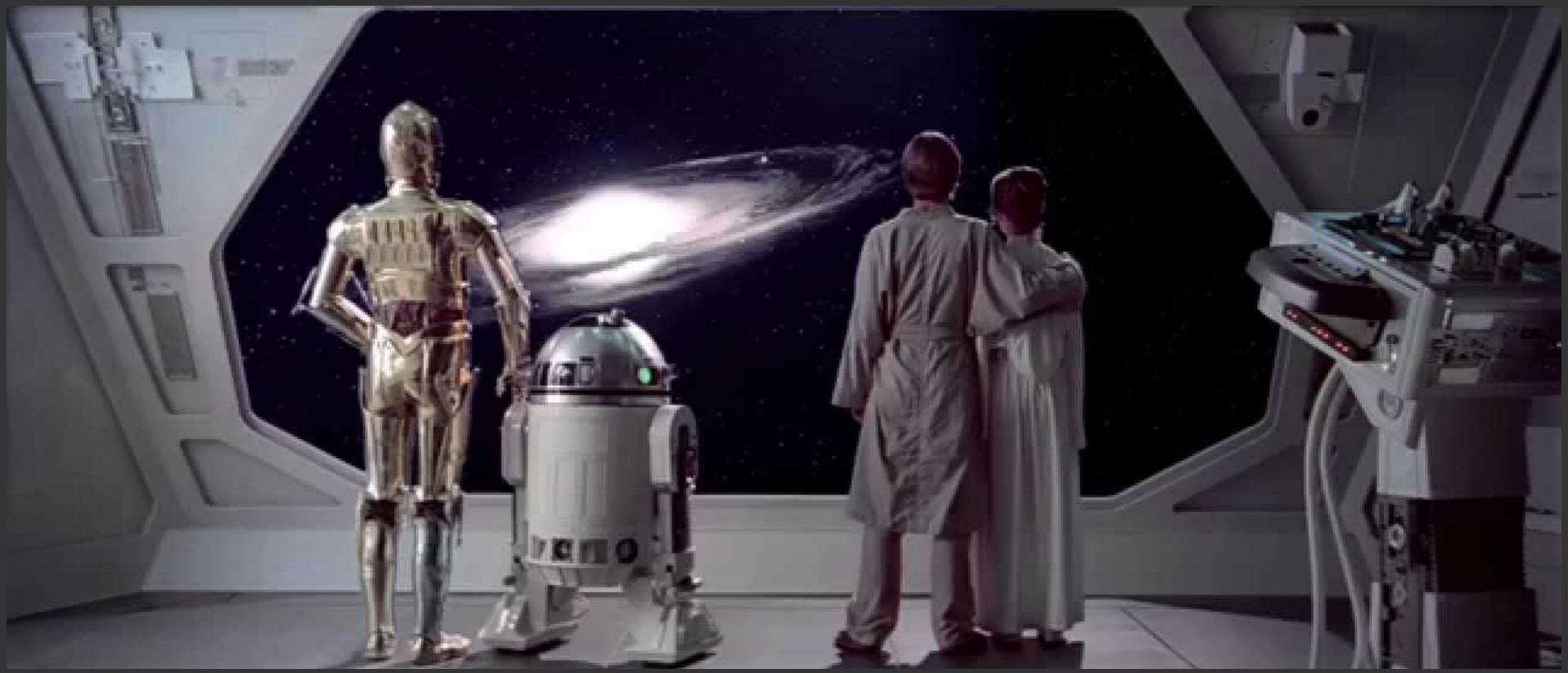
* Se plantea una ecuación por cada estado, como unión de n términos. Cada término representa un tipo de cadena que parte del estado; por lo tanto, existe un término por cada arco saliente. Estos términos son la concatenación del símbolo que rotula al arco y la variable correspondiente al próximo estado.
* En la ecuación planteada para el estado aceptador se agrega el término λ.
* Se despejan las ecuaciones según el lema: Si $\omega = \beta + \gamma\omega$ entonces $\omega = \gamma^*\beta$
* La ER se encuentra en el estado inicial del AF.

Ejercicio con Método II: Sea $\Sigma = \{a, b\}$ y $L = \{w \in \Sigma^* | w \; tiene \; 3k + 1 bs, \; k \geq 0\}$

$$M = (S, \Sigma, \delta, s_0, F)$$
$$S = \{s_0, s_1, s_2\}$$
$$\Sigma = \{a, b\}$$
$$F = \{s_1\}$$

δ	a	b
s_0	s_0	s_1
s_1	s_1	s_2
s_2	s_2	s_0

>>> Gracias!



Bibliografía

1. Introduction to Automata Theory, Languages, and Computation - Hopcroft et. al 2007 (3er ed.)
2. Teoría de la Computación - Gonzalo Navarro 2011.
3. Fundamentos de Cs. de la Computación - Juan Carlos Augusto 1995.